

PROTOTIPO HOMOS 2.0:
HERRAMIENTA SOFTWARE PARA EL MODELADO Y SIMULACIÓN
BASADO EN OBJETOS Y REGLAS

ANGEL MARIA VALDES PEÑA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA

2011

PROTOTIPO HOMOS 2.0:
HERRAMIENTA SOFTWARE PARA EL MODELADO Y SIMULACIÓN
BASADO EN OBJETOS Y REGLAS

ANGEL MARIA VALDES PEÑA

Trabajo de grado presentado para optar al título de Ingeniero de Sistemas

Director: Ing. HUGO HERNANDO ANDRADE SOSA
Codirector: Ing. MERLY SULGEY GÓMEZ SÁNCHEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA

2011

Dedicatoria

*Para todos aquellos interesados en la herramienta software Homos
y
la metodología de modelamiento basado en objetos y reglas, MBOR.*

Agradecimientos

Doy infinitas gracias a Nuestro Señor Jesucristo bendito por permitirme llegar a este momento de la vida, por los padres que tengo, los hermanos, los amigos, las experiencias, las oportunidades y las demás cosas que me ha dado, que aunque en su mayoría me resulten desconocidas, en su sabiduría así las ha dispuesto.

Doy gracias a mis padres Angel Maria y Ana Vicenta, a mis hermanos Ana Milena, Elizabeth y Miguel Angel, a mis sobrinos Juan Camilo & Santiago y a todos los amigos que con su comprensión, soporte y ayuda lograron que finalmente redactara y terminara este trabajo.

Doy gracias a los ingenieros Oscar Lozano Avellaneda y Carmen Duarte Mogotocoro quienes con su tesis de grado hicieron posible encontrar una idea que explorar y mejorar.

Doy gracias a mi director de proyecto el Ing. Hugo Hernando Andrade Sosa quien me permitió la oportunidad de investigar sobre la MBOR y Homos 1.0, por el soporte y la buena energía que me dió, por haber creído en mí todo este tiempo.

Doy gracias a la Ing. Merly Sulgey Gomez Sánchez por su ayuda y orientación.

Doy gracias a la Ing. Hina Luz Garavito Robles por su cercanía, por su permanente motivación y apoyo.

Doy gracias a el Ing. Oscar Infante por su interés y apoyo incondicional.

Y a todos aquellos que participaron directa e indirectamente con sus ideas y trabajo en la realización de este proyecto y la redacción final de este documento.

¡Muchas Gracias!

Sin la presencia de todos estos amigos, este proyecto no habría sido posible.

Contenido

1	Problema de investigación	34
1.1	Descripción del problema	34
1.2	Formulación del problema	36
1.3	Justificación	36
1.3.1	Impacto	36
1.3.2	Viabilidad	37
1.3.3	Entidades interesadas en el proyecto	37
1.4	Delimitación conceptual	38
1.5	Objetivos	40
2	Marco teórico	41
2.1	Metodología de modelamiento basada en objetos y reglas	41
2.2	Marco referencial	43
2.3	Marco conceptual	45
2.3.1	Autómatas celulares y teoría de objetos	45
2.3.2	Teoría de grafos	45
2.3.3	Teselados	45
2.3.4	Algebra de procesos	46
2.3.5	Pensamiento combinatorio	46
3	Diseño metodológico	47
3.1	Tipo de investigación	47
3.2	Población	47

3.2.1	Objeto de estudio	47
3.3	Técnicas e instrumentos para la recopilación de datos	47
3.3.1	Metodología de investigación	48
3.4	Prueba piloto	50
3.5	Herramientas metodológicas adicionales	50
3.5.1	Metodología de desarrollo de software	50
4	Presentación y análisis de resultados	51
4.1	Descripción de resultados	51
4.1.1	Organización del documento	51
4.1.2	Proceso de investigación realizado	53
4.1.3	Resultados Teóricos obtenidos	53
4.1.4	Resultados prácticos obtenidos	55
4.1.5	Resultados software	55
4.2	Análisis de resultados obtenidos	58
5	Propuesta de diseño general de una siguiente versión del software Homos	59
5.1	¿Qué características nuevas deben ser incluidas en la MBOR y la herramienta software para aumentar sus capacidades de representación?	60
5.1.1	Representación escrita de modelos BOR, LBOR	60
5.1.2	útiles de modelado	60
5.1.3	Cambios en la representación del diagrama BOR	67
5.1.4	Implementación de nuevas funciones en la herramienta software	68
5.2	¿Cuales son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?	69
5.3	¿Cómo mantener el espíritu original de la aplicación ante estos cambios?	71
5.4	El espíritu de la MBOR y Homos 1.0	71
5.4.1	Evolución continuada de la herramienta software Homos	71
5.4.2	Proyección de investigación a largo plazo	71
6	Conclusiones	74
6.1	Propósitos cumplidos	74
6.2	Respuesta a inquietudes de investigación	75

7	Recomendaciones	77
7.1	Elementos de continuidad	77
7.2	Por hacer	78
7.2.1	Acciones inmediatas pendientes de realización	79
7.3	Mirada al futuro	79
7.3.1	por hacer grf	80
7.4	Continuidad de investigación	80
	Bibliografía	82
A	Elementos de análisis del software Homos 1.0	96
A.1	Contacto con la herramienta software	99
A.1.1	Licencia	99
A.1.2	Idiomas disponibles	99
A.1.3	Presentación y adquisición del software	100
A.1.4	Detalles de instalación del software	100
A.1.5	Modos de funcionamiento de la herramienta	102
A.1.6	Resultados obtenidos	104
A.2	Recursos y materiales disponibles	105
A.2.1	Archivos utilizados por la aplicación	105
A.2.2	Documentación particular de la aplicación	106
A.2.3	Entradas en el registro de Windows	109
A.2.4	Documentos y referencias externas conocidas	112
A.2.5	Resultados obtenidos	112
A.3	Análisis funcional	114
A.3.1	Pruebas de software	114
A.3.2	Resultados obtenidos de las pruebas de software	116
A.3.3	Métricas de código fuente	128
A.3.4	Estructura interna y algoritmos usados	131
A.3.5	Aplicaciones adicionales	137
A.3.6	Fallas implícitas en el software	140

A.3.7	Herramientas de evaluación de métricas usadas en este documento	141
A.3.8	Resultados obtenidos	143
A.4	Análisis de Interfaz gráfica	146
A.4.1	Elementos componentes	147
A.4.2	Descripción de los elementos que conforman la Interfaz	147
A.4.3	Interfaz gráfica y secuencia de operaciones en modelado con MBOR	157
A.4.4	Pertenencia de iconos en interfaz gráfica	165
A.4.5	Resultados obtenidos	171
A.5	Análisis de la interfaz de automatización OLE	174
A.5.1	Propósito de la interfaz	174
A.5.2	Documentación sobre la interfaz	174
A.5.3	Instalación y desinstalación de la interfaz OLE en Homos 1.0	175
A.5.4	Resumen de operaciones disponibles con la interfaz	176
A.5.5	Etapas en la creación de un modelo mediante la interfaz OLE	177
A.5.6	Organización de funciones disponibles según etapas de implementación del modelo	178
A.5.7	Descripción de funciones y procedimientos disponibles	184
A.5.8	Método de construcción de modelos script	194
A.5.9	Detalles de implementación práctica de modelos script	196
A.5.10	Ejemplo de implementación de un modelo script	200
A.5.11	Extensión de capacidades de modelado mediante programación	203
A.5.12	Resultados del análisis	208
A.6	Análisis a la Metodología BOR	209
A.6.1	Fundamentos de la Metodología de modelamiento basada en objetos y reglas MBOR	209
A.6.2	proceso de modelamiento	218
A.6.3	Limitaciones presentes en la MBOR	221
A.6.4	Protonotación y análisis de transiciones	221
A.6.5	Sub elementos dentro del proceso de modelamiento	221
A.6.6	Resultados del análisis	266
A.7	Análisis al diagrama BOR	272

A.7.1	Elementos gráficos de la notación	272
A.7.2	Observaciones generales	273
A.7.3	Resultados del análisis	275
B	Elementos de evaluación del Software Homos 1.0	280
B.1	Evaluación mediante criterio interno	281
B.1.1	Primer formato interno de evaluación de software	281
B.1.2	Segundo formato interno de evaluación de software	285
B.2	Evaluación mediante criterio externo	291
B.2.1	Resultados obtenidos	291
B.3	Comparación con respecto a programas similares	297
B.3.1	Water	297
B.3.2	WorldMaker	299
B.4	Resultado de evaluaciones	300
B.5	¿Cuáles son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?	302
B.6	Criterios de mínimos de evaluación de Homos 1.0	302
B.6.1	¿Cuáles son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?	304
B.6.2	¿Que es evaluar?	306
B.6.3	¿Que evaluar de Homos 1.0?	307
C	Grafómatas: Autómatas de movimiento	309
C.1	Origen y descripción básica	310
C.1.1	Definición de grafómata	311
C.2	Grafómatas y construcciones relacionadas	313
C.2.1	Construcción manual de un grafómata	316
C.2.2	Configuración de un debugrafo	317
C.2.3	Construcción del dual de un grafómata	319
C.3	Selección de funciones del objeto Tgrafómata	330
C.3.1	Funciones útiles en la construcción manual de grafómatas	330
C.3.2	Funciones de construcción de un grafómatas	331

C.3.3	Funciones relacionadas con la construcción del grafomata dual	332
C.3.4	Funciones de uso del grafomata	333
C.3.5	Funciones de búsqueda de vértices en el espacio de teselado	333
C.3.6	Movimiento sobre el espacio teselado	334
C.3.7	Ejemplo de uso del objeto Tgrafomata	335
C.4	Estructuras implementadas	337
C.4.1	Grafómata 3.3.3.3.3, malla de triángulos equiláteros	339
C.4.2	Grafómata dual 3.3.3.3.3	340
C.4.3	Grafómata 6.6.6, malla de hexágonos	341
C.4.4	Grafómata dual 666	342
C.4.5	Grafómata 3.3.3.4.4	343
C.4.6	Grafómata dual 3.3.3.4.4	345
C.4.7	Grafómata el Cairo	346
C.4.8	Grafómata dual el Cairo, 3.3.4.3.4	348
C.4.9	Grafómata 4.8.8	349
C.4.10	Grafómata dual 4.8.8	351
C.4.11	Grafómata 3.6.3.6	352
C.4.12	Grafómata dual 3.6.3.6	354
C.4.13	Grafómata 3.3.3.3.6	355
C.4.14	Grafómata dual 3.3.3.3.6	357
C.4.15	Grafómata 3.4.6.4	359
C.4.16	Grafómata dual 3.4.6.4	361
C.4.17	Grafómata 4.6.12	362
C.4.18	Grafómata dual 4.6.12	365
C.4.19	Grafómata 3.12.12	366
C.4.20	Grafómata dual 3.12.12	369
C.4.21	Grafómata Variación en triángulos	370
C.4.22	Grafómata dual variación en triángulos	372
C.4.23	Grafómata Variación en 4.8.8	373
C.4.24	Grafómata dual variación en 4.8.8	374

C.4.25	Grafómata variación en 3.12.12	375
C.4.26	Grafómata dual variación 3.12.12	376
C.4.27	Grafómata variación el Cairo	377
C.4.28	Grafómata dual variación el Cairo	379
C.4.29	Grafómata pentágonos y octágonos	380
C.4.30	Grafómata dual pentágonos y octágonos	382
C.4.31	Grafómata pentágonos y rombos	383
C.4.32	Grafómata dual pentágonos y rombos	385
C.4.33	Grafómata Cuadrados y hexágonos	386
C.4.34	Grafómata dual cuadrados y hexágonos	388
C.4.35	Grafómata Subcuadros o Waffles	389
C.4.36	Grafómata dual subcuadros	391
C.4.37	Grafómata Molinetes	392
C.4.38	Grafómata dual molinetes	394
C.4.39	Grafómata triángulos y hexágonos	395
C.4.40	Grafómata dual triángulos y hexágonos	397
C.4.41	Grafómata 4.4.4.4, malla cuadrada, vecindad de Von Newman	398
C.4.42	Grafómata vecindario de Moore 9 celdas	399
C.4.43	Grafómata vecindario de Moore 25 celdas	400
C.4.44	Grafómata vecindario de Moore 49 celdas	401
C.4.45	Generador de vecindarios de Moore	402
C.5	Caracterización de grafómatas	404
C.5.1	Conectividad entre vértices de un teselado	408
C.6	Reconstrucción de un teselado a partir de su grafómata	409
C.6.1	Determinación de la malla regular o de repetición	409
C.6.2	Algoritmo de teselado	412
C.6.3	Uso del algoritmo de teselado	417
C.6.4	Teselado óptimo	419
C.6.5	Razones de uso del teselado en 3 etapas en el ambiente	420
C.6.6	Teselados y vecindarios producidos por los grafómatas disponibles	422

C.7	Extensión y expansión de uso del concepto de grafomata	430
C.7.1	Relaciones entre teselados	430
C.7.2	Teselas y costuras	431
C.7.3	Fenomenos de dualizacion	433
C.7.4	Mas allá de Tgrafomata	434
C.7.5	Beneficios de los grafomatas	436
C.8	Conclusiones obtenidas	437
D	SPRT: Sistema de producción de reglas de transición	438
D.1	Génesis de un sistema de representación de transiciones	439
D.1.1	Cantidad de conocimiento en Naturaleza de las transiciones	440
D.1.2	Representación simbólica	441
D.1.3	Acción de cambio o transición	443
D.1.4	Dimensiones de evaluación	448
D.2	Fundamentos de un SPRT	449
D.2.1	Protonotación	449
D.2.2	Componentes	449
D.2.3	Primitivas de transición	449
D.2.4	Reglas de transición	450
D.2.5	Patrones de transición	451
D.3	Sistemas de SPRT	453
D.3.1	Características generales	457
D.4	SPRT de símbolos - nSmD	459
D.4.1	Significado operativo de las primitivas de transición de un SPRT	463
D.5	SPRT de referencias de símbolos - nRD	468
D.5.1	Primitivas de transición en sistemas de referencias	468
D.5.2	Sistemas de notación	476
D.5.3	Formas de representación de modelos	476
D.6	Sistemas SPRT 2mSmD - Estructuras presentes en reglas de transición de sistemas de referencias	477
D.6.1	Patrones de transicion de sistemas nSmD completos	479

D.6.2	notación transparente	486
D.6.3	Representación transparente de un modelo BOR	486
D.6.4	resumen de estructuras sprt encontradas	487
D.7	El SPRT de la MBOR y Homos 1.0	488
D.8	extensiones de protonotacion	497
E	Útiles de investigación	499
E.1	AegesPHP: Abrir, Editar, Guardar y Ejecutar Scripts PHP	503
E.1.1	Descripción de la interfaz	503
E.1.2	Contenido por defecto del cuadro de texto del editor	505
E.1.3	Notas de implementación	505
E.1.4	Flujo de trabajo con el editor AegesPHP	507
E.1.5	Sobre el desarrollo del editor	507
E.2	GSPRT: Generador de Sistemas Producción de Reglas de Transición	517
E.2.1	Reporte de resultados	518
E.2.2	Resultados de uso del script GSPRT y documentación asociada	520
E.2.3	Matriz SPRT de los sistemas nSmD, 2mSmD y nRD reportados	520
E.2.4	SPRT nSmD: Sistemas de transición entre símbolos	522
E.2.5	SPRT nRD: Sistemas de referencias de símbolos	556
E.2.6	SPRT 2mSmD: Límites de representación de patrones de transición	561
E.2.7	Código fuente del script GSPRT	568
E.3	Prueba de uso de la interfaz OLE de Homos 1.0	584
E.3.1	Revisión y mantenimiento de la Interfaz OLE	585
E.3.2	Ajuste del parámetro probabilidad de las reglas existentes en Homos 1.0	586
E.3.3	Cambios realizados a las funciones presentes en la interfaz OLE	587
E.4	Ejemplos de completitud en modelado con la MBOR	588
E.4.1	Ejemplo de modelo con regla de transición de probabilidad variable	589
E.4.2	Ejemplo de modelo con regla de transición de probabilidad variable y retardo en cambio de información	596
E.4.3	Ejemplo de control simultáneo de dos modelos	602
E.4.4	Recreación del modelo Predador - Presa presente en el manual de usuario de Homos 1.0	615

E.4.5	Recreación del modelo Predador - Presa del Picture Simulator	623
E.5	Algoritmo de teselado implementado en METAPOST	631
E.6	Hojas de cálculo y resultados de investigación	635
E.6.1	Prueba de relaciones numéricas entre características de Grafómatas	635
E.6.2	Caracterización numérica de los SPRT	635
E.6.3	Prueba práctica de caracterización en diferencias de modelos BOR	638
E.7	Diagramas de flujo y guías de código fuente	654
E.8	Primer prototipo para Homos 2.0	657
E.8.1	Productos de investigación integrados en el prototipo software PH2	657
E.8.2	Desarrollo del prototipo	657
E.8.3	Alcances y limitaciones del prototipo	659
E.8.4	Descripción de elementos implementados en la interfaz gráfica de PH2	661
E.8.5	Descripción de elementos de menú no implementados en PH2	678
E.8.6	Instrucciones de uso y prueba de manejo del prototipo	684
E.8.7	Componentes del prototipo	687
E.8.8	Modelo UML	688
E.8.9	Recursos relacionados	690
E.8.10	Conclusiones	691
F	Anexos en CD-ROM	693
F.1	Recursos relacionados con el desarrollo del proyecto de investigación	693

Lista de tablas

5.1	Representación tabular de transiciones de un modelo BOR	65
5.2	Caracterización en diferencias de modelos BOR	66
5.3	Actividad de primitivas y patrones presentes en el modelo	66
5.4	Componentes y representaciones disponibles para la MBOR	68
A.1	Versiones del software Homos 1.0	101
A.2	Seriales conocidos de Homos 1.0	103
A.3	Documentos y referencias conocidos de Homos 1.0	113
A.4	Resumen de métricas de código fuente de Homos 1.0	128
A.5	Resumen de métricas de código del Visor de Homos	138
A.6	Fallas conocidas en la VCL de Delphi 3	142
A.7	Herramientas de análisis y evaluación de código Pascal / Delphi	143
A.8	Identificadores CLSID de la Interfaz OLE	175
A.9	Estructura de relaciones entre funciones en módulos disponibles vs operaciones específicas en etapa <i>tal</i>	178
A.10	Etapas de edición, métodos de construcción del modelo.	182
A.11	Etapas de simulación: métodos de control del modelo.	183
A.12	Integración de elementos componentes de la MBOR	214
A.13	Integración entre elementos componentes de la MBOR	215
A.14	descripcion	219
A.15	Simbolos del protolenguaje Homos	227
A.16	Conocimiento previo sobre un estado de cambio	235
A.17	Cambios de estado fundamentales	237

A.18 Transiciones entre estados ?, A	237
A.19 Transiciones entre estados ?, A, B	237
A.20 Transiciones entre estados ?, A, B	238
A.21 Sistema con dos transiciones simultáneas entre estados ?, A, B, C	239
A.22 Sistema con dos transiciones simultáneas entre estados ?, A, B, C	240
A.23 Analisis de reglas de transicion mediante primitivas	246
A.24 Construcción de reglas de transición mediante primitivas	246
A.25 Ejemplo de nuevas reglas de transicion	247
A.26 Exploración de nuevas reglas de transición	249
A.27 Exploración de nuevas reglas de transición a partir de la protonotación	251
A.28 expresiones en el lenguaje?	254
A.29 Representación de Formas de pensamiento en HOMOS	259
A.31 Tabla comparativa entre Dinámica de sistemas y MBOR	267
A.32 Algunas limitaciones presentes en Homos 1.0	268
A.34 Algunas limitaciones presentes en Homos 1.0	269
A.36 Algunas limitaciones presentes en Homos 1.0	270
A.38 Integración entre elementos componentes de la MBOR	271
A.39 Semántica gráfica de reglas en la forma final de representación en el diagrama BOR	277
B.1 Primera evaluación interna 1998, Descripción del software	282
B.2 Primera evaluación interna, Aspectos funcionales del software	283
B.3 Primera evaluación interna, Aspectos técnicos y estéticos	283
B.4 Primera evaluación interna, Aspectos pedagógicos	283
B.5 Primera evaluación interna, Esfuerzo congntivo y observaciones	284
B.6 Primera evaluación interna, Observaciones encontradas	284
B.7 Segunda evaluación interna: Identificación del entorno, descripción del software	287
B.8 Segunda evaluación interna, Aspectos funcionales del software, utilidad	288
B.9 Segunda evaluación interna, Aspectos pedagógicos	288
B.10 Segunda evaluación interna, Aspectos técnicos y estéticos	289
B.11 Segunda evaluación interna, Recursos didácticos	289
B.12 Segunda evaluación interna, Esfuerzo congntivo	289

B.13 Segunda evaluación interna, Valoración global	290
B.14 Segunda evaluación interna, Observaciones encontradas	290
B.15 Criterios de evaluación: Fabricante	292
B.16 Criterios de evaluación: Usuario	292
B.17 Evaluación de software - Modelo y entradas	293
B.19 Evaluación de software - Ejecución	294
B.20 Evaluación de software - Pruebas & eficiencia	294
B.21 Evaluación de software - Animación	295
B.22 Evaluación de software - Salidas	296
B.23 Comparación de características compartidas entre Wator y Homos 1.0	298
B.24 Comparación de características compartidas entre WorldMaker 2.0 y Homos 1.0	300
C.1 Estructura de grafómata y dibúgrafo asociado	338
C.2 Distribución de vértices, trazos y aristas de los grafómatas y dibúgrafos disponibles en PH2	407
C.3 Etapas de teselado, cantidad de operaciones y duracion de proceso	421
D.1 representacion	442
D.2 Metáforas de transición	444
D.3 Expresiones de transición y estado de conocimiento de la dimensión de evaluación	446
D.4 Expresiones de transición y estado de conocimiento de la dimensión de evaluación	447
D.5 Patrones de transición fundamentales	450
D.6 Primitivas de transición fundamentales	450
D.8 Patrones de transición en primitivas de transición	451
D.7 Formas de notación de reglas de transición	451
D.9 Notación de patrones de transición	452
D.10 Determinación de patrones a partir de reglas de transición	454
D.11 Determinación de patrones en reglas de transición: Diferencias de representación y relaciones entre símbolos	455
D.12 Determinación de patrones en reglas de transición: Indices e Invariabilidad de representación	455
D.13 Clasificación de SPRT según primitivas de transición	457

D.14 Distribución de reglas de transición independientes en sistemas de SPRT nSmD con selección de primitivas	460
D.15 Distribución de reglas de transición independientes en sistemas de SPRT nSmD con selección de primitivas	461
D.16 Distribución de reglas de transición independientes en sistemas de SPRT nSmD con selección de primitivas	462
D.17 Matriz general de primitivas de transición	463
D.18 Matriz de primitivas de transición e identificador de vacío	464
D.19 Selección de primitivas y subconjuntos SRT de un 2S2D	465
D.20 Construcción de SRTs según selección de subconjuntos de primitivas de transición en un 2S2D	466
D.21 Distribución de reglas de transición independiente en SPRT nSmD completos	467
D.22 Distribución de patrones de transición en SPRT nSmD completos	469
D.23 Distribución de diferencias entre filas de la tabla	470
D.24 Interpretación matricial de primitivas de transición del SPRT 1RD y 2RD	472
D.25 Significado de Primitivas de transición de un SPRT de referencias	473
D.26 Interpretación matricial de primitivas de transición del SPRT 3RD	475
D.27 Ejemplos de metáforas de transición en términos de reglas de transiciones	478
D.28 Patrones, metáforas y transparencias de 2S1D	480
D.29 Patrones, metáforas y transparencias de 4S2D	480
D.30 Patrones, metáforas y transparencias de 6S3D - Patrones 1..16	481
D.31 Patrones, metáforas y transparencias de 6S3D - Patrones 17..29	482
D.32 Patrones, metáforas y transparencias de 6S3D - Patrones 30..42	483
D.33 Patrones, metáforas y transparencias de 6S3D - Patrones 43..52	484
D.34 Representación básica de reglas de transición de la MBOR en formalismo de transición de SPRT	489
D.35 Conjunto completo de reglas independientes y patrones de transición presentes en Homos 1.0	491
D.36 Primitivas de transición presentes en MBOR y Homos 1.0	492
D.37 Patrones y variantes de transición presentes en la MBOR de Homos 1.0	494
D.38 Representación de movimiento y ejemplo de acciones mediante referencias	496
D.39 Secuencias OEIS y SPRT	498

E.1	Efecto de selección de primitivas en un SPRT	536
E.2	Matriz de máximos de permutación en SPRT de símbolos	571
E.3	Componentes y representaciones disponibles en ejemplos de Modelado BOR	588
E.4	Modelo BOR de OLE_EJ_cristales_pulso.php	590
E.5	Caracterización en diferencias de OLE_EJ_cristales_pulso.php	591
E.6	Actividad de primitivas y patrones presentes en OLE_EJ_cristales_pulso.php	591
E.7	Modelo BOR de OLE_EJ_din_voto_retardo.php	597
E.8	Caracterización en diferencias de OLE_EJ_din_voto_retardo.php	598
E.9	Actividad de primitivas y patrones presentes en OLE_EJ_din_voto_retardo.php	598
E.10	Modelo BOR de OLE_EJ_pulso_m1_m2.php - Bacterias en probeta	603
E.11	Caracterización en diferencias del modelo #1, Bacterias en probeta	604
E.12	Actividad de primitivas y patrones presentes en Bacterias en probeta	604
E.13	Modelo BOR de OLE_EJ_pulso_m1_m2.php - Factor ambiental	608
E.14	Caracterización en diferencias del modelo #2, Factor ambiental	609
E.15	Actividad de primitivas y patrones presentes en Factor ambiental	609
E.16	Modelo BOR de OLE_predador_presa_6rt.php	616
E.17	Caracterización en diferencias de OLE_predador_presa_6rt.php	617
E.18	Actividad de primitivas y patrones presentes en OLE_predador_presa_6rt.php	617
E.19	Modelo BOR de OLE_predador_presa_7rt.php	624
E.20	Caracterización en diferencias de OLE_predador_presa_7rt.php	625
E.21	Actividad de primitivas y patrones presentes en OLE_predador_presa_7rt.php	625
E.22	Caracterización numérica de grafómatas	636

Lista de figuras

1.1	Delimitación conceptual	39
3.1	Metodología de exploración holista	48
4.1	Plan de proyecto, cuerpo de investigación y estructura del documento	52
4.2	Plan de proyecto, cuerpo de investigación y estructura del documento	54
5.1	Ejemplos de aplicación conceptual de Zonas en ambiente	70
5.2	Proyección	73
A.1	Cuerpo de análisis realizado sobre Mbor y Homos 1.0	98
A.2	Visualización del comportamiento del modelo mediante curvas poblacionales	107
A.3	Selección de elementos del modelo y opciones de impresión disponibles	108
A.4	Resultados reportados y opciones de impresión (a)	110
A.5	Resultados reportados y opciones de impresión (b)	111
A.6	Especialización de reglas de transición en modelo UML de Homos 1.0	136
A.7	Estructura del menú principal de la aplicación	147
A.8	Estructura de pasos de construcción de un modelo en Homos 1.0	158
A.9	Proceso de modelado y prototipado continuo	162
A.10	Guía de acciones disponibles en interfaz gráfica de Homos 1.0	164
A.11	Elementos de imagen en interfaz gráfica	168
A.12	Imágenes en Recursos	169
A.13	El Pensador de Rodin	170
A.14	Referencias directas al pensador en Homos 1.0	170

A.15 Etapas de implementación de un modelo con la interfaz OLE	177
A.16 Valores del parámetro Direccion	189
A.17 Funciones y etapas de creación de un modelo mediante la interfaz OLE	195
A.18 Mensajes de error en uso de interfaz OLE	200
A.19 Estructura y representación gráfica de reglas de transición en la MBOR	217
A.20 Estructura y representación gráfica de reglas de transición en la MBOR	266
A.21 Componentes del diagrama BOR	272
A.22 Notación gráfica de una clase objeto	276
A.23 Ejemplo 1 de diagrama BOR	278
A.24 Ejemplo 2 de diagrama BOR	279
 B.1 Ambiente de Wator	 297
B.2 Ambiente en WorldMaker	299
 C.1 Razon de uso de 3 o mas aristas por vértice	 312
C.2 Grafomatas y construcciones relacionadas	314
C.3 Relaciones entre elementos de grafomatas	315
C.4 Determinación del dibúgrafo de un grafómata	318
C.5 Proceso simplificado de construcción de estructuras duales	319
C.6 Forma de ordenamiento de aristas en cada vértice del sistema	320
C.7 Determinacion de antípodas del sistema	321
C.8 Calculo de vertices duales	323
C.9 Proceso de calculo de vertices duales	324
C.10 Construcción de la estructura dual a partir de vertices duales	329
C.11 Ejecución de ejemplo de uso de Tgrafomata	335
C.12 Patrón gráfico del grafómata 3.3.3.3.3	339
C.13 Patrón gráfico del grafómata dual 3.3.3.3.3	340
C.14 Patrón gráfico del grafómata 6.6.6	341
C.15 Patrón gráfico del grafómata dual 6.6.6	342
C.16 Patrón gráfico del grafómata	343
C.17 Patrón gráfico del grafómata dual 3.3.3.4.4	345

C.18 Patrón gráfico del grafómata el Cairo	346
C.19 Patrón gráfico del grafómata dual 3.3.4.3.4	348
C.20 Patrón gráfico del grafómata 4.8.8	349
C.21 Patrón gráfico del grafómata dual 4.8.8	351
C.22 Patrón gráfico del grafómata 3.6.3.6	352
C.23 Patrón gráfico del grafómata dual 3.6.3.6	354
C.24 Patrón gráfico del grafómata 3.3.3.3.6	355
C.25 Patrón gráfico del grafómata dual 3.3.3.3.6	358
C.26 Patrón gráfico del grafómata 3.4.6.4	359
C.27 Patrón gráfico del grafómata dual 3.4.6.4	361
C.28 Patrón gráfico del grafómata 4.6.12	364
C.30 Patrón gráfico del grafómata 3.12.12	366
C.29 Patrón gráfico del grafómata dual 4.6.12	367
C.31 Patrón gráfico del grafómata dual 3.12.12	369
C.32 Patrón gráfico del grafómata variación en triángulos	370
C.33 Patrón gráfico del grafómata dual variación en triángulos	372
C.34 Patrón gráfico del grafómata variación en 4.8.8	373
C.35 Patrón gráfico del grafómata dual variación en 4.8.8	374
C.36 Patrón gráfico del grafómata variación en 3.12.12	375
C.37 Patrón gráfico del grafómata dual variación en 3.12.12	376
C.38 Patrón gráfico del grafómata variación el Cairo	377
C.39 Patrón gráfico del grafómata dual variación el Cairo	379
C.40 Patrón gráfico del grafómata pentágonos y octágonos	380
C.41 Patrón gráfico del grafómata dual pentágonos y octágonos	382
C.42 Patrón gráfico del grafómata pentágonos y rombos	383
C.43 Patrón gráfico del grafómata dual pentágonos y rombos	385
C.44 Patrón gráfico del grafómata cuadrados y hexágonos	386
C.45 Patrón gráfico del grafómata dual cuadrados y hexágonos	388
C.46 Patrón gráfico del grafómata subcuadros	389
C.47 Patrón gráfico del grafómata dual subcuadros	391

C.48 Patrón gráfico del grafómata Molinetes	392
C.49 Patrón gráfico del grafómata dual Molinetes	394
C.50 Patrón gráfico del grafómata triángulos-hexágonos	395
C.51 Patrón gráfico del grafómata dual triángulos-hexágonos	397
C.52 Patrón gráfico del grafómata vecindario de Von Newman	398
C.53 Patrón gráfico del grafómata vecindario Moore de 9 celdas	399
C.54 Patrón gráfico del grafómata vecindario Moore de 25 celdas	400
C.55 Patrón gráfico del grafómata vecindario Moore de 49 celdas	401
C.56 Diagrama de algoritmo de construcción de vecindarios de Moore	402
C.57 Grafómata, dibúgrafo y malla de repetición	410
C.58 Malla regular en el grafómata base y su dual	411
C.59 Algoritmo de relleno de superficies	413
C.60 Ejemplo de ejecución del algoritmo de relleno de superficie	416
C.61 Relleno del espacio mediante uso serial del algoritmo de relleno	418
C.62 Relacion entre relleno optimo y ejes de movimiento de algoritmo de teselado	419
C.63 Vecindario triangular	423
C.64 Vecindario Von Newman	423
C.65 Vecindario hexagonal	423
C.66 Vecindario 3.3.3.4.4	424
C.67 Vecindario 3.6.3.6	424
C.68 Vecindario 3.12.12	424
C.69 Estrella de David, Vecindario 3.3.3.3.6	425
C.70 Vecindario 3.4.6.4	425
C.71 Vecindario 4.6.12	425
C.72 Cuadrados y octágonos, Vecindario 4.8.8	426
C.73 Cuadrados y hexágonos , Vecindario Ojo de buey	426
C.74 Cuadrados y trapecios, Vecindario Waffles	426
C.75 Vecindario el Cairo	427
C.76 Variación en Vecindario el Cairo	427
C.77 Vecindario de Molinetes	427

C.78 Variación en Vecindario de triángulos	428
C.79 Estrellas de 4 puntas, variación en Vecindario de 4.8.8	428
C.80 Estrellas de 6 puntas, Variación en Vecindario 3.12.12	428
C.81 Vecindario de triángulos y hexágonos	429
C.82 Vecindario de pentágonos y rombos	429
C.83 Vecindario de octágonos y pentágonos	429
C.84 Ejemplo de relaciones entre de teselados implementados	430
C.85 Ejemplo de bordes de costura, paso entre áreas de grafómatas	432
C.86 Fenómenos de dualización de grafómatas	433
D.1 Desarrollo del concepto de SPRT	439
D.2 diseño de SPRT	449
D.3 Definición completa de la matriz SPRT nSmD	458
D.4 Sistemas de Símbolos: Matriz de SPRT nSmD	459
D.5 Sistemas de Referencias: Vector de SPRT nRD	468
D.6 Cobertura de dimensiones de evaluacion y primitivas de transición en un SPRT de referencias	471
D.7 Reglas de transición e interfaz gráfica de Homos 1.0	489
E.1 Utiles de investigación	500
E.2 Guía de instalación y uso de AegesPHP	508
E.3 Interfaz gráfica del script aegesphp.php	509
E.4 Sección de la Matriz SPRT comprendida por 4S3D	542
E.5 Vector de SPRT nRD comprendidos por 3RD	559
E.6 Homos 1.0 como servidor de aplicaciones	584
E.7 Resultados de simulación de OLE_EJ_cristales_pulso.php	589
E.8 Representación de diagrama BOR del modelo OLE_EJ_cristales_pulso.php	592
E.9 Representación gráfica de transiciones de OLE_EJ_cristales_pulso.php	593
E.10 Resultados de simulación de OLE_EJ_din_voto_retardo.php	596
E.11 Representación de diagrama BOR del modelo OLE_EJ_din_voto_retardo.php	599
E.12 Representación gráfica de transiciones en OLE_EJ_din_voto_retardo.php	599
E.13 Representación de diagrama BOR del modelo #1, Bacterias en probeta	605

E.14 Representación gráfica de transiciones de Bacterias en probeta	606
E.15 Representación de diagrama BOR del modelo #2, Factor ambiental	607
E.16 Representación gráfica de transiciones de Factor ambiental	610
E.17 Población de bacterias en OLE_EJ_pulso_m1_m2.php	611
E.18 Factor ambiental en OLE_EJ_pulso_m1_m2.php	611
E.19 Representación de diagrama BOR del modelo predador presa de seis reglas	618
E.20 Representación gráfica de transiciones de OLE_predador_presa_6rt.php	619
E.21 Resultados de simulación de OLE_predador_presa_6rt.php	620
E.22 Representación de diagrama BOR del modelo predador presa de siete reglas	626
E.23 Representación gráfica de transiciones de OLE_predador_presa_7rt.php	627
E.24 Resultados de simulación de OLE_predador_presa_7rt.php	628
E.25 Comportamiento poblacional de OLE_EJ_cristales_pulso	642
E.26 Valores de aleatoriedad en OLE_EJ_cristales_pulso	642
E.27 Actividad de reglas de transición en OLE_EJ_cristales_pulso	643
E.28 Actividad de clases en OLE_EJ_cristales_pulso	643
E.29 Actividad de primitivas en OLE_EJ_cristales_pulso	644
E.30 Permanencia y cambio de primitivas en OLE_EJ_cristales_pulso	644
E.31 Actividad total de primitivas en OLE_EJ_cristales_pulso	645
E.32 Actividad de primitivas como transiciones fundamentales en OLE_EJ_cristales_pulso	645
E.33 Actividad de patrones de transición en OLE_EJ_cristales_pulso	646
E.34 Actividad total de patrones en OLE_EJ_cristales_pulso	646
E.35 Comportamiento poblacional de OLE_predador_presa_7rt	647
E.36 Valores de aleatoriedad en OLE_predador_presa_7rt	647
E.37 Actividad de reglas de transición en OLE_predador_presa_7rt	648
E.38 Actividad de clases en OLE_predador_presa_7rt	648
E.39 Actividad de primitivas en OLE_predador_presa_7rt	649
E.40 Permanencia y cambio de primitivas en OLE_predador_presa_7rt	649
E.41 Actividad total de primitivas en OLE_predador_presa_7rt	650
E.42 Actividad de primitivas como transiciones fundamentales en OLE_predador_presa_7rt	650
E.43 Actividad de patrones de transición en OLE_predador_presa_7rt	651

E.44 Actividad total de patrones en OLE_predador_presa_7rt	651
E.45 Características de los diagramas de flujo implementados en este proyecto	655
E.46 Ejemplo de Guia de código	656
E.47 Productos de investigación y primer prototipo de Homos 2.0	657
E.48 Estructura del menú principal del prototipo Homos 2.0	667
E.49 Estructura del submenú Ambiente en el prototipo Homos 2.0	668
E.50 Estado inicial del prototipo de Homos 2.0	669
E.51 Registro de momentos y controles de desarrollo de la simulación	669
E.52 Visualizar opciones de movimiento de objetos en espacio según geometría [Ctrl+B] . . .	670
E.53 Visualizar áreas de influencia espacial de objetos según geometría [Ctrl+D]	670
E.54 Visualizar áreas de influencia y opciones de movimiento según geometría [Ctrl+M] . . .	671
E.55 Visualizar objetos en espacio libre [Ctrl+L]	671
E.56 Ambiente en panorámica cercana, 11x	672
E.57 Ambiente en panorámica máxima, 60x	672
E.58 Visualización de Zona 3: Reloj de arena	673
E.59 Visualización de Zona 4: Anillo cuadrado	673
E.60 Presentar la última ubicación de objeto [Ctrl+U]	674
E.61 100.000 objetos en ambiente	674
E.62 Objetos en ambiente como puntos / círculos uniformes [Shift+1]	675
E.63 Objetos en ambiente como círculos coloreados según su clase [Shift+2]	675
E.64 Objetos en ambiente identificados serialmente [Shift+3]	676
E.65 Área de influencia inmediata de objetos coloreada según su clase [Shift+4]	676
E.66 Objetos en ambiente según imagen de clase [Shift+5]	677
E.67 Objetos en ambiente como hormigas [Shift+6]	677
E.68 Guía de acciones disponibles en el prototipo de Homos 2.0	686
E.69 Estructura de unidades componentes del prototipo PH2.exe	687
E.70 Diagrama de clases de PH2	688
E.71 Diagrama de dependencias	689
E.72 Muestra de diagramas de flujo del código fuente de PH2	690

Lista de Scripts

A.1	Control elemental de errores en scripts PHP	196
A.2	Creación y terminación de una instancia de automatización	197
A.3	Ejemplos de ciclos de control	197
A.4	Ciclo de presentación de resultados	199
A.5	Implementación de un modelo de voto simple	201
A.6	Ejemplo de modelo con pulso	203
A.7	Ejemplo de modelo con retardo	204
A.8	Definición de dos modelos en un mismo script	205
A.9	Ciclo de control principal y simulación síncrona de dos modelos	206
E.10	Editor AegesPHP	510
E.11	Configuración de GSPRT: SPRT completos	517
E.12	Ejemplos de configuración de GSPRT para un SPRT de referencias	522
E.13	Configuración de GSPRT para un SPRT 4S2D incompleto	535
E.14	Configuración de GSPRT para un SPRT de referencias	556
E.15	Generador de Sistemas de Producción de Reglas de Transición	574
E.16	Un modelo de crecimiento de cristales con pulso de regeneración poblacional	594
E.17	Un modelo de voto con retardo	600
E.18	Ejemplo de manejo paralelo de 2 modelos	612
E.19	Recreación del modelo Predador-Presa	621
E.20	Recreación del modelo Predador-Presa de siete reglas de transición	629
E.21	Implementación del algoritmo de relleno de superficies en METAPOST	631

Lista de Reportes GSPRT

E.1	SPRT 1S1D de vacío	523
E.2	SPRT 2S1D de símbolos distintos	524
E.3	SPRT 2S1D de un símbolo y vacío	525
E.4	SPRT 2S2D de símbolos distintos	526
E.5	SPRT 2S2D de símbolo y vacío	527
E.6	SPRT 3S2D	529
E.7	SPRT 3S3D	531
E.8	SPRT 4S2D con selección de primitivas	537
E.9	SPRT 4S2D	539
E.10	SPRT 4S3D	543
E.11	SPRT 1RD	557
E.12	SPRT 2RD	558
E.13	SPRT 3RD	559
E.14	SPRT 6S3D	561
E.15	SPRT 8S4D	563

RESUMEN

TÍTULO:

PROTOTIPO HOMOS 2.0: HERRAMIENTA SOFTWARE PARA EL MODELADO Y SIMULACIÓN BASADO EN OBJETOS Y REGLAS *

AUTOR:

ANGEL MARIA VALDES PEÑA **

PALABRAS CLAVE:

Simulación, Metodología de modelamiento basado en objetos y reglas [MBOR], Sistemas de producción de reglas de transición [SPRT], Teselaciones regulares y Grafómatas, Proceso de innovación aplicado.

CONTENIDO:

Las herramientas software al igual que sus contrapartes del mundo real requieren de ajustes, revisiones, mantenimiento y en casos especiales reparación, reconstrucción y reorientación de uso en procura de mejorar sus capacidades. Para Homos 1.0 y su metodología ha llegado el momento de innovar, mediante este proceso se busca extender la vida útil del objeto tecnológico.

Con investigación y la aplicación práctica del conocimiento obtenido a través de análisis, exploración, expansión de temas claves y síntesis de elementos teóricos se determinaron los alcances y limitaciones existentes en la MBOR y la herramienta software; con la información adquirida se establecen e implementan útiles teóricos y prácticos que demuestran el impacto y viabilidad de las novedades planteadas.

Se propone el uso de Sistemas de producción de reglas de transición [SPRT] como refundación de la metodología de modelamiento basada en objetos y reglas [MBOR], estos sistemas aportan formalidad y generalidad para el estudio y construcción de reglas de transición. Se sienta un precedente sobre el uso de Grafómatas para la representación de divisiones del espacio y la construcción de teselaciones regulares y se presenta un diseño general para la realización de futuras versiones de la herramienta software que faciliten la materialización de los nuevos elementos propuestos.

Se espera que el aporte realizado conduzca el desarrollo de herramientas de simulación inspiradas en la MBOR, permita obtener resultados analíticos nuevos y productos derivados que fortalezcan el uso práctico y aplicado de la representación dinámica de fenómenos mediante objetos y reglas.

*Proyecto de grado.

**Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Hugo Hernando Andrade Sosa. Codirector: Merly Sulgey Gómez Sánchez.

ABSTRACT

TITLE:

HOMOS PROTOTYPE 2.0: SOFTWARE TOOL FOR MODELING AND SIMULATION BASED ON OBJECTS AND RULES *

AUTHOR:

ANGEL MARIA VALDES PEÑA **

KEYWORDS:

Simulation, Modeling methodology based on objects and rules [MBOR], Production systems of transition rules [PSTR], Regular tessellations and Grafómatas, Applied innovation process.

DESCRIPTION:

Software tools like their real world counterparts require adjustments, servicing, maintenance and in special cases, repair, reconstruction and use reorientation seeking to improve their capacity. For Homos 1.0 and its methodology is time to innovate. Through this process seeks to extend the life of the technological object.

Through research and practical application of knowledge gained through analysis, exploration, expansion of key issues and synthesis of theoretical elements were determined the scope and limitations in the MBOR and the software tool. With the acquired information, it is possible to establish, design and implement theoretical and practical tools that demonstrate the impact and viability of raised developments.

We propose the use of Production systems of transition rules [PSTR] as a new foundation for the modeling methodology based on objects and rules [MBOR]. This system provides formality and generality for the study and construction of transition rules. We set a precedent for the use of "Grafómatas" as a representation of space divisions and for the construction of regular tessellations and a general design is presented for the realization of future versions of the software tool that ease the materialization of the new proposed elements.

It is expected that the contribution made leads to the development of MBOR inspired simulation tools, to obtain new analytical results and derived products that strengthen the practical and applied use of the dynamic representation of phenomena through objects and rules.

*Undergraduate research thesis.

**Faculty of Physical Mechanical Engineering. School of Systems Engineering and Informatics. Director: Hugo Hernando Andrade Sosa. Codirector: Merly Sulgey Gómez Sánchez.

Introducción

motivo de inv
como se desarrolló
que demuestra la inv
cuales son los objetivos de inv
como se organiza el cont de inv

La introducción de un trabajo de investigación deberá ser clara, simple, escrita en lengua informativa. Deja de lado apreciaciones personales, giros o adjetivaciones. Es un planteo de lo que es tu investigación. Como el "copete periodístico" deberá contestar a estas preguntas;

-Qué es lo que has hecho -Para qué lo has hecho -Por qué (quién te lo ha solicitado -Cuando- el tiempo insumido -Dónde (o sea tocas la Isla de Pascua) -Cómo. .Acá van los métodos. Entrevistas, encuestas, consultas, guías de investigación. Tú sabrás que utilizaste. -Quiénes intervinieron en el proyecto -Obstáculos que se presentaron en la investigación.

Creo que va completo. Recuerda: la introducción debe ser corta, muy clara, decir de qué se trata, pero no adelantes nada de lo que viene después.

presentar la investigación realizada
innovar con respecto a mbor y homos
pruebas de la realización de un valioso aporte a la metodología y la hs

Este trabajo tiene como proposito, con la presentacion por medio de este

En la descripcion del problema de investigacion

Presentación del marco teórico, referencial y conceptual

diseño metodologico

analisis de resultados obtenidos

Se concretan en los resultados de mayor impacto logrados como resultado del proceso de investigacion Como resultado del proceso de investigacion realizado La propuesta

conclusiones y recomendaciones

El grueso del trabajo esta compuesto por el cuerpo de anexos producidos durante la investigacion realizada, que da soporte formal y _ a las distintos elementos integrantes de la propuesta de diseño de siguientes versiones de homos presentada en este documento.

Se presenta en la seccion de bibliografía la documentación que

Problema de investigación

1.1. Descripción del problema

El grupo Simon ha creado a lo largo de su historia productiva herramientas software orientadas hacia la aplicación práctica de la dinámica de sistemas y el pensamiento sistémico en la educación. Uno de sus éxitos más relevantes en este campo es la familia software Evolución 3.x - 4.x y sus descendientes, los proyectos MAC.

Homos 1.0 fue una novedad bien recibida en su momento, pero debido a motivos aún desconocidos no recibió igual apoyo que otros proyectos realizados por el grupo. Es un hecho lamentable reconocer que el proceso de desarrollo de la herramienta ha sido letárgico¹. Se presume que algunas de las causas hicieron factible ese evento fueron:

- Falta de una política de desarrollo de aplicaciones en el grupo que coordinara la continuidad de la investigación sobre la herramienta y su metodología; y agregada a ésta, una idea entre muchas, un posible plan de incentivos con el cual *atraer* investigadores al área de estudio.
- Ausencia de personal docente y estudiantil interesado en el desarrollo de la herramienta, que se tradujo en una menor fuerza de trabajo y recursos que permitieran dar continuidad al proyecto.
- Inexistencia de un plan explícito de mantenimiento preventivo y correctivo del software, tanto en la tesis de Homos como en la política de gestión de aplicaciones por parte del grupo Simon.
- Poca aplicación directa del software como herramienta pedagógica en el medio educativo, debido a los rechazos por su complejidad aparente², el cambio de paradigma por parte de algunos usuarios, la falta de personal ilustrado en el tema, un pobre sistema de soporte y ayudas al usuario incorporado en la herramienta, junto con sus particulares debilidades

¹Comprendido desde 1998 hasta 2011. Inactividad en lo relacionado a cambios o actualizaciones en el software y su metodología.

²Que parten del supuesto inicial común entre los usuarios de una abrumadora complejidad en el manejo del software, entre otras razones.

mediáticas³ y multimedia. Este último elemento condujo al grupo a la creación del proyecto Micrho [IMS00, RC08, CD00], que extiende las funcionalidades del programa en ese sentido y permite la construcción de micromundos con fundamento en la MBOR. Pese a los esfuerzos realizados⁴, para ambos software en términos de aceptación general no se obtuvo mayor éxito⁵.

Como consecuencia del receso, se presentan actualmente en la herramienta y su metodología:

- Presencia de fallas de varios niveles de importancia y aspectos funcionales perfeccionables en el software, ausencia de facilidades que mejoren los procesos de modelado, el uso de la herramienta y de valor agregado.
- Una difusión limitada del software, destinada principalmente a círculos académicos.
- Un número reducido de trabajos de aplicación teórica y práctica fundados en la herramienta y su metodología.
- Un aumento en el número de usuarios insatisfechos con el software actual, que gradualmente dan paso a un *olvido* o remplazo de la metodología y sus implicaciones en favor de otras tecnologías y representaciones que suplan sus necesidades.

Poner al día la herramienta y enderezar su curso actual requiere de una masiva intervención en varios niveles:

- A nivel de software: Acción de mantenimiento software urgente para la herramienta Homos 1.0.
- A nivel metodológico: Actualización e incorporación de los elementos necesarios para dar flexibilidad a la herramienta, al proceso de modelado y nueva vida al proyecto Homos.
- A nivel administrativo: Se requiere un cambio en la forma en que el grupo realiza sus procesos de producción de software.
- A nivel social: Una promoción activa acompañada de un proceso de trabajo y experimentación alrededor de la herramienta.
- A nivel educacional, pedagógico y práctico en el aula: Es necesaria la creación de materiales apropiados según el caso.

Teniendo en mente la amplitud y lo trascendente del problema para el grupo Simon y como parte de una solución a futuro, este proyecto se circunscribirá al nivel metodológico, a la evolución conceptual de la herramienta software, Homos 2.0. Esto debido a que la innovación⁶ es la operación que representa a largo plazo la mejor alternativa de aporte a la herramienta software y su temática.

³Se subraya la ausencia de otras interfaces de manejo en la herramienta.

⁴Esta capa adicional de software tiene sus propias limitaciones y desventajas, que se sumaron a las ya presentes en Homos.

⁵Se presentaron experiencias exitosas durante demostraciones académicas. Explicaciones sobre las razones por las cuales se producen estos resultados se verán mas adelante en este documento, específicamente en el cuerpo de análisis en anexo A en la página 96.

⁶"Innovación es la secuencia de actividades por las cuales un nuevo elemento es introducido en una unidad social con la intención de beneficiar la unidad, una parte de ella o a la sociedad en conjunto. El elemento no necesita ser enteramente nuevo o desconocido a los miembros de la unidad, pero debe implicar algún cambio discernible o reto en el *status quo*." [WF90]

1.2. Formulación del problema

El aspecto investigativo de este proyecto se centrará en dar respuesta a las siguientes inquietudes, surgidas del proceso de análisis preliminar, la observación del problema y la delimitación propuesta:

- Con respecto al análisis y evaluación de la herramienta:
¿Cuales son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?
- Con respecto a la metodología de modelado:
¿Qué características nuevas deben ser incluidas en la MBOR y la herramienta software para aumentar sus capacidades de representación?,
¿Cómo mantener el espíritu original de la aplicación ante estos cambios?

1.3. Justificación

En relación directa con la historia de desarrollo de la herramienta, las razones por las cuales se genera este proyecto son:

- Dar continuidad al proceso de evolución de la herramienta software como tal, presentando las innovaciones a las que haya lugar obtenidas durante el proceso de investigación.
- Servir de incentivo a desarrolladores, al personal del grupo Simon y en general a quienes se interesen en esta línea de investigación, al promover y apoyar por este medio la continuidad de los estudios sobre la herramienta y su metodología en el futuro.

1.3.1. Impacto

Se espera que el desarrollo de este proyecto influya positivamente como apoyo e impulso a la metodología basada en objetos y reglas [MBOR], a la herramienta software Homos y a las aplicaciones software desarrolladas en la universidad: La evolución de una herramienta de simulación favorece su usabilidad y aplicabilidad, la aparición de nuevos modelos implementados con el software, la creación de proyectos de investigación relacionados, el acercamiento y el reconocimiento por parte de los usuarios al trabajo realizado en el grupo Simon y la Universidad Industrial de Santander. Este aporte se traduce a largo plazo en continuidad del desarrollo de la herramienta así como la producción de avances significativos en su área particular de investigación.

1.3.2. Viabilidad

Para la realización técnica de este proyecto se requieren conceptos sobre probabilidad y estadística, teoría de autómatas celulares y tecnología orientada a objetos, programación, estructuras de datos, elementos de la ingeniería de software y fundamentos de la MBOR, corazón del proyecto. Algunos de estos conocimientos ya han sido adquiridos durante la carrera, otros serán profundizados durante la ejecución de este proyecto. Se cuenta con los fundamentos teóricos, las capacidades económicas y logísticas necesarias para su elaboración, además de la voluntad del grupo en hacer de este proyecto una realidad. Los requerimientos técnicos necesarios para su ejecución, su impacto social y el aporte investigativo, lo hacen meritorio como trabajo de grado en Ingeniería de Sistemas.

1.3.3. Entidades interesadas en el proyecto

Grupo Simon de investigaciones en modelado y simulación⁷

Es un colectivo de estudiantes, profesores y egresados de la Escuela de Ingeniería de Sistemas e Informática de la UIS que desarrollan una labor de investigación adscrita a la línea de Pensamiento Sistémico, Modelado y Simulación en los programas de Maestría en ingeniería área Informática y Pregrado, cuya misión es construir un proyecto de cambio profesional, pedagógico y social alrededor de la Ingeniería de Sistemas, guiados por una concepción sistémica útil en el desarrollo de proyectos de Investigación interdisciplinarios.

⁷Sintetizado de http://www.uis.edu.co/portal/investigacion/grupos/simon_uis/simon.html

1.4. Delimitación conceptual

La metodología de modelamiento basada en objetos y reglas [MBOR] y la herramienta software que la implementa, Homos 1.0 son objetos de estudio de este proyecto. Las acciones a desarrollar se centrarán en la satisfacción de las preguntas de investigación mediante la realización práctica los útiles previstos en los objetivos general y específicos planteados en la sección 1.5 en la página 40.

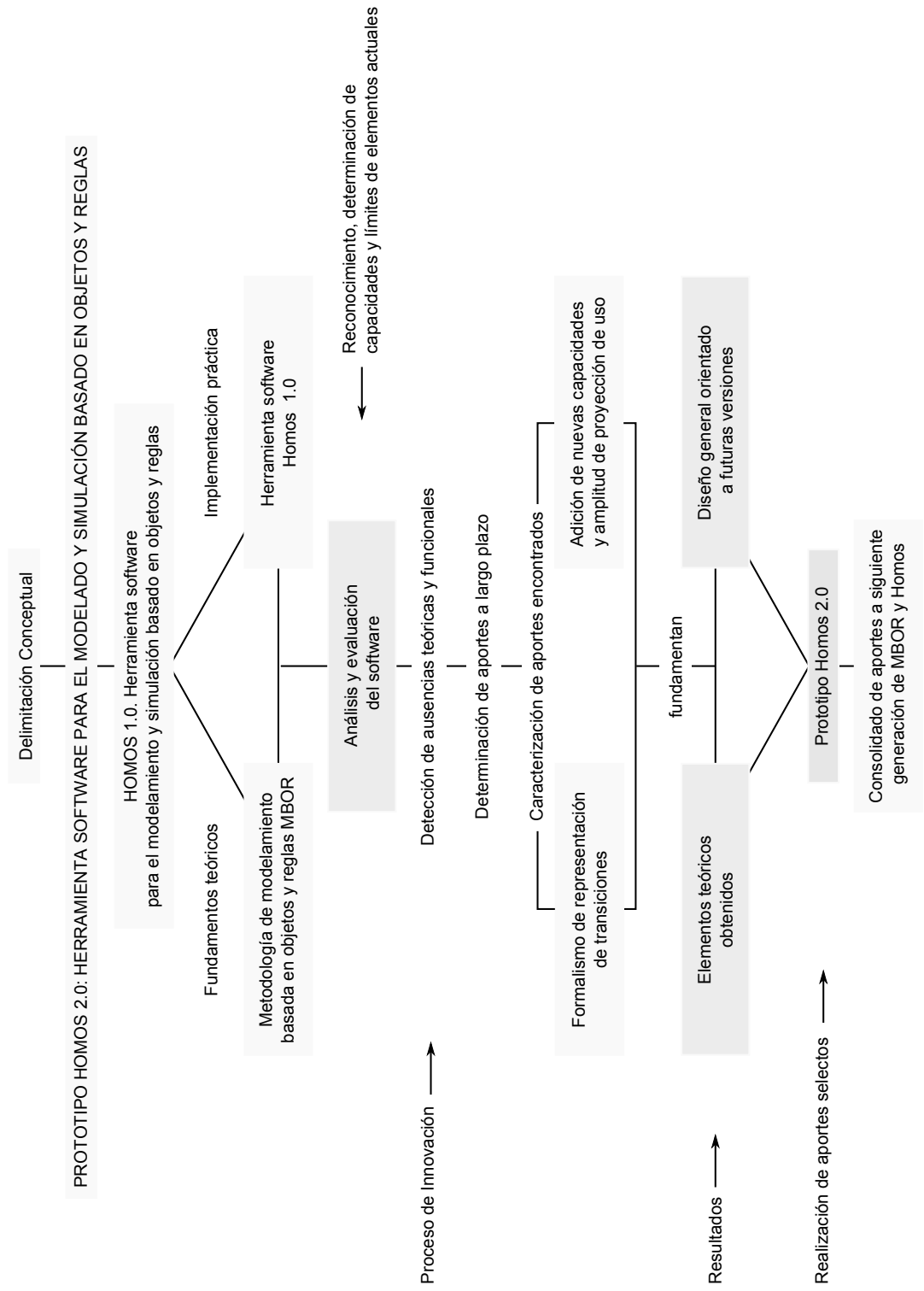
Como se muestra en la figura 1.1 en la página siguiente, la investigación busca fortalecer teóricamente a la MBOR con mejoras en la representación de reglas de transición y la ampliación del espacio de aplicación de la herramienta proponiendo e implementado según sea el caso, adiciones o cambios en las capacidades funcionales del software. La idea global de todo el conjunto de acciones apunta a la búsqueda de conocimiento y la determinación de aportes que beneficien a largo plazo a la metodología y al software que la soporte.

Tomando como referencia el modelo de fases de innovación G.I.F.T. presentado en [Pic09] que consiste fundamentalmente en la búsqueda de variedad y frescura de las ideas [*Generate*], la determinación de nuevos usos para las ideas existentes [*Imagine*], la adaptación de ideas externas o ajenas a su campo de trabajo [*Frame*] y la prueba de los conceptos generados junto con un compartir las ideas obtenidas [*Test*]; se hace una adaptación del modelo al propósito de investigación de este proyecto configurando un proceso de innovación particular en el que se establecen las siguientes cuatro etapas:

1. Determinación de las capacidades y límites presentes en la metodología y la herramienta software mediante la aplicación de un cuerpo de análisis y evaluación de software ajustado a las necesidades particulares del objeto de estudio.
2. Búsqueda de ausencias teóricas y funcionales existentes en la metodología y el software.
3. Selección de entre las ausencias detectadas de aquellas cuyo impacto sea de largo plazo para la metodología y la herramienta software. Se hará especial énfasis en la selección de aquellos elementos relacionados con la representación de transiciones y la adición de nuevas capacidades funcionales en el software.
4. Construcción de soporte teórico: caracterización, descripción en detalle con grados de profundización variable según corresponda de los elementos seleccionados a ser implementados.

Durante las operaciones de innovación a realizar se tendrá en cuenta la sinergia entre los distintos elementos considerados, esto es con el propósito de dar unidad y continuidad entre las múltiples facetas componentes del objeto de estudio. A partir de los elementos teóricos y el conjunto de funciones obtenidos a incorporar, se determinará un diseño o propuesta de solución a futuro de la problemática principal y de las preguntas de investigación planteadas en este proyecto. Se espera que al final del proceso de innovación se obtengan productos teóricos y prácticos, algunos de los frutos obtenidos se integrarán posteriormente en un prototipo software que permita demostrar la viabilidad práctica de los resultados obtenidos.

Figura 1.1: Delimitación conceptual



1.5. Objetivos

Objetivo general

Desarrollar un prototipo software que implemente innovaciones para una segunda versión de la herramienta software HOMOS según los elementos teóricos propuestos durante el proceso de investigación.

Objetivos específicos

Analizar y evaluar la herramienta Software HOMOS 1.0

Realizar un documento de análisis y evaluación de la herramienta software HOMOS 1.0 que proporcione un referente teórico para el presente proyecto y sirva de documentación interna sobre este software para el grupo Simon.

Investigar y proponer un diseño general de la herramienta para futuras versiones

Realizar un documento de análisis y diseño general de la herramienta para una siguiente versión del software que contemple mejoras a elementos encontrados en el documento de análisis y evaluación de la versión actual del software según los resultados obtenidos durante el proceso de investigación.

Desarrollar un prototipo software de la aplicación

Realizar un prototipo software que implemente innovaciones propias de una siguiente versión del software según los resultados obtenidos durante la investigación.

Marco teórico

2.1. Metodología de modelamiento basada en objetos y reglas

La metodología de modelamiento basada en objetos y reglas [MBOR] es el fundamento teórico de la herramienta software Homos. Se la describe con detalle en capítulo 1.3 de la tesis de Homos 1.0 [MA98]. Establece el objetivo principal del software, los elementos que constituyen el modelo y sus útiles de modelado.

La metodología implementa una forma de descripción de la relación temporal de hechos o interacciones entre entidades u objetos y el ámbito en el cual se presentan. Se centra en lo que acontece en la dinámica de interacción más que en el detalle y la descripción funcional de los elementos componentes de la situación en estudio. Esto hace que Homos sea fundamentalmente una herramienta de simulación fenomenológica.

El principio funcional de la metodología es la fusión de algunos conceptos de autómatas celulares con teoría de objetos. Los autómatas celulares aportan las siguientes características:

- Un espacio matricial finito que almacena hasta un objeto por celda, cada una de forma cuadrada.
- Define el concepto de vecindad como las celdas inmediatas a la ubicación donde está actualmente el objeto en evaluación. La MBOR establece el uso de la vecindad de Moore de 9 celdas, una celda central en evaluación y sus 8 vecinas circundantes.
- Un espacio limitado mediante bordes o ambiente. Los bordes definen la forma de confinamiento de los objetos presentes en el ambiente, hacen que el espacio sea finito, infinito o cíclico.
- Establecen la forma en la cual se realiza la dinámica del modelo. El proceso consiste en la elección sistemática de los objetos en el espacio de simulación donde para cada uno se determina su interacción particular con respecto a los demás objetos presentes en las celdas vecinas según ciertas relaciones o reglas de transición previamente establecidas.

La teoría de objetos aporta la tipificación de los objetos o instancias según categorías específicas o clases. Este conjunto de elementos hace posible la representación y simulación en términos de objetos y reglas. La metodología de modelamiento define los siguientes elementos:

El ambiente: Lugar donde se realiza la interacción entre los objetos, es donde se produce la realización de la ejecución de la simulación o paisaje dinámico.

Las clases-objeto: Definen de los distintos tipos de objetos disponibles presentes en el modelo. Las instancias en el ambiente de las clases objeto son los objetos, componen la población de cada tipo de clase presente en el modelo.

Las reglas de transición: Configuran las relaciones o interacciones entre las clases-objeto del modelo, en el ambiente representan las acciones a realizar entre objetos según la clase a la que pertenezcan. Tienen inspiración biológica, se caracterizan mediante la elección particular de las clases que intervienen en la transición y la asignación de la prioridad y probabilidad de ejecución. Las reglas de transición previstas por la metodología son:

Decadencia: El objeto deja de existir para la simulación.

Transformación: El objeto cambia de clase.

Retracción: El objeto reduce su cantidad en una unidad.

Expansión: El objeto aumenta su cantidad en una unidad.

Eliminación: Un objeto hace que otro deje de existir.

Neutralización: Dos objetos del ambiente simultáneamente dejan de existir.

Alteración: Un objeto cambia de clase en presencia de otro objeto.

Movimiento: El objeto se desplaza en el ambiente una cantidad determinada de pasos.

Reproducción: La presencia de dos objetos hace posible a un tercer objeto.

A partir de las definiciones dadas y las relaciones básicas entre objetos, se formaliza la estructura y la dinámica del modelo. La metodología de modelamiento establece un proceso de modelado o secuencia de construcción y ejecución de modelos en cuatro etapas:

1. La adquisición de información sobre el fenómeno sistema a modelar.
2. La implementación del fenómeno en la herramienta mediante el uso de representaciones intermedias¹ hasta llegar a una construcción en términos de clases-objeto y reglas o modelo BOR², según lo establecido en la metodología.
3. La simulación del modelo BOR construido en la herramienta software.

¹Modelo en lenguaje prosa: una descripción textual conceptual detallada del fenómeno en estudio y la descripción de objetos y acciones: un mecanismo con el cual se igualan los elementos componentes de las expresiones prosa en términos de conceptos de objetos y reglas.

²Basado en [clases-]Objetos y Reglas.

4. El ajuste de sensibilidad de los parámetros previstos en el modelo con el propósito de mejorar u observar cambios en el comportamiento obtenido.

La metodología establece la forma y tipo de representación de las clases y los objetos en la herramienta software, designa una notación gráfica para algunas de las reglas de transición existentes y caracteriza a Homos con un modelo matemático que es propio del mundo de los autómatas celulares. Se profundiza el estudio de la metodología en la sección A.6 y su representación grafica en la sección A.7.

2.2. Marco referencial

El estudio de los sistemas dinámicos y las herramientas creadas para su análisis y modelado han sido claves en la búsqueda de soluciones de los problemas más diversos y han tenido gran éxito porque al resaltar los elementos más significativos de un fenómeno, permiten su entendimiento y explicación³ [ADE⁺01, p190]. Esta simplificación incluye supuestos sobre ciertos aspectos de los sistemas modelados, plasmados implícitamente dentro de ellos pero sin representación dentro del vocabulario propio de la metodología usada. Un ejemplo en [DN99]⁴ y [Lom99]. A la par de este planteamiento, se han hecho importantes críticas al método tradicional relacionadas con el origen del modelado, donde se cuestiona el uso de las ecuaciones diferenciales como único punto de partida para el modelado dinámico [Tof84, TM87, Lom99]⁵.

Con esto en mente, surgió la necesidad de contar con herramientas y metodologías nuevas para la representación de la realidad que incluyan aspectos omitidos por la metodología, que permitan acercar el lenguaje de representación al lenguaje natural y las representaciones al objeto paisaje observado, como la influencia del espacio en el desarrollo del fenómeno [Lom92]⁶[DN99, AMA00], además de la presentación de un enfoque nuevo de modelamiento dinámico.

Inquietos por estas ideas y tras varios meses de investigación, en 1998 el grupo Simon presentó a Homos 1.0: Herramienta software para el modelamiento y la simulación basado en objetos y reglas[MA98]. Su aporte más importante fué el desarrollo de la metodología de modelamiento basada en objetos y reglas MBOR, que está fundada en la combinación de características de los autómatas celulares y la tecnología orientada a objetos. Mediante una metáfora, la MBOR permite identificar los elementos del fenómeno a modelar como objetos que interactúan siguiendo un conjunto de reglas de comportamiento, distribuidos en un espacio determinado donde se desarrolla su dinámica.

Entre los antecedentes al desarrollo de la herramienta, se cuentan las aplicaciones software WaTor

³"Ninguna representación considera todos los elementos del fenómeno, siempre hay elementos no asumidos o *excluidos* para la [su] representación."

⁴"We use these [three different model implementations of the processes represented by the Lotka-Volterra predator-prey equations] models to investigate the system dynamics suppressed by the assumptions contained in differential-equation-based models; specifically, we examine the effects of space and demographic stochasticity."

⁵Sugieren el uso de autómatas celulares y la representación mediante individuos para el modelado de sistemas dinámicos.

⁶Desde el punto de vista de las falencias presentes en el estudio de la ecología.

[Dew84]⁷ y Picture Simulator [CFNF94]⁸, las cuales implementaban el modelo predador presa de Lotka-Volterra [May73], e incluían facilidades para cambiar su comportamiento mediante la variación de algunos de sus parámetros. Homos superó ampliamente las funcionalidades presentes en esas aplicaciones, al ser construida como un entorno práctico de modelado. En Homos es posible caracterizar el ambiente en el cual se desarrollará la simulación, los objetos que intervendrán y sus relaciones de interacción⁹; pueden definirse diferentes escenarios de simulación junto con la presentación de los valores obtenidos en ejecución mediante tablas de datos y gráficas.

Homos está planteado en un modelo centrado en los procesos del pensamiento [AP98], ha sido pensado como una herramienta de modelado de sistemas dinámicos y de trabajo para la informática educativa, al hacer posible la representación de modelos sistémicos en términos de objetos y reglas. Este enfoque contrasta con el modelo educativo tradicional, usualmente¹⁰ dirigido hacia la instrucción y la transmisión de conocimientos.

Posteriormente como una expansión natural de su línea de software, pensando en ampliar las capacidades didácticas de Homos y en potencializar sus capacidades como herramienta TIC¹¹, al permitir mediante una interfase multimedia e hipermedia animada la integración de diversos contenidos en los modelos educativos existentes en Homos, El grupo Simon crea en el 2000 a Micrho 1.0 [CD00], una aplicación para la creación de micromundos fundados en la MBOR. Con esta herramienta “... no sólo se trata de promover habilidades del pensamiento a través del juego con el computador y de la creación de ambientes de aprendizaje, sino que también permita la apreciación de ideas formales y de conceptos, se trata de que el niño se comprometa con modelos de aprendizaje cognitivos y de desarrollo que inspiren el aprendizaje por exploración y que se interesen por la interpretación y explicación de las relaciones del individuo con su mundo exterior.”[ACD00].

En 2007 se realizó la Propuesta informática para la educación soportada en el modelado basado en objetos y reglas [SS07]. En ella se establecen pautas para el uso de Homos como herramienta TIC, específicamente lo relacionado con los requerimientos y las características de los ambientes informáticos con los cuales mejorar la experiencia de aprendizaje con el software. Siguió a esta propuesta en 2008, Micrho 2.0 [RC08]. Esta versión implementa algunos de los elementos propuestos en [SS07], se centró en la mejora del control administrativo de sesiones de uso en el software y la actualización del aspecto gráfico de la interfaz con la cual se desarrollan micromundos soportados con Homos 1.0.

La presente tesis continúa el trabajo realizado en 1998 acerca de los conceptos y fundamentos sobre los cuales descansa la MBOR y la herramienta software. Se espera que los resultados obtenidos fortalezcan, renueven y hagan de Homos una herramienta relevante dentro del área de modelado

⁷Idea original por A.K. DEWDNEY en *Sharks and fish wage an ecological war on the toroidal planet Water*, publicado en Scientific American Diciembre 1984 e implementado por Dr. WARREN L. KOVACH en 1993. <http://www.kovcomp.co.uk/wator/index.html>

⁸Producto software del proyecto *Pictorial modelling of dynamic systems* por ANTONIO S. CAMARA et all, publicado en System Dynamic Review, Volumen 10 número 4, Winter 1994.

⁹Manejadas en la terminología de Homos como las reglas de transición o simplemente reglas.

¹⁰Justificable en épocas anteriores a la masificación del computador y su rol como herramienta educativa. Recientemente las actitudes conservadoras o innovadoras de los educadores hacen la diferencia [Kof03].

¹¹Tecnologías de Información y Comunicación, comprenden los desarrollos en metodologías multimedia, hardware, software, internet, plataformas móviles y demás avances tecnológicos relacionados.

y simulación de sistemas dinámicos.

2.3. Marco conceptual

2.3.1. Autómatas celulares y teoría de objetos

Uso de aut y t. de objetos en este proyecto

2.3.2. Teoría de grafos

definición de un grafo

uso de t de grafos en este proyecto

matroides y teoría de grafos

2.3.3. Teselados

clasificación de teselados

periodicos

- regulares [3,6] [4,4][6,3]

quasi-regular

semi-regular

no regular = voronoi

ejemplo de un algoritmo de teselado hexagonal en ver Quasi-regular Tessellation of Hexagons

- uniformes, semirregulares o de arquimedes

demiregulares

irregulares ver teselado_clases_05.pdf

aperiodicos

pinwheel y penrose RAdin! http://ma.utexas.edu/mp_arc/c/95/95-25.ps.gz

no regulares

tesela

teselado

http://www.utadeo.edu.co/comunidades/estudiantes/ciencias_basicas/geometria/teselado_clases_05.pdf

existen teselados

en este proyecto se hace uso general de algunos teselados periodicos, regulares e irregulares. se implementan los teselados de :

recursos sobre teselados en

<http://www.geom.uiuc.edu/software/tilings/>

<http://www.ics.uci.edu/~eppstein/junkyard/>

2.3.4. Algebra de procesos

2.3.5. Pensamiento combinatorio

razonamiento_combinatorio.pdf

combinatoria = forma avanzada de conteo

es una forma de pensamiento matemático dependiente de la teoria combinatoria

En principio, el pensamiento combinatorio puede abstraerse como un considerar dentro de un conjunto de alternativas. El pensamiento combinatorio como forma distinguible de pensamiento, es la capacidad de pensar en todas las combinaciones y variantes posibles de un fenómeno.

Se usa esta forma de pensamiento como una herramienta de solución de problemas, al permitir mediante un enfoque sistemático, ordenado, una proceso que facilite la comprensión y entendimiento del objeto de estudio. Ejemplos claros de su aplicación son el estudio que se hace en química de los compuestos moleculares y de polímeros, cadenas DNA y RNA en biología [ESW97].

Según [RG11] se han identificado cuatro niveles de pensamiento combinatorio. Estos son:

1. Investigación de algunos casos: diferenciación entre fenómenos.
2. Conteo completo de casos: independencia del contexto en el que surgen los casos.
3. Generación sistemática todos los casos posibles: generalización y abstracción.
4. Cambiar un problema combinatorial por otro: conceptualización, homologación con estructuras conocidas para llegar a la solución de problemas. Al llegar a este nivel se ha alcanzado una comprensión del razonamiento combinatorio.

Esta forma de pensamiento aplica al tema de estudio de este proyecto, ya que se presenta implícitamente durante el proceso de construcción un modelo BOR. Mediante la combinación de un conjunto finito y conocido de reglas de transición se busca expresar la dinámica de un fenómeno.

Diseño metodológico

3.1. Tipo de investigación

La investigación a desarrollar en este proyecto es de tipo sistémico - propositivo. Para conseguir este objetivo es necesaria una caracterización del objetivo de estudio, mediante un proceso de determinación de los elementos ausentes para luego realizar la etapa de presentación de propuestas. Estas se sustentan en los resultados obtenidos de la investigación, adjuntando las caracterizaciones teóricas y prácticas obtenidas durante el desarrollo de este proyecto.

3.2. Población

3.2.1. Objeto de estudio

Esta constituida por dos objetos de estudio que son: la metodología de modelamiento basada en objetos y reglas, junto con todos los elementos característicos y útiles teóricos y la herramienta software Homos 1.0. De esta última se busca una comprensión íntima de sus aspectos fundamentales.

3.3. Técnicas e instrumentos para la recopilación de datos

Estos son los medios propuestos por esta investigación para la obtención de datos e información de las características presentes en la herramienta software y la MBOR. El sentido de aplicación de medios usados es buscar desde lo particular a lo general.

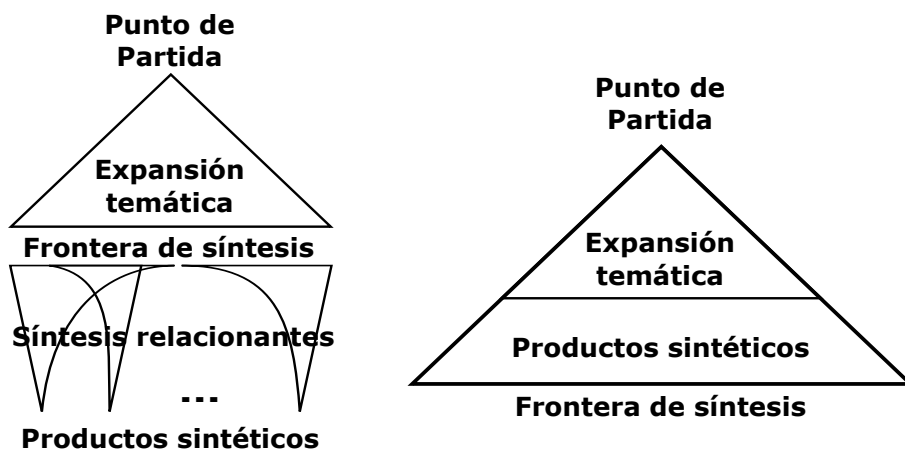
- Realización de procesos de análisis a la metodología de modelamiento en sus componentes principales:

- Desarrollo de un cuerpo de análisis sobre la herramienta software y la metodología MBOR.
 - Evaluación a la herramienta software.
- Determinación de elementos clave
 - Propuesta de elementos a incorporar, surgidos como conclusiones luego del analisis realizado
 - Determinación de los aportes mas importantes mediante propuesta de elementos nuevos y contraste con elementos ya existentes, esta etapa corresponde con la profundización de los elementos propuestos.
 - Integración de los elementos propuestos al cuerpo teórico de la metodología y la organización funcional del software.

Con estos elementos en mente, se plantean en general las distintas operaciones del proceso de investigación.

3.3.1. Metodología de investigación

Figura 3.1: Metodología de exploración holista



1. Punto de partida: tema a explorar, objeto definido de estudio, proviene de un proceso previo de Selección del tema a profundizar.
2. Expansión del todo temático: Búsqueda y exploración de las distintas facetas del objeto. entre mas facetas mejor, aunque hace mas lento el proceso. Se realiza mediante análisis, se estructura en un cuerpo analisis que facilita el ordenamiento de la exploración complementado con evaluaciones.

3. Frontera de síntesis: borde imaginario que define hasta donde se ha llegado con el proceso de exploración temática. para determinar el fin del proceso de inv se establece una frontera objetivo. La llegada a esta frontera se determina mediante el logro o no de los objetivos de investigación propuestos.
4. Búsqueda de conexiones y Síntesis relacionantes: A partir de los elementos obtenidos en la etapa de exploración, se busca la conexión entre los distintos elementos obtenidos del proceso de exploración, se presupone posible la separación de módulos o unidades cuando sea posible aquellos elementos que sean afines. independientemente del objeto de estudio, el proceso produce frontera de síntesis y uno o mas módulos temáticos. vigente así como solo el borde de exploración puede producir solo frontera de exploración,
5. Productos sintéticos obtenidos: son el resultado de las generalizaciones, abstracciones, teorizaciones de los elementos encontrados en el proceso de expl temática. Los módulos o unidades de conexiones obtenidas constituyen los productos del proceso de investigación. se presuponen inicialmente en la frontera objetivo, pero puede bien darse el caso de llegar a otros elementos inicialmente no previstos. como resultado del proceso, los productos obtenidos resultan naturalmente integrados al punto de partida del cual provienen.

La exploración holista es fractal, en cuanto a que al iniciar nuevos procesos de exploración tienen la misma forma, se dan nuevas unidades (sirven de fundamento a nuevas exploraciones) y fronteras de síntesis. al observar una cadena de exploraciones de este tipo se observan relaciones triángulo representa la totalidad presente en el punto de partida

3.3.1.1. Construcción de documentos

En general se usará el siguiente método para la elaboración de los contenidos presentados en este documento:

1. Determinación del tema a tratar: Selección de un tema específico, se refiere al contenido principal del documento, sobre lo que se va a tratar.
2. Generación de ideas, lluvia de ideas, expansión del tema. Según la cantidad de conocimiento sobre el tema se recurrirá a una fase previa de ilustración sobre el tema. dependiendo de si existen o se posee conocimiento semejante al requerido para tratar el tema
3. Detección y selección de ideas principales.
4. Organización de ideas mediante modelo o diagrama. la representación gráfica y la explicación de los elementos obtenidos facilita la generación de conexiones entre subtemas, permite construir progresivamente resultados, definiciones, expansiones, detectar falencias y fallos y demás operaciones de razonamiento que permiten establecer el ordenamiento lógico de las ideas a representar.

5. Concretar ideas en papel: Paso de las ideas concentradas de las etapas anteriores al papel, explicación y expansión de las ideas segun el orden establecido en la conexión y configuración de las ideas obtenidas del proceso. generalmente de la idea principal a las secundarias, jerárquicamente, relacionando conceptos previamente presentados, introduciendo ideas nuevas a medida que sean requeridas.

3.4. Prueba piloto

En este proyecto se presentará una prueba de funcionamiento del elemento o característica obtenido mediante investigación. Se espera que algunos de los elementos creados tengan naturaleza teórica, y cuando así sea, se presentan ejemplos de aplicación de los conceptos presentados.

3.5. Herramientas metodológicas adicionales

3.5.1. Metodología de desarrollo de software

Para el proceso de desarrollo del software a realizar por esta tesis, se elige el ciclo de vida de prototipado evolutivo. Según [Fl601, Apéndice 2, Ciclos de vida del desarrollo de software] este se caracteriza por:

- Iniciar el desarrollo del software desde partes previamente seleccionadas pasando al resto a partir de estas partes. Generalmente se comienza con las de mayor riesgo en el sistema.
- Permitir durante el proceso iterativo la determinación de los demás requisitos que completan el diseño, los cuales son incorporados / realimentados al sistema a medida que progresa el proyecto.
- No se descarta código, se reutiliza o modifica a medida que avanza el proyecto.
- Continúa el proceso hasta que se considera que ha llegado al punto de madurez, cuando es suficientemente bueno y se puede entregar como producto final.

Al igual que el prototipado simple, el prototipado evolutivo se ajusta al propósito general de esta tesis de proponer y probar funcionalidades, realizar actividades exploratorias y prueba de conceptos, en concordancia con lo planteado en los objetivos general y específicos.

Presentación y análisis de resultados

4.1. Descripción de resultados

Con la información obtenida mediante la investigación realizada, se introdujeron conceptos nuevos para la metodología de modelamiento, se obtuvieron resultados en los elementos más importantes, se encontraron elementos tales que permiten ampliar el alcance de los elementos teóricos actuales, tanto en la construcción de modelos como en la representación de espacios.

Se incrementaron las capacidades de representación de modelado en cuanto a su capacidad para describir relaciones mediante reglas de transición. ya no se determinaron los elementos que tanto productos teóricos y prácticos se encuentran en los anexos.

Fundamentos de grafómatas, análisis y construcción sistema de producción de reglas de transición la propuesta incluye, implementa y responde a las preguntas de investigación planteadas en donde se describe un diseño general para futuras herramientas y se presenta el lineamiento general de los prototipos realizados.

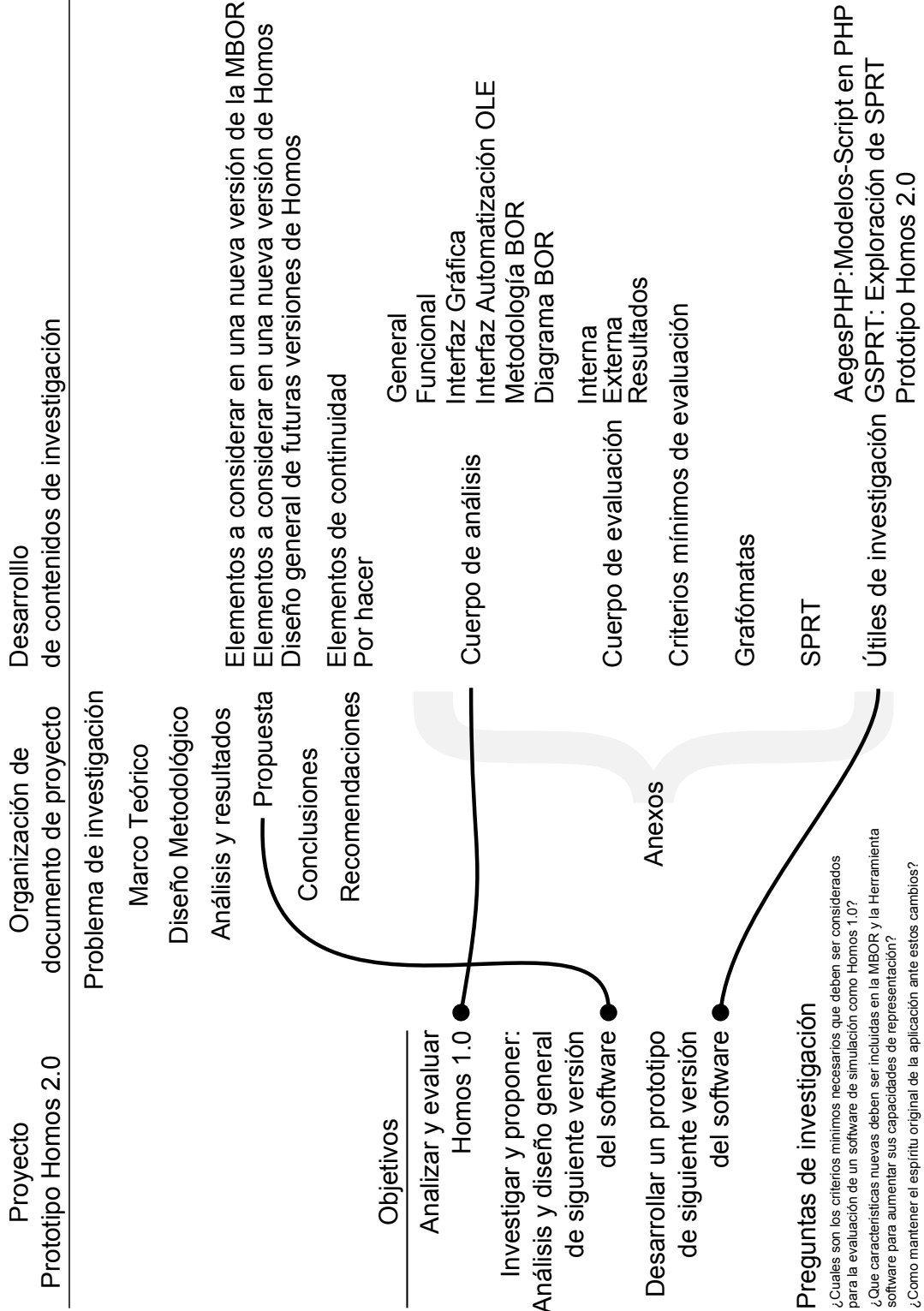
Se logró la caracterización de las reglas de transición de la MBOR actual en términos de los elementos teóricos encontrados en esta tesis, propuesta de inclusión de elementos funcionales nuevos en la herramienta software, derivados de experiencia de análisis, de desarrollo del software y de uso aplicado del software.

4.1.1. Organización del documento

Un nuevo marco teórico para la MBOR y las herramientas software Homos

a partir de los aportes teóricos realizados por este proyecto a la mbor y homos, se presenta un fondo teórico que da soporte teórico..

Figura 4.1: Plan de proyecto, cuerpo de investigación y estructura del documento



4.1.2. Proceso de investigación realizado

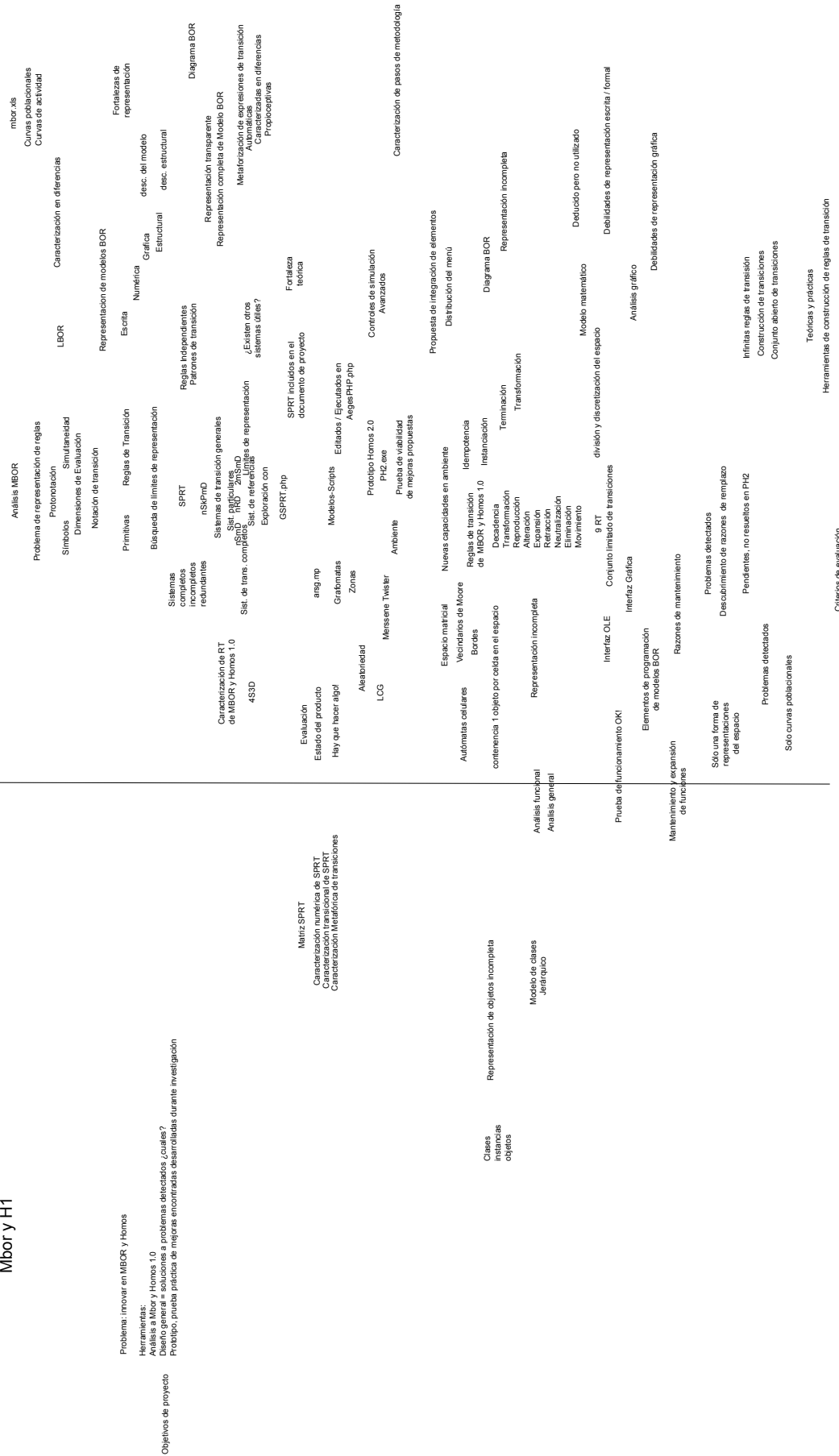
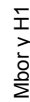
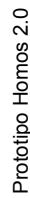
la figura 4.2 contiene una descripción grafica del proceso de investigación realizado

4.1.3. Resultados Teóricos obtenidos

Este es un compendio de los resultados teoricos obtenidos en el desarrollo de esta tesis:

- Desarrollo y evolucion de un formalismo escrito particular para la representacion de reglas de transición
- Desarrollo de un cuerpo de sistemas de produccion de sistemas de transicion, trabajo derivado del punto anterior, que se ofrece ahora como fundamento teórico principal para el desarrollo de transiciones
- Paso del modelo restringido de reglas de transición de Homos 1.0 a un modelo general, organizado y extensible de producción de reglas de transición, los SPRT.
- Representacion escrita y grafica de modelos basados en objetos y reglas de transicion adicionales a los previstos en la sección de nomenclatura prevista en la página 23 de la tesis de Homos 1.0.
- Caracterización de reglas de transición mediante primitivas y patrones de transición, metáforas automáticas, ecuaciones en diferencias y la inclusión de varios tipos de representación gráfica y escrita
- Descripción del proceso de modelado descrito por la metodologia usando la herramienta software.
- Caracterización de los SPRT mediante estudio de la distribucion de independencia de reglas y determinacion de patrones de transicion existentes
- Desarrollo de una estructura de grafo grafómata, caracterizada mediante elementos menores
- Implementacion de nuevas formas de representacion de reglas de transicion, tanto notacional escrita y gráfica.
- Expansión del modelo matemático presentado en página 24 la tesis de Homos 1.0, ampliado mediante representaciones alternativas de reglas de transicion usando sistemas SPRT, caracterizacion de reglas de transición mediante ecuaciones en diferencias por paso de simulación.
- Desarrollo de un algoritmo para la construccion automática de grafomatas duales a partir de un grafomata dado.
- Desarrollo de un algorimo para el relleno de superficies con grafómatas.

Figura 4.2: Plan de proyecto, cuerpo de investigación y estructura del documento



4.1.4. Resultados prácticos obtenidos

Este es un compendio de los resultados prácticos obtenidos en el desarrollo de esta tesis:

- Desarrollo de una guía para el desarrollo de modelos mediante scripts, haciendo uso del servidor de automatización OLE implementado en la herramienta software.
- Desarrollo de un modelo escrito para la representación de clases
- Desarrollo de un modelo escrito para la representación de modelos BOR a nivel de representación transicional y en ecuaciones de diferencias
- Generación del cuerpo de reglas de transición general de los sistemas SPRT 1S1D, 2S1D, 3S2D, 4S2D y 4S3D, todos usados como referencia a temas internos e incorporados al texto en la sección de anexos.
- Generación de nuevos tipos de curvas relacionadas con el modelo adicionalmente a las poblacionales. Se introducen las curvas de actividad, un simil al concepto de flujos presente en la dinámica de sistemas.
- Generación del cuerpo de reglas de transición general de los sistemas SPRT 1RD, 2RD, 3RD, 4RD, 5RD y 6RD. Los tres primeros sistemas han sido incorporados al cuerpo del documento, los restantes como anexo en el CD que acompaña esta tesis.
- Generación del cuerpo de reglas de transición general de los sistemas SPRT 2mSmD 4S2D, 6S3D, 8S4D, todos incluidos como anexo en el CD que acompaña esta tesis.
- Se presenta una propuesta de formato gráfico alternativo para los diagramas BOR

4.1.5. Resultados software

Este es el compendio de actividades y productos software obtenidos durante el desarrollo de la investigación y desarrollo aplicado sobre la MBOR y la herramienta software Homos 1.0. Los elementos presentados se entrelazan entre si, sinérgicamente con los distintos temas presentados en este documento.

- Mantenimiento y reparación de la interfaz OLE de Homos 1.0. Operación requerida y realizada durante la etapa de análisis del software
- Compilación de una última versión Homos 1.0 [1.0.0.73]. Esta versión no requiere serial y contiene los últimos cambios realizados durante el mantenimiento, sirvió de base para el desarrollo de los ejemplos de modelos mediante scripts presentados en este documento.
- Desarrollo de un script en PHP, AegesPHP.php que sirvió de herramienta de desarrollo de los distintos scripts PHP generados durante el proceso de investigación.

- Realización de pruebas de uso de la interfaz OLE mediante el desarrollo de modelos-scripts en PHP usando a Homos 1.0 [1.0.0.50 y 1.0.0.73] como servidor de aplicaciones.
- Desarrollo de un script en PHP, GSPRT.php que permite configurar, construir, explorar y caracterizar sistemas de producción de reglas de transición o SPRT.
- Desarrollo de un script en Metapost, arsg.mp que permite ver el resultado gráfico de funcionamiento del algoritmo de relleno del ambiente usando estructuras de grafómatas. Este script se creó a partir del algoritmo existente en el prototipo software.
- Desarrollo e implementación de un prototipo software de Homos 2.0 en etapas tempranas de desarrollo, que implementa algunas de las ideas presentadas en este documento:
 - Diseño del menú de la aplicación que muestra en forma integrada los elementos principales y característicos de una nueva versión de la herramienta software.
 - Uso y prueba de uso de zonas mediante presentación gráfica y conteo de objetos en ambiente.
 - Pruebas básicas de representación de objetos en ambiente.
 - Implementación de una opción rudimentaria de selección entre tipos de generadores de aleatoriedad, usando valores producidos por un algoritmo LCG¹ y el generador de números aleatorios Mersenne Twister presentado en [MN98].
 - Prueba de funcionamiento de opciones de manejo de la simulación:
 - Presentación de un menú de simulación experimental, parcialmente funcional que incluye:
 - Opciones de control de la velocidad de operación de la simulación.
 - Detención, Avance y retroceso de la simulación.
 - Implementación del concepto de grafómatas mediante:
 - La construcción de una librería de 25 estructuras gráficas² relacionadas con teselaciones regulares + 5 vecindarios de moore de distinto alcance.
 - Desarrollo de un conjunto de funciones básicas de manejo y de construcción automatizada de elementos característicos de los grafómatas.
 - Uso aplicado de los grafómatas para la ubicación en el espacio de los objetos contenidos en el ambiente de simulación.
 - Prueba de opciones gráficas de representación del espacio mediante grafómatas en el ambiente de simulación..
 - Prueba de operaciones de rotación, traslación y vista panorámica del ambiente de simulación.

¹Linear Congruential Generator, mediante la función `Random(LimitPlusOne : Integer)` implementada internamente en Delphi.

²En esta versión del software, generadas manualmente.

El ciclo de vida con el cual se desarrollaron los distintos productos software presentados es evolutivo. A partir de una idea básica se construye un prototipo inicial, se dan mejoras, expansiones y correcciones hasta llegar a la estabilidad; que se alcanza luego de muchas pruebas de uso y obtención de resultados deseados. Los distintos productos presentados no están en su forma definitiva, pero cumplen el propósito para el cual fueron creados.

4.2. Análisis de resultados obtenidos

La representación transparente de los modelos BOR propuesta en este proyecto supera a los elementos de notación gráfica proporcionados por la MBOR y homas 1.0, así como la propia representación gráfica de modelos BOR propuesta

Propuesta de diseño general de una siguiente versión del software Homos

En este apartado se da respuesta a las inquietudes de investigación consignadas en la formulación del problema planteado para esta tesis:

Se determina el conjunto de criterios usados para la evaluación de Homos 1.0, destilando de entre ellos los que se consideran mínimos necesarios para su análisis y evaluación.

Se presenta una propuesta de inclusiones, modificaciones y expansiones de características a implementar en la mbor y en futuras versiones de la herramienta software.

Las distintas afirmaciones realizadas y el fundamento de los distintos elementos propuestos descansan sobre el cuerpo de analisis y el trabajo teorico descrito en los anexos que componen este documento.

el anexo que habla sobre los sistemas de producción de reglas de transicion [SPRT] sirve de

fundamento de notacion escrita de reglas de transición

cuerpo de teorías sobre las cuales descansa la representacion transicional de los modelos bor

el anexo de grafomatas

el el apartado utiles de investigacion se muestran ejemplos practicos de algunos de los elementos presentados en esta propuesta, ejemplos de representacion de modelos BOR usando caracterizacion diferencial y

el diseño es la solucion propuesta a un problema

problemas encontrados en mbor y homos 1.0

aporte teorico con mejoras y cambios en la metodologia

aporte a la herramienta software indicando funcionalidades nuevas y su integracion sinergica con respecto de la metodologia sobre la que se fundamenta

5.1. ¿Qué características nuevas deben ser incluidas en la MBOR y la herramienta software para aumentar sus capacidades de representación?

problema: la mbor y homos requieren de nuevas características, las existentes son insuficientes porque se requieren cosas nuevas?

hay

Debilidades de representacion

Escrita

Gráfica

Teóricas

se desconocen los limites de representacion, hasta donde se puede llegar con transiciones??

5.1.1. Representación escrita de modelos BOR, LBOR

estas construcciones teoricas alteran la forma de uso de la metodologia en varias formas

proporcionan un fundamento matematico inmediato mediante la caracterizacion en diferencias.

se presentan a continuacion elementos que permiten establece de manera formal un Lenguaje basado en objetos y reglas [LBOR] con el cual expresar y construir modelos BOR.

en cada apartado se presenta

5.1.2. útiles de modelado

modelado con mbor

modelo prosa

modelo de objetos y reglas. MOR?

sistema de reglas de transición en notación de transición

representación transparente de reglas de transición

ecuaciones en diferencias.

metáforas descriptivas.

curvas de comportamiento poblacional.

curvas de actividad.

5.1.2.1. Modelo de clases

con el proposito de aumentar la capacidad de representacion de transiciones en la LBOR, se plantea una representación jerarquica como fundamento par ala representacion de un modelo de clases para un modelo BOR. este se estructura de la siguiente forma:

subclase

clase general

Clase

SubClase

SubsubClase

nombre del conjunto mayor

nombre de subconjunto

superclase

clase

subclase

subsubclase

representacion en forma jerárquica

//////////

G:\HDE\recuperacion\J\Z_deF\programas\modelolobosyconejos2.txt

Fuente en Pascal 5.1: Paso entre vértices del grafomata

```
1 modelo "lobos y conejos"
2 Documentacion "modelo realizado para presentar las capacidades del lenguaje de representacion texto
  de Homos2, en este caso el modelo predador presa"
3 Documentacion "estilo reglas internas a objetos"
4
5 Definicion
6 Documentacion "definicion de clases objeto, reglas y ambiente de simulacion"
7
8 clase_objeto Lobo
9   Documentacion "definicion de la clase lobo, depredador local"
10  imagen lobo.bmp
11  color azul
12  fin clase
13
14
15  regla mover
16    tipo movimiento
17    documentacion "Movimiento en el espacio"
18    ejecucion 10%
19    prioridad 1
```

```

20 fin regla
21 regla reproducir
22 tipo alteracion
23 documentacion "Continuidad de la especie"
24 ejecucion 10%
25 prioridad 1
26 fin regla
27 regla morir
28 tipo decadencia
29 documentacion "Muerte por vejez u otras causas"
30 ejecucion 10%
31 prioridad 1
32 fin regla
33
34
35 clase_objeto Conejo
36 Documentacion "definicion de la clase conejo, presa"
37 imagen conejo.bmp
38 color rojo
39 fin clase
40
41 ambiente Bosque
42 largo 200 celdas
43 ancho 200 celdas
44 limite infinito
45 fin ambiente
46
47 escenario "condiciones favorables"
48 usa ambiente bosque
49 usa objetos conejo y lobo
50 condiciones
51 iteraciones 1000 pasos
52 ubicar 100 conejos aleatoriamente en bosque
53 ubicar 10 lobos aleatoriamente en bosque
54 fin condiciones
55 end escenario
56
57 implementacion
58
59 iniciar escenario, simular escenario y terminar escenario si conejos = 0 o lobos = 0
60 fin modelo
61
62 definicion simular x
63 iniciar x
64 simulacion x
65 fin definicion
66
67 *****
68 modelo "lobos y conejos"
69 Documentacion "modelo realizado para presentar las capacidades del lenguaje de representacion texto
    de Homos2, en este caso el modelo predador presa"
70 Documentacion "estilo reglas internas a objetos"
71
72 Definicion
73 Documentacion "definicion de clases objeto, reglas y ambiente de simulacion"
74
75 clase_objeto Lobo
76 Documentacion "definicion de la clase lobo, depredador local"
77 imagen lobo.bmp
78 color azul
79 regla mover
80 tipo movimiento
81 documentacion "Movimiento en el espacio"
82 ejecucion 10%
83 prioridad 1
84 fin regla
85 regla reproducir
86 tipo alteracion
87 objeto alterador lobo
88 objeto alterado conejo
89 objeto resultado lobo
90 documentacion "Continuidad de la especie"
91 ejecucion 10%
92 prioridad 1
93 fin regla
94 regla morir
95 tipo decadencia

```

```

96   documentacion "Muerte por vejez u otras causas"
97   ejecucion 10%
98   prioridad 1
99   fin regla
100 fin clase
101
102 clase_objeto Conejo
103 Documentacion "definicion de la clase conejo, presa"
104 imagen conejo.bmp
105 color rojo
106 regla mover
107   documentacion "Movimiento en el espacio"
108   ejecucion 10%
109   prioridad 1
110   fin regla
111 regla reproducir
112   documentacion "Continuidad de la especie"
113   ejecucion 10%
114   prioridad 1
115   fin regla
116 regla morir
117   documentacion "Muerte por vejez u otras causas"
118   ejecucion 10%
119   prioridad 1
120   fin regla
121 fin clase
122
123 ambiente Bosque
124   largo 200 celdas
125   ancho 200 celdas
126   limite infinito
127 fin ambiente
128
129 escenario "condiciones favorables"
130 usa ambiente bosque
131 usa objetos conejo y lobo
132 condiciones
133   iteraciones 1000 pasos
134   ubicar 100 conejos aleatoriamente en bosque
135   ubicar 10 lobos aleatoriamente en bosque
136 fin condiciones
137 end escenario
138
139 implementacion
140   iniciar escenario, simular escenario y terminar escenario si conejos = 0 o lobos = 0
141 fin modelo
142
143 definicion simular x
144   iniciar x
145   simulacion x
146 fin definicion

```

//////////

5.1.2.2. Zonas

matematicas bordes

<http://grasshopperresources.blogspot.com/2010/05/matematicas-esenciales-para-diseno.html>

5.1.2.3. Representación de transiciones mediante notación de transición

Inclusión dentro de la metodología de los conceptos de

primitivas

reglas

patrones

expresiones de transicion de cada elemento,

metaforas asociadas

- diferenciales,
- automáticas,
- Descripciones automaticas, construidas a partir de la naturaleza de las primitivas que las componen
- de significado semántico arbitrario o propioceptiva¹: establecidas por los modeladores que dan sentido a su uso dentro del modelo

5.1.2.4. Representación escrita de modelos BOR mediante notación de transición

5.1.2.5. Caracterización en diferencias de modelos BOR

Un primer ejemplo de aplicación de caracterizacion de modelos bor en excel

5.1.2.6. Medios de representacion de transiciones

5.1.2.7. Representación transparente de modelos de transición

Con el propósito de hacer visible el proceso de cambio presente en los modelos,

Sirve de paso intermedio entre la representación tabular y la notación de orden y adición de reglas a la presentación grafica como diagrama BOR. La estructura general de la representación es la visualización del contenido de las transiciones haciendo uso de la notación prevista para cada patron de transición al que está asociado del que se deriva cada regla de transición presente.

Se fundamenta en el coeficiente de ejecución o actividad de la regla de transición identificado con el número de la regla de transición a la que pertenece que multiplica a la expresión de transición en notación compacta, la representación consiste en una sumatoria de términos, donde cada término es una regla de transición expresada como una multiplicación entre el coeficiente de ejecución o actividad de la regla y la expresión de transición particular de la regla de transición.

Ejemplos de esta forma de representación de transición de modelos bor se presentan en el anexo en las ecuaciones E.11E.12E.13E.14E.15E.16.

¹que se entiende aquí como la capacidad del modelador para comprender la función de la regla de transición en el modelo expresándola en sus propios términos, corresponde con la descripción de lo que el modelador entiende que hace la regla de transición dentro del modelo

Tabla 5.1: Representación tabular de transiciones de un modelo BOR

<i>Tabla de Símbolos</i>	<i>Id de Regla</i>	<i>Descripción según Notación de transición</i>	<i>Metáfora descriptiva en prosa</i>	<i>% Probabilidad de ejecución</i>	<i>\$ Prioridad de ejecución</i>	<i>Evaluación de celdas vecinas según configuración de regla</i>
s_1	r_1	Notación de orden $s_1, s_2 : s_k, s_n$	Descripción de la regla de transición r_1	$i \geq 0$	$0 \leq j \leq 255$	Descripción en función de la relación entre símbolos estructurada en la regla de transición.
$\vdots$	$\vdots$		$\vdots$			
s_n	r_k	Notación de adición $s_1 : s_n + s_2 : s_2$	Descripción de la regla de transición r_k			
Supuestos iniciales del modelo						
<i>Tabla de meta-entidades Metasímbolos</i>	<i>Id de Regla inicial</i>	<i>Descripción según Notación de transición</i>	<i>Metáfora descriptiva en prosa</i>	<i>#s_n Cantidades iniciales</i>	<i>Distribución de s_n en el ambiente</i>	
j_1	q_1	$j_1 : s_1$	Descripción del supuesto q_1	$0, 1, n, \infty$	Según formas implementadas: Uniforme, aleatoria, agrupada... etc.	
$\vdots$	$\vdots$					
j_m	q_w	$j_m : s_n$	Descripción del supuesto q_w			

Elementos adicionales de caracterización del formato tabular e implementación del modelo:

- Descripción del ambiente, tamaño, forma, geometría usada... total de objetos insertados y su distribución en el espacio.
- Descripción de la notación de transición usada para describir las reglas de transición del modelo, descripción de las abreviaturas incluidas en la tabla.
- Descripción del SPRT al que pertenece el modelo implementado. La cantidad de símbolos es la descrita en la tabla de símbolos del modelo. Las meta-entidades se presentan con propósitos de completitud sobre el origen de los símbolos mostrados en la representación escrita y el modelo de clases asociado.
- Inclusión de otros elementos o características que permitan la descripción del modelo, no contemplados en este formato.

Tabla 5.2: Caracterización en diferencias de modelos BOR

Id de Regla	Regla según notación de transición	Descripción de aportes poblacionales según primitivas presentes en r_k		Nombre de coeficientes de ejecución	Acción de primitivas $\times$ Actividad de reglas	Nivel*, Id., Símbolo, Clase	Flujo*, Sumatoria de aportes poblacionales
		Acción bruta de primitiva	Acción neta de primitiva	Actividad de regla	Aporte neto de primitiva	Población	Actividad de población
r_1	$s_2, s_2 : s_2, s_1$	$-s_2 - s_2 + s_2 + s_1$	$-s_2 + s_1$	$n_1 R_1$	$-n_1 s_2 + n_1 s_1$	s_1	$\sum n_i s_1 = n_1 + n_k + \dots$
r_2	$s_1, s_2 : s_2, s_1$	$-s_1 - s_2 + s_2 + s_1$	0	$n_2 R_2$	0	s_2	$\sum n_i s_2 = n_k - n_1 + \dots$
$\vdots$				$\vdots$		$\vdots$	
r_k	$s_1 : s_2$	$-s_1 + s_2$	$-s_1 + s_2$	$n_k R_k$	$-n_k s_1 + n_k s_2$	s_m	$\sum n_i s_m$

<i>Tendencia de reglas</i>	$Tr = \sum p_i R_k$	<i>Representación transparente</i>

Tabla 5.3: Actividad de primitivas y patrones presentes en el modelo

Id de primitiva	Primitiva según notación de transición	Actividad de primitiva	Tipo de primitiva	Id de patrón	Patrón según notación de transición	Actividad de patrón
p_1	$s_1 : s_2$	$\sum n_i p_1$	Cambio	P_1	$S_1 : S_1$	$\sum n_i P_1$
$\vdots$				$\vdots$		
p_h	$s_n : s_n$	$\sum n_i p_h$	Permanencia	P_z	$S_1, S_1 : S_1, S_2$	$\sum n_i P_z$
<i>Tendencia de primitivas</i>		$TP = \sum p_i$	$\frac{\sum cambio}{\#p} + \frac{\sum permanencia}{\#p}$	<i>Tendencia de patrones</i> $TP = \sum P_i$		

Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k .

Tr, Tp, TP son abreviaturas de tendencia de reglas, de primitivas y de patrones del sistema

Una vez conocidas las reglas de transición del sistema, el contenido de las tablas 5.2 y 5.3 son autogenerables.

Esta tablas son un ejemplo de automatización interna requerida de una forma de manipulables mediante automatización interna representacion intermedia a usar dentro del proceso de modelado

A futuro, la presentación de la acción bruta de primitivas en la tabla puede obviarse, ya que la acción neta es la relevante para construir las actividades de población. Se incluye parincluir este elemento como una opción en la generación de la tabla de caracterización diferencial del modelo

#p representa la cantidad de primitivas en uso con las que se construyen las reglas de transición del sistema

C y P representan la tendencia global de cambio y permanencia presentes en el modelo.

en la sección TAL se muestran ejemplos aplicados de uso de estas tablas, con el proposito incentivar y mejorar la comprension de su uso los valores de diferencias presentados en estas tablas [] se calculan por cada paso de simulación. Con los valores obtenidos es posible presentar *curvas de actividad* de cada uno de los elementos presentados.

$$\begin{aligned}
Modelo &= \sum_{i=1}^m n_i R_i \\
&= n_1 R_1 + n_2 R_2 + \cdots + n_k R_m \\
&= n_1 \left[\begin{array}{cc} S_i & \rightarrow - \\ - & S_j \end{array} \right] + \cdots + \\
&\quad + n_g \left[S_j \xrightarrow{c} S_k \right] + \cdots + \\
&\quad + n_h \left[S_k \rightarrow S_l \right] + \cdots
\end{aligned} \tag{5.1}$$

5.1.2.8. ejemplos de diagramas

reglas de transicion (conjunto) -> representaciones transparentes (ecuacion = suma de (partes)) -> diagramas (totales o partes)

5.1.3. Cambios en la representación del diagrama BOR

usando la representacion transparente como una expresion intermedia entre la notacion de reglas de transicion y la expresion grafica del diagrama,

uso de representaciones vectoriales para la construccion grafica de diagramas bor, se propone usar el formato *.SVG Scalable vector graphics de amplia aceptacion y soporte en la industria como primera opción para este tipo de representaciones.

Tabla 5.4: Componentes y representaciones disponibles para la MBOR

Elementos originales	Incorporaciones propuestas	Detalle
Modelo Prosa	Representación tabular del modelo BOR	Descripción del modelo Modelo según las reglas de transición existentes.
	Representación gráfica de transiciones	Representación iconográfica de las reglas de transición del modelo. Fácil de leer.
	Caracterización en diferencias del modelo BOR	Soporte matemático complementario al modelo de automata previsto en Homos 1.0
	Representación Transparente del modelo	Unión entre descripción matemática y expresión de transiciones. Precursor de los diagramas BOR
Diagrama BOR	Diagrama BOR simbólico, pictográfico y mixto	Propuesta de remplazo, mejoras en la representación de transiciones
Ejemplo de resultados de simulación Curvas poblacionales		Ambiente, gráficas de comparación, vistas, resultados en archivos texto y representación binaria de un proyecto.
		Valores poblacionales en función del tiempo
	Curvas de actividad poblacional	Cantidad de relaciones poblacionales
	Curvas de actividad de reglas	Cantidad de aplicación de reglas
	Curvas de actividad de primitivas	Cantidad de aplicación de primitivas
	Curvas de actividad de patrones de transición	Cantidad de aplicación de patrones
	Modelos Script (en PHP)	Representación del modelo en forma no tradicional, flexible.

5.1.4. Implementación de nuevas funciones en la herramienta software

Todo cambio en la metodología representa necesariamente una expresión funcional en la herramienta software.

5.1.4.1. Soporte a las formas de representación escrita y a los cambios planteados en el diagrama BOR

5.1.4.2. Representación de las divisiones en el espacio mediante grafómatas

5.1.4.3. Zonas en el ambiente

ejemplo conceptual de zonas en ambiente

ejemplos de zonas han sido implementados en el prototipo presentado en esta tesis. en el anexo tal se presenta mas informacion al respecto

5.1.4.4. Narración de simulación de modelos

5.1.4.5. Determinación de Hitos de simulación

5.1.4.6. Creación integrada de micromundos

5.1.4.7. Caracterizacion de reglas de transición

5.1.4.8. Inclusión de nuevos sistemas de generación de valores aleatorios

En el prototipo PH2.exe se implementa un ejemplo de esto, permitiendo la eleccion etre las rutina de numeros aleatorios de delphi y merssene twister

otros algoritmos a posibilidades de inclusion

5.1.4.9. Propuestas de extensión a la notación de transiciones

sobre el lenguaje de sprt, extension para el soporte de lenguaje de progrmacion

realizacion de condicionamiento, estructuras de control

acceso a variables

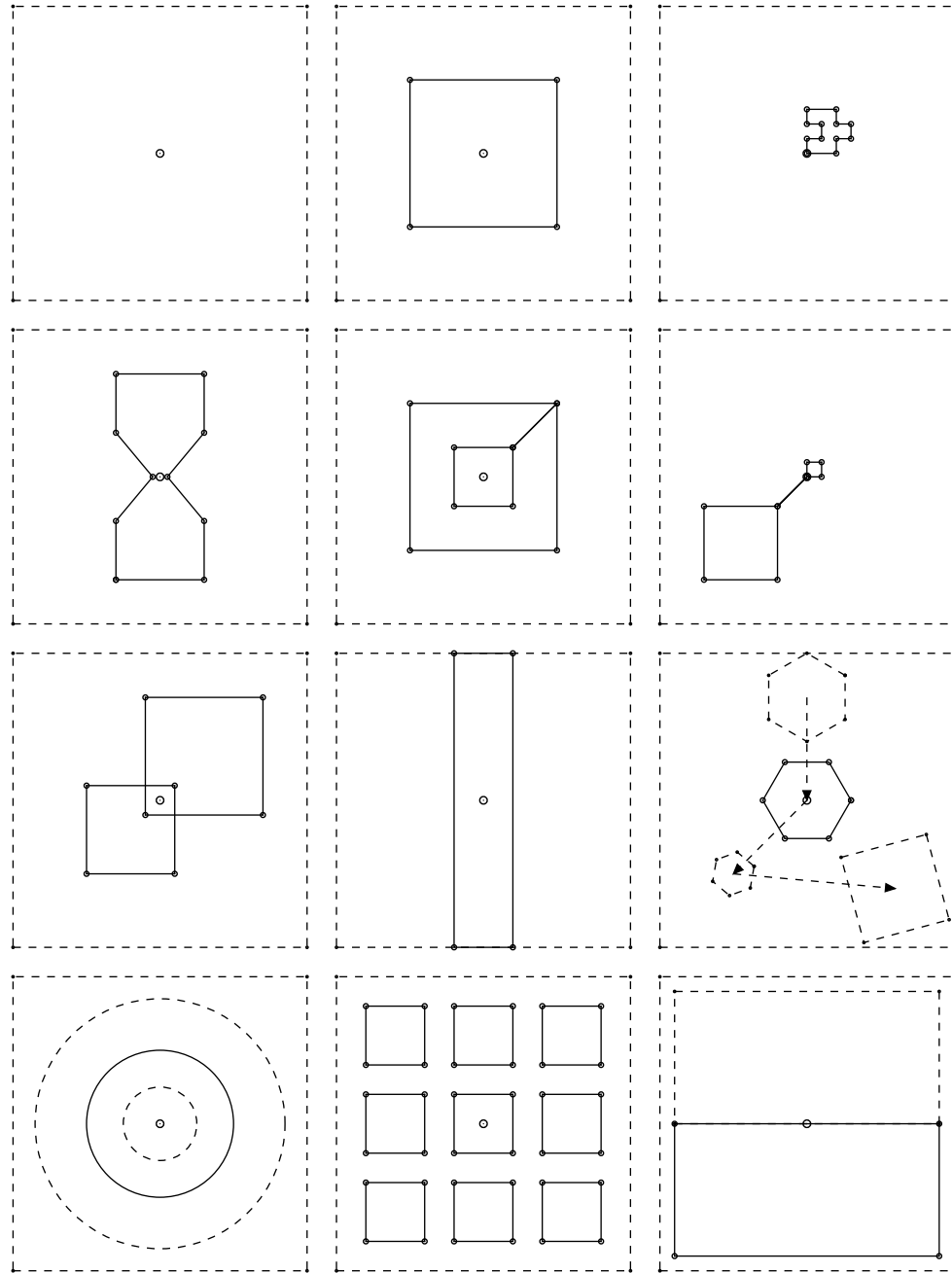
acceso a funciones

5.2. ¿Cuales son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?

problema: como evaluamos a homos 1.0?

que evaluar?

Figura 5.1: Ejemplos de aplicación conceptual de Zonas en ambiente



En la figura se muestran 12 ejemplos distintos de zonas. De arriba a abajo de izquierda a derecha:
 Espacio vacío: sin uso de zonas como en la figura E.50 en la página 669, zona cuadrada, zona de forma abstracta, Reloj de arena: ver figura E.58 en la página 673,
 Anillo cuadrado: ver figura E.59 en la página 673, Relación entre zonas cuadradas separadas, Solapamiento de zonas cuadradas, franja en el espacio, concepto de zonas móviles, zonas circulares concéntricas, zonas organizadas como cuadras en una ciudad y zona como separación de medios, P.ej: separación de agua y aire.
 Fuente: Autor.

5.3. ¿Cómo mantener el espíritu original de la aplicación ante estos cambios?

A continuacion se describe el impacto de los cambios propuestos a implementar en la MBOR y Homos 1.0

se describe el proposito u objetivo fundamental dela herramienta ta a en es

problema: que incluir en mbor y homos que represente una novedad pero que no cambie su proposito general

5.4. El espíritu de la MBOR y Homos 1.0

el autor de este documento considera que los cambios planteados aunque son fundamentales

respetan el espiritu de la aplicación

para que fue construida la aplicación

puede hacer ahora mas y mejor con las extensiones y cambios planteados

5.4.1. Evolución continuada de la herramienta software Homos

5.4.2. Proyección de investigación a largo plazo

Propuesta de planes de continuidad del proyecto de investigacion

1. creacion de soporte teorico

- a)* Profundizacion de la caracterizacion teorica de los SPRT y aplicaciones practicas
- b)* Desarrollo de los grafomatas

2. creacion de la herramienta

a) Motor de simulacion

- 1) desarrollo del lenguaje de simulación
- 2) desarrollo de interfaces para el uso aplicado del simulador

b) Motor grafico

- 1) Ambiente y entorno gráfico
- 2) Especializacion de representacion de divisiones del espacio y grafomatas de uso general

3. creacion de contenidos

a) Desarrollo de Mircromundos con la herramienta software

4. proyectos de mantenimiento y continuidad

a) Mantenimiento a la herramienta software

b) plantear planes regulares de mantenimiento de la herramienta software

c) reuniones de busqueda de aplicaciones, desarrollo de nuevos ejemplos, fortalecimiento teorico y avance de la MBOR

d) Derivacion de versiones reducidas o especializadas del motor de simulacion y la herramient software

alcances y limitaciones encontrados en la mbor y homos 1.0

superacion de los alcances y limitaciones encontrados, propuesta

Figura 5.2: Proyección

Soportados en fundamentos de MBOR

Proyección de actividades de desarrollo de la aplicación a largo plazo

Nivel teórico	Fortalecimiento de fundamentación teórica
	Profundización de conceptos asociados
	soporte a mejoras metodológicas

Nivel práctico	Aplicación de conceptos en elementos funcionales en el software
	Construcción de herramienta software modular
	Mejora en creación de micromundos
	mejora de productos
	Implementacion de

Nivel aplicado	Creación de modelos
	Creación de micromundos
	Paso a desarrollos derivados

Nivel autoreferente	Actividades de mantenimiento y seguimiento

Conclusiones

Una vez concluida esta investigación, es relevante presentar un compendio de pensamientos que redondean la experiencia académica vivida durante el desarrollo del proyecto

Ver alcances del proyecto y recomendaciones para futuros desarrollos p101 TH1

C_mo_hacer_una_conclusi_n.pdf

6.1. Propósitos cumplidos

Se encontró un fundamento teórico sólido para la mecánica de transiciones usada como principio de construcción de modelos por parte de la metodología de modelamiento basada en objetos y reglas.

- Se definieron sus componentes básicos, se dio una interpretación de cada elemento presentado y un alcance de aplicación de dichos elementos.
- Con los elementos establecidos se pueden construir, comprender mejor su funcionamiento y aplicar con efectividad sistemas de reglas de transición, se establecieron pautas generales para futuras versiones de la herramienta, estas pautas van desde recomendaciones de lo inmediatamente posible hasta mejoras en el alcance aplicativo.
- Se dio inicio a una nueva forma de ver el espacio para la herramienta.
- Se establecieron los fundamentos básicos de construcción de grafómatas,
- Se dio inicio al proceso de exploración y formalización de estructuras regulares de división del espacio para el uso particular de la herramienta.
- Agrega variedad representacional en múltiples formar.

Se encontraron algunos de los límites más importantes realización técnica, implementación practica a futuro en general, se dieron pasos importantes y suficientes al propósito de generar innovación en la metodología y su proyección como herramienta software, ampliando el horizonte de aplicación de sus herramientas conceptuales, encontrando los limites a los cuales se encuentra sujeta por la misma esencia de sus favoreciendo

6.2. Respuesta a inquietudes de investigación

este proyecto da respuestas a las inquietudes planteadas en las prueguntas de investigacion mediante los siguientes elementos: Sobre mejoras ala herramienta software, Sobre el espiritu de la aplicación, cual es el espiritu de la aplicacion, el espiritu de Homos 1.0?. Luego del analisis se afirma sobre la herramienta software que:

- Es constructivista por ser un simulador es en si misma tiene una naturaleza dinamica de su contenido.
- Consta de etapa de edicion y de realizacion o simulacion del modelo construido con los utiles previstos con la herramienta.
- El proposito educativo integrar una forma de pensar sistematica en la construccion de modelos.
- Proposito educativo aplicado depende del objetivo del modelador.
- La descripción que hace de lo observado en la naturaleza y capes fenomenologica.

Una característica de la aplicacion es que es constructivista, puesto que permite construir conceptos e ideas. la metodologia como tal no cambia, hace uso de utiles nuevos donde existian espacios vacios.

conclusiones:

El autor de este documento considera que se alcanzó el trabajo realizado realizó verdadera innovacion:

se brindo apoyo teorico y practico a la mbor y a homos

se reinicio el trabajo iniciado en 1998

El aspecto investigativo de este proyecto se centrará en dar respuesta a las siguientes inquietudes, surgidas del proceso de análisis preliminar, la observación del problema y la delimitación propuesta:

- Con respecto al análisis y evaluación de la herramienta:
 - ¿Cuales son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?

- Con respecto a la metodología de modelado:
 - ¿Qué características nuevas deben ser incluidas en la MBOR y la herramienta software para aumentar sus capacidades de representación?,
 - ¿Cómo mantener el espíritu original de la aplicación ante estos cambios?

Objetivo general

Desarrollar un prototipo software que implemente innovaciones para una segunda versión de la herramienta software HOMOS según los elementos teóricos propuestos durante el proceso de investigación.

Analizar y evaluar la herramienta Software HOMOS 1.0

Realizar un documento de análisis y evaluación de la herramienta software HOMOS 1.0 que proporcione un referente teórico para el presente proyecto y sirva de documentación interna sobre este software para el grupo Simon.

Investigar y proponer un diseño general de la herramienta para futuras versiones

Realizar un documento de análisis y diseño general de la herramienta para una siguiente versión del software que contemple mejoras a elementos encontrados en el documento de análisis y evaluación de la versión actual del software según los resultados obtenidos durante el proceso de investigación.

Desarrollar un prototipo software de la aplicación

Realizar un prototipo software que implemente innovaciones propias de una siguiente versión del software según los resultados obtenidos durante la investigación.

Recomendaciones

En [SS04] se propone una arquitectura de software para ambientes de modelado y simulación de eventos discretos.

Esta es una propuesta para una nueva versión, por lo cual la recomendación principal es considerar este estudio en el caso que se implemente dicha versión. En forma global, este proyecto aconseja:

- Profundizar en los elementos teóricos propuestos para la MBOR.
- Integrar los útiles de modelado propuestos en este trabajo de grado a la MBOR.
- Continuar el desarrollo de la herramienta software Homos 2.0 en el contexto de los cambios presentados por este proyecto.
- Continuar la evolución de los útiles de modelado desarrollados en esta tesis.

A continuación se muestran en detalle las caracterizaciones de los elementos recomendados.

7.1. Elementos de continuidad

Terminar el proceso de construcción automática de grafómatas iniciado en este proyecto e incorporar los resultados obtenidos a las futuras herramienta software homos. Construir el atlas de grafómatas, una guía de estructuras, [grafomatas regulares]. Una siguiente versión del herramienta software requiere del cambio del tipo de gráficos usados. se recomienda :

- Implementar el soporte y hacer uso de imágenes vectoriales para representar gráficamente a los objetos en el ambiente.
- Específicamente se propone hacer uso del formato Scalar Vector Graphics, se hace énfasis en SVG, porque es un tipo de documento amplia aceptación y soporte basado en XML, un estándar web w3c, fácil de editar y manejar.

- Establecer un formato abierto soportado por multiples browsers.
- Realizar cambios de panoramica y zoom al ambiente, conservando la forma de la imagen.
- Dar Flexibilidad y mejora estetica de las imagenes, el formato *.bmp e *.ico de 16x16 bits es insuficiente para presentar detalles gráficos.
- Dar soporte a formatos viejos, permitir nuevos tipos de imagen, especialmente vectoriales que no pierden su forma ante redimensiones.
- Modelos de produccion o uso, deberian contar con imagenes vectoriales como parte de su expresion gráfica.
- Realizar una biblioteca de imagenes de diferentes motivos, a usanza de la provista por Homos 1.0.

7.2. Por hacer

Se encontró un fundamento teórico sólido para la mecánica de transiciones usada como principio de construcción de modelos por parte de la metodología de modelamiento basada en objetos y reglas. Se definieron sus componentes básicos, se dio una interpretación de cada elemento presentado y un alcance de aplicación de dichos elementos.

Con los elementos establecidos se pueden construir, comprender mejor su funcionamiento y aplicar con efectividad sistemas de reglas de transición, se establecieron pautas generales para futuras versiones de la herramienta, estas pautas van desde recomendaciones de lo inmediatamente posible hasta mejoras en el alcance aplicativo:

- Se dio inicio a una nueva forma de ver el espacio para la herramienta.
- Se establecieron los fundamentos básicos de construcción de grafómatas,
- Se dio inicio al proceso de exploración y formalización de estructuras regulares de división del espacio para el uso particular de la herramienta.
- Agrega variedad representacional en múltiples formar.
- mejorar cada uno de los algoritmos asociados con la construccion y manipulacion de grafomatas.
- encontrar la manera de crear automaticamente grafomatas de teselaciones regulares.

Implementar la forma final de la herramienta software Homos 2.0 que integre sinérgicamente los elementos planteados en esta tesis y otros elementos de utilidad que puedan encontrarse en siguientes investigaciones y desarrollos futuros.

Se encontraron algunos de los límites más importantes realización técnica, implementación practica a futuro en general, se dieron pasos importantes y suficientes al propósito de generar innovación en la metodología y su proyección como herramienta software, ampliando el horizonte de aplicación de sus herramientas conceptuales, encontrando los limites a los cuales se encuentra sujeta.

7.2.1. Acciones inmediatas pendientes de realización

- Validar exitosamente el código fuente del script AegesPHP.php
http://validator.w3.org/#validate_by_input+with_options
- <http://validator.w3.org/>

7.3. Mirada al futuro

Esta es una propuesta para una nueva versión, por lo cual la recomendación principal es considerar este estudio en el caso que se implemente dicha versión.

Terminar el proceso de construcción automática de grafómatas iniciado en este proyecto e incorporar los resultados obtenidos a las futuras herramienta software homos. Construir el atlas de grafómatas, una guía de estructuras [grafómatas regulares].

- Una siguiente version del herramienta software requiere del cambio del tipo de graficos usados.
- Se recomienda implementar el soporte y hacer uso de imagenes vectoriales para representar graficamente alos objetos en el ambiente.
- Especificamente se propone hacer uso del formato Scalar Vector Graphics, se hace énfasis en SVG
- Formatos de salida, ampliar aceptación y soporte, porque es un tipo de documento basado en XML,un estandar web w3c, facil de editar y manejar.
- Formato abierto soportado por multiples browsers.
- Realizar cambios de panoramica y zoom al ambiente, conservando la forma de la imagen
- Flexibilidad y mejora estetica de las imagenes, el formato *.bmp e *.ico de 16x16 bits es insuficiente para presentar detalles gráficos.
- Dar soporte a formatos viejos, permitir nuevos tipos de imagenes, especialmente vectoriales que no pierden su forma ante redimensiones
- Realizar una biblioteca de imagenes de diferentes motivos, a usanza de la provista por Homos 1.0, con Modelos de produccion o uso, deberian contar con imagenes vectoriales como parte de su expresion grafica, establecidos como un formato abierto.

- ———hacer / implementar nuevos grafomatas para usar!

para la creacion de nuevas formas de grafomatas

formas especificas = mapa vial de una ciudad

formas generales = teselas regulares, p.ej: teselas de poligonos regulares, teselas periodicas y aperiodicas etc..

7.3.1. por hacer grf

Se cuenta con el fuente de los grafomatas base, falta el fuente de los grafomatas duales:

determinar el codigo fuente de los grafomatas duales. haciendo ingeniería inversa de los datos obtenidos, leer la estructura de un grafomata dual y expresarla en codigo fuente

fenomenos que se presentan al usar grafomatas como rellenos de superficies limitadas: líneas de costura

implementar mallas irregulares, teselados aperiodicos como los teselados de penrose (penrose tilings)

ya que son objetos graficos en condicioes iguales a los grafomatas, entonces son suceptivles de realziar en ellas las ideas propuestas para los grafomatas, tendran tambien costuras, ...

para esta adicion, se espera que el nivel de dificultad de implementacion aumente, debido a su innata no regularidad.

superficies esfericas

<http://www.ventrella.com/Alife/Cells/GlidersOnSpheres.pdf>

aperiodicos

<http://stephencollins.net/penrose/pdfs/Penrose.pdf>

7.4. Continuidad de investigación

guia de acciones recomendadas para la continuidad del proceso de inv y dess del soft homos y su metodología

———

se propone una iniciativa a largo plazo en la cual se constituya a homos como plataforma para el aprendizaje mediante juego

game-learning

Serious Games in business

draft

<https://www.inter-actief.utwente.nl/studiereis/pixel/files/indepth/YmeJoustra.pdf>

desarrollo de juegos serios

juegos serios (con proposito) y efectividad de aprendizaje

planes de investigacion en juegos serios

<http://seriousgames.msu.edu/>

<http://www.artdes.salford.ac.uk/our-courses/creative-games-msc/>

http://es.wikipedia.org/wiki/Juego_serio

enfoque de modelado orientado a juegos serios!

<http://www.seriousgames.org/>

http://www.seriousgames.org/presentations/serious-games-taxonomy-2008_web.pdf

- grafomatas,
- Los grafomatas implementados en esta version del prototipo usan valores enteros para determinar sus ubicaciones. Se espera que en siguientes iteraciones de desarrollo del prototipo que los grafomatas tambien hagan uso de valores flotantes.
- se reduce el uso de estructuras de listas enlazadas o matrices, se usan solo referencias numericas a ubicaciones en el espacio y la estructura del grafomata como funcion enrutadora en el teselado
- editor de grafomatas! con capacidades iniciales de edicion, subdivision y fusion de grafomatas

Bibliografía

- [ACD00] Andrade, Hugo, Amaury Cabarcas y Néstor Díaz: *MICRHO: Herramienta para el desarrollo de habilidades del pensamiento, soportado en la metodología de modelamiento basado en objetos y reglas*. En *Coloquio La Informática y los Jóvenes, Cuba*, 2000.
- [ADE⁺01] Andrade, Hugo, Isaac Dynner, Angela Espinosa, Hernán López Garay y Ricardo Sotaquirá: *Pensamiento Sistémico: Diversidad en Búsqueda de Unidad*. Ediciones UIS, 2001.
- [Agu96] Aguilar, Luis Joyanes: *Programación orientada a objetos*. McGraw-Hill, 1996.
- [Ake04] Akerkar, Salil R.: *Analysis and Visualization of Time-varying Data Using the Concept of Activity Modeling*". Tesis de Doctorado, Department of Electrical and Computer Engineering - University of Arizona, 2004.
- [Ala02] Alarcón, Gabriela González: *¿Qué observar cuando se evalúa software? Una propuesta para la evaluación didáctica de software educativo*. En *Congreso Iberoamericano Informática Educativa*, 2002.
- [Als09] Alshwaikh, Jehad: *Diagrams as interaction: The interpersonal (meta)function of geometrical diagrams*. En *Proceedings of the British Society for Research into Learning Mathematics* 29(1), Marzo 2009.
- [AMA98] Avellaneda, Oscar Lozano, Carmen Duarte Mogotocoro y Hugo Hernando Andrade: *HOMOS 1.0: Herramienta software para el modelamiento y simulación basado en objetos y reglas*. *Revista Iberoamericana de Informática educativa*, Mayo-Agosto(17):5-15, 1998.
- [AMA00] Andrade, Hugo Hernando, Carmen Duarte Mogotocoro y Oscar Lozano Avellaneda: *Paisajes Dinámicos con Homos 1.0*. *Revista Iberoamericana de Informática educativa*, Mayo-Agosto(17):5-15, 2000.
- [AN02] Andrade, Hugo y Ximena Marcela Navas: *Ingeniería de Sistemas: Realidad virtual y aprendizaje*. *Revista Ingenierías UIS*, 1(1):2-4, 2002.
- [AP98] Andrade, Hugo y Carlos Parra: *Esbozo de una propuesta de modelo educativo centrado en los procesos de pensamiento*. En *Cuarto Congreso Iberoamericano de Informática Educativa, Brasil*, 1998.

- [Ara92] Aracil, Javier: *Introducción a la dinámica de sistemas*. Alianza Editorial, Tercera edición, 1992.
- [AS95] Andrade, Hugo y Ricardo Sotaquirá: *Pensamiento de sistemas y dinámica de sistemas para el modelamiento de fenómenos de diversa naturaleza*. Universidad Industrial de Santander, 1995.
- [AS07] Alonso-Sanz, Ramón: *A Structurally Dynamic Cellular Automaton with Memory in the Triangular Tessellation*. *Complex Systems*, 17:1–15, 2007.
- [Bah11] Bah, Tavmjong: *Inkscape: Guide to a Vector Drawing Program*. SourceForge Community Press series. Prentice Hall, Cuarta edición, 2011.
- [Bar90] Barker, Richard: *CASE Method: Tasks and Deliverables*. Addison Wesley, 1990.
- [Bay94] Bays, Carter: *Cellular Automata in the Triangular Tessellation*. *Complex Systems*, 8:127–150, 1994.
- [Bay05] Bays, Carter: *A Note on the Game of Life in Hexagonal and Pentagonal Tessellations*. *Complex Systems*, 15:245–252, 2005.
- [Bay09] Bays, Carter: *Cellular automata on triangular, pentagonal and hexagonal tessellations*, 2009.
- [BB10a] Baetens, Jan M. y Bernard De Baets: *Continuous cellular automata on irregular tessellations: mimicking steady-state heat flow*. 2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC 2011), 1:53–59, 2010.
- [BB10b] Baetens, Jan M. y Bernard De Baets: *Phenomenological study of irregular cellular automata based on Lyapunov exponents and Jacobians*. *Chaos*, 20:15, 2010.
- [BdL09] Buchholz, Ralph H. y Warwick de Launey: *An Edge-Minimization Problem for Regular Polygons*. *the electronic journal of combinatorics*, 16:1–30, 2009.
- [Ber76] Bertalanfy, Ludwing Von: *Introducción a la teoría general de los sistemas*. Fondo de cultura económica, 1976.
- [Bez08] Bezos, Javier: *Ortotipografía y notaciones matemáticas*, Julio 2008. <http://www.tex-tipografia.com/archive/ortomatem.pdf>, Versión 0.15 2008-07-16.
- [Bir59] Bird, Otto: *Peirce's Theory of Methodology*. *Philosophy of Science*, 26(3):187–200, Julio 1959.
- [Bla98] Blackwell, Alan F.: *Metaphor in Diagrams*. Tesis de Doctorado, University of Cambridge, 1998.
- [Boo92] Boohan, R.: *WorldMaker: an object-based approach to computer modelling*. En *Proceedings of the European Conference about Information Technology in Education, University of Barcelona.*, páginas 30–39, Noviembre 1992.
- [Bor01] Borland: *Delphi 6 for windows Developer's Guide*. Sams, 2001.

- [BPS01] Bergstra, J.A., A. Ponse y S.A. Smolka (editores): *Handbook of Process Algebra*. Elsevier, 2001.
- [Bru09] Brualdi, Richard A.: *Introductory Combinatorics*. Pearson Education Asia Limited and China Machine Press, Quinta edición, 2009.
- [BVS⁺93] Björner, A., M. Las Vergnas, B. Sturmfels, N. White y G. Ziegler: *Oriented Matroids*, 1993.
- [Cam91] Camara, Antonio S.: *Multidimensional modelling for water resources management in Decision Support Systems: Water Resources Planning*. NATO ASI Series, G. 26:97–112, 1991.
- [Can97] Cantú, Marco: *Mastering Delphi 3*. SYBEX, Segunda edición, 1997.
- [Cas96] Castro, A. P.: *Dynamic water quality modeling using cellular automata*. Tesis de Doctorado, Virginia Tech Blacksburg, VA, 1996.
- [CD00] Cabarcas, Amaury y Néstor Díaz: *MICRHO, Micromundo para el desarrollo del pensamiento sistémico soportado en el Modelamiento basado en Objetos y Reglas*. Tesis de Grado, Universidad Industrial de Santander, UIS, 2000.
- [Cer00] Cerda, H.: *La evaluación como experiencia total*. Editorial Magisterio, 2000.
- [CF00] Chung, A. J. y A. J. Field: *A Simple Recursive Tessellator for Adaptive Surface Triangulation*. *Journal of graphics, gpu, and game tools*, 5(3):1–9, 2000.
- [CFC96] Camara, Antonio S., Francisco C. Ferreira y P. Castro: *Spatial simulation modelling*. En *Spatial Analytical Perspectives in Social and Environmental Sciences*, volumen Delft: Society for Computer Simulation, páginas 201–212, 1996.
- [CFD⁺91] Camara, Antonio S., Francisco C. Ferreira, Paulo Diogo, Pedro Gonçalves y Joao P. Silva: *Multimedia system dynamics models for environmental education*. En Kerchoff, E. (editor): *Proceedings of the European Simulation Symposium, Society of Computer Simulation*, páginas 227–280, 1991.
- [CFLS90] Camara, Antonio S., Francisco C. Ferreira, D. P. Loucks y M. J. Seixas: *Multidimensional simulation applied for water resources management*. *Water Resources Research*, 29(9):1877–1886, 1990.
- [CFNF94] Camara, Antonio S., Francisco C. Ferreira, Edmundo Nobre y Jose E. Fialho: *Pictorial modelling of dynamic systems*. *System Dynamic Review*, 10(4):361–373, Winter 1994.
- [CIT01] CITE: *WorldMaker 2000: User Manual*. University of Hong Kong, 2001.
- [CS95] Cypher, Allen y David C. Smith: *KidSim: End User Programming of Simulations*. En *Proceedings of CHI*, 1995.
- [CS96] Cornell, Gary y Troy Strain: *Programación en Delphi*. McGraw-Hill, 1996.
- [CS05] Cook, David A. y James M. Skinner: *How to Perform Credible Verification, Validation, and Accreditation for Modeling and Simulation*. The Aegis Technologies Group, Inc., 2005.

- [CV04] Costanza, Robert y Alexey Voinov: *Landscape Simulation Modeling. A Spatially Explicit, Dynamic Approach*. Springer-Verlag, 2004.
- [CY90a] Coad, Pete y Edward Yourdon: *OOA - Object oriented analysis*. Prentice Hall, 1990.
- [CY90b] Coad, Pete y Edward Yourdon: *OOD - Object oriented design*. Prentice Hall, 1990.
- [Dan85] Dancy, Jonathan P.: *An Introduction to Contemporary Epistemology*. Wiley-Blackwell, 1985.
- [Dew84] Dewdney, Alexander Keewatin: *Sharks and fish wage an ecological war on the toroidal planet Wa-Tor*. Scientific American, 251(6):14–22, Diciembre 1984.
- [DG88] Dougherty, Edward R. y Charles R. Giardina: *Mathematical methods for artificial intelligence and autonomous systems*. Prentice Hall, 1988.
- [DH00] Doggett, Michael y Johannes Hirche: *Adaptive View Dependent Tessellation of Displacement Maps*. En *SIGGRAPH / Eurographics Hardware Workshop*, 2000.
- [Die05] Diestel, Reinhard: *Graph Theory*. Springer, Tercera edición, 2005.
- [Dim02] Dimitrakopoulou, Angelique: *MODELLINGSPACE User requirements - Requirements, Design Principles & Specifications document (RDPS)*. School of Tomorrow. University of Aegean, Marzo 2002. http://ltee.org/adimitr/wp-content/uploads/2007/10/16-d01_modellingspace.pdf, Project IST-2000-25385.
- [dM00] Meulen, Meine Van der: *Definitions for Hardware/Software Reliability Engineers*. Springer, 2000.
- [dMyE00] Evaluación, Grupo de Medición y: *Marco de Interpretacion de Resultados*. ICFES - Subdireccion de Aseguramiento de la Calidad Grupo de Medición y Evaluación Educativa, 2000.
- [DN99] Donalson, Douglas D. y Roger M. Nisbet: *Population dynamics and spatial scale: effects of system size on population persistence*. Ecology, 80(8):2492–2507, 1999.
- [Eco96] Ecopetrol: *Plan estratégico para el desarrollo de sistemas de información orientados a objetos - reutilización y estándares*. Ingenieros de sistemas Ltda - Ecopetrol - Distrito centro oriente, 1996.
- [ESP05] ESPASA: *Diccionario de la lengua española*. Espasa-Calpe, S.A., Pozuelo de Alarcón, edición, Septiembre 2005.
- [ESP06] ESPASA: *Diccionario de sinónimos y antónimos*. Editorial:Espasa-Calpe, S.A. Pozuelo de Alarcón, Abril 2006.
- [ESW97] Ellman, Jonathan, Barry Stoddard y Jim Wells: *Combinatorial thinking in chemistry and biology*. Proceedings of the National Academy of Sciences of the United States of America, 94(7):2779–2782, 1997.

- [FH01] Flache, A. y R. Hegselmann: *Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics*. Journal of Artificial Societies and Social Simulation, 4(4):1, 2001.
- [Fló01] Flórez, Luis Carlos Gómez: *Planeación de proyectos, un enfoque para ingeniería de Sistemas e informática*. Universidad Industrial de Santander, UIS, Primera edición, 2001.
- [Ger90] Gerling, R. W.: *Classification of Triangular and Honeycomb Cellular Automata*. Physica A, 162:196–209, 1990.
- [GMS94] Goossens, Michel, Frank Mittelbach y Alexander Samarin: *The LaTeX Companion*. Addison-Wesley, 1994.
- [GO97] Goodman, Jacob E. y Joseph O'Rourke (editores): *Handbook of Discrete and Computational Geometry*. Discrete Mathematics Its Applications. CRC Press, Primera edición, 1997.
- [GO04] Goodman, Jacob E. y Joseph O'Rourke (editores): *Handbook of Discrete and Computational Geometry*. Discrete Mathematics Its Applications. CRC Press, Segunda edición, 2004.
- [Goe04] Goemans, Michel: *Topics in Combinatorial Optimization, Spring 2004*. <http://www-math.mit.edu/goemans/18997-CO/topics-co.html>, 2004.
- [GR99] Goossens, Michel y Sebastian Rahtz: *The LaTeX Web Companion - Integrating TEX HTML and XML*. Addison-Wesley, 1999.
- [Grä07] Grätzer, George: *More Math Into LaTeX*. Springer, Cuarta edición, 2007.
- [Gra98] Graells, Pere Marquès: *Diseño y evaluación de programas educativos*, 1998.
- [Gra01] Graells, Pere Marquès: *Ficha de catalogación y evaluación multimedia*, 2001. Universidad Autonoma de Barcelona.
- [Gra02] Graells, Pere Marquès: *Evaluación y selección de software educativo*. En II Congreso Nacional de Nuevas Tecnologías y Necesidades Educativas Especiales. Universidad Autónoma de Barcelona, 19-21 de septiembre 2002.
- [GRM97] Goossens, Michael, Sebastian Rahtz y Frank Mittelbach: *The LaTeX Graphics Companion - Illustrating documents with TEX and PostScript*. Addison-Wesley, 1997.
- [GS87] Grünbaum, Branko y G.C Sheppard: *Tilings and patterns*. W. H. Freeman and Company, 1987.
- [GSD⁺01] Ganguly, Niloy, Biplab K Sikdar, Andreas Deutsch, Geoffrey Canright y P Pal Chaudhuri: *A Survey on Cellular Automata*. <http://www.cs.unibo.it/bison/publications/CAsurvey.pdf>, 2001.
- [Güt98] Güting, Ralf Hartmut: *Tutorial in Spatial Database Systems*, 1998. <http://mif.vu.lt/cs2/en/courses/ispatal/files/>.

- [Gut91] Gutowitz, Howard A.: *Transients, cycles and complexity in cellular automata*. Physical Review A, 44(12):7881–7884, 1991. citeseer.ist.psu.edu/gutowitz91transients.html.
- [GW93] Gruber, P.M. y J.M. Wills (editores): *Handbook of Convex Geometry - Volume B*. North-Holland, 1993.
- [HBVC07] Hlupic, Vlatka, Vesna Bosilj-Vuksic y Vlatko Ceric: *Criteria for the Evaluation of Business Process*. En *Interdisciplinary Journal of Information, Knowledge, and Management*, volumen 2, 2007.
- [Hlu01] Hlupic, Vlatka: *Simulation Software: a survey of academic and industrial users*. International Journal of Simulation, 1(1-2):1–11, 2001.
- [HMW07] Hammam, Yasser, Antoni Moore y Peter Whigham: *The dynamic geometry of geographical vector agents*. Computers, Environment and Urban Systems, 31:502–519, 2007.
- [Hob89] Hobby, John D.: *MetaPost Publications*, 1989. <http://ect.bell-labs.com/who/hobby/mppubs.html>.
- [Hob92a] Hobby, John D.: *Introduction to MetaPost*. En *EuroTEX '92 Proceedings*, páginas 21–36, Septiembre 1992.
- [Hob92b] Hobby, John D.: *A User's manual for MetaPost*. Computing Science Technical Report 162, AT&T Bell Laboratories, 1992.
- [Hob93] Hobby, John D.: *Drawing Graphs with MetaPost*. Computing Science Technical Report 164, AT&T Bell Laboratories, 1993.
- [HS74] Hirsch, Morris W. y Stephen Smale: *Differential Equations, Dynamical Systems and Linear Algebra*. Academic Press, Primera edición, 1974.
- [HSD04] Hirsch, Morris W., Stephen Smale y Robert L. Devaney: *Differential Equations, Dynamical Systems and an introduction to chaos*, volumen 60 de *Pure and Applied Mathematics*. Academic Press, Segunda edición, 2004.
- [ICO98] ICONTEC: *Documentación, Presentación de tesis, trabajos de grado y otros trabajos de investigación*. Instituto Colombiano de Normas Técnicas ICONTEC, Cuarta edición, 1998.
- [ICO08a] ICONTEC: *Compendio Tesis y Otros trabajos de Grado*. Instituto Colombiano de Normas Técnicas ICONTEC, Quinta edición, 2008.
- [ICO08b] ICONTEC: *Documentación, Presentación de tesis, trabajos de grado y otros trabajos de investigación*. Instituto Colombiano de Normas Técnicas ICONTEC, Sexta edición, 2008.
- [Ita94] Itami, Robert M.: *Simulating spatial dynamics: cellular automata theory*. Landscape and Urban Planning, 30:27–47, 1994.
- [Jho09] Jhonson, Will: *Matroids*. http://www.math.washington.edu/~morrow/336_09/papers/Will.pdf, Junio 2009.

- [Joh02] Johansson, Björn: *Discrete Event Simulation - present situation and future potential*. Informe técnico, Department of Product and Production Development - Chalmers University of Technology, 2002.
- [Jou99] Journal, Dr. Dobb's: *Dr. Dobb's Essential Books on Algorithms and Data Structures*. Miller Freeman, 1999.
- [Jou10] Joustra, Y.: *Serious games in Bussines*. www.inter-actief.utwente.nl/.../YmeJoustra.pdf, 2010.
- [Juo11] Juozapavicius, A.: *Spatial database sistems*. Vilnius University, 2011. Course Notes.
- [Kan02] Kan, Stephen H.: *Metrics and Models in Software Quality Engineering*. Addison-Wesley, Segunda edición, 2002.
- [KB84] Kolman, Bernard y Robert C. Busby: *Estructuras de matemáticas discretas para la computación*. Prentice Hall, 1984.
- [KD99] Kopka, Helmut y Patrick W. Daly: *A Guide to LaTeX*. Addison-Wesley, Tercera edición, 1999.
- [Knu84] Knuth, Donald E.: *The TeXbook*. Addison-Wesley, 1984.
- [Knu86] Knuth, Donald E.: *The METAFONTbook*. Addison-Wesley, 1986.
- [Knu97] Knuth, Donald E.: *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997, ISBN 0201896842.
- [Kof03] Koffman, Hugo: *Nuevas tecnologías en la enseñanza: ¿Continentes o herramientas de la cultura humana?* OEI - Revista Iberoamericana de Educación - Sección de los lectores, none:none, 2003.
- [Kul94] Kulpa, Zenon: *Diagrammatic Representation And Reasoning*. Machine Graphics & Vision, 3:77–103, 1994.
- [Kul01] Kulpa, Zenon: *Diagrammatic Representation for Interval Arithmetic*. Linear Algebra and Its Applications,, 1(324):55–80, 2001.
- [Kul08] Kulpa, Zenon: *What is diagrammatics?*, 2008. <http://www.ippt.gov.pl/~zkulpa/diagrams/fpptd/WID.pdf>.
- [KZ01] Kaucic, Branko y Borut Zalik: *Quasi-regular Tessellation of Hexagons*. Journal of Computing and Information Technology - CIT 9, 3:227–232, 2001. <http://hrcak.srce.hr/file/69434>.
- [Lak92] Lakoff, George: *The Contemporary Theory of Metaphor*. En *Metaphor and Thought*. Cambridge University Press, 1992.

- [Lam94] Lamport, Leslie: *LaTeX: A Document Preparation System - User Guide and Reference Manual*. Addison-Wesley, Segunda edición, 1994. Versión 2.47.
- [LC94a] Ludwing, Michael y Michelle Courant: *Modelling interactions between autonomous agents*. En *Proceedings of the System Dynamics Conference, SD'94: Exploring the Boundaries*, Stirling, Scotland, 11 - 15 July 1994.
- [LC94b] Lumetta, Steven S. y David E. Culler: *Mantis User's Guide, Version 1.0*. Informe técnico UCB/CSD-94-828, EECS Department, University of California, Berkeley, Agosto 1994.
- [LCS88] Larkin, Timothy S., Raymond I. Carruthers y Richard S. Soper: *Simulation and object-oriented programming: the development of SERB*. SIMULATION, 51(3):93–100, 1988.
- [Lis00] Lischner, Ray: *Delphi in a Nutshell*. O'Reilly, 2000.
- [LLLT00] Law, N., S. C. Li, R. Li y P. Tang: *WorldMaker: Making Collaborative Exploration of Complex Systems Accessible to Children*. En *Proceedings of the 4th Global Chinese Conference on Computers in Education*, páginas 448–455, 2000. Singapore.
- [IMS00] Álvarez, Amaury Cabarcas, Néstor M. Díaz Mariño y Hugo Hernando Andrade Sosa: *MICRHO, Micromundo para el desarrollo del pensamiento sistémico soportado en el Modelamiento basado en Objetos y Reglas*. Artículo presentado en el Coloquio la Informática y los Jóvenes. La Habana, Cuba., 1:12, 2000.
- [Lom92] Lomnicki, Adam: *Population Ecology from the Individual Perspective*. En De Angelis, D. y L. Gross (editores): *Individual based models and approaches in ecology. Population, communities and ecosystems.*, páginas 3–17. Chapman & Hall, New York, EEUU, 1992.
- [Lom99] Lomnicki, Adam: *Individual-based models and the individual-based approach to population ecology*. Ecological Modelling, 115:191–198, 1999.
- [LT98a] Laurini, Robert y Derek Thompson: *Fundamentals of Spatial Information Systems*. Academic Press, 1998.
- [LT98b] Law, Nancy y Eric W. C. Tam: *WORLDMAKER (HK) - an Iconic Modelling Tool for Children to Explore Complex Behaviour*. En *ICCE98 - International Conference on Computers in Education*, 1998.
- [LY03] Lince, Emiliani y Mario Yeneris: *Evolución 3.5, Herramienta software para el modelamiento y simulación con dinámica de sistemas*. Tesis de Grado, Universidad Industrial de Santander, 2003.
- [MA98] Mogotocoro, Carmen Duarte y Oscar Lozano Avellaneda: *HOMOS 1.0. Herramienta software para el modelamiento y simulación basado en objetos y reglas*. Tesis de grado, Universidad Industrial de Santander, 1998.
- [Man96] Manning, Michelle M.: *Delphi 2.0 Guía Oficial de Borland*. Borland Press - Prentice Hall, 1996.

- [Mar07] Margenstern, M.: *Cellular Automata in Hyperbolic Spaces*. Old City Publishing, 2007.
- [May73] May, Robert M.: *Stability and Complexity in Model Ecosystems*. Princeton University Press, Princeton, NJ, 1973.
- [May76] May, Robert M.: *Simple mathematical models with very complicated dynamics*. *Nature*, 261:459–467, 1976.
- [May01a] Mayor, Pablo Grech: *Introducción a la Ingeniería - Un enfoque a través del diseño*. Pearson Educación de Colombia Ltda, Primera edición, 2001.
- [May01b] Mayor, Pablo Grech: *Introducción a la ingeniería: un enfoque a través del diseño*. Pearson Education, 2001.
- [Mey09] Meyers, Robert A.: *Encyclopedia of Complexity and Systems Science*. Springer, 2009.
- [MGB⁺04] Mittelbach, Frank, Michael Goossens, Johannes Braams, David Carlisle y Chris Rowley: *The LaTeX Companion - Tools and Techniques for Computer Typesetting*. Addison-Wesley, Segunda edición, 2004.
- [MN98] Matsumoto, Makoto y Takuji Nishimura: *Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.
- [MO94] Martin, James y James J. Odell: *Análisis y diseño orientado a objetos*. Prentice Hall Hispanoamericana, 1994.
- [Mon00] Montagut, Martha Vitalia Corredor: *Principios de inteligencia artificial & sistemas expertos*. Universidad Industrial de Santander, UIS, 2000.
- [MS01] McGregor, John y David Sykes: *A Practical Guide to Testing Object-Oriented Software*. Addison Wesley, 2001.
- [Nar97] Narayanan, N. Hari: *Diagrammatic Communication: A Taxonomic Overview*. Informe técnico, Visual Information, Intelligence & Interaction Research Group. Department of Computer Science & Engineering, Auburn University, Alabama 36849-5347. USA., 1997.
- [NPH98] Nikoukaran, Jalal, Ray J. Paul y Vlatka Hlupic: *Criteria for simulation software evaluation*. En D.J. Medeiros, E.F. Watson, J.S. Carson y M.S. Manivannan (editores): *Winter Simulation Conference, Proceedings of the 30th conference on Winter Simulation*, páginas 339–406. Department of Information Systems and Computing Brunel University Uxbridge, Middlesex UB8 3PH, UNITED KINGDOM, 1998.
- [oEEE98] Electrical, The Institute of y Inc. Electronics Engineers: *IEEE Recommended Practice for Software Requirements Specifications*. IEEE, 1998.
- [Ox92] Oxley, James G.: *Matroid Theory*. Oxford Univ. Press, 1992.

- [Pic09] Pick, Stephen: *Four Steps to Innovation: Lessons Learned from the Intelligence Community*. Organization Development Practitioner, 41(3):37–41, 2009.
- [Pop92] Popken, Douglas A.: *An object-oriented simulation environment for air base logistics*. Simulation, 59(5):328–338, 1992.
- [PR02] Pfeifle, Julian y Jörg Rambau: *Computing Triangulations Using Oriented Matroids*. Informe técnico, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2002. <http://www.zib.de/Publications/Reports/ZR-02-02.pdf>.
- [Pre93] Pressman, Roger S.: *Ingeniería del software, un enfoque práctico*. McGraw-Hill, Tercera edición, 1993.
- [Pre98] Pressman, Roger S.: *Ingeniería del Software, Un Enfoque Práctico*. McGraw-Hill Interamericana, Cuarta edición, 1998.
- [Pre02] Pressman, Roger S.: *Ingeniería Del Software, Un Enfoque Práctico*. McGraw-Hill Interamericana, Quinta edición, 2002.
- [PV92] Purcell, Edwin J. y Dale Varberg: *Cálculo con geometría analítica*. Prentice Hall, Sexta edición, 1992.
- [Qui08] Quintero, María Victoria Vargas: *Sentido pedagógico de la evaluación. Estudio de caso en la Facultad de Ciencias de la Pontificia Universidad Javeriana*. Universitas Scientiarum, 13:252–257, 2008.
- [Rac90] Raczynski, Stanislaw: *PASION: object-oriented simulation on the PC*. En *Proceedings of the '1990 System Dynamics Conference', Massachusetts*, páginas 908–917, 1990.
- [Rad85] Radin, Charles: *Tiling, periodicity, and crystals*. Journal of Mathematical Physics, 26(6):1342–1344, Junio 1985.
- [Rad94a] Radin, Charles: *Miles of Tiles*. Mathematics Department University of Texas, 1994.
- [Rad94b] Radin, Charles: *The Pinwheel Tilings of the plane*. The Annals of Mathematics, 139(3):661–702, 1994.
- [Rad00] Radin, Charles: *Statistical Sym*, 2000.
- [RC08] Ramos, John Fredy Ortiz y Diana Carolina Suárez Celis: *MICRHO 2.0, Micromundo para el Desarrollo del Pensamiento Sistémico Soportado en el Modelamiento Basado en Objetos y Reglas*. Tesis de Grado, Universidad Industrial de Santander, 2008.
- [Öre81] Ören, Tuncer I.: *Concepts and Criteria to Assess Acceptability of Simulation Studies: A Frame of Reference*. Communications of the ACM, 24(4):180–189, 1981. Simulation Modeling and Statistical Computing.
- [Rei98a] Reisdorph, Kent: *Sams Teach Yourself Delphi 4 in 21 Days*. Sams Publishing, 1998.

- [Rei98b] Reisdorph, Kent: *Sams Teach Yourself Delphi 4 in 21 Days*. Sams Publishing, 1998.
- [RG95] R. Graham, M. Grötschel, L. Lovász (editor): *Handbook of Combinatorics*. Elsevier, 1995.
- [RG11] Rezaiea, Mani y Zahra Gooya: *What do I mean by combinatorial thinking?* *Procedia - Social and Behavioral Sciences*, 11:122–126, 2011.
- [RNM03] Rus, Ioana, Holger Neu y Jürgen Münch: *A Systematic Methodology for Developing Discrete Event Simulation Models of Software Development Processes*. En *ProSim Workshop '03. Software Process Simulation Modeling*, 2003.
- [Roe01] Roegel, Denis: *Space geometry with METAPOST*. *TUGboat*, 22(4):298–314, 2001.
- [RP94] Ruth, Mattias y Frederick Pieper: *Modeling spatial dynamics of sea-level rise in a coastal area*. *System Dynamic Review*, 10(4):375–389, Winter 1994.
- [Rub98] Ruble, David A.: *Análisis y diseño práctico para sistemas cliente/servidor con GUI*. Prentice Hall, 1998.
- [RW92] Radin, Charles y Mayhew Wolff: *Space tilings and local isomorphism*. *Geometriae Dedicata*, 42:355–360, 1992.
- [RY] Rätzmann, Manfred y Clinton De Young.
- [Sar00] Sargent, Robert G.: *Verification, Validation, And Accreditation Of Simulation Models*. En Joines, J. A., R. R. Barton, K. Kang y P. A. Fishwick (editores): *Proceedings of the 2000 Winter Simulation Conference*, 2000.
- [Sar06] Sarjoughain, Hessam S.: *Model Composability*. En *Proceedings of the 2006 Winter Simulation Conference*, 2006.
- [Sch08] Schulz, Klaus Peter: *The nature of innovation and implications on innovation management*. En *Insightful Encounters - Regional Development and Practice-Based Learning*. *Conference on Regional Development and Innovation Processes March 5th-7th, Porvoo- Borgå, Finland*, 2008. <http://www.congreszon.fi/@Bin/1680410/3.Innovation.Schulz.pdf>.
- [SD04] Shiyuan, Hu y Li Deren: *Vector Cellular Automata Based Geographical Entity*. En *Geoinformatics 2004. Proc. 12th Int. Conf. on Geoinformatics*. *Geospatial Information Research: Bridging the Pacific and Atlantic University of Gävle, Sweden, 7-9 June*, 2004.
- [SE03] Schneider, Philip J. y David H. Eberly: *Geometric Tools for Computer Graphics*. Elsevier Science (USA), 2003.
- [SEF⁺03] Stafford, Kagan Erdil, Emily Finn, Kevin Keating, Jay Meattle, Sunyoung Park y Deborah Yoon: *Software Maintenance As Part of the Software Life Cycle*. Tufts University, 2003. from Comp180: Software Engineering.
- [SF06] Sosa, Hugo Hernando Andrade y Luis Carlos Gómez Flórez: *Tecnologías informáticas en la escuela*. División de publicaciones UIS, Primera edición, 2006.

- [SF09] Sosa, Hugo Hernando Andrade y Luis Carlos Gómez Flórez: *Tecnologías informáticas en la escuela*. División de publicaciones UIS, Cuarta edición, 2009.
- [Sie99] Sierra, Luis Javier Jaramillo: *Serie: Aprender a investigar - Módulo 1: Ciencia, Tecnología, Sociedad y desarrollo*. ICFES, Tercera edición, 1999.
- [Slo85] Sloan, S. W.: *A point-in-polygon program*. Advanced Engineering Software, 7(1):45–47, 1985.
- [Slo11a] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org>, Home site.
- [Slo11b] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A000004>, The zero sequence.
- [Slo11c] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A000012>, The simplest sequence of positive numbers: the all 1's sequence.
- [Slo11d] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A000312>, Number of labeled mappings from n points to themselves. An endofunction.
- [Slo11e] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A001372>, Number of mappings or mapping patterns from n points to themselves; number of endofunctions.
- [Slo11f] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A005993>, Number of multidigraphs with loops on 2 nodes with n arc.
- [Slo11g] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A007318>, Pascal's triangle read by rows.
- [Slo11h] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A008794>, Square repeated.
- [Slo11i] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A016754>, Odd squares. Also centered octagonal numbers.
- [Slo11j] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A050927>, Number of directed multigraphs with loops on 3 nodes with n arcs.
- [Slo11k] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A050929>, Number of directed multigraphs with loops on 4 nodes with n arcs.
- [Slo11l] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A052171>, Number of directed multigraphs with loops on infinite set of nodes with n arcs.
- [Slo11m] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A061711>, Product of first n terms of an arithmetic progression with first term n and common difference n.

- [Slo11n] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A136564>, Array read by rows: Number of directed multigraphs with loops with n arcs, k vertices and no vertex of degree 0.
- [Slo11o] Sloane, N. J. A.: *The On-Line Encyclopedia of Integer Sequences*, 2011. <http://oeis.org/A144047>, Limiting values of A136564.
- [Soc90] Society, IEEE Computer: *IEEE Std 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology*. IEEE Computer Society, 1990.
- [SP00] Shi, W. y M. Y. Cheung Pang: *Development of Voronoi-based cellular automata: an integrated dynamic model for geographical information systems*. International Journal of Geographical Information Science, 14:455–474, 2000.
- [SPS06] Sarjoughian, Hessam S., Sivakumar Palaniappan y Anil Sawhney: *Application of Devs Framework in Construction Simulation*. En *Proceedings of the 2006 Winter Simulation Conference*, 2006.
- [SS04] Sarjoughian, Hessam S. y Ranjit K. Singh: *Building Simulation Modeling Environments Using Systems Theory and Software Architecture Principles*. En *Advanced Simulation Technology Symposium (ASTC)*, Abril 2004.
- [SS07] Sánchez, Merly Sulgey Gómez y Hugo Hernando Andrade Sosa: *Propuesta informática para la educación soportada en el modelado basado en objetos y reglas*. Tesis de magister en ingeniería, Área en informática y Ciencias de la Computación., Universidad Industrial de Santander, 2007.
- [SSS00] Spiegel, Murray R., John Schiller y R. Alu Srinivasan: *Probabilidad y Estadística*. Schaum Outline. McGraw-Hill, Segunda edición, 2000.
- [ST02] Sklar, David y Adam Trachtenberg: *PHP Cookbook*. O'Reilly Media, Primera edición, Noviembre 2002.
- [Ste80] Stevens, Peter S.: *Handbook of Regular Patterns: An Introduction to Symmetry in Two Dimensions*. MIT Press, Junio 1980.
- [Str91] Stroustrup, Bjarne: *What is .Object-Oriented Programming?* En *Proceedings of 1st European Software Festival*, 1991.
- [STS03] Syropoulos, Apostolos, Antonis Tsolomitis y Nick Sofroniou: *Digital Typography using LaTeX*. Springer, 2003.
- [Tar99] Tarazona, Javier Trujillo: *Generación y consumo de energía bajo demanda, modelado utilizando la herramienta software Homos para el modelamiento y simulación basado en objetos y reglas*. Documento interno, Marzo 1999.
- [Teo96] Teodoro, Vitor Duarte: *Introdução ao modellus*. Faculdade de ciências e Tecnologia, Universidad de Lisboa, Junio 1996.

- [TM87] Toffoli, Tommaso y Norman Margolus: *Cellular Automata Machines: A new environment for modeling*. MIT Press, 1987.
- [Tof84] Toffoli, Tommaso: *Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics*. *Physica*, 10D:117–127, 1984.
- [Tru98] Truemper, Klaus: *Matroid Decomposition*. Leibniz, Plano, Texas, 1998.
- [TVVB02] Tewoldeberhan, Tamrat W., Alexander Verbraeck, Edwin Valentin y Gilles Bardonnnet: *An evaluation and selection methodology for discrete-event simulation software*. En *Proceedings of the 2002 Winter Simulation Conference*, 2002.
- [Ven09] Ventrella, J.: *Earth Day 2009 - A Spherical Cellular Automaton*. <http://www.ventrella.com/EarthDay/EarthDay.html>, 2009.
- [Ven10] Ventrella, Jeffrey: *Glider Dynamics on the Sphere: Exploring Cellular Automata on Geodesic Grids*. *Journal of Cellular Automata*, 6(2-3):245–256, 2010.
- [Voß09] Voß, Herbert: *Mathematiksatz in LATEX*. LOB-media.de, Berlin/Heidelberg, 2009, 2009.
- [Voß10] Voß, Herbert: *Math mode*. The UK TeX Archive, 2010. <http://www.tex.ac.uk/tex-archive/info/math/voss/mathmode/Mathmode.pdf>.
- [Wel88] Welsh, D.: *Matroids and their applications*, 1988.
- [WF90] West, Michael A. y James L. Farr: *Innovation and creativity at work: psychological and organizational strategies*. Jhon Wiley and Sons Inc., 1990.
- [Whi32] Whitney, Hassler: *Non-separable and planar graphs*. *Trans. Amer. Math. Soc.*, 34:339–362, 1932.
- [Wil73] Wilson, R. J.: *An introduction to matroid theory*. *The American Mathematical Monthly*, 80:500–525, 1973.
- [WISB86] Watanabe, Y., M. Ito, T. Soma y T. Betsumiya: *Non-periodic Tessellation with Eight-fold Rotational Symmetry*. En *Science on Form: Proceedings of the First International Symposium for Science on Form*, 1986.
- [You93] Yourdon, Edward: *Análisis y Diseño Estructurado Moderno*. Prentice Hall, 1993.
- [Zam99] Zamir, Saba (editor): *Handbook of Object Technology*. CRC Press, 1999.
- [ZPK00] Zeigler, Bernard P., Herbert Praehofer y Tag Gon Kim: *Theory of Modeling and Simulation*. Academic Press, Segunda edición, 2000.

Apéndice A

Elementos de análisis del software

Homos 1.0

Introducción

Dentro de los objetivos planteados en este proyecto, se proyecta un analisis global a la herramienta software y su metodologia asociada, la MBOR. El analisis se realiza de esta forma para obtener la mayor cantidad de datos posible acerca del objeto de estudio. Se pretende revelar los insumos de cambio o elementos principales, aquellos que al ser modificados causen un impacto trascendente en la herramienta y su metodologia. Esto corresponde con la etapa de generacion de ideas del método G.I.F.T. presentado en la delimitacion conceptual propuesta para esta investigacion en la sección 1.4 en la página 38.

Al final del proceso se obtendrá el soporte teórico y práctico que fundamente los elementos presentados en la propuesta de diseño general a contener en el capítulo 5 en la página 59.

Estructura del cuerpo de análisis

El proceso de analisis a aplicar es particular al objeto de estudio. La Figura A.1 en la página 98 presenta la estructura del cuerpo de analisis a realizar sobre la herramienta software y su metodologia. El orden de aplicacion del analisis va desde lo superficial a lo mas profundo, buscando de esta forma implementar una aproximacion al objeto de estudio en semejanza con la idea planteada en el *diagrama del Iceberg* usado en la dinámica de sistemas[ADE⁺01, p228].

La forma del cuerpo de analisis busca preservar las interrelaciones entre cada una de las unidades propuestas. En el diagrama de la figura A.1 en la página 98 el sentido de las flechas marca el camino hacia el proposito funcional que da sentido a toda la estructura de analisis. La integracion entre unidades pretende permitir la busqueda de una solucion que tenga en cuenta la complementario

de cada apartado y dar de esta forma soporte integral a los objetivos específicos planteados así como establecer soluciones a las inquietudes de investigación planteadas en este proyecto.

El orden de acceso al objeto de estudio es el siguiente: Lo primero que se establece es un análisis general, el cual permite conocer que es lo que se ofrece y como ven otros a la herramienta software. El análisis funcional pretende facilitar la comprensión de la implementación misma del software, los alcances y las limitaciones actuales presentes en la solución establecida o diseño actual del util de modelado.

El estudio de las interfaces permite saber de que se dispone y que posible realizar con la herramienta. Este se realiza en dos grandes bloques: Se hace repaso a los elementos que componen la expresión gráfica del producto, con el propósito de encontrar los elementos que expliquen su forma actual. Posteriormente se hace una revisión de la interfaz de automatización OLE disponible en la aplicación, la idea es revelar las faltas y las capacidades existentes en esta extensión programable del software.

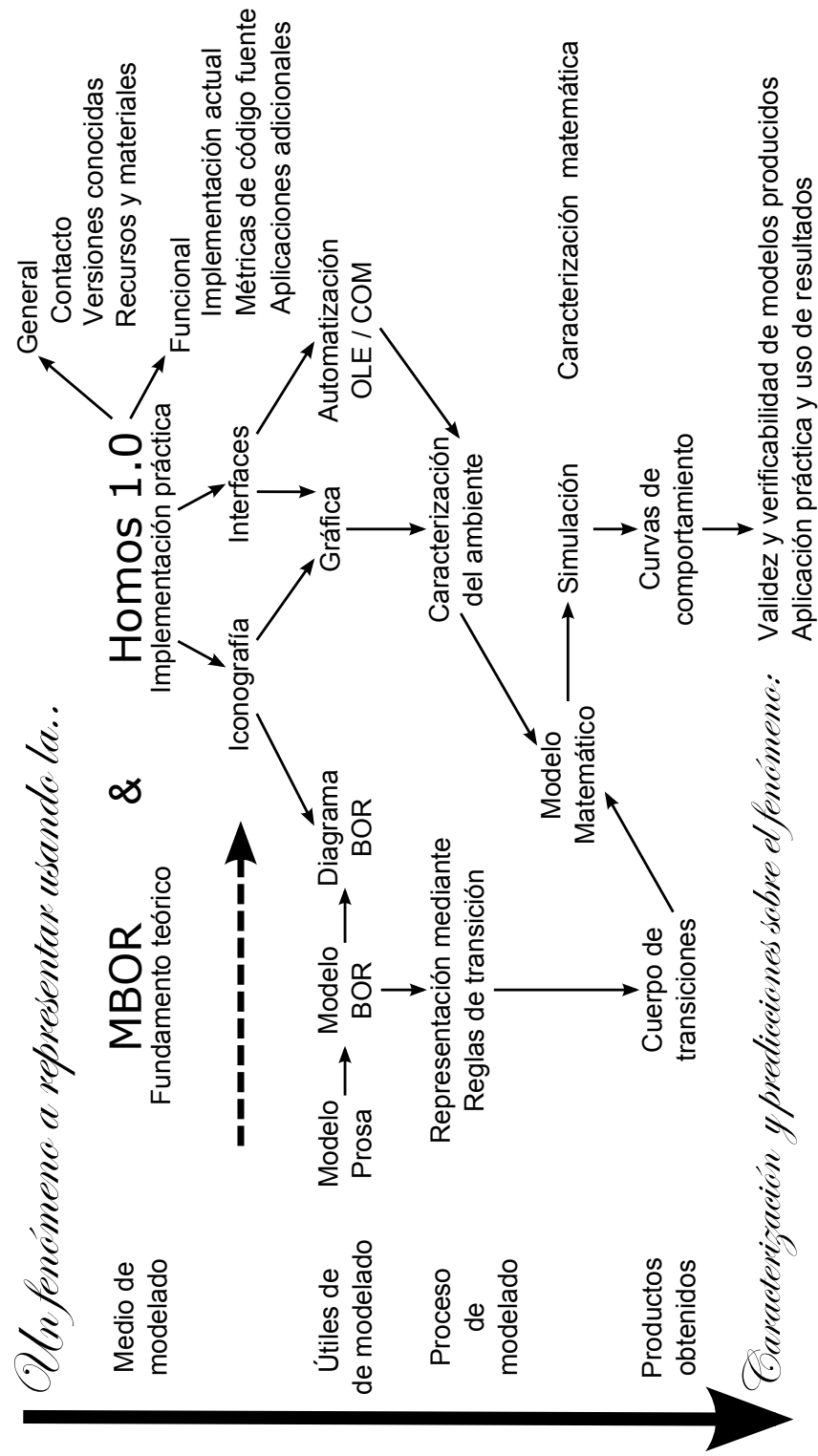
Ya en lo profundo, se expone el núcleo mismo de la herramienta software: la metodología de modelamiento basada en objetos y reglas. Se espera determinar lo que la caracteriza, para así establecer con fundamento los cambios o posibles nuevas incorporaciones a las que haya lugar. El diagrama BOR aparece al final, no porque este al fondo de todo, sino porque se interpreta mejor si se cuenta con los elementos obtenidos del análisis a la metodología.

En la figura A.1 también se hace visible el contexto general de análisis: la transformación o lo que se logra en términos del estudio de un fenómeno cualquiera, desde su existencia como un problema externo pasando por su tránsito dentro del sistema que comprende el método y la implementación que es posible realizar del fenómeno en la herramienta software hasta los distintos resultados o caracterizaciones logradas del fenómeno producidas por la realización del proceso metodológico.

Al final de cada elemento presente en el cuerpo de análisis se incluyen los resultados obtenidos.

Figura A.1: Cuerpo de análisis realizado sobre Mbor y Homos 1.0

Cuerpo de Análisis



Fuente: Autor.

A.1. Contacto con la herramienta software

El análisis realizado en este apartado comprende la revisión de la licencia vigente, los idiomas ofrecidos, la forma de obtención del software, los detalles de des/instalación del software y los modos de funcionamiento disponibles de la herramienta. El objetivo de este análisis es establecer lo que se puede conocer solo con el contacto *superficial* entre los usuarios y la herramienta software. A partir de esta relación pueden determinarse los elementos que se consideren faltantes en la etapa de interacción humana más externa con la actual versión del software.

A.1.1. Licencia

Homos 1.0 es software propietario, se requiere autorización para hacer uso del software en forma comercial o industrial. Informalmente, el Grupo Simon [GS] promueve la herramienta software como gratuita y de libre aplicación solo si se usa para fines académicos. Esta flexibilidad no se refleja en el texto de la licencia que acompaña el software. El texto de la licencia se ofrece en español e inglés según corresponda. La licencia no existe como un documento separado, está incorporada dentro del instalador y se muestra durante el proceso de instalación de la aplicación. Los datos de contacto presentados en la licencia aun están en vigencia.

Texto de la licencia en español [Sic]

Homos 1.0 Herramienta Software para el Modelamiento y Simulación Basado en Objetos y Reglas es propiedad intelectual de la Universidad Industrial de Santander UIS; del Grupo SIMON de Investigaciones adscrito a la Escuela de Ingeniería de Sistemas e Informática de la UIS; del director del proyecto Profesor Hugo Hernando Andrade Sosa y de los desarrolladores del mismo, Carmen Elena Duarte Mogotocoro y Oscar Armando Lozano Avellaneda.

Los derechos de propiedad intelectual y uso industrial y comercial de Homos 1.0, se rigen por las leyes Colombianas y por los reglamentos internos de la Universidad Industrial de Santander.

Carmen Elena Duarte Mogotocoro
Oscar Armando Lozano Avellaneda
Hugo Hernando Andrade Sosa

Grupo SIMON de Investigaciones
Escuela de Ingeniería de Sistemas e Informática
UNIVERSIDAD INDUSTRIAL DE SANTANDER
Carrera 27 Calle 9 Ciudad Universitaria
Bucaramanga (Colombia)
Teléfono 6344000 Ext 2681
Email : simon@uis.edu.co

A.1.2. Idiomas disponibles

El proyecto se desarrolló en español e inglés. Existe un archivo binario distinto para cada versión compilada.

A.1.3. Presentación y adquisición del software

La aplicación puede adquirirse en los idiomas disponibles directamente en las instalaciones del GS¹, incluidas en un Cd recopilatorio titulado *Pensamiento sistémico, modelamiento y simulación* [CdPS] el cual contiene otros títulos de software del grupo², junto con documentos sobre fundamentos teóricos y ejemplos³. En el presente de este documento, solo la versión en español del software⁴ puede descargarse directamente de la página web del grupo en:

http://www.uis.edu.co/portal/investigacion/grupos/Simon_uis/software/sHomos.exe

A.1.4. Detalles de instalación del software

El programa instalador ocupa aproximadamente 3.4 MB y una vez finalizada la instalación usa cerca de 7.7 MB de espacio en disco⁵. Al inicio del proceso de instalación se requiere la introducción de un número serial⁶.

Los requerimientos mínimos necesarios para su correcto funcionamiento son los siguientes⁷:

CPU Pentium de 100 MHz, 16 MB de Memoria RAM, Espacio libre en Disco Duro de 10 MB, Monitor SVGA 800x600, Windows 95, Además requiere el Ratón para manejar la aplicación. Se recomienda el uso de una CPU Pentium de 200 MHz, 32 MB de Memoria RAM e Impresora.⁸

Por defecto el programa se instala en la carpeta C:\program files\UIS\⁹, dentro se crean las carpetas Ayudas, Proyectos¹⁰ y Recursos que contienen el archivo ayuda.hlp, un conjunto de 12 proyectos ejemplo¹¹ listos para usar y un grupo de 21 imágenes prediseñadas de motivos animales para representar clases objeto en los proyectos.¹²

Durante la instalación se crean los accesos directos a la aplicación principal, al Visor de Homos, el archivo de ayudas y las entradas en el registro necesarias para la identificación del software y la asociación de los tipos de archivo *.prh y *.rep con la herramienta software.

¹Universidad Industrial de Santander, Escuela de Ingeniería de Sistemas e Informática, Grupo SIMON de Investigaciones Teléfono:(7)6344000 ext 2681 TeleFax: 6343377 e-mail : simon@uis.edu.co y handrade@uis.edu.co

²Evolucion 20a, Evolucion 2.x - 3.x - 4.x, Michro 1.0 - 2.0 y Simuis 1.0

³http://www.uis.edu.co/portal/investigacion/grupos/simon_uis/publicaciones/cdedu.html

⁴Versión 1.0.0.23

⁵Los elementos y valores indicados en esta parte del documento corresponden a los realizados por la instalación de la versión web de la herramienta software.

⁶1998-1142-1127-9191 o 1998-1142-1127-0000 para las versiones sin interfaz OLE en español e inglés respectivamente. Posteriores proyectos software del GS establecen elementos de seguridad con palabra clave solo para los contenidos de los proyectos, no para la instalación de las aplicaciones. Ver la tabla A.2 en la página 103.

⁷Información de requisitos mínimos de instalación consignada en el texto de la ayuda de Homos.

⁸Como se verá mas adelante en este documento, se requieren prestaciones mucho mayores de hardware que las establecidas en la tesis de Homos 1.0 para hacer uso práctico del software.

⁹Los programas más recientes del grupo se instalan en C:\Archivos de programa\SIMON

¹⁰Según los autores de Homos 1.0, "El modelo se convierte en proyecto al ser introducido en la herramienta de modelamiento. Por tanto se considera a proyecto y modelo como sinónimos". Ver "¿qué es un proyecto Homos?" en la ayuda de Homos 1.0.

¹¹Que incluyen: El modelo predador-presa: lobos y conejos, el crecimiento de una población de ranas, la propagación del fuego, el reloj de arena, modelo de una infección versiones 1 y 2, un juego de billar, un modelo de colas simple, la maratón, un modelo de producción de crispeta, el ciclo de agua y un proyecto vacío.

¹²Ver la figura A.12 en la página 169.

A.1.4.1. Versiones disponibles

Existen actualmente en 5 versiones del software Homos 1.0: Demo, en español e inglés sin interfaz OLE, en español con esta interfaz implementada y una versión de mantenimiento realizada durante el desarrollo de este proyecto. Todas las versiones listadas excepto la última coexisten en el CdPS del GS. Solo se distribuye públicamente en el sitio web del GS la versión en español sin interfaz OLE.

Tabla A.1: Versiones del software Homos 1.0

Propósito de uso	Versión	Idioma	¿OLE?	¿Setup?	Medio	Ubicación del software en el CdPS del Grupo Simon. ❶
Demo	1.0.0.23	Español	No	Si	CD	..\InsHomos\demo
Profesional	1.0.0.23	Español	No	Si	CD y Web	..\InsHomos\Español
Profesional	1.0.0.23	Inglés	No	Si	CD	..\InsHomos\Ingles
<i>Completa</i>	1.0.0.50	Español	Si	No	CD	..\Homos\bin\ejecutables
<i>Completa revisada</i>	❷ 1.0.0.73	Español	Si	No	CD	..\Anexos\Homos1.0

❶ La raíz de cada directorio listado es \Cd Homos.

❷ Aporte realizado por esta tesis debido a operaciones de mantenimiento de la Interfaz OLE. Se indican los cambios realizados en las sección E.3.1 en la página 585. Los seriales disponibles de cada versión se presentan en la tabla A.2 en la página 103.

Fuente: Autor.

A.1.4.2. Instalación de Homos 1.0 en versión completa

Para hacer uso de la versión completa de Homos 1.0 realice los siguientes pasos:

1. Verificar que se tenga instalado Homos 1.0. Instalar la versión del software disponible en la Web, la 1.0.0.23. con ello se garantiza la presencia de las últimas ediciones de documentación y ejemplos predefinidos para la herramienta.
2. Reemplazar el ejecutable principal instalado con el archivo `homos.exe`:
 - a) Contenido en la versión 1.0.0.50 o completa presente en el CdPS. Cambiar el valor del número serial editando manualmente el registro de windows con `regedit.exe` mediante la incorporación de la siguiente entrada al registro:

Serial en registro de Windows

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\UIS]
[HKEY_LOCAL_MACHINE\SOFTWARE\UIS\Homos]
[HKEY_LOCAL_MACHINE\SOFTWARE\UIS\Homos\1.0.0.50]
"Name"="Windows Angelical"
"Company"="Heaven"
"Serial"="1998-0212-0430-7474"
```

- b) Con la versión 1.0.0.73 incluida en CD de distribución de este proyecto. Este ejecutable no requiere activación o número serial.
3. Registrar la automatización OLE de la aplicación. Se realiza en la carpeta de instalación mediante esta línea de comando:

```
homos.exe /regserver
```

Permanecen en el misterio las razones por las cuales no se distribuyó públicamente la versión completa del software desde el momento de su realización.

A.1.4.3. Desinstalación del software

El software no cuenta con una utilidad propia de desinstalación, es necesario recurrir a la opción *Agregar o quitar programas* del panel de control de Windows. También es posible remover la aplicación mediante el borrado manual de todos los archivos y carpetas creados durante la instalación, junto con la eliminación de las siguientes entradas en el registro:

```
HKLM\Software\UIS\Homos, HKCR\ .rep y HKCR\ .prh13
```

Adicionalmente si se ha instalado la versión completa del software, será necesario desactivar la automatización OLE con:

```
homos.exe /unregserver
```

A.1.5. Modos de funcionamiento de la herramienta

A.1.5.1. Versión demo y profesional

Estos modos de funcionamiento de la aplicación fueron implementados con el propósito diferenciar presentaciones del producto, pensando en un emprendimiento comercial con la herramienta software¹⁴. Con el paso del tiempo dicha iniciativa no prosperó, dando paso a la actual herramienta gratuita y de libre uso para fines académicos. Es necesario contactar al GS para obtener una licencia especial en caso de usar la herramienta con otros propósitos, distintos a los previstos en la licencia¹⁵.

Versión Demo: Cuando el programa se ejecuta en modo demo, la herramienta no permite:

- Guardar proyectos.

¹³Esta información corresponde con los elementos creados por una instalación estándar del software. HKLM, HKCR son acrónimos de HKey Local Machine y HKey Clases Root, secciones componentes del registro de Windows.

¹⁴Que corresponde con la creación de la aplicación interna Seriales.

¹⁵"Los derechos de propiedad intelectual y uso industrial y comercial de Homos 1.0, se rigen por las leyes Colombianas y por los reglamentos internos de la Universidad Industrial de Santander". Ver Licencia de Homos 1.0 en la página 99.

- Usar el repositorio: almacenar, recuperar clases y reglas.
- Guardar las series de datos presentadas en las tablas de datos.

Al intentar una de estas acciones, se obtiene este mensaje: *“Esta versión de Demostración no permite ejecutar esta acción. Adquiera la versión original enviando un e-mail a simon@uis.edu.co”*. Para saber en que modo se encuentra funcionando la aplicación basta con observar la ventana *Acerca de...* Allí aparece explícitamente enunciado el estado de la versión del software, si es demo o profesional.

Versión Profesional: Cuando la herramienta se encuentra en modo profesional, las funciones incorporadas descritas en los manuales y la tesis de Homos están disponibles. Si se ha instalado la versión completa de la herramienta, entonces se dispone también de la interfaz OLE.

A.1.5.2. Seriales de instalación

Se listan a continuación los números seriales conocidos para la instalación de las distintas versiones existentes del software. El valor establecido durante la instalación determina si la herramienta se ejecuta en modo demo o profesional. Los seriales varían de acuerdo con la versión en la que fueron implementados.

Tabla A.2: Seriales conocidos de Homos 1.0

Versión	Descripción	Serial Demostración	Serial Profesional	Fecha
Demo	Demo en español	1998-9111-4291-1127	-	12/08/98
Profesional	Versión en Inglés	-	1998-1142-1127-0000	10/08/98
	Versión en Español disponible en la web	1998-9111-4291-1127*	1998-1142-1127-9191	25/08/98
Completa	Ultima compilación en CdPS	1998-9111-4291-1127*	1998-0212-0430-7474	17/10/98

* Implementado pero no funcional, desactivado internamente dentro del software

Fuente: Autor.

Aunque la funcionalidad demo se encuentra implementada internamente en todas las versiones conocidas del software¹⁶, la activación de este modo no se encuentra operativa en las versiones profesionales y completa. Existe en el CdPS un instalador del software que requiere de ese valor, se encuentra en ... \Cd Homos\InsHomos\demo.

Los valores de los números seriales de cada uno de los modos presentados se obtuvieron mediante el uso de un *desensamblador*, el PE explorer - Disassembler versión 1.99 R3 de HEAVENTOOOLS, en su versión de evaluación por 30 días.

¹⁶Exceptuando la 1.0.0.73

A.1.5.3. Estructura y manejo de números seriales

Los cuatro primeros dígitos de la secuencia de cada serial representan el año de realización del producto. Los demás valores numéricos son de origen arbitrario.

La aplicación *seriales* incluida en el CdPS solo realizaba labores de registro, no generaba secuencias numéricas que pudieran ser usadas como seriales de instalación. El servicio de instalación previsto en la aplicación no hace validación alguna de estos valores, por lo tanto es posible instalar la aplicación con una entrada texto cualquiera como clave; al ejecutar el software se presenta el mensaje de error “*El producto no es legal*” y se detiene todo intento de inicio de la aplicación. Luego de la instalación y para solucionar este problema, se requiere editar manualmente en el registro de windows el número del serial de instalación usando los valores de claves conocidas o realizar nuevamente el proceso de instalación con el serial adecuado según corresponda.

La validación de registro que realiza el software se resume a una comparación de valores entre un serial fijo internamente codificado y el número almacenado en el registro de Windows.

A.1.6. Resultados obtenidos

A partir de los elementos detectados se establece que:

- Se requiere actualizar el texto de la licencia, en cuanto a las limitaciones u oportunidades de los usuarios con la herramienta desde el punto de vista legal.
- En terminos de estandares actuales, el proceso de instalacion es realmente insuficiente. Se requiere de un instalador que cierre el circulo de vida de la aplicacion.
- El desorden de versiones existentes corresponde con un proceso de desarrollo no convencional del software. Se recomienda hacer disponibles las ultimas compilaciones del software, junto con material de ayuda que explique los cambios realizados.
- No se descarta el uso comercial de la herramienta, pero se desaconseja el enfoque mediante licencias y numeros seriales. Se propone a cambio la apertura del codigo fuente y el uso de licencias de tipo creative commons¹⁷ y la generacion de ingresos causados por capacitacion y cursos, creacion y realizacion de productos y servicios alrededor del proceso de modelado.

¹⁷<http://creativecommons.org/>

A.2. Recursos y materiales disponibles

Este apartado cubre los distintos insumos, productos y subproductos generados por la herramienta software Homos 1.0. Por ser recursos asociados al software o su metodología, se adjunta un listado de publicaciones directamente relacionadas con la MBOR o la herramienta software.

A.2.1. Archivos utilizados por la aplicación

A.2.1.1. Archivos primarios

Archivos de proyecto Homos *.prh: Son el tipo de archivos principales de la herramienta, contienen el soporte digital de los modelos creados con ella. Se ubican inicialmente en la carpeta `Proyectos`¹⁸, que es el directorio de trabajo por defecto.

Características:

- La aplicación no recurre a esquemas de compresión como parte del proceso de creación de archivos.
- Contiene un registro que es usado con fines de descripción del modelo. En este espacio también se almacena la contraseña para el proyecto.
- Usa una estructura dinámica para el almacenamiento de los elementos que conforman el modelo, los objetos están listados en el orden con el cual fueron creados; en este mismo orden son guardados y recuperados.
- En esta versión se contemplan la existencia de múltiples clases objeto, reglas y escenarios, estos últimos incluyen una tabla de datos y varias vistas por cada escenario.
- Existe un único ambiente por proyecto que almacena la ubicación y el tipo de clase objeto de los objetos instanciados en el modelo.
- Para las acciones de E/S se usan las funciones de archivos estándar de Delphi, `TFileStream`, `TWriter` y `TReader` para el manejo de archivos. El contenido de los documentos generados por la aplicación puede ser manipulado con versiones posteriores de Delphi sin mayores contratiempos.
- La estructura dinámica de almacenamiento refleja las posibilidades ofrecidas por la herramienta y su flexibilidad.

El archivo `repositorio.rep`: Ubicado en el directorio raíz de la aplicación, es el depósito digital de los objetos y reglas enviadas al repositorio. Se crea un único archivo `repositorio.rep` por cada instalación de Homos y es administrado por la herramienta. En general esta funcionalidad no ha sido diseñada para que existan simultáneamente múltiples instancias de este archivo durante la ejecución de la aplicación.

¹⁸... \UIS\Homos\Proyectos

A.2.1.2. Archivos secundarios

Archivos de imágenes: Mediante la opción *Exportar/General/Gráfica* de las gráficas de comparación es posible guardar las imágenes de las curvas producidas en formatos *. bmp, *. wmf y *. emf. Es una facilidad heredada de las funciones incorporadas en el objeto Teechart usado en la herramienta.

Archivos de texto *.txt: Creables desde los diálogos de escenarios, tablas y gráficas, permiten guardar las series de datos producidas por el simulador, su propósito es servir de fuente de datos para la creación de gráficas de comparación, un útil para analizar datos y obtener conclusiones.

A.2.1.3. Archivos de recursos

Imágenes de identificación de clases objeto: Son archivos gráficos en formato *. bmp de 16×16 hasta 32×32 píxeles con paleta de 256 colores o más, ubicados en la carpeta recursos. El conjunto de figuras incluida en la herramienta es ampliable por el usuario con la adición de nuevas imágenes usando el mismo formato.

Cuando son utilizadas en los proyectos, los archivos gráficos seleccionados se incorporan al archivo *. prh respectivo.

Otros archivos de recursos: Los archivos Homos.avi y Homos ani, un vídeo y un cursor animado, son usados por el programa principal con fines promocionales y estéticos.

A.2.2. Documentación particular de la aplicación

Entendiendo documentación como *“Cualquier información escrita o gráfica que describe, define, especifica, reporta o certifica actividades, requerimientos, procedimientos o resultados”*[Soc90], el proyecto cuenta con los siguientes elementos:

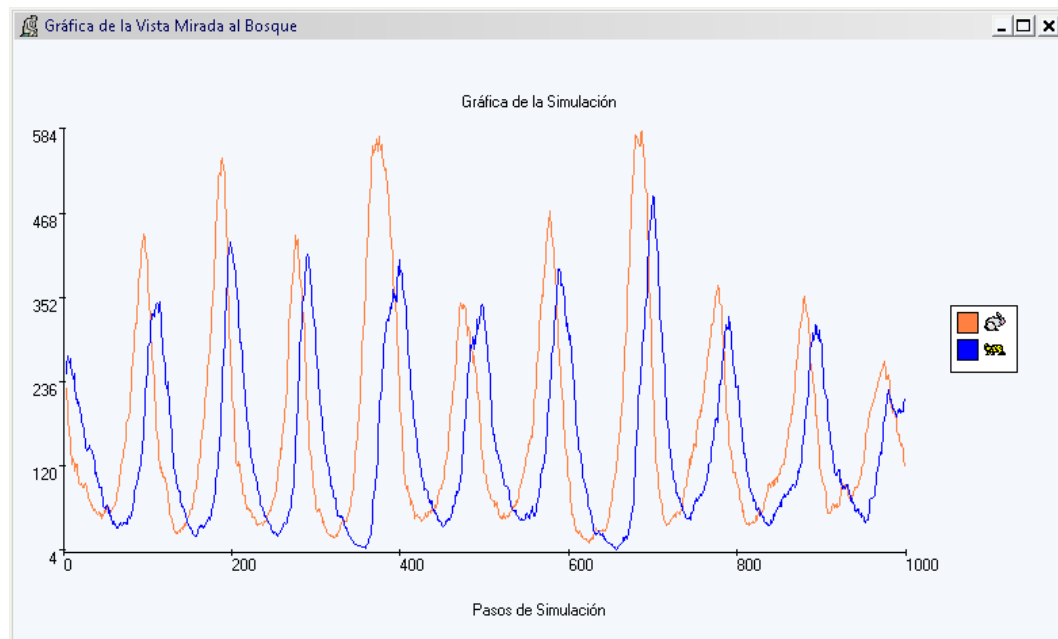
A.2.2.1. Material Impreso

Tesis de Homos: Fuente original de este trabajo, consulta obligada para aquellos que quieran saber más sobre Homos y su metodología. Se encuentra en la biblioteca de la UIS.

Manual de usuario: En el se describen en profundidad los detalles y el uso de la interfaz gráfica de Homos, acompañados de un apéndice sobre la MBOR y un ejemplo de aplicación, el modelo predador – presa *“Lobos y conejos”*. Consta de 86 páginas y hace parte de la tesis de grado de Homos 1.0. Se desconoce si se han realizado reimpresiones separadas de este documento por parte del Grupo Simon.

Diagrama de clases: Describe la estructura de clases que componen la aplicación. Anexo presente en la tesis de homos.

Figura A.2: Visualización del comportamiento del modelo mediante curvas poblacionales



A.2.2.2. Resultados Gráficos

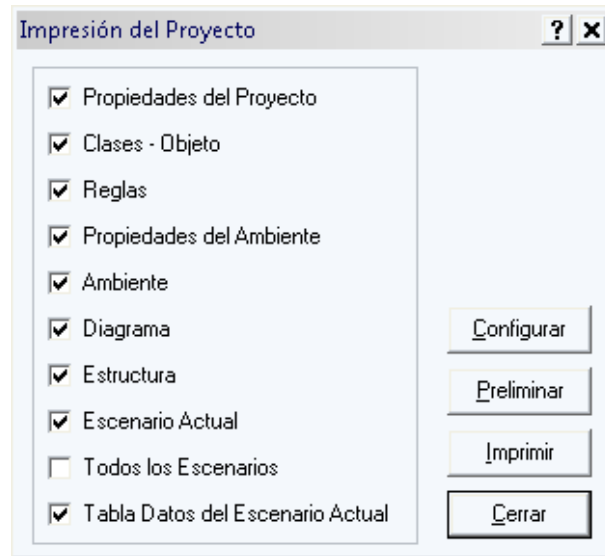
La aplicación presenta los diagramas BOR, el estado dinámico de objetos en el ambiente durante la simulación, las vistas y las gráficas de comparación presentes en los escenarios. Ver la figura A.2. No se incluye en este grupo los resultados de impresión por que no implican un cambio en la forma de ver los datos.

A.2.2.3. Resultados Impresos

Las opciones de impresión comprenden la presentación en papel de las propiedades del proyecto. Esta conformada por el listado de clases objeto y reglas presentes en el modelo, el listado de propiedades del ambiente, las imágenes del ambiente, el diagrama BOR y el árbol de proyecto del modelo junto con las propiedades establecidas por el escenario actual. Adicionalmente pueden listarse las propiedades de todos los demás escenarios creados y la presentación de la tabla de datos del escenario actual. Los elementos aparecen plasmados en el papel en el orden en el que aparecen en el listado, cada elemento puede ser seleccionado individualmente e impreso por separado.

En la figuras A.4 en la página 110 y A.5 en la página 111 se presenta un ejemplo de resultados de ejecución del proyecto Predador-Presa de A.S. Camara.prh según las opciones de impresión mostradas en la figura A.3. El proyecto al que se hace referencia hace parte por defecto de la instalación estándar de la herramienta software.

Figura A.3: Selección de elementos del modelo y opciones de impresión disponibles



Fuente: Autor.

A.2.2.4. Archivo de grabación de resultados de simulación

Es un archivo de texto formateado generado en los cuadros de diálogo de escenario y usado en las gráficas de comparación como fuente de datos.

Formato del archivo de grabación: En la primera línea del archivo aparece la palabra *Paso* seguida del nombre de las clases objeto que sirven de columnas en la tabla de datos. Las filas restantes están compuestas por el identificador de paso de simulación y los valores poblacionales de cada clase objeto. Los datos en las filas están separados entre si mediante comas, los registros están delimitados por retornos de carro.

A.2.2.5. Ayudas en línea

`Homos.hlp`: Es la versión digital del manual de usuario presente en la tesis de Homos. Está incluida en la carpeta ayudas de la herramienta. Ocupa cerca de 2.5 MB de disco.

A.2.3. Entradas en el registro de Windows

Almacenamiento del número serial usado en la versión 1.0.0.50

```
[HKEY_LOCAL_MACHINE\SOFTWARE\UIS]
[HKEY_LOCAL_MACHINE\SOFTWARE\UIS\Homos]
[HKEY_LOCAL_MACHINE\SOFTWARE\UIS\Homos\1.0.0.50]
"Name"="Windows XP" "Company"="Microsoft" "Serial"="1998-0212-0430-7474"
```

Asociación de archivos con la aplicación

```
[HKEY_CLASSES_ROOT\.prh]
@="ProyectoHomos"
[HKEY_CLASSES_ROOT\.prh\ShellNew]
"FileName"="ProyectoHomos.prh"
[HKEY_CLASSES_ROOT\.rep]
@="RepositorioHomos"
```

Tipo de librería

```
[HKEY_CLASSES_ROOT\TypeLib\{B9EF32C3-39C4-11D2-AA12-444553540000}]
[HKEY_CLASSES_ROOT\TypeLib\{B9EF32C3-39C4-11D2-AA12-444553540000}\1.0]
@="Librería Homos"
[HKEY_CLASSES_ROOT\TypeLib\{B9EF32C3-39C4-11D2-AA12-444553540000}\1.0\0]
[HKEY_CLASSES_ROOT\TypeLib\{B9EF32C3-39C4-11D2-AA12-444553540000}\1.0\0\win32]
@="C:\\Program Files\\UIS\\Homos\\Homos.exe"
[HKEY_CLASSES_ROOT\TypeLib\{B9EF32C3-39C4-11D2-AA12-444553540000}\1.0\FLAGS]
@="0"
[HKEY_CLASSES_ROOT\TypeLib\{B9EF32C3-39C4-11D2-AA12-444553540000}\1.0\HELPDIR]
@="C:\\Program Files\\UIS\\Homos\\Homos.exe"
```

Registro de la interfaz

```
[HKEY_CLASSES_ROOT\Interface\{B9EF32C4-39C4-11D2-AA12-444553540000}]
@="ISistema"
[HKEY_CLASSES_ROOT\Interface\{B9EF32C4-39C4-11D2-AA12-444553540000}\ProxyStubClsid]
@="{00020424-0000-0000-C000-000000000046}"
[HKEY_CLASSES_ROOT\Interface\{B9EF32C4-39C4-11D2-AA12-444553540000}\ProxyStubClsid32]
@="{00020424-0000-0000-C000-000000000046}"
[HKEY_CLASSES_ROOT\Interface\{B9EF32C4-39C4-11D2-AA12-444553540000}\TypeLib]
@="{B9EF32C3-39C4-11D2-AA12-444553540000}" "Version"="1.0"
```

Identificadores de Clase

```
[HKEY_CLASSES_ROOT\CLSID\{B9EF32C5-39C4-11D2-AA12-444553540000}]
@="Sistema de Homos"
[HKEY_CLASSES_ROOT\CLSID\{B9EF32C5-39C4-11D2-AA12-444553540000}\LocalServer32]
@="C:\\Program Files\\UIS\\Homos\\Homos.exe"
[HKEY_CLASSES_ROOT\CLSID\{B9EF32C5-39C4-11D2-AA12-444553540000}\ProgID]
@="Homos.Sistema"
[HKEY_CLASSES_ROOT\CLSID\{B9EF32C5-39C4-11D2-AA12-444553540000}\TypeLib]
@="{B9EF32C3-39C4-11D2-AA12-444553540000}"
[HKEY_CLASSES_ROOT\CLSID\{B9EF32C5-39C4-11D2-AA12-444553540000}\Version]
@="1.0"

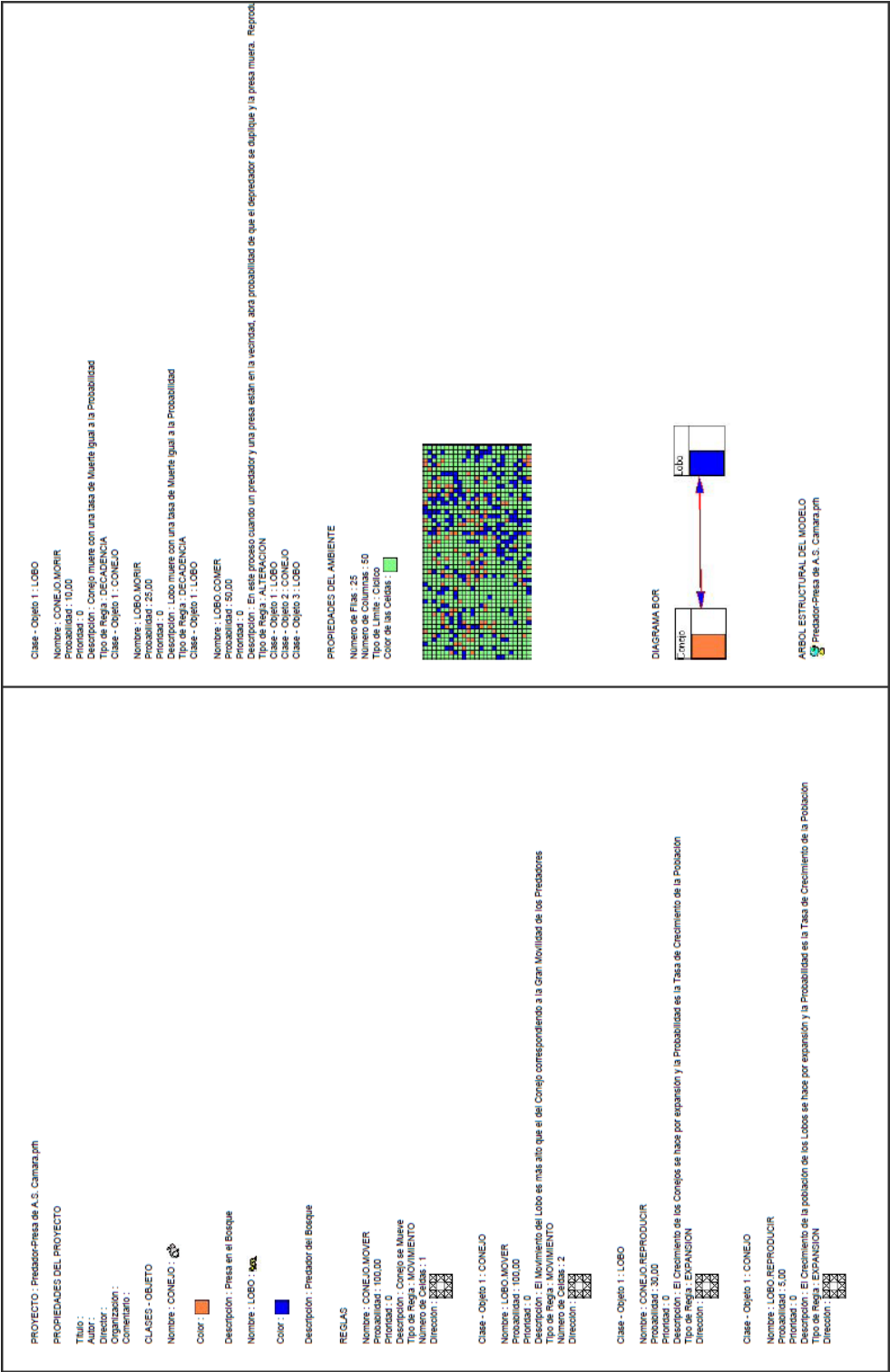
[HKEY_CLASSES_ROOT\Homos.Sistema]
@="Sistema de Homos"
[HKEY_CLASSES_ROOT\Homos.Sistema\Clsid]
@="{B9EF32C5-39C4-11D2-AA12-444553540000}"
```

No se listaron otras entradas pertenecientes a la sección HKEY_LOCAL_MACHINE que están asociadas a la interfaz OLE porque son copias de las entradas presentes en HKEY_CLASSES_ROOT¹⁹. En la versión en inglés de la herramienta, la entrada de registro principal se almacena en:

[HKEY_LOCAL_MACHINE\SOFTWARE\UIS\Homos\1.0.0.51]

¹⁹Datos obtenidos usando el Editor de registro de Windows XP, regedit.exe Versión 5.1

Figura A.4: Resultados reportados y opciones de impresión (a)



La imagen hace parte del archivo de recurso... \anexos\Homos 1.0\Resultados_impresos_en_Homos_1.0. pdf.
Fuente: Autor.

Figura A.5: Resultados reportados y opciones de impresión (b)

Ubicar 10 objetos de la Clase - Objeto : LOBO en forma Aleatoria	
VISTAS DEL ESCENARIO	
Vista : Mirada al Bosque	
Trajectory # 1 = Conejo	
Trajectory # 2 = Lobo	
TABLA DE DATOS DEL ESCENARIO : Bosque	
Paso	Conejo Lobo
1	10
2	113
3	122
4	139
5	141
6	156
7	169
8	184
9	210
10	233
11	243
12	261
13	309
14	336
15	359
16	378
17	418
18	458
19	470
20	485
21	509
22	554
23	593
24	611
25	619
26	631
27	630
28	653
29	630
30	655
31	621
32	611
33	598
34	593
35	585
36	539
37	484
38	472
39	418
40	455
41	337
42	302
43	266
44	243
45	203
46	166
47	150
48	126
49	116
50	97
51	81
52	83
53	67
54	57
55	46
56	51
57	290
58	49
59	42
60	36
61	38
62	31
63	31

La imagen hace parte del archivo de recurso ... \anexos\Homos 1.0\Resultados_impresos_en_Homos_1.0.pdf.
Fuente: Autor.

A.2.4. Documentos y referencias externas conocidas

Los documentos presentes en este listado están relacionados con el desarrollo histórico de la MBOR y de la herramienta software Homos 1.0. En la tabla A.3 se indica el aporte principal de cada uno.

A.2.5. Resultados obtenidos

Los distintos elementos presentados se ajustan a las descripciones dadas en la tesis y manual de Homos 1.0. Las opciones de impresion de Homos 1.0 son algo para rescatar en futuras versiones de la herramienta software.

En cuanto a la documentacion, para mas de 10 años de existencia se observa poca actividad en cuanto a referencias a la metodologia y la herramienta software, tal y como se estableció previamente en el planteamiento de problema de este proyecto. Hasta ahora, el progreso de la herramienta software se ha dado solo mediante actividades principalmente generadas por trabajos de grado. El grupo Simon necesita incrementar la cantidad de trabajos de grado relacionados o establecer otro medio para la continuidad del trabajo.

Tabla A.3: Documentos y referencias conocidos de Homos 1.0

Título	Fecha	Autor	Origen	Resumen
Homos 1.0: Herramienta software para el modelamiento y simulación basado en objetos y reglas.	Agosto 1998	Carmen Elena Duarte Mogotocoro Oscar Armando Lozano Avellaneda	Trabajo de grado	Documento origen de la herramienta y su metodología. Trabajo de grado y manual de usuario.
Homos 1.0: Herramienta software para el modelamiento y simulación basado en objetos y reglas.	1998	Carmen Elena Duarte Mogotocoro Oscar Armando Lozano Avellaneda Hugo Hernando Andrade Sosa	Artículo presentado en IV congreso RIBIE, Brassilia.	Descripción de Homos 1.0, fundamentos teóricos, reseña sobre la ausencia del espacio en otros sistemas de representación dinámicos.
Paisajes Dinámicos con Homos 1.0.	Agosto 2000	Carmen Elena Duarte Mogotocoro Oscar Armando Lozano Avellaneda Hugo Hernando Andrade Sosa	Artículo en Revista de enseñanza y tecnología de Mayo-Agosto 2000. Pág 5-15.	Descripción de la MBOR, Desarrollo de un modelo de epidemia en la herramienta. Contiene una breve reseña sobre diferencias en estructuras de modelamiento con respecto a Evolución 2.0a.
Micrho, Micromundo para el desarrollo del pensamiento sistémico soportado en el Modelamiento basado en Objetos y Reglas.	2000	Amaury Cabarcas Álvares Néstor M. Díaz Mariño	Trabajo de Grado	Micromundo para el modelamiento basado en objetos y reglas. Hace uso directo del código fuente de homos dentro de la aplicación.
Micrho, Micromundo para el desarrollo del pensamiento sistémico soportado en el Modelamiento basado en Objetos y Reglas.	2000	Amaury Cabarcas Álvares Néstor M. Díaz Mariño Hugo Hernando Andrade	Artículo presentado en el Coloquio la Informática y los Jóvenes. La Habana, Cuba.	Soporte sobre las razones por las cuales ofrecer micromundos soportados por MBOR. Una solución al problema ¿Como hacer llegar a homos a las masas?.
Propuesta informática para la educación soportada en el modelado basado en objetos y reglas.	Nov. 2007	Merly Sulgey Gómez Sánchez Hugo Hernando Andrade Sosa	Tesis de magister en ingeniería, Área en informática y Ciencias de la Computación.	Propuesta para hacer uso aplicado de la MBOR en el medio educativo. Contiene resultados de distintas evaluaciones sobre la aplicación de la MBOR y Micrho 1.0 en el aula de clase.
Micrho 2.0, Micromundo para el Desarrollo del Pensamiento Sistémico Soportado en el Modelamiento Basado en Objetos y Reglas.	Nov. 2008	John Fredy Ortiz Ramos Diana Carolina Suarez Celis	Trabajo de Grado	Segunda versión del micromundo para el modelamiento basado en objetos y reglas. Aporta consideraciones de diseño para la realización practica de la micromundización de modelos de Homos 1.0.
Prototipo Homos 2.0: Herramienta software para el modelado y simulación basado en objetos y reglas.	Dic. 2011	Angel Maria Valdes Peña	Trabajo de Grado	El presente documento. Compendio de estudios sobre la herramienta y su metodología asociada. Revisión a la MBOR, Propuesta de una nueva versión del software.

A.3. Análisis funcional

En este apartado de analisis se contemplan las siguientes acciones sobre el software:

- Se realiza un reporte de los resultados obtenidos de una bateria de pruebas de software,
- Gracias a tener acceso al codigo fuente de Homos 1.0, se realiza un reporte de metricas de codigo fuente en la aplicacion y una revision a la estructura interna de la aplicación.
- Se realiza un listado de aplicaciones adicionales asociadas a la herramienta software, se establecen algunos elementos de falla causados por el compilador de la aplicacion y se presenta un listado de herramientas usadas en el proceso de analisis del software.

Al final se establecen sintetizan los resultados obtenidos del proceso de analisis funcional realizado.

A.3.1. Pruebas de software

Son el conjunto de procedimientos por los cuales se comprueba y evalúa la calidad de un software. Generalmente hacen parte del ciclo de desarrollo ya que permiten conocer el estado de la aplicación, corroborar que cierto aspecto del software se encuentre dentro de los límites de trabajo o prestaciones deseado, encontrar las fallas y errores momentáneos que posteriormente serán corregidos durante las labores del ciclo de mantenimiento.

De acuerdo con la naturaleza del software a probar y el momento de desarrollo en el que se encuentre el software, se aplicarán uno o más tipos de pruebas simultáneamente. Según su propósito existen varios tipos, entre ellos:

Pruebas de sanidad o de humo ★: Típicamente son un intento de pruebas iniciales para determinar si una nueva versión del software esta ejecutándose suficientemente bien como para realizar sobre ella otras pruebas de mayor escala.

Pruebas exploratorias ★: Pruebas informales de software que no se basan en planes de prueba o casos de prueba, los probadores aprenden del software durante el proceso.

Prueba de unidades ★: Es la prueba de menor escala, estudia funciones o módulos de código particulares. Es típicamente realizada por el programador y no por los probadores o *Testers*, ya que requiere un detallado conocimiento del código y diseño interno del programa. No siempre es fácil de realizar a menos que la aplicación tenga una arquitectura bien diseñada y código estructurado, podría requerir el desarrollo de módulos particulares de prueba o de aprovechamiento.

Pruebas de integración ★: Pruebas que combinan partes de la aplicación para determinar si ellas funcionan correctamente conjuntamente. Las partes pueden ser módulos, aplicaciones individuales, cliente/servidor, en una red... etc. Este tipo de pruebas es especialmente relevante para sistemas cliente/servidor y sistemas distribuidos.

Pruebas de integración incremental: Pruebas continuas de una aplicación cuando una nueva funcionalidad ha sido adicionada, requiere que los bloques que componen la aplicación y sus funciones sean suficientemente independientes para trabajar separadamente antes de que todas las partes de la aplicación estén completas o que los módulos de pruebas estén explícitamente desarrollados. Son realizadas por programadores o por probadores.

Pruebas funcionales ★: Son pruebas de tipo caja negra adaptadas especialmente a los requerimientos funcionales de una aplicación. Este tipo de pruebas debe ser realizado solo por probadores; supone que los programadores revisan que su código trabaja bien antes de liberarlo.

Pruebas de instalación y desinstalación ★: Prueba completa, parcial o actualización del proceso de instalación/desinstalación.

Pruebas de sistema ★: Prueba de caja negra que esta basada en la totalidad de los requerimientos y especificaciones. Cubre todas las partes combinadas de un sistema.

Pruebas de fin a fin: Similares a las pruebas de sistema. Corresponden a la escala *macro* de las pruebas. Involucra pruebas en un ambiente completo de aplicación de una situación que imite el uso en el mundo real, como interactuar con una base de datos, usar la red de comunicaciones o interactuar con otros hardware, aplicaciones o sistemas según sea el caso.

Pruebas de carga ★: Prueba una aplicación durante situaciones de carga pesada. Permite determinar en que momento el sistema degrada o falla su tiempo de respuesta.

Pruebas de estrés ★: Término usado conjuntamente con pruebas de carga y ejecución. Describe pruebas en las cuales el sistema es sometido a cargas inusualmente pesadas, alta cantidad de repeticiones, acciones o entradas, valores numéricos extremos, etc.

Pruebas de ejecución ★: Esta relacionada con las pruebas de estrés y de carga. Idealmente la prueba de ejecución y similares está definida en la documentación de requerimientos, la aceptación de calidad ²⁰ y los planes de prueba del software.

Pruebas de recuperación ★: Prueba cuan bien se recupera un sistema ante caídas, fallas de hardware u otros problemas catastróficos.

Pruebas de usabilidad ★: Prueba la amigabilidad del sistema frente al usuario. Es claramente subjetiva y depende de los usuarios a los que este destinada. Para hacerlas más objetivas se emplean entrevistas de usuario, encuestas, grabaciones de video, sesiones de usuario y otras técnicas. Programadores y probadores no son usualmente apropiados como probadores de usabilidad.

Pruebas de aceptación: Pruebas finales basadas en las especificaciones dadas por el usuario final o el comprador, generalmente se realizan durante un periodo de tiempo limitado.

²⁰QA, Quality Assurance

Pruebas de aceptación de usuario: Determinan si el software es satisfactorio para un usuario final o un comprador.

Pruebas dirigidas según el contexto ★: Son pruebas orientadas por la comprensión del ambiente, la cultura y el uso proyectado del software.

Pruebas de regresión: Son pruebas que se realizan después de correcciones o modificaciones del software o su ambiente. Puede ser difícil determinar cuántas de estas son necesarias, especialmente cerca del fin del ciclo de desarrollo. Las herramientas de pruebas automatizadas pueden ser especialmente útiles para este tipo de pruebas.

Pruebas de compatibilidad ★: Prueban cuan bien se ejecuta el software en un combinación particular de hardware, software, sistema operativo, red, etc.

Pruebas alfa: Pruebas de una aplicación cuando el desarrollo esta cerca de su terminación. Cambios menores en el diseño podrían presentarse como resultado de dichas pruebas. Es típicamente realizada por usuarios finales, pero no por programadores o probadores.

Pruebas beta: Pruebas aplicadas cuando el desarrollo y las pruebas principales de funcionamiento están terminadas. Los errores y problemas finales necesitan ser encontrados antes de la liberación completa del software. Es típicamente realizada por usuarios finales, pero no por programadores o probadores.

Pruebas de mutación: Pruebas dedicadas a determinar si un conjunto de datos de prueba o casos de prueba es útil para deliberadamente introducir cambios en comportamiento del código (Errores). Se comparan estos resultados con los obtenidos usando los datos originales de prueba para determinar si los errores son detectados. Una apropiada realización de esta prueba requiere amplios recursos computacionales.

Pruebas de comparación ★: Compara las debilidades y fortalezas de un software contra productos de la competencia.

Esta es una lista no exhaustiva de las pruebas de software más conocidas [Pre02]. Se marcaron con ★ las pruebas realizadas durante el desarrollo del proyecto, según la conveniencia de su aplicación para este análisis.

A.3.2. Resultados obtenidos de las pruebas de software

El propósito de toda prueba de software es encontrar defectos²¹. Los defectos a los que se hace referencia tienen que ver con la desviación del funcionamiento esperado del software en relación a lo esperado por los usuarios. En general todos los defectos resultan perjudiciales para la aplicación,

²¹Debido a la falta de consenso general sobre la definición de falta, error y falla (*fault, error and fail*)[dM00], en esta tesis se usa la palabra defecto para denotar todo elemento interactivo, funcional y de implementación presente en el software que lo aleje del ideal de calidad establecido por el fabricante, las normas internacionales y el uso del elemento según el sentido común.

siendo peores aquellos que se convierten en causal de rechazo al uso por parte de los usuarios ya que representan un costo máximo para cualquier aplicación.

Integran este resumen los defectos encontrados más frecuentes y fácilmente reproducibles. De entre los defectos más graves encontrados sobresalen las *violaciones de acceso*, ya que cuando se presentan generalmente son seguidas de la terminación abrupta del programa. También durante las pruebas se encontraron errores de memoria y mensajes de punteros inválidos, pero por su infrecuencia y la dificultad para determinar los pasos necesarios para reproducirlos, no se incluyeron dentro de estos resultados. El autor considera que la lista presentada es suficientemente significativa para los propósitos de ilustración de este análisis.

En cada defecto listado y cuando exista la posibilidad, se hará notar una solución conocida para superar el problema. Si ésta no se indica debe asumirse que no es posible evitar la situación problemática en la presente versión del software. El propósito es hacer notar que existen soluciones para algunos de los inconvenientes que presenta actualmente la herramienta, pero también que estas soluciones alternativas surgen como remedio a las imperfecciones existentes en el software. Los defectos aquí presentados hacen parte de aquellos que son visibles luego de la entrega del producto y generalmente son consecuencia de descuidos, problemas imprevistos o no cubiertos durante la etapa de diseño. Procesos de mantenimiento posteriores a la entrega del software permiten corregir o al menos reducir sus efectos.

A.3.2.1. Equipo de pruebas

Las características del hardware en las que se realizaron las pruebas son:

- Equipo Pentium 2 de 350 Mhz, 384 Mb de ram, 30 Gb de disco duro con Windows 98 SE, quemadora de 52x.
- Equipo Dell Pentium 4 de 2.8 Ghz, 256 Mb de ram, 80 Gb de disco duro con XP SP2, unidad de CDROM 48x.
- Equipo Pentium D de 3.2 Ghz, 2 Gb de ram, 350 Gb de disco duro con XP SP2, quemadora de 52x.

Estos equipos no se usaron simultáneamente, han sido empleados en diferentes momentos durante el desarrollo de este proyecto. El equipo Dell fué facilitado por la EISI²² en la sala JAVS 4, las otras máquinas fueron obtenidas por parte del desarrollador de este proyecto.

A.3.2.2. Pruebas de carga en caja negra

- Crecimiento lineal en la cantidad de memoria usada por la aplicación:

El consumo de este recurso es directamente proporcional al tamaño del modelo y al tiempo de planeado de ejecución o número de pasos. La causa principal es el crecimiento lineal de las

²²Escuela de Ingeniería de Sistemas e Informática, UIS.

tablas de datos. Una mayor complejidad del modelo implica un mayor consumo de este recurso por paso de simulación. Es posible mejorar esta situación evitando o reduciendo el uso de las vistas durante la simulación.

- Elevado consumo de CPU durante simulación:

La aplicación hace uso intensivo de la CPU y no existe facilidad en la herramienta que permita su control o administración. Del análisis de código realizado se evidencian ineficiencias en algunos de los procesos internos de cálculo.

- Necesidad de mejora en algoritmos para obtener tiempos de ejecución más rápidos:

Durante las pruebas realizadas se comprobó que no hay una mejoría notable en la velocidad de procesamiento de un proyecto determinado aún usando hardware de mejores prestaciones.

- Eficiencia en costos de almacenamiento:

Este es un aspecto en el cual la herramienta es eficiente. Cuenta con un tamaño de instalador y de instalación reducidos. Gracias al diseño aplicado en la estructura de datos y el tamaño mínimo definido para las figuras en los archivos *.prh, hace innecesario el uso de estrategias de compresión o similares para reducir su tamaño.

- Costo de recursos GDI:²³ en Windows 95, 98 y ME

Se han encontrado ineficiencias y debilidades en la administración de estos recursos, recordando también que esta es una de las limitaciones más importantes en esos sistemas operativos²⁴. En versiones superiores de Windows (NT, XP, Vista, 7) no se han reportado estas incompatibilidades.

A.3.2.3. Resultados de pruebas funcionales sobre la interfaz gráfica

- Maximización incompleta de la ventana principal de la aplicación:

Al maximizar la ventana de la aplicación, esta no se ajusta al marco del escritorio de Windows. Deja un espacio vacío en la parte superior, haciendo que la parte inferior de la ventana este siempre dentro del área de la barra de herramientas de Windows.

- Enfoque siempre activo de uno cualquiera de los elementos la barra de zoom accionada por el ratón:

Cuando el enfoque no está dirigido sobre un menú de la aplicación, se asigna por defecto a uno de los elementos que componen la barra de zoom. La selección de objetos en el ambiente no hace diferencia sobre esta situación.

²³Graphics Device Interface: interfaz de dispositivo gráfico de Windows, contiene todas las funciones y estructuras necesarias para la manipulación de contenido gráfico de y hacia cualquier dispositivo de salida conectado al ordenador: pantalla, impresora, plotter, etc. Las familias NT, XP y Windows 2000 tienen manejo ilimitado de recursos GDI.

²⁴Ver Windows says it's "Dangerously Low on System Resources" en <http://www.annoyances.org/exec/show/article07-104>

- Proceso de creación de reglas ininterrumpible luego de establecer el tipo de regla:

Luego de iniciar la creación de una regla no está permitido abortar este proceso, al intentarlo el mensaje es *“No puede cancelar la creación de la regla”*. Esta restricción aplica para la creación de todas las reglas sin excepción.

Solución: Terminar la creación de la regla, modificarla o eliminarla de acuerdo al propósito usuario.

- Cambio de la información definida en la regla de alteración durante edición:

Luego de crear una regla de transición, si se accede a ella y en vez de aceptar se da cancelar para salir del cuadro de diálogo de edición de la regla, el valor de la propiedad segunda clase de la regla cambia. Se inicia en el valor de la primera clase objeto creada en el modelo.

Posible Solución: Evitar editar la regla de alteración una vez que ha sido establecida.

- Límite en el número de clases objeto a los que se puede acceder manualmente desde la barra de clases objeto:

Desde la barra de clases objeto es posible acceder directamente hasta 33 clases objeto en resolución de 800x600, 45 en 1024x768, 61 en 1440x900 y así sucesivamente.

Solución 1: Incluir las clases no listadas programáticamente desde la lista para ubicar objetos de cada clase dentro del ambiente en el cuadro de diálogo de escenario.

Solución 2: Ir a insertar objetos, elegir la clase fuera de alcance e incluir uno o más de las instancias del objeto.

En ambos casos se hará necesaria una reubicación manual de los objetos insertados. Esta operación variará dependiendo de las necesidades de ubicación de objetos del desarrollador en ese modelo en particular.

- Abortar la creación de una clase objeto no la elimina del listado de clases objeto disponibles:

Al cancelar el proceso de creación de una clase objeto esta no desaparece de la lista, la clase es creada y asignada con los valores establecidos por defecto para esta operación.

Solución: Eliminar la clase objeto abortada manualmente.

- No hay autonumeración de nombres de reglas, clases, archivos, escenarios y otros identificadores de texto enumerables

Se ofrece un único valor en cada caso, que corresponde al nombre del tipo de elemento en edición seguido de un 1. por ejemplo Clase1, Escenario1. Las reglas no ofrecen nombres por defecto.

- Acción de edición de imagen inconclusa:

Al hacer click sobre la imagen aparece el mensaje *“inicia la aplicación o ventana que permite la edición de la imagen”*. No se incluye como editar o cambiar esta opción en el programa.

Solución: Abrir manualmente el editor de imágenes preferido.

- Captura de imágenes del diagrama BOR solo con herramientas externas:

No se dispone de una facilidad en la herramienta que permita salvar la imagen gráfica del diagrama BOR. Se requiere para ello de una aplicación externa.

- Defectos de imagen cuando se realizan cambios en la distribución del espacio del diagrama BOR.

Al mover el separador entre el espacio del diagrama BOR y el árbol de transiciones, aumentando el espacio en el diagrama, se observa una línea fantasma sobre el árbol de transiciones. Esto se da especialmente al cambiar el estado de maximización del formulario de la aplicación.

- Claustrofobia en diagramas BOR:

La característica de reducción de espacio automática en el diagrama BOR hace que el objeto que este más hacia abajo y a la derecha determine el tamaño total del diagrama.

- Inconsistencia en la ubicación de la dirección de aplicación de reglas en diagrama BOR:

Al ajustar la posición de las clases objeto en el diagrama, la ubicación de la flecha que indica la orientación de aplicación de la regla cambia de lugar, se ubica en el extremo opuesto del objeto a la que apunta, sin perder el sentido de aplicación. Esto sucede cuando el elemento apuntado se alinea horizontalmente a la derecha de la otra clase objeto.

Solución: Reubicar la clase objeto apuntada hasta que se produzca un leve desnivel entre los objetos relacionados. Esto hace que la punta de flecha se dirija a la clase objeto destino de la regla correctamente.

- Posibilidad de pérdida de clases objeto en el diagrama BOR:

Si se toma una clase objeto en el diagrama BOR y se la envía a un espacio más allá de los bordes superior e izquierdo de la ventana del diagrama, la clase objeto manipulada se hace inalcanzable permanentemente.

Posible solución: Ninguna, abstenerse de realizar dicha operación. Si ya se ha producido esta situación, entonces evitar salvar el modelo para que al reabrirlo, este aparezca en la ubicación anterior al traslado.

- Posibilidad de desintegrar del diagrama BOR:

Si al realizar la edición del diagrama BOR el área del diagrama es mayor que el espacio del área de la ventana que lo muestra, el algoritmo de redibujado falla al deslizar verticalmente la imagen. La integridad gráfica del diagrama se pierde.

Solución: Reubicar las clases objeto para que se encuentren dentro de la zona visible del diagrama.

- Posible inutilización del software por limitación en hardware:

Es imprescindible la presencia del ratón para usar completamente la herramienta, sin él es imposible organizar a gusto los objetos en el espacio del diagrama BOR, ya que no existe un proceso de organización automática disponible para estos elementos, también se dificulta la inserción personalizada de objetos en el ambiente.

A.3.2.4. Prueba funcional general

- Es posible simular modelos vacíos:

Se puede iniciar una simulación de un modelo que no contiene clases objetos y reglas de transición.

- La terminación de la simulación no es inmediata:

Dependiendo de la cantidad de cálculos a realizar por paso de simulación, es posible que la aplicación no responda por unos segundos.

- Necesidad de terminar previamente la simulación antes de cerrar el programa:

El mensaje *“debe terminar la Simulación antes de cerrar”* se hará visible si se pretende terminar el programa durante la simulación.

- Necesidad de guardar un nuevo proyecto al momento de crear el modelo:

Al crear un nuevo archivo de proyecto se abre una ventana *“guardar como”* inmediatamente. No es posible crear un documento nuevo sin salvarlo primero

Solución: Abrir un documento de proyecto existente y realizar las modificaciones necesarias hasta que se alcance la forma requerida. Luego salvar el proyecto con el nombre deseado.

- Pregunta de guardado de proyecto actual aun si no se han realizado cambios:

Esta pregunta se hace siempre, la aplicación contempla el salvado del documento actual antes de su finalización. No hay opción o configuración que permita cambiar este comportamiento.

- No se encuentra el último archivo abierto por la aplicación:

Si se abre un archivo de proyecto en una carpeta diferente a *“raíz del programa /proyectos”* al hacer click en el elemento de archivos recientes del menú se hace visible el siguiente mensaje: *“error en el archivo del proyecto Homos. No existe o Esta ocupado”*.

Solución: Guardar los archivos de proyecto en la carpeta indicada para ese propósito.

- La creación de escenarios no es inmediata:

Es necesaria la presencia de un escenario activo (seleccionado) para realizar una simulación. No existen escenarios por defecto.

- Excepción al manejar un modelo con 2 o más escenarios:

En determinadas circunstancias se presenta un *"Exception ElistError - List index out of bounds"*.

- Imposible eliminar vistas asociadas a un escenario específico:

Las lista de vistas sigue inalterada luego de realizar el procedimiento de borrado ofrecido por la aplicación.

- Problema aleatorio de impresión:

En determinadas circunstancias es posible que la aplicación no pueda realizar el proceso de impresión reportando que no puede crear un determinado archivo.

- Espacio insuficiente en ambiente y bloqueo de simulación:

Se presenta al usar en el escenario la opción de ubicación automática de objetos. Con cada simulación esta opción va llenando el ambiente de objetos hasta que se produce este mensaje: *"en el ambiente no hay espacio suficiente para ubicar los objetos seleccionados en el escenario"*. Luego de este mensaje no es posible iniciar nuevamente la simulación, aun cerrando el proyecto actual y cambiándolo por otro.

Solución: Borrar todo el contenido del ambiente antes de cada simulación. Si el mensaje se ha hecho presente, reiniciar la aplicación.

- Redimensionamiento dinámico de ambiente permitido:

Al reducir dinámicamente el tamaño del ambiente de simulación se presenta un *"Exception ElistError - List index out of bounds"*.

Posible solución: Ninguna, abstenerse de realizar dicha operación.

- Posibilidad de interactuar con el modelo dinámicamente durante la simulación:

Es posible insertar clases objeto en el ambiente y alterar el comportamiento esperado del modelo. Este es en la mayoría de los casos un comportamiento no deseado del simulador.

- Posibilidad de alterar la integridad de la lista de ubicación automática en escenario:

Se presenta al agregar una clase objeto en la lista para ubicar del escenario y luego cambiar manualmente el nombre de esa clase. El cambio en el nombre no se actualiza en cada una de las listas internas del programa, generando inconsistencias y posibles violaciones de acceso.

Posible solución: Ninguna, abstenerse de realizar dicha operación.

- Violación de acceso y error de memoria durante simulaciones de larga duración:

Se presenta al ejecutar una simulación por largos periodos de tiempo.

Solución: Establecer un numero de pasos razonable para los procesos de simulación.

- Violación de acceso al presentar la gráfica de vista de cambios:

Se presenta cuando se realizan cambios en el nombre de una de las clases objeto que sirve como fuente de datos a una vista. El problema se hace visible al iniciar la simulación o al acceder a dicha vista.

Solución: Evitar hacer esto. Si se necesita cambiar el nombre de una clase en esas condiciones entonces revisar las ventanas de vistas presentes en el modelo y verificar que dicho nombre de clase no se encuentre en uso por alguna de esas ventanas. Si lo está, entonces remover dicha clase de las vistas en las que aparezca, realizar el cambio y reinsertar la clase donde se la necesite. Si se guarda el documento previamente al momento de cambiar de ventana o simular, la falla se hace permanente en la estructura de datos del archivo de proyecto. Al abrir este documento se presenta nuevamente la violación de acceso.

- Regla de neutralización selectiva:

Cuando al usar esta regla, se establece que la clase neutralizante y la clase neutralizada son clases objeto distintas, la clase neutralizante sobrevive durante la simulación en la mayoría de los casos ignorando la esencia misma de la regla. Esto no sucede cuando la clase neutralizante y neutralizada pertenecen a una misma clase.

- Regla de reproducción invasiva:

Al proponer un modelo con un subespacios cerrados, la clase que ejecute la regla de reproducción burla dicho cerco e invade espacios que de otra forma serían inalcanzables para esa clase objeto.

- Regla de retracción invasiva:

Al aplicar esa regla sobre una clase objeto, las demás clases objeto presentes en el ambiente son afectadas por su acción, aunque su definición establezca que solo se ven afectados los objetos de la clase a la que se aplica.

- El algoritmo de simulación actual no hace diferencias al calcular clases-objeto con o sin reglas:

Una clase objeto que tenga reglas asociadas (clase activa) es en la práctica más lenta de procesar que una que no tiene reglas asociadas (clase inerte) y por esa razón es posible pensar que un ambiente lleno de clases inertes simule más rápido que uno compuesto por clases activas. Durante las pruebas se encuentra que no hay diferencias significativas entre estos ambientes opuestos y la forma en que el simulador trata estas distintas clases objeto, lo que se traduce en un aumento del uso de la cantidad de CPU necesaria para la simulación.

- La velocidad de simulación es dependiente del tamaño del ambiente planteado en el modelo:

Para un proyecto que no implique cambios poblacionales al ejecutarlo en ambientes de tamaños diferentes, se simula mucho más rápido en un ambiente de dimensiones reducidas que en uno extenso.

El contenedor fundamental de los objetos en el ambiente es una matriz. Esta matriz es recorrida punto a punto, evaluando la existencia o no de objetos y realizando las diferentes evaluaciones según lo establecido por las reglas fijadas para el modelo. Se pierde tiempo de simulación en la evaluación de las distintas celdas vacías, es posible pensar en otros tipos de almacenamiento para evitar este inconveniente.

- No hay validación de consistencia o integridad de la estructura de datos en los archivos de modelo *.prh:

Si los datos del modelo presentan algún tipo de inconsistencia, al cargar el archivo se presenta una violación de acceso y desde ese punto en adelante la estabilidad del programa se encuentra comprometida.

- Es posible burlar la seguridad de los archivos *.prh:

Mediante un editor de texto como Notepad es posible ver la contraseña establecida en el archivo ya que sobre este dato no se hace encriptamiento. No existen mecanismos adicionales en la herramienta que refuercen este aspecto.

Solución: Manejar un esquema de seguridad externo mediante el uso de herramientas de compresión y encriptamiento de archivos, según las necesidades del usuario / desarrollador.

- Problema de sincronía del archivo repositorio.rep:

Al abrir 2 o más instancias de Homos simultáneamente durante la ejecución se presenta el siguiente mensaje “cannot create file ... /repositorio.rep”.

Solución: Homos no ha sido diseñado para ejecuciones múltiples simultáneas. Es posible ejecutar copias de la aplicación en carpetas separadas, pero se recomienda abstenerse de realizar dicha operación.

- Problema de asignación de parámetros en reglas en la interfaz OLE:

Durante la revisión de código se encontró que la interfaz no asigna el valor del parámetro Prioridad internamente en las reglas de transición Expansión, Retracción, Transformación; por lo tanto ellas funcionan con el valor predeterminado [Prioridad = 1] al momento de ser usadas dentro de una simulación iniciada con esta interfaz.

- Métodos en la interfaz OLE con ausencia de controles internos sobre datos:

Existen métodos en la interfaz que requieren de una planificación previa para hacer uso seguro de ellos. Por ejemplo, al usar el método `ObjetosInsertar` es posible producir un *desbordamiento* de la capacidad del ambiente de contener objetos. La *solución* posible a esta situación es llevar la cuenta de los espacios vacíos para realizar una inserción segura de objetos. Esto es algo que debería hacer por sí mismo el algoritmo que realiza la inserción, no el usuario o una de las capas externas al software. Ver el análisis a la interfaz OLE en la sección A.5 en la página 174.

A.3.2.5. Facilidades ausentes en esta versión del software

La herramienta cumple su propósito la creación de modelos basados en objetos y reglas. Permite su simulación y la obtención de resultados en cada una de sus ejecuciones. Sin embargo, durante la vida útil del software se ha hecho evidente que hay falencias en algunos de los aspectos que componen al software.

- No hay cuadro de diálogo u otro elemento en el programa que permita guardar la configuración actual de la herramienta:

No se dispone de una función que almacene las preferencias o configuración del usuario, por ejemplo en el registro de Windows o en archivos *.ini.

- Ausencia de facilidades de edición como rehacer y deshacer:

No es posible para los usuarios hacer uso de este tipo de operación en esta versión de la herramienta. El costo máximo de equivocarse en un paso de creación es rehacer la totalidad del modelo.

Solución: Guardar copias del documento de proyecto actual antes de realizar cambios críticos. Trabajar y hacer cambios con la última copia.

- Ausencia de otras opciones de configuración en escenario:

Permitir variar el tipo de límite presente en el ambiente de simulación, el color y otras características adicionalmente a la opción de cambio de zoom y tipo de representación de imágenes.

- Ausencia de una facilidad que permita establecer inicios diferentes en escenarios:

La herramienta permite establecer escenarios diferentes para un mismo ambiente, pero no es posible establecer varios ambientes que usen un mismo escenario.

Solución 1: Se define el proyecto y se establece el escenario. Se cambia el ambiente tantas veces como sea necesario.

Solución 2: Crear un proyecto *fuentes* y copiarlo tantas veces como ambientes distintos deseen ser probados. En este proyecto origen se halla especificado el escenario de pruebas planteado.

- Ausencia de opciones de edición en diagramas BOR:

Hay pocas opciones de manejo de las clases objeto y sus relaciones en los diagramas BOR. Están ausentes facilidades como la selección de varios elementos simultáneamente o su acceso y manipulación mediante teclado.

- Ausencia de mejores controles de simulación:

No existe una forma directa de realizar el avance paso a paso de la simulación o la terminación automática de la simulación ante determinada situación poblacional en el ambiente.

Avance paso a paso temporizado: Es factible emular la simulación en cámara lenta, mediante los siguientes pasos: Crear una clase objeto inerte²⁵, aumentar el área del ambiente para acomodar instancias de esa clase y aislarlas de los demás objetos presentes en el ambiente del modelo. A mayor cantidad de instancias de esta clase inerte, es mayor el tiempo que tarda en completarse un paso de simulación. Como esta clase es un agregado a las demás presentes en el modelo puede manejarse su número a gusto del desarrollador, según sus necesidades. Cabe anotar que esta es una solución claramente ineficiente en términos de uso de la CPU. Lo contrario también es cierto, al reducir el número de instancias de clases objeto presentes en el ambiente, mayor será la velocidad con que se calcule cada paso de simulación.

- Vista condensada de los datos y configuraciones presentes en el modelo:

Ausencia de una facilidad que permita ver en forma condensada los componentes del modelo; fácil acceso a clases, reglas, parámetros establecidos en el modelo e indicaciones del estado del simulador.

- Intuitividad en el manejo e la interacción con la herramienta:

La herramienta tiene algunos aspectos poco intuitivos en su manejo, particularmente en lo relacionado con los datos calculados y el uso de las gráficas de comparación. Ausencia de controles de terminación de la simulación programables o al menos relacionados con la cantidad de instancias de clase presentes en la simulación, elementos que mejorarían la experiencia de uso.

- Enriquecer el aspecto multimedia de la herramienta:

La aplicación no hace uso de audio o video adicionales, tanto en la documentación o como medios externos adicionales a los modelos, excepto los incorporados con fines de presentación de la herramienta.

La presentación visual de la interacción de los objetos en el ambiente de simulación y el diagrama BOR del modelo son el aspecto gráfico más destacado de la aplicación.

- Limitaciones en la expresividad del modelado:

Ausencia de un formalismo escrito explícito para la expresión de los modelos, ya sea en forma puramente descriptiva, con apoyo matemático o mediante un lenguaje propio. Poca variedad temática en las reglas de interacción, actualmente todas son de inspiración biológica.

- Propósitos documentales:

Aspecto documental ineficiente en cuanto a su propósito y utilidad. Está limitado, no tiene una buena integración con el modelo y generalmente es ignorado, tomado por innecesario o una curiosidad entre los desarrolladores. No existe una relación claramente definida entre el modelo en lenguaje prosa y la implementación del modelo en la herramienta.

²⁵Que no tiene reglas asociadas

- Necesidad de límites de manejo operacional en métodos de la interfaz OLE:

Algunas funciones de construcción del modelo pueden ser usadas durante procesos posteriores de simulación, con resultados imprevistos y caídas de la aplicación si se hace uso inapropiado de ellas.

- Ausencia de algunas funciones básicas en la interfaz OLE:

No se cuenta actualmente con funciones que permitan conocer el valor del paso actual de simulación, permitir crear reportes texto o gráficos de algunos elementos componentes del modelo, entre otras ausencias notables. Ver el análisis a la interfaz OLE en la sección A.5 en la página 174

A.3.2.6. Conclusión y resultados obtenidos

La razón por la cual existe esta amplia variedad de elementos en esta sección del análisis se debe principalmente al prolongado periodo de uso de la aplicación sin acciones de mantenimiento. El GS no cuenta con recursos suficientes para dedicarse completamente al desarrollo continuado de aplicaciones.

Sobresale sin lugar a dudas los defectos en el funcionamiento de algunas de las reglas de transición implementadas en el simulador. Aunque los problemas detectados no impiden su funcionamiento, reducen en buena medida la fiabilidad y la certeza con respecto al comportamiento esperado de las reglas al momento de ser usadas en un modelo cualquiera.

A.3.3. Métricas de código fuente

Son un conjunto de medidas de software que permiten obtener una vista clara del código que se está desarrollando. Permiten identificar riesgos potenciales, entender el estado inmediato del proyecto y llevar registro de avance del desarrollo del software[Pre02]. Para la actividad de pos desarrollo realizada en este análisis, la siguiente tabla se presenta un resumen de las métricas de código fuente relevantes obtenidas del programa:

Tabla A.4: Resumen de métricas de código fuente de Homos 1.0

Métrica	Σ
Archivos	94
Subrutinas	924
Unidades compiladas	88
Formularios	45
Líneas blancas	2023
Líneas de código LC	17308
Líneas de comentario LCOM	734
Porcentaje de código comentado LCOM/LC en %	4.2
Total de Líneas	19764
Declaraciones Constantes (Declarative Statements)	4147
Declaraciones Ejecutables (Executable Statements)	7600
Tamaño final ejecutable en Bytes	1868288

Fuente: Autor.

Del resumen presentado pueden resaltarse los siguientes elementos:

- Tiene un índice de líneas de código comentadas de $\approx 4\%$. Esto hace particularmente difícil la revisión y el mantenimiento del código de la aplicación.
- El tamaño en código de la aplicación es reducido. El total de subrutinas es elevado con respecto a la cantidad de líneas de código efectivas, suponiendo una cantidad uniforme aproximada de 18 líneas de código por cada una. Indica una cantidad apreciable de subdivisión del trabajo pero también puede representar una repetición importante de subrutinas dispersas por todo el código. Como lo demuestra la observación directa del código, Cualquier iniciativa de mantenimiento requerirá de una fase previa de ordenamiento y simplificación de subrutinas, que implica a su vez posibles cambios en la estructura interna del software.

Durante el proceso de investigación del desarrollo original del software no se encontró prueba de la realización de algún tipo de métricas u otro tipo de análisis de código durante su construcción. No existe registro de tales actividades.

En un apartado posterior dentro de esta misma sección se enuncia el conjunto de herramientas usadas para la realización de este análisis. El reporte aquí presentado es un resumen de la suma parcial de elementos seleccionados de los informes ofrecidos por cada herramienta utilizada.

A.3.3.1. Reporte de análisis de código

A continuación se listan algunos de los elementos encontrados en el análisis que van en claro detrimento de la aplicación, destacando los más importantes.

Legibilidad del código

- Se manejan una nomenclatura propia en nombres de variables y demás estructuras, por ejemplo: "TxtNumeroFilas", "FrmRegistro". Pero también se usan los nombres autogenerados de los objetos de Delphi como "label14", "Shape13".
- Poca documentación interna, gran parte del código sin comentar (96 %).

Cumplimiento de convenciones

- 135 campos de clase declarados en forma no estándar.
- 7 identificadores con nombres no apropiados manejo de nomenclaturas no estándar en algunas variables.
- 9 clases con constructores visibles con nombres no estándar.

Inconsistencias en uso de convenciones propias del lenguaje

- 5 nombres de unidades erróneas.
- 297 casos de inconsistencias dentro de identificadores usados en la aplicación. Por ejemplo se da homonimia de clases entre TEntero en UTabladatos y TEntero en UAutomata. Mismo nombre, diferente estructura.

Reducción de código

- 66 funciones procedimiento, 88 llamadas a métodos usadas solo una vez dentro de las unidades y 45 identificadores establecidos y referenciados solo una vez.

Existencia de código anulado

- Hay código bloqueado en la aplicación, sin comentarios visibles del porqué de su desactivación.

Existencia de código extra

- De la evaluación del código fuente se encuentra que hay inclusión de módulos sin uso práctico al propósito de la aplicación. Específicamente existe un *eastern egg*²⁶ relacionado con los créditos del programa en *Facercade.pas*, que da acceso a un juego 4 en línea descrito en *Fjuego.pas*. Este código es funcional en todas las versiones conocidas de la herramienta.
- Existen atributos y métodos de clases que no se usan dentro de la aplicación. Un ejemplo en la página 194.

Reutilización de código

- Hay poca reutilización del código, se observa repetición de funciones principalmente en las secciones que definen las reglas de transición y su manejo en los formularios.

Manejo de variables

- Existen en la aplicación variables instanciadas no usadas, por ejemplo “*hjkghjk1*” en *Fgraficacomparación.pas*.

Código riesgoso

- 9 Ciclos *for*, 4 *repeats* y 40 condiciones *if* en corto circuito.
- 17 bloques *begin-end* vacíos.
- Error en la Creación de un objeto en *FHomos*.
- Uso de constructores o destructores sin herencias *inherited* en 38 casos.
- 31 casos de variables declaradas nunca usadas.

Manejo de memoria - *Memory leaks*

- 39 llamadas a Liberación de objetos sin medidas de protección, usando el comando *Free*.

Los valores presentados se obtuvieron del análisis automatizado del código fuente realizado por varias aplicaciones, las cuales se listan en la tabla A.7.

A.3.3.2. Código fuente existente y versiones de la aplicación

Debido problemas de almacenamiento del código fuente ajenos a la voluntad de los desarrolladores originales o el GS, solo quedan del proyecto Homos 1.0 el código fuente de la versión completa en español, la profesional en inglés y los instaladores de las distintas versiones compiladas del software, todas incluidas dentro del CdPS.

²⁶Ctrl+Alt+Shift+botón derecho del ratón sobre el botón *video* del la ventana *acerca de*.

A.3.4. Estructura interna y algoritmos usados

Con el propósito de conocer sobre los fundamentos sobre los cuales se construyó el software Homos 1.0, se presenta a continuación un resumen del análisis de los elementos de diseño existentes en la solución final implementada en el software Homos 1.0. El análisis se realizó mediante una revisión del código fuente usando diagramas de flujo y del diagrama de clases UML de la aplicación.

A.3.4.1. Exposición de algunos supuestos presentes en el actual diseño del software

Se detectaron los siguientes supuestos de desarrollo en la construcción de la aplicación: se listaron solo los aspectos relevantes para esta sección del análisis:

- Al observar la caracterización de las reglas de transición en el sistema se hace notar que:
 - Existe un tipo de formulario para cada tipo de regla y una unidad de código por cada tipo de regla, con diferencias mínimas entre unidades. Véase la figura A.6 en la página 136.
 - Se distinguen unas de otras por sus parámetros, la cantidad de clases requeridas²⁷ y si requieren establecer la dirección de acción y el número de pasos.
 - Cada una tiene un tipo distinto de acción final, esto es lo que causa la representación especializada de cada regla en el sistema.
 - Las funciones de transición f y g que caracterizan a las reglas en el simulador realizan cada una:
 - ◊ $aplicar(g)$ = mover/instanciar + eliminar + transformar un objeto.
 - ◊ $aplicar(f)$ = eliminar y transformar objetos.
 - El método usado para la representación de reglas de transición mediante subclases de un tipo principal de regla de transición es dinámicamente inflexible; La única forma de implementar nuevos tipos de reglas de transición es realizando cambios a nivel del código fuente.
 - El comportamiento real de las reglas de transición está codificado externamente al cuerpo de las propias reglas: Las transiciones que caracterizan a cada regla se encuentran implementadas internamente dentro del código de las funciones que establecen el funcionamiento de simulador. Esto agrava el estado del punto anterior.
 - No se previeron facilidades en el software para ampliar sin consecuencias estructurales la cantidad de tipos de reglas de transición disponibles para el proceso de simulación.
- Al observar la caracterización de los objetos y las clases objeto en el sistema se hace notar que:
 - Los objetos de simulación no son objetos reales, su implementación actual los hace actuar como registros en una lista encapsulados por las definiciones presentes en `Tobjeto`.

²⁷Entre cero y tres clases-objeto

- Las definicion de las clases objeto se encuentra aislada de las reglas asi que un cambio en esta seccion del codigo no implica una alteracion en el diseño actual de las reglas.
- Se implementó una asociación débil entre objetos y clases-objeto:
 - Se hace posible implementar caracterizaciones adicionales a los objetos distintas de las atribuibles por la clase objeto a la que se asocie.
 - Plantea la posibilidad de implementar caracterizaciones especiales particulares de los objetos asi construido
 - El objeto referido puede ser de cualquiera de las clase-objeto existentes o de *ninguna*²⁸.
 - Cada objeto hace referencia a solo una de las clases-objeto existentes, esta relacion no se hace notar explicitamente por ninguno de los diagramas de clases presentados. La relacion entre clases y objetos es un artilugio lógico.
- La clase que define a las clases-objeto contiene en su interior la descripcion de cada objeto presente en el sistema, consistente en su color de clase, ubicacion en el espacio, nombre y valores iniciales
- Por defecto de todos los objetos de la simulacion se ubican en la primera celda del ambiente, en [1,1].
- Existe una asociacion entre instancias de reglas de transicion y clases-objeto, pero la operacion inversa no esta implementada, conocer desde la clase objeto cuales son las reglas de transicion en las que participa. Esta relación simplificaría muchos de los procesos realizados por el algoritmo que concreta la evolucion del sistema.
- La clase Tobjeto implementa los atributos pasoentrada y pasosalida, pero no se hace uso de estos atributos y sus métodos en ninguna parte de la aplicación. Su existencia supone la intención de permitir el cálculo del tiempo de vida de simulación de los objetos.
- Sobre el almacenamiento de datos usado en el sistema se hace notar que:
 - Un supuesto clave del desarrollo de la aplicacion en general es que se hará *uso de pocas instancias* de cada uno de elementos de modelado disponibles. Esto se deduce de como estan constituidas las distintas fuciones dentro del sistema.
 - En general el formato binario usado es inflexible, no facilita el cambio o actualización del formato: El diseño implementado es impractico a largo plazo: Cambiar o alterar la configuracion de una regla implica crear el codigo especifico para esa regla, alterar el diseño del archivo que contiene el modelo y recompilar la aplicacion ante cada cambio relevante, generando por esta via incompatibilidad entre versiones.
 - El almacenamiento de datos se realiza en formato serial y fijo. Alterar este orden en el software o en un documento ya salvado hace imposible la carga del archivo-modelo en la aplicación. Esto hace que modificar el código fuente de homos 1.0 produzca inevitablemente la necesidad de implementar codigo de adaptación de formatos de archivo entre distintas versiones.

²⁸Objetos sin tipo, Esto último tipo es posible por la estructura presente pero no se hace uso de dicha capacidad en esta version del software.

- Cada tipo principal de datos dentro de la aplicacion tiene una clase que la representa. Se establecen métodos de almacenamiento particular en cada tipo de clase de datos presente en el sistema.
- El proceso de simulación aunque ordenado funcionalmente en su operacion interna, no realiza operaciones en forma modular: No existe un objeto simulador como tal, para representar el funcionamiento del simulador existe una coleccion de funciones y los datos que requiere para su funcionamiento están dispersos entre las las distintas clases presentes en la estructura.
- No es posible establecer valores por defecto o configuraciones distintas a las que se encuentran codificados internamente dentro de la aplicación.

A.3.4.2. *Cuellos de botella relevantes detectados*

Se presentan los elementos mas destacados, en su mayoria corresponden con subtarear del proceso de simulacion. En algunos casos se ofrece solucion a la problematica planteada.

- Se usa profusamente dentro de los algoritmos implementados en la aplicacion búsquedas seriales y ciclos anidados o burbujas. Son efectivas en su accionar, se destacan en ese sentido los procesos de simulacion y la busqueda de objetos en el ambiente. De lo observado en el codigo puede deducirse que pueden realizarse estos procesos en forma mas eficiente.
- El proceso de busquedas en vecindarios realizado en `TSimulador.DarObjetosVecindadEnT` hace 8 busquedas seriales sobre $k \leq n$ objetos con k como el total de objetos evaluados por cada ciclo / celda. Se propone acelerar el proceso paralelizando la búsqueda. Lo optimo seria que cada ubicacion del espacio indicara los objetos que contiene o a los que hace referencia sin necesidad de procesos de busqueda.
- No hay una clasificacion de los objetos segun su funcion dentro de la simulacion. Si un objeto hace parte de la configuración de requisitos de ejecución de alguna de las reglas de transicion entonces el objeto es *activo*. Un objeto que no tiene reglas de transicion asociados es un objeto *inerte*. La funcion `PasarSiguienteEstado` de la unidad de simulación procesa todos los objetos disponibles como si cada uno de ellos fuera activo. La busqueda de objetos a simular deberia restringirse solo al conjunto de objetos activos. La clasificación de los objetos es estática y puede conocerse desde antes del inicio de la simulacion, las relaciones presentes en el modelo permiten establecer la diferencia entre activo e inerte en los objetos.
- Se realiza una eliminacion serial de listas de objetos en cada paso de simulación: hay un *constante* proceso de liberacion de la memoria usada por las listas de objetos `ListaT1` (estado anterior) y `ListaT1` (estado siguiente) al final de cada paso de simulacion. El metodo implementado aumenta el tiempo de procesamiento en cada paso de simulacion. Este proceso puede realizarse en forma optima mediante el uso de una estructura de datos que no requiera reiniciar constantemente los datos almacenados.

- *fmatriz* es la matriz que representa al ambiente de simulación. Esta estructura es *simulada* mediante listas. El acceso a una ubicación dentro de una lista en Delphi es muchísimo más lento que el de una matriz. Se presume que la razón de la adaptación se debió a reducción de uso de memoria. En la actualidad ya no es necesaria tal precaución.
- La unidad *ulistareglas* define a *Tlistareglas* un objeto en el cual se realiza búsqueda de reglas de transición mediante comparación de cadenas de caracteres. Compara serialmente el nombre de una regla de transición dada contra los demás nombres disponibles hasta hallar o fallar la búsqueda. Esta función recibe frecuentes llamadas de funciones vitales para la simulación como es el caso de *TAutomata.DarReglasPropias*, una función con la cual se determinan las reglas vigentes para el vecindario actual. El objeto *Tlistareglas* es una solución a un elemento imprevisto en el diseño: hacer uso de un índice numérico particular para cada regla implementada en el modelo. Se sugiere deshabilitar toda búsqueda mediante cadenas e implementar en código los elementos necesarios para el uso de índices numéricos en las reglas de transición. Lo óptimo sería evitar toda búsqueda y hacer uso de enlaces directos entre estructuras de datos.
- Al aumentar la cantidad de reglas de un modelo se ralentiza enormemente el proceso de simulación de la aplicación. Internamente el algoritmo implementado repasa la totalidad de las reglas de transición cada vez que se requiere determinar el conjunto de reglas aplicables a un objeto. La ejecución de algunas reglas de transición tales como reproducción o neutralización son más lentas que otras por la cantidad de objetos relacionados en sus configuraciones.

A.3.4.3. Otras desventajas presentes de la actual implementación del software

Del análisis realizado se destacan, en orden de relevancia:

- Hay falencias en el proceso de cálculo de la aplicación: No hay método implementado en la herramienta que evite el uso total de la CPU durante la simulación.
- El proceso de simulación no hace uso explícito de técnicas multihilo adaptadas especialmente para el programa, durante la ejecución de los cálculos solo hay un hilo dedicado al proceso de simulación. Este hilo se crea correctamente, pero no es destruido al terminar el proceso, ni al cerrar el proyecto en curso, acumulándose a cada ejecución, usurpando recursos y desestabilizando la aplicación, esto último es válido para entornos Windows 9x.
- Repetición innecesaria de formularios para las reglas. Estas constan de un núcleo básico de propiedades y unos pocos valores que marcan diferencias individuales. Estos casos son subsanables con la presentación u ocultamiento de los elementos propios que caracterizan las diferencias de dichas reglas, sin la creación en código de múltiples formularios.
- El manejo no uniforme de cadenas, que conducen a un sistema incompleto de translación de mensajes y la necesidad de modificar el código fuente, tal y como sucedió cuando se compiló el programa para crear la versión en inglés de la herramienta.

A.3.4.4. Documentos de soporte

Con propósitos de documentación, se anexa la estructura UML establecida por los autores de Homos 1.0. Esta se encuentra en:

`contenido\anexos\Homos 1.0\Modelo_UML_Homos_1.0_original\UML_Diagrama_clases_original.pdf`

También se incluye una descripción automatizada del diagrama de clases, que incluye los paquetes/unidades y las relaciones existentes entre las clases disponibles²⁹. Los resultados se presentan en:

`contenido\anexos\Homos 1.0\Modelo_UML_Homos_1.0.0.73`

Finalmente se adjuntan los diagramas de flujo del código fuente del software Homos 1.0 en sus versiones 1.0.0.23, 1.0.0.50, 1.0.0.73 en `contenido\anexos\Homos 1.0`

Los diagramas de flujo mostrados se agrupan en guías, La estructura de los diagramas y las guías son particulares de este proyecto. Se describen sus características en la sección E.7 en la página 654.

²⁹Generada con Eldean ESS-Model 2.2

A.3.5. Aplicaciones adicionales

A.3.5.1. El Visor de Homos

Junto con el ejecutable principal, se instala en la carpeta raíz del programa el visor de Homos, una aplicación que permite la ejecución con limitaciones de modelos creados con la herramienta. Su propósito es presentar los modelos sin riesgo de alteración permanente sobre la estructura y parámetros establecidos en el proyecto.

Puede ejecutarse independientemente de la instalación principal del programa, no requiere una instalación preexistente de Homos en la máquina de prueba. En la distribución de Homos se incluye la versión 1.0 de esta aplicación

Características

- Comparte todas las características de ejecución de la simulación encontradas en Homos, pero con las limitaciones necesarias para evitar la modificación del modelo.
- Su funcionamiento no requiere el uso de entradas en el registro de Windows.

Facilidades

- Permite la presentación de estadísticas de objetos presentes en la simulación y en regiones seleccionadas en el ambiente
- Permite la creación y manipulación de gráficas de comparación junto con el guardado de las series de datos calculadas.
- La generación de vistas temporales sobre los datos actuales y la impresión del modelo, excluyendo la presentación del diagrama BOR y el árbol estructural.
- Permite la creación de tablas, vistas y gráficos de comparación.

Restricciones funcionales En el menú principal se suprimen las entradas edición y proyecto, se alteran los demás elementos de la interfaz así:

- Se eliminan las opciones de guardado en el menú Archivo.
- Se agregan los accesos a escenarios y vistas en el menú Simulación. No es posible crear escenarios nuevos ni editarlos.
- Se elimina el acceso al repositorio en la opción de menú Ventana.
- No permite ver el diagrama BOR asociado al modelo.

- Suprime algunos botones de las barras de herramientas. Las modificaciones implementadas son:
 - No se presenta las barras de clases objeto ni la de posición en el ambiente
 - En la barra de edición de ambiente solo está activo el botón de estadísticas del ambiente
 - En la barra de archivo están presentes los botones abrir, cerrar e imprimir.
 - En la barra de proyecto están disponibles los botones de acceso al ambiente, los escenarios, las vistas e iniciar la simulación.

La barra de zoom y modo está completa.

Aunque el menú ayudas esté completo, no es funcional en este programa; en la ventana de *Acerca de...* se excluye la presentación del video de Homos.

Métricas de código fuente disponibles En la tabla A.5 se presenta un resumen de las métricas principales del Visor de Homos:

Tabla A.5: Resumen de métricas de código del Visor de Homos

Métrica	Valores	$1 - \frac{V_v}{V_h}$ en %
Archivos	86	8.5
Subrutinas	869	6.0
Unidades compiladas	53	39.8
Formularios	23	48.9
Líneas blancas	1755	13.2
Líneas de código LC	14446	16.5
Líneas de comentario LCOM	501	31.7
Porcentaje de código comentado LCOM/LC en %	3.4	19.0
Total de Líneas	16563	16.2
Declaraciones Constantes (Declarative Statements)	3701	10.8
Declaraciones Ejecutables (Executable Statements)	6033	20.6
Tamaño final ejecutable en bytes	1631232	12.7

V_v = Valor de métrica obtenido para el Visor de Homos

V_h = Valor de métrica obtenido para Homos 1.0

$1 - \frac{V_v}{V_h}$ Diferencia porcentual entre los valores hallados, relacionan los datos obtenidos en las métricas para el visor [V_v] y la aplicación principal [V_h].

Fuente: Autor.

Para los propósitos de comparación de la tabla resumen, un valor grande de diferencia es algo positivo siendo la única excepción a este criterio el conteo de las líneas de código comentadas. Los

valores cercanos a cero representan *igualdad* en términos numéricos para la métrica señalada en ambas aplicaciones, valores mayores del 20 % representan diferencias apreciables. Valores cercanos a 100 % representan la ausencia del elemento analizado por la métrica.

En cuanto a las diferencias presentes entre el visor y la aplicación principal, se destaca el menor uso de formularios y unidades por parte del visor de Homos, aunque con casi igual densidad de líneas. Las diferencias menores entre los demás valores indican la reutilización de código usada entre aplicaciones,

A partir de los valores de diferencias hallados se destacan:

- Un menor uso de formularios y unidades [40 %], que se debe a la limitación de prestaciones ofrecida por el visor comparada con respecto a la aplicación principal.
- Reorganización de subrutinas en los formularios y unidades disponibles sin mucha variación en cuanto a la cantidad de líneas de código usadas [6 % y 17 %].
- Hay una diferencia del 12.7 % en el tamaño final del ejecutable, que se considera baja si se tiene en cuenta la cantidad de limitaciones presentes en esta aplicación con respecto al programa principal.
- Esta aplicación tiene un [19 %] menos de código comentado con respecto al contenido presente en el código fuente de Homos 1.0.

Se hace notar que no se hace referencia a este programa dentro del manual de usuario, en el archivo de ayuda o en la tesis de Homos.

A.3.5.2. Otras aplicaciones relacionadas

Las aplicaciones listadas en este apartado tienen relación con el presente proyecto como utilitarios independientes, descendientes del desarrollo original de la aplicación. En general estos programas no cuentan con soporte al usuario por parte del GS, tampoco se encuentran disponibles para descarga.

- **SERIALES**

Esta aplicación es un administrador de números de serie, creada con el objetivo de llevar registro y control de las copias vendidas de Homos 1.0. De uso privado por el GS, permitía llevar en forma simple la contabilidad de las licencias comercializadas del software. Actualmente sirve solo a propósitos históricos, como una idea por desarrollar. Se encuentra en el CdPS en ... \Cd Homos\InsHomos\Seriales.

- **EDIHOMOS**

Es un editor de imágenes-iconos para Homos creado por Oscar Anibal Miranda Mercado en el 2000. Está hecho en Visual Basic y su propósito es facilitar la creación de los iconos a usar como imágenes de clases en la herramienta. Actualmente se encuentra asociado a la herramienta en el CdPS en ... \Editor Imagenes.

No se hace mención de su existencia en la bibliografía de la herramienta o la página web del GS. Es un programa muy poco conocido entre los usuarios de la aplicación.

- MICRHO 1.0

Es un micromundo fundado en el modelado basado en objetos y reglas. Es el resultado de la tesis *Micrho, Micromundo para el desarrollo del pensamiento sistémico soportado en el Modelamiento basado en Objetos y Reglas* elaborado por Amaury Cabarcas, y Néstor Díaz en el 2000. Es una capa gráfica multimedia en la cual se crea un micromundo respaldado por uno o más modelos BOR realizados con Homos.

Como tal es independiente de Homos ya que incluye en sí mismo el código principal del simulador, adaptado a sus propias necesidades. Los modelos creados en Homos pueden ejecutarse en Micrho, restando al desarrollador los demás aspectos y elementos necesarios para completar el micromundo.

Aparece consignado en el CdPS en ... \Instaladores MicrhoAlfa. No se encuentra disponible para la descarga en la página web del GS.

- Editor de caminos

Esta es una aplicación creada durante el desarrollo del proyecto Micrho, su finalidad es facilitar el proceso de construcción de micromundos. Se encuentra en el CdPS en ... \Prototipo Final Micrho\Editor

- MICRHO 2.0

Segunda versión de un micromundo fundado en el modelado basado en objetos y reglas. Desarrollado en la tesis *Michro 2.0, Micromundo para el Desarrollo del Pensamiento Sistémico Soportado en el Modelamiento Basado en Objetos y Reglas*³⁰ elaborado por John Fredy Ortiz Ramos y Diana Carolina Suárez Celis en 2008. Es una aplicación independiente de Homos que implementa una capa con la cual se hace posible el uso de los modelos de Homos como micromundos, con toda la carga administrativa y de control requerida para ese proceso. Aporta experiencia de uso e ideas de diseño, sugerencias sobre funcionalidades por realizar y varias recomendaciones a considerar por este proyecto. No se encuentra disponible para la descarga en la página web del GS.

A.3.6. Fallas implícitas en el software

No todas las fallas encontradas en los programas son causadas por la *falta* de dedicación de sus desarrolladores, también existen los problemas heredados de construcciones previas. Los compiladores son herramientas software que requieren mantenimiento y actualizaciones. Podemos mencionar que el propio VCL de Delphi 3³¹ tiene sus debilidades, errores e inconsistencias, las cuales se trasladan al código compilado y que no tienen nada que ver con la estructura interna o

³⁰[Sic] Se supone se refiere a la segunda versión de Micrho, situación causada por el nombre del título del proyecto de grado en el que fue desarrollado.

³¹Ver <http://smartbear.com/products/development-tools/performance-profiling/leaks/leaksd3/>

los algoritmos usados para el desarrollo de la aplicación. Homos no es la excepción, parte de las fallas del manejo de GDI podrían ser causadas por estas herencias. Ver la tabla A.6 en la página siguiente

También pueden encontrarse en AutomatedQA otros listados de errores de memoria presentes para otras plataformas: Microsoft MFC, Visual C++ 6 y Visual Studio .Net Borland VCL versiones 3, 4, 5, 6, 7. Borland C++ Builder versiones 3, 4, 5, 6. No aparecen las últimas versiones de estas aplicaciones, pero es presumible que también sean agregadas en estas listas en el futuro.

Existe información adicional sobre esta temática en Internet, como ejemplo la sección "*Delphi Fixups, Patches & Updates*:" de Richey's DELPHI-BOX <http://www.inner-smile.com/delhiph.phtml> y directamente con el fabricante en el Delphi Developer Support de Borland [mediante WaybackMachine³²]:

<http://web.archive.org/web/20020623091620/http://info.borland.com/devsupport/delphi/fixes/3update/index.html>

La cual contiene información relevante sobre Delphi 3. En la época en la cual se concluyó el proyecto Homos no existía esta información, así que se asume que no se han incorporado y compilado dichos cambios.

A.3.7. Herramientas de evaluación de métricas usadas en este documento

Actualmente se encuentran en el mercado herramientas gratuitas y comerciales que realizan tareas de análisis, depuración de código fuente, binarios y demás actividades relacionadas con el desarrollo de aplicaciones para programadores en Delphi. Algunas de ellas se usaron para obtener información sobre el código fuente de Homos 1.0, notando que de las herramientas de pago listadas se aplicaron en sus versiones demo o shareware³³ disponibles. Ver tabla A.7 en la página 143

Algunas de estas herramientas o sus funcionalidades ya están incorporadas en las últimas versiones de Delphi, lo que ahorra costos, facilita el trabajo de depuración de errores y una mejor construcción de software.

Durante el desarrollo del proyecto Homos 1.0 se usó la herramienta WithClass en versiones 95 y 98 de Microgold para generar los diagramas de clases. La ausencia de actividades de medida en este software se comprende mejor si se observa que Delphi 3 no cuenta con herramientas integradas para reportar métricas, diagramas de clase o ejecutar tareas de análisis del código fuente.

Para realizar estas labores se requería de herramientas externas de análisis con las que no se contaba en ese tiempo.

³²Un servicio de registro histórico provisto por <http://www.archive.org/>

³³Software que generalmente ofrece funcionalidad completa pero por pocos días, luego de los cuales es necesario registrar el producto o desinstalar la aplicación.

Tabla A.6: Fallas conocidas en la VCL de Delphi 3

Known Delphi 3 VCL Memory Leaks

Archivos	Procedimiento	Descripción	Versión
controls.pas	ThintWindow.WMNCPaint	ThintWindow does not free a Device Context in it's WM_NCPAINT handler.	D4
forms.pas	UnhookHintHooks New procedure added	UnhookHintHooks does not close the handle of the HintThread MakeObjectInstance does not free a chunk of 4096 bytes.	D4, ND4
printers.pas	Tprinter.Destroy	After a call to SetPrinter, the Tprinter class forgets to free the DeviceMode global memory handle.	D4
checklist.pas	New procedure added	TcheckListBox does not free TcheckListBoxDataWrapper classes allocated for each checked item in a list box	D4
classes.pas	Unit finalization	When creating a thread, a window class - ThreadWindow - is created but never destroyed	D4
comctrls.pas	TrichEditStrings.LoadFromFile TrichEditStrings.SaveToFile	The TrichEditStrings fail to free the Fconverter class for every call to LoadFromFile and SaveToFile	D4
registry.pas	Tregistry.CloseKey	If LazyWrite = False, the Tregistry class does not close the CurrentKey	ND4
graphics.pas	Tbitmap.SetPixelFormat Tbitmap.ReadDIB(5486) Tbitmap.ReadDIB(5507)	For 8 bit Bitmaps, a palette is not freed. Tbitmap.ReadDIB when used for 8bit bitmaps leaks a bitmap Handle Memory overrun with certain types of DIBs	D4, D4.02, D4

Delphi 3 Demos Leaks

ipcthrd.pas	TIPCThread.Destroy TclientDirectory.Destroy	TIPCThread does not free FconnectEvent TclientDirectory does not free a palette – Tmutex class	NA
rxmain.pas	TmainForm.FormDestroy	The main form does not free the Splitter class	NA

NA = No arreglado

D4 = Resuelto en Delphi 4

ND4 = No arreglado en Delphi 4, solucionado en una versión posterior.

Fuente: Autor.

Tabla A.7: Herramientas de análisis y evaluación de código Pascal / Delphi

Software	Descripción	Página fabricante
Understand for Delphi Versión 1.4 Build 340 by Scientific Toolworks Inc. [STI] 1997-2006	Analizador de código, métricas y reporte de estado del código.	http://scitools.com/products/ understand/delphi/product.php
Pascal Analyzer 3.0 e Icarus de Peganza	Analizadores de código, métricas, reportes del estado del código.	www.peganza.com
Visustin 3.06 de Aivosto	Creador de diagramas de flujo.	www.aivosto.com/visustin.html
Eyebol Delphi Code Analyzer 1.30d de Eyebol	Analizador de código fuente, buscador de errores lógicos.	www.eyebol.com
Aqtime 3.16 Y 4.9 de AutomatedQA	Analizador de código fuente, buscador de errores lógicos.	www.automatedqa.com
ModelMaker 8.4 de ModelMaker Tools BV	Herramienta de modelado UML, ingeniería inversa, métricas, búsqueda de errores, integrable al IDE de Delphi.	www.modelmakertools.com
ESS-Model 2.2 de ELDEAN	Herramienta de código abierto gratuita, convierte código a diagramas de clase UML y documentación Html.	www.essmodel.com www.eldean.se/essmodel www.sourceforge.net/projects/essmodel
WithClass 8.5 de Microgold	Herramienta de modelado UML.	www.microgold.com
PE Explorer V1.99 R3 de Heaventools	Herramienta avanzada de análisis de binarios, desensamblador.	www.pe-explorer.com
ProDelphi V19.0 for Win32	Profiler, acelerador de aplicaciones. Facilita la eliminación de “ <i>cuellos de botella</i> ” en las aplicaciones.	www.prodelphi.de/indexpd.htm
Process Explorer 10.02	Utilidad de monitoreo de aplicaciones para Windows.	www.sysinternals.com http://technet.microsoft.com/ en-us/sysinternals/bb896653
MemProof 0.9.5.0	Heap memory and resource “leaks” debugger.	www.automatedqa.com

Fuente: Autor.

A.3.8. Resultados obtenidos

En general, se observa una falta de seguimiento en cuanto al desarrollo del software en varios niveles. Entre los elementos encontrados se destaca que:

- Se requieren labores de mantenimiento para el código fuente de la herramienta software Homos 1.0.
- Algunas de las funcionalidades que caracterizan la simulación tienen problemas en su funcionamiento. Se requiere de mantenimiento al sistema de simulación de la aplicación para corregir las distintas fallas encontradas.
- Existen elementos en la interfaz gráfica de la aplicación sobre los cuales es posible realizar cambios en su funcionamiento y estructura³⁴.
- Plantear nuevas reglas de transición con el código fuente que implementa a Homos 1.0 implica cambiar la forma en la cual funciona internamente el simulador. En otras palabras, en el caso de Homos hay necesidad de reconstruir la aplicación para implementar nuevas formas de transición.

Aunque hay una clara tarea de mantenimiento con respecto al software Homos 1.0, el autor de este documento considera que NO es viable la realización de actualizaciones o cambios mayores en el software. En vez de eso, se propone una recreación de Homos 1.0 o la generación de una herramienta completamente nueva en la cual sea posible hacer uso de una generalización práctica de la representación de las reglas de transición y su accionar, el sustento a esta afirmación se describirá en detalle en secciones posteriores en este documento. El problema de la flexibilización de las reglas de transición tiene cota superior, existe un límite para su desarrollo práctico. Pensando en una nueva aplicación, se sugiere:

- Evitar repetir los problemas de diseño encontrados. Una segunda versión del software debe dinamizar la construcción de reglas con una forma de construcción interna diferente, flexible.
 - Diseñar el software teniendo en cuenta grandes cantidades de elementos. Tener en cuenta que es posible esperar el uso de grandes cantidades de cualquiera de los recursos disponibles, especialmente de objetos en el ambiente.
 - Diseñar el software de tal forma que exista una clara separación entre capas, esto es, independencia de la interfaz gráfica del estado interno de la aplicación.
 - Hay necesidad de reducir / unificar la interfaz gráfica que permite la construcción de las distintas estructuras de representación. Se sugiere la implementación de una única interfaz para la creación y edición de reglas de transición.
 - Se requieren ajustes en el funcionamiento interno del proceso de simulación descrito por el algoritmo matemático para hacerlo más flexible y que permita cambios. El autor de este documento apuesta por una refundación total del sistema de transiciones, más simple que la implementación actual.
- Se requiere aleatoriedad de acceso y soporte a múltiples estructuras en el formato de almacenamiento de la aplicación. Se sugiere un formato abierto para la descripción de

³⁴Esto se complementa con los resultados presentados en el análisis de la interfaz gráfica en la sección A.4.5 en la página 171.

los contenidos de los modelos, que permita la transparencia en las definiciones planteadas en archivos y que permita cargar algunos elementos seleccionados del modelo.

- Implementar un mecanismo de manejo de versiones entre archivos de proyecto. Es factible suponer que una siguiente version del software pueda manipular y hacer uso de la estructura de datos establecida en versiones anteriores.

A.4. Análisis de Interfaz gráfica

Este apartado trata acerca de los diferentes utiles previstos en la interfaz grafica de herramienta software y su integracion sinergica para el logro de la creacion efectiva de modelos fundados en el paradigma de transiciones planteado en la MBOR. El analisis de la interfaz grafica de Homos 1.0 se realizará en tres niveles distintos:

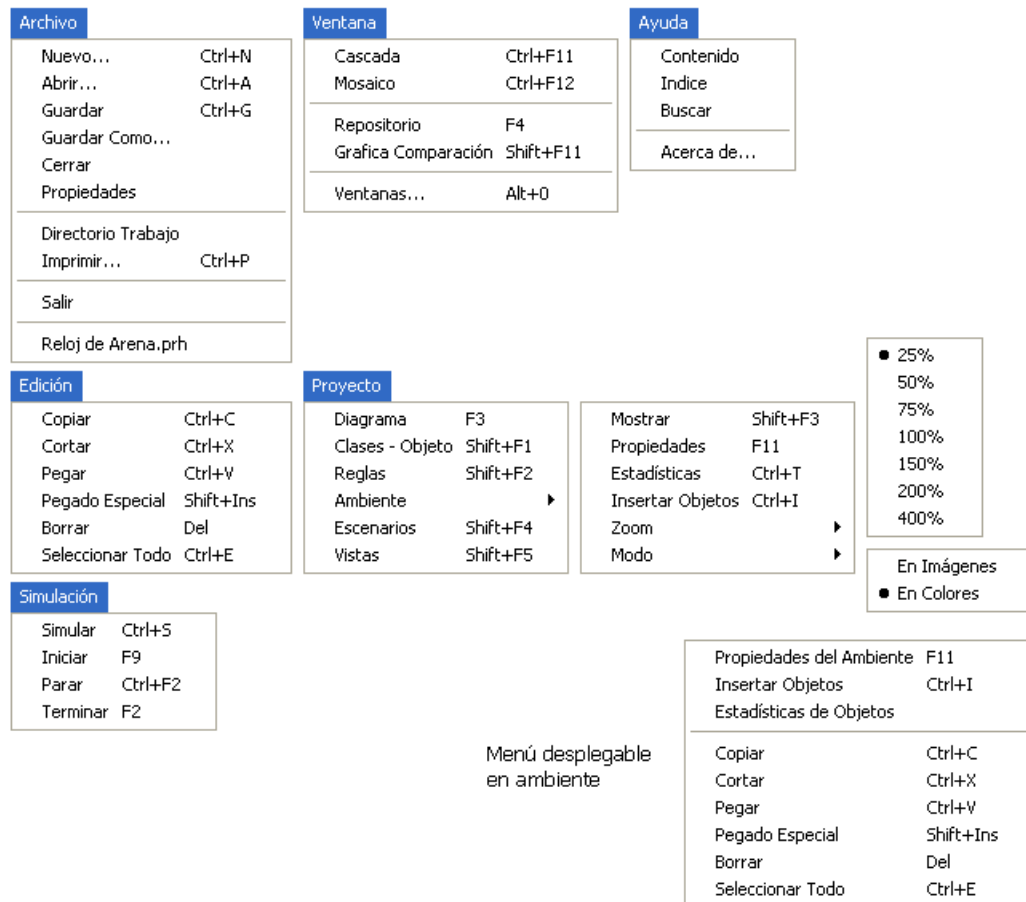
- Descripción de elementos en interfaz: El desarrollo de este analisis no busca repetir elementos ya explicados en la trabajo de grado o el manual de Homos 1.0. Se pretende encontrar e integrar los elementos descriptivos faltantes a que haya lugar con el propósito de complementar la información prevista en los documentos anteriormente citados.
- Interfaz y secuencia de operaciones de modelado: Este apartado busca integrar los procesos de modelado con las distintas facilidades presentes en la interfaz gráfica.
- Iconos en interfaz grafica: Pretende establecer elementos con los cuales mejorar y facilitar la representación grafica de clases-objeto y demas elementos propios del modelado grafico en la herramienta software.

La sinergia entre los aspectos analizados permite establecer pautas que permitan mejorar el aspecto grafico del programa, las distintas funciones ofrecidas y la fluidez de los pasos necesarios para el desarrollo de modelos en la herramienta software.

A.4.1. Elementos componentes

La interfaz gráfica de Homos 1.0 consta principalmente de un menú principal y un grupo de barras de herramientas con botones de operaciones, ventanas y cuadros de diálogo especializados. El diseño de la interfaz gráfica de la aplicación sigue los estándares propios de los programas Windows, en cuanto a colores, formas de los controles, tipos de letra y demás características asociadas, en concordancia con los elementos establecidos en la tesis de Homos 1.0.

Figura A.7: Estructura del menú principal de la aplicación



Fuente: Autor.

A.4.2. Descripción de los elementos que conforman la Interfaz

Las ventanas de ambiente, el diagrama BOR y vistas son de tipo MDI³⁵ maximizadas, Las demás ventanas y cuadros de diálogo son modales. La interacción con la herramienta es visual y los comandos se imparten principalmente con el ratón y atajos de teclado.

³⁵Multiple document interface

A.4.2.1. Menú principal

Da acceso a todas las funciones implementadas en Homos.

Facilidades:

- Permite crear, abrir, guardar, imprimir y cerrar proyectos.
- Permite establecer el directorio de trabajo de la aplicación.
- Presenta y da acceso al último proyecto abierto por la aplicación.
- Permite la edición del ambiente mediante las opciones de copiar, cortar, pegar el contenido de las celdas y la selección de todas las celdas.
- Permite el acceso a las ventanas de diagrama, clases objeto, reglas, ambiente, escenario y vistas.
- Presenta las opciones de inicio, pausa y terminación de la simulación.
- Permite la organización de ventanas en cascada o mosaico y presenta el listado de ventanas activas.
- Presenta el repositorio, las gráficas de comparación, las ayudas incorporadas en la herramienta y los créditos.
- Da acceso a las propiedades del proyecto, que es una breve documentación que consta de una hoja de resumen y una de estadísticas.
 - En la hoja resumen se consigna el título del proyecto, el dato de los nombres de autor(es), director de proyecto, institución, fecha de creación del documento y comentarios. Consignar esta información sobre el modelo es opcional.
 - En estadísticas se presenta el número de clases, reglas y escenarios creados en el proyecto.
 - Permite asignar una contraseña de seguridad para el proyecto. Esta se hará permanente luego de salvar el documento. No es posible acceder al contenido de un proyecto si no se introduce la clave correcta, este comportamiento es funcional para la aplicación principal y para el visor de Homos.

A.4.2.2. Barras de herramientas

Operan sobre el ambiente, poseen una movilidad mínima y están organizadas según su función:

- **Barra de archivo:** Permite crear, abrir, guardar, cerrar e imprimir proyectos de Homos. Es la única presente y activa desde el inicio de la aplicación.
- **Barra de proyecto:** Da acceso a las ventanas de diagrama BOR, manejo de clases objeto y de reglas, el ambiente, los escenarios y las vistas. Permite hacer visible la barra de simulación. Se activa al abrir exitosamente un proyecto.

- **Barra de edición de ambiente:** Permite el acceso a las opciones de edición de propiedades del ambiente, inserción de objetos, presentación de estadísticas, copiar, cortar y pegar dentro del ambiente, la reflexión y rotación de elementos previamente seleccionados. Esta y las barras descritas a continuación aparecen incorporadas dentro de la ventana ambiente.
- **Barra de zoom y modo:** Permite modificar el zoom y elegir el modo gráfico de los objetos en el ambiente
- **Barra de clases objeto:** Contiene el puntero de selección y la lista de clases objeto definidas disponibles a para insertar en el ambiente.
- **Barra de posición:** Presenta en pantalla la posición de la celda sobre la que esta ubicado el ratón dentro del ambiente.

A.4.2.3. Barra de flotante de simulación

Es la única barra de herramientas no estándar de la aplicación, ya que aparece anclada en la parte superior derecha de la ventana principal. Permite ejecutar, pausar y terminar la simulación. Presenta el estado actual del simulador y una barra de progreso que indica cuantos pasos de simulación han sido calculados. Esta barra no es reubicable pero es posible ocultarla mediante un botón ubicado en su extremo izquierdo. Aparece cuando se inicia la simulación, al usar el elemento de menú o el botón asignado en la barra de herramientas para ese fin.

A.4.2.4. El Ambiente

Es una ventana que presenta el lugar en el espacio en el cual se lleva acabo la simulación.

Características:

- Tiene forma rectangular, con tamaño mínimo de 1x1 hasta un máximo de 256 x 256 celdas cuadradas.
- Contiene barras de herramientas propias para las labores de edición del ambiente.

Facilidades:

- Cambiar el tamaño del ambiente especificando el número de filas y columnas.
- Asignación del tipo de limite a usar en el ambiente, cíclico, finito e infinito, según lo definido para el espacio dentro de los autómatas celulares.
- Cuenta con un Zoom entre un 25 % y un 400 % con opción de modo de presentación de objetos en colores o imágenes.
- Permite establecer el color de fondo de las celdas del ambiente.
- Permite la inserción de objetos, hasta el máximo de la capacidad asignada.

- Permite la inclusión de instancias de clases objeto distribuidos en forma agrupada, aleatoria y uniforme.
- Permite cortar, copiar y pegar objetos de áreas seleccionadas.
- Permite rotar a 90, 180 y 270 grados, hacer reflexiones verticales y horizontales sobre áreas seleccionadas.
- Con el botón “*estadística de objetos*” es posible ver su cantidad de instancias por cada clase, el número de celdas vacías y el total de objetos incluidos en el área de selección activa dentro del ambiente. Si no existe una selección activa, se calculan estos valores para todo el espacio.

Debilidades:

- Permite ubicar objetos en forma personalizada solo mediante el uso del ratón.

A.4.2.5. Los Escenarios

Permiten establecer las condiciones de simulación del proyecto. Sus datos se introducen en el programa mediante un cuadro de diálogo creado para tal fin. Pueden definirse tantos escenarios como sean necesarios para mostrar la versatilidad o restricciones propias del modelo.

Características:

- Permite establecer el número de pasos de simulación a usar durante la ejecución del modelo. El valor máximo permitido es $\text{High(Integer)} = 2^{31} - 1 = 12147483647$.
- Permite establecer variaciones particulares al valor de probabilidad de ejecución de las reglas, cuya finalidad es exhibir las posibles alteraciones en el comportamiento del modelo ante situaciones específicas.
- Permite la creación y eliminación de vistas.

Facilidades:

- Establece el zoom y el modo de presentación de objetos por defecto para el ambiente.
- Permite guardar los datos generados por la ejecución del modelo en un archivo previamente especificado en formato texto³⁶.
- Permite la adición automática de nuevos objetos distribuyéndolos en forma agrupada, aleatoria o uniforme en el ambiente.

El proceso de simulación requiere que previamente se determine cual de los escenarios listados es el escenario activo.

³⁶Ver recursos y materiales, en la sección A.2 en la página 105

A.4.2.6. Las Vistas

Son ventanas donde se presentan, mediante curvas en el plano, los comportamientos o cambios poblacionales de las clases objeto existentes en el modelo. El desarrollador puede definir tantas vistas como sean necesarias para representar uno o mas aspectos particulares del comportamiento del modelo en un escenario especifico. Cada gráfica tiene su propia tabla de datos, generadas a partir de una única tabla de datos interna, calculada durante la simulación.

Características:

- El eje X representa el tiempo de simulación, limitado por el número de pasos determinados por el escenario activo.
- El eje Y maneja las cantidades poblacionales de las diferentes clases objeto presentes en el modelo.
- Cada vista es única, las vistas son exclusivas del escenario donde fueron creadas.

Facilidades:

- Permite establecer títulos para las gráficas, el manejo de las escalas de los ejes, la presentación de convenciones y la cuadrícula del plano.
- Permite ver la tabla de datos asociada con la gráfica y la impresión directa de la gráfica activa.

A.4.2.7. Tabla de datos

Es una tabla que presenta por cada paso de simulación la cantidad de instancias que hay por cada clase objeto presente en el modelo.

Características:

Presenta en forma tabulada los valores poblacionales actuales de simulación. Los valores de paso de simulación identifican las filas, las columnas están compuestas por los valores poblacionales de las clases objeto seleccionadas en la vista a la que la tabla pertenece.

Facilidades:

Permite salvar los datos tabulados en un archivo de texto³⁷.

A.4.2.8. Gráficas de comparación

Son una utilidad incorporada en la herramienta que permite visualizar en forma gráfica las diferencias entre los comportamientos poblacionales resultado de distintas sesiones de simulación del modelo. Estas gráficas tienen mejores y mas completas opciones de presentación que las vistas porque son una instancia funcional del objeto Teechart 3.0 , que es parte de las librerías gráficas de Delphi 3.

³⁷Ver recursos y materiales, sección A.2.2.4 en la página 108.

Características:

Mismas características de presentación de información que las observadas en las vistas: el eje X maneja el tiempo de simulación, el eje Y maneja valores poblacionales. Difieren de las vistas en que son gráficas independientes de los escenarios o cualquier otro elemento del proyecto. Si se cuenta con datos de simulaciones, es posible presentarlos sin necesidad de tener abierto el proyecto origen de los datos. Los cambios en la configuración de las gráficas de comparación no se incluyen dentro del modelo de información propio del archivo de proyecto.

Facilidades en el manejo de gráficas:

- Permite abrir los archivos de texto generados por la opción guardar de la tabla de datos, con las columnas de datos obtenidas permite agregar o eliminar las series que serán presentadas en la gráfica de comparación activa³⁸.
- Permite adicionar, duplicar, eliminar o cambiar el orden de presentación de las series así como establecer el tipo de gráfico a usar en varios estilos: usando líneas estándar y rápida, mediante barras horizontales y verticales, usando áreas, por puntos, usando diagrama de porcentajes, burbujas, diagrama de Gant, campos vectoriales y usando figuras regulares. Todas estas opciones gráficas son dibujables en forma 3D.
- Cuenta con facilidades para enviar la imagen al portapapeles o exportar la imagen en varios formatos gráficos: *.bmp, *.wmf o *.emf.
- Permite la configuración de la profundidad lateral de las vistas 3d.
- Permite establecer los colores, el fondo, el borde, el relleno y el ancho de las paredes laterales del gráfico, observables cuando se presentan en 3D.
- Permite configurar los colores de fondo, el estilo de los bordes, la alineación, fuentes y visibilidad del texto del título de la gráfica.
- Permite establecer el estilo, el contenido texto, los colores, la fuente, la visibilidad, el tamaño, la forma, las texturas de las líneas de división, las sombras y la ubicación relativa respecto de la gráfica de las leyendas de las series presentadas.
- Permite usar scroll (movimiento lateral) de la gráfica en dirección a los ejes.
- Incluye un zoom animado de inmersión a zonas rectangulares definidas mediante el ratón dentro del área gráfica.
- Permite configurar el aumento de los detalles de la gráfica.
- Permite establecer las formas de biseles interiores, exteriores, el ancho, la imagen, colores y rellenos del fondo del gráfico.
- Permite establecer márgenes laterales en valores de porcentaje y la visibilidad de las líneas de márgenes.
- Permite la paginación de la imagen y su escalado, incluyendo la impresión por secciones de la gráfica.

³⁸Ver recursos y materiales, sección A.2.2.4 en la página 108.

- Permite la presentación preliminar del gráfico, el acceso a las propiedades de la impresora y la elección de la orientación del papel a usar durante la impresión.
- Permite configurar las etiquetas de los ejes X y Y en varios estilos y con formatos de texto, además de configurar las marcas de eje y sus longitudes.
- Permite la configuración, ubicación y formato de presentación de ejes, su visibilidad y el manejo de escalas incluyendo vistas logarítmica e invertida de valores en el eje Y.

Facilidades en el manejo de series:

- Permite la edición de propiedades de series individuales, manejando los formatos texto y gráficos de los datos de la serie.
- Permite establecer punteros de ratón distintivos sobre las series de datos.
- Permite la elección de ubicaciones para la presentación de los textos de los ejes X y Y.
- Permite configurar el formato gráfico de marcas: su visibilidad, forma geométrica y varios formatos para la presentación de sus valores texto.
- Permite realizar operaciones con la fuente de datos: reemplazarlos por valores aleatorios, eliminar los datos o usar funciones sobre los datos. Esta última opción se encuentra deshabilitada, durante las pruebas de este análisis no se encontró la forma de acceder a dicha funcionalidad.

Estas son las facilidades accesibles directamente desde el gráfico de comparación. Cabe anotar que este listado no aparece descrito en el manual de usuario o en la tesis de Homos. Para conocer más detalles sobre este objeto, ver: *Manual de desarrollador con teechart 3.0* incluido en la carpeta teechart de la instalación de Delphi 3.

A.4.2.9. El Diagrama BOR

Esta ventana presenta las relaciones presentes entre las clases objeto y las reglas establecidas en el modelo.

Características:

En el lado izquierdo se muestra la estructura gráfica del modelo; en el lado derecho se presenta el árbol de proyecto, el cual lista las clases del modelo y la definición de las reglas: nombre, tipo y clases objeto a los que ellas se apliquen. Dentro del diagrama, la forma de las conexiones y sus colores obedecen a los elementos de modelado establecidos en la tesis de Homos 1.0.

Facilidades:

- Creación de clases objeto y reglas mediante click derecho sobre el fondo del área gráfica del diagrama.
- Edición y eliminación de las clases objeto mediante click derecho del ratón sobre el elemento gráfico que los representa en el diagrama.

- Edición de Reglas mediante doble click derecho sobre el nombre de la regla en el árbol de proyecto.

Desventajas:

- La reubicación de objetos en el diagrama solo es posible mediante el uso del ratón.

Elementos establecidos en la notación gráfica como las diferencias entre líneas simples y dobles; la presentación del nombre, probabilidad y prioridad de la regla como parte de su descripción y las esquinas redondeadas para las clases, no hacen parte de la implementación actual del diagrama BOR en la herramienta. Se profundiza sobre la naturaleza del diagrama BOR en la sección A.7 en la página 272.

A.4.2.10. Sobre los cuadros de diálogo

Se encargan de la recepción y filtrado de los datos del programa.

Características

- No admiten la repetición de identificadores. Se comprueba que cada nombre de regla, clase objeto, escenario, vista y gráfica a usar en el programa es único.
- Se valida que el formato, el tipo y el intervalo del dato introducido sea apropiado para cada campo numérico de la aplicación; se usa la coma como separador de punto decimal.
- Cualquier desviación de los formatos establecidos es alertada al usuario con un mensaje apropiado para cada caso, permaneciendo en ese estado hasta que se realicen los ajustes necesarios.
- Presenta un botón de ayuda junto al botón de cierre de la ventana. Sirven de guía rápida sobre los elementos presentes en el cuadro de diálogo activo.

A.4.2.11. Cuadro de diálogo de clases objeto

Permite la edición de las propiedades de las clases objeto definidas para el modelo

Características:

Permite establecer el nombre de clase, la imagen y color de identificación junto con una breve descripción de su objetivo en el modelo.

Facilidades:

- Permite Crear, modificar y eliminar clases objeto.
- Presenta el nombre y la imagen distintiva de todos los objetos actualmente integrados en el modelo.
- Permite enviar y traer clases objeto del repositorio. En esta operación, es opcional el envío y la recuperación de las reglas que estén relacionadas con dichas clases al repositorio.

A.4.2.12. Cuadro de diálogo de reglas

Permite la edición de las reglas definidas para el modelo. El cuadro está compuesto por los campos necesarios para asignar las propiedades particulares a cada tipo de reglas presente en la aplicación.

Características:

Todas las reglas tienen en común los siguientes parámetros:

- Nombre de la clase objeto a la que se aplica la regla o *primera clase*.
- La probabilidad de ejecución, comprendida entre 0 % y 100 %.
- La prioridad de aplicación de la regla, comprendida entre 0 y 255.
- Un campo de texto que sirve para dar una breve descripción sobre la utilidad de la regla.

La regla de decadencia esta completamente satisfecha con estos parámetros. Esta regla no permite la edición del campo de prioridad porque su valor es constante y es igual a cero.

Las reglas que tienen componente en el espacio adicionan a las anteriores:

- Dirección en el espacio: conjunto de 8 casillas de marcado que establecen las direcciones de movimiento.

Este parámetro hace parte de la definición de las reglas de Expansión, Retracción y Movimiento, adicionando en esta última el número de celdas a desplazarse.

Las reglas que tienen componente de influencia sobre otras clases adicionan a las anteriores:

- La clase objetivo o *segunda clase*: es la clase con la cual se tiene interacción y sobre la que se aplican los efectos de la regla

Hace parte de las reglas de Alteración, Eliminación, Neutralización, Transformación y Reproducción.

La regla de reproducción incluye un parámetro adicional llamada *clase resultante* que representa a una tercera clase, efecto de la aplicación de dicha regla.

Facilidades:

- Permite crear, modificar y eliminar reglas, además de enviar y traer reglas del repositorio.
- Presenta el listado de las reglas actualmente integradas en el modelo.

Al hacer doble click sobre la lista de reglas disponibles del modelo pasa a la edición de sus parámetros.

En los ejemplos contenidos por el manual y en algunos los proyectos de ejemplo, se sugiere la notación "*Nombre de la clase objeto.función de la regla*" para los nombres de las reglas. Esta forma de notación no es alentada en el software y generalmente no es la estándar entre sus usuarios modeladores. Lo común es usar el verbo o el nombre de la acción que representa esa regla.

Debilidades:

Los distintos cuadros de diálogo de reglas no hacen evaluación de las clases objeto relacionadas por la regla. Por ejemplo es posible aplicar una regla de transformación de una clase a si misma, así como que una clase se altere en si misma. En ambos casos significa que la clase no cambia al aplicar la regla. Mas que una debilidad es una flexibilidad inicialmente no deseada.

A.4.2.13. El Repositorio

Es una facilidad del software que permite la reutilización de clases objetos y reglas en los modelos. Pretende ahorrar trabajo al modelador, al permitir guardar parte de las estructuras de los modelos que se usen con mayor frecuencia.

Características:

- Los datos del repositorio son persistentes, no están asociados a los proyectos creados por la herramienta³⁹.
- Durante la ejecución del programa, almacena las clases objetos y las reglas asociadas a ellos en memoria, cada elemento está presentado en hojas separadas dentro del cuadro de diálogo de repositorio.
- Para almacenar una regla, las clases objeto a las que están relacionadas deben existir previamente en el repositorio.

Facilidades:

- Presenta el listado de las clases objeto y las reglas contenidas en el repositorio
- Permite eliminar clases y reglas en el repositorio usando el cuadro de diálogo de repositorio.
- En los cuadros de diálogo de clase objeto y reglas se realizan las operaciones de enviar y traer elementos al repositorio.

En el manual aparece asociado con el tema del manejo de reglas, pero no se hace énfasis en ese documento sobre su utilidad práctica. Esta funcionalidad es generalmente es ignorada por los modeladores.

Por razones de diseño, no es posible saber a priori que reglas están asociadas a una clase determinada en el repositorio; se hace necesario el uso de una notación como *“Nombre de la clase objeto.Función de la regla”* en las reglas para ver estas asociaciones.

³⁹Ver recursos y materiales, sección A.2 en la página 105.

A.4.3. Interfaz gráfica y secuencia de operaciones en modelado con MBOR

A.4.3.1. Descripción general del proceso de construcción de modelos en Homos 1.0

Para un modelador el trabajo con la herramienta está compuesto por una fase conceptual y cuatro fases aplicadas, cada una caracterizada por el tipo de acciones que se realizan sobre el modelo.

La secuencia de operaciones del proceso se ilustra en figura A.8 en la página siguiente. En orden normal de aparición, estas fases son:

Idea y Conceptualización, Fase de desarrollo conceptual del modelo: Comprende las distintas operaciones relacionadas con la captura de elementos de la realidad y su conversión en conceptos tangibles, tales como los resultados escritos de la observación del fenómeno, la creación de un modelo prosa asociado⁴⁰.

Debido a la naturaleza *abstracta* de estos desarrollos, en general no hay soporte en software para este tipo de operaciones. En la MBOR y la actual herramienta, no existe soporte para la generación o apoyo a la creación *mecanizada* de modelos prosa, una guía de patrones de modelos⁴¹, procesos de conversión automática entre las descripciones presentes del modelo prosa en elementos de la metodología BOR, entre otras posibles operaciones. Por esta razón no se contempla⁴² como una fase explícita dentro del proceso de construcción con los útiles previstos por la herramienta software.

La génesis de la idea y la conceptualización de un fenómeno en un modelo y sus aplicaciones derivadas, obedece generalmente a intereses creados o particulares sobre el fenómeno a representar.

Fase de edición del modelo: Comprende el uso de las facilidades de creación y edición de elementos componentes del modelo en la herramienta. Esta fase configura el inicio de un proceso de *prototipado continuo*⁴³, especialmente cuando se está desarrollando un modelo desde cero.

Fase de simulación y adaptación: Consta de uno o más ciclos de modificación del modelo que involucran la generación y evaluación de paisajes dinámicos de prueba, con los cuales el modelador corrobora el *ajuste* de la estructura implementada como modelo con respecto al comportamiento proyectado o deseado.

Fase de reportes y resultados: Parte final de proceso de construcción del modelo. Consiste en la recuperación de los distintos datos obtenidos durante la simulación para su posterior análisis y estudio. Generalmente los desarrolladores y modeladores son los encargados de la realización de las distintas fases ya presentadas.

Es una labor adicional de los creadores del modelo, el orientar el resultado final hacia los

⁴⁰Esto último según la metodología BOR. En general se refiere a descripciones, representación formal en alguna notación escrita o gráfica.

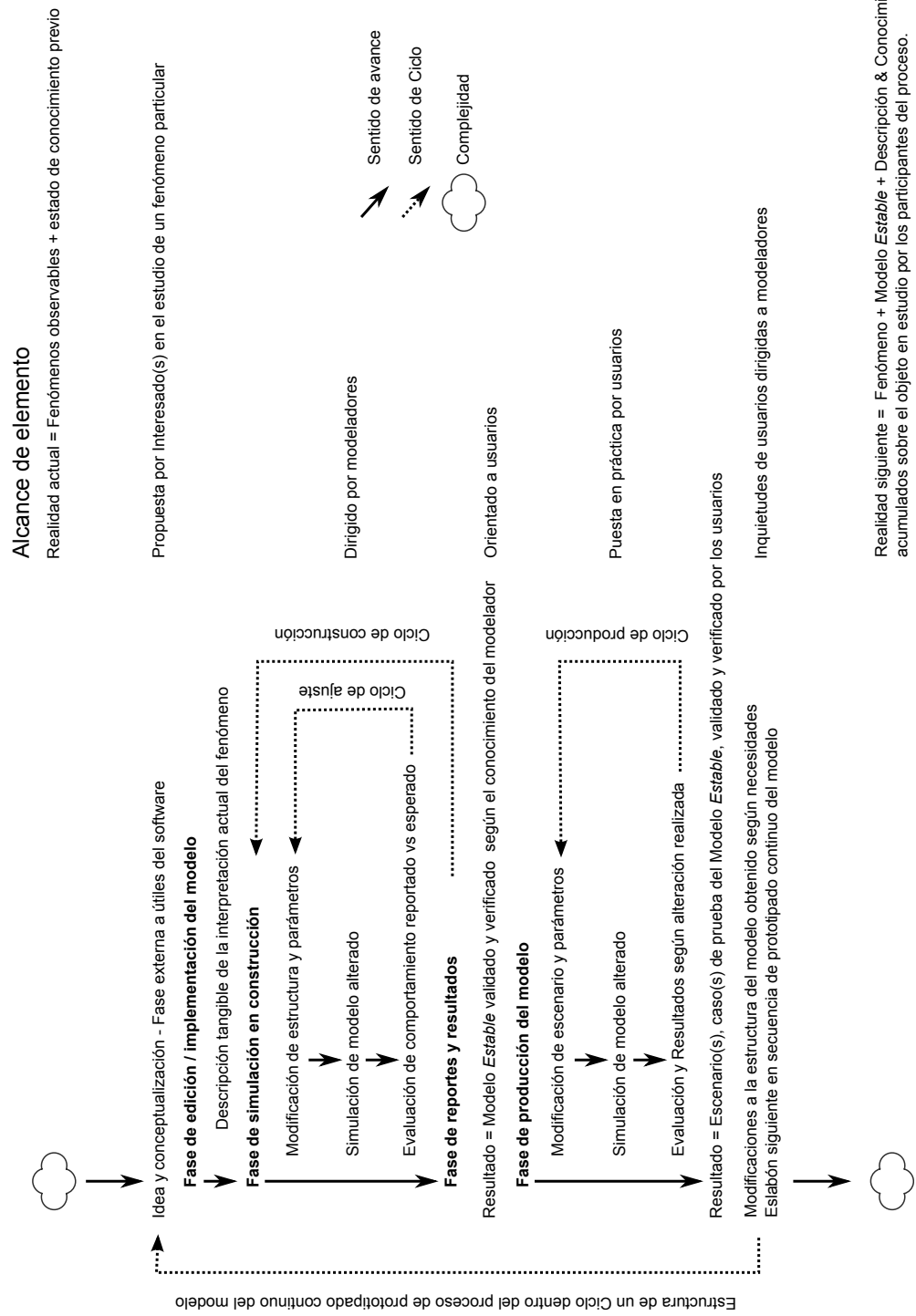
⁴¹Por ejemplo, estructuras como "*Tragedy of commons*" en dinámica de sistemas.

⁴²No se resalta como una fase que pueda ser llevada a cabo o al menos facilitada con los útiles previstos actualmente en el software.

⁴³Véase la figura A.9 en la página 162.

Figura A.8: Estructura de pasos de construcción de un modelo en Homos 1.0

Desarrollo de modelos en Homos 1.0



Fuente: Autor.

usuarios. Para facilitar esta tarea se sugiere el uso de nombres de reglas, clases objeto, escenarios y demás parámetros que faciliten su lectura y no den lugar a confusiones de identificación. También ayuda a la comprensión del modelo la inclusión de detalles relacionados a cada elemento y en las propiedades generales del proyecto, la presentación de varias vistas cuando esto sea posible, la ordenación gráfica de las relaciones y clases objeto presentados en el diagrama BOR.

Al completar en secuencia las fases de simulación y reportes se configura una iteración dentro del ciclo de construcción del modelo. Se llega al final de este proceso cuando se obtiene un modelo ajustado a los lineamientos teóricos o conceptuales deseados por el modelador. En este punto el modelo creado se hace resistente a cambios de estructura, logra una forma estable⁴⁴.

Fase de producción: Se alcanza una vez que se ha conseguido un modelo estable que reproduzca el comportamiento deseado de acuerdo con el nivel de ajuste al fenómeno establecido por el modelador. La acción más frecuente en esta fase será llevada a cabo por los usuarios del modelo, consiste en la edición de parámetros y la creación de uno o más escenarios que permitan visualizar los distintos comportamientos posibles en el modelo. Cada alteración o cambio de los parámetros estará acompañada de la ejecución del modelo y la inmediata evaluación de los resultados obtenidos.

Cada terminación de una secuencia de EDICIÓN → SIMULACIÓN → EVALUACIÓN del modelo estable en esta fase configura un iteración dentro del ciclo de producción o uso del modelo. Se obtiene de estas iteraciones la creación de material documental, uno o más escenarios y la experiencia en la mecánica de realización y evaluación de pruebas funcionales del modelo. Todo esto es paralelo a un incremento en la experiencia de modelado y la práctica de trabajo con la herramienta.

De este proceso también se obtienen diversos elementos que inducen ideas de cambio a largo plazo en la estructura del modelo. Estas observaciones dan pie al siguiente conjunto de acciones de edición que configuran la próxima versión del modelo.

El orden de los pasos presentados en la metodología y el ciclo de vida del modelo obedece a una secuencia lógica de operaciones que tienen como objetivo la construcción efectiva de al menos una versión funcional del modelo. La percepción de las distintas fases y el paso entre etapas es realizado generalmente en forma inconsciente por el modelador del proyecto.

La llegada a cada fase dentro de una iteración del proceso de prototipado del modelo indica un avance concreto en la construcción del modelo. La finalización de un ciclo completo del proceso de modelado establece la creación de una nueva versión del modelo.

El doble ciclo interno presentado dentro del diagrama de la figura A.10 entre FACILIDADES $\Rightarrow$ SIMULACIÓN $\Rightarrow$ REPORTES corresponde con una secuencia de construcción más general, la cual se muestra en la figura A.9 en la página 162.

⁴⁴Es una estabilidad *temporal*. Los conocimientos sobre el fenómeno y los útiles que se usan en su construcción - representación hacen posible la permanencia del modelo.

A.4.3.2. Usuarios y fase de producción de modelo

Un usuario de la aplicación y más apropiadamente de un modelo implementado en la herramienta, realizará inicialmente uno o más ciclos de simulación para explorar y posteriormente verificar⁴⁵ el ajuste del modelo al fenómeno en estudio. Posteriormente hará acciones de edición de parámetros en escenario seguidos de ciclos de simulación y obtención de resultados, con los cuales determinará los cambios que predice el modelo para la configuración particular de parámetros que haya establecido. Véase figura A.9 en la página 162.

Un usuario interesado en el porqué de los resultados obtenidos del modelo querrá ilustrarse más sobre el tema y realizar alteraciones o cambios más profundos, hasta llegar a la estructura misma del modelo, según sean sus necesidades. Al hacer esto, el usuario pasa de observador a agente activo del proceso de modelado, se transforma en modelador.

Se enriquece por este medio mediante un proceso completamente autoreflexivo y autodirigido, la experiencia de trabajo del usuario con el modelo y con la herramienta. Es también raíz para un nuevo eslabón dentro del ciclo de prototipado continuo del modelo y a nivel del usuario, punto de partida para procesos automotivados de investigación.

A.4.3.3. Descripción del proceso de prototipado continuo de un modelo

La figura A.9 contiene una descripción gráfica sobre las acciones y los cambios en el estado de las ideas sobre el fenómeno que se dan en cada ciclo durante la construcción de un modelo.

Antes de iniciar el proceso, existe un estado de conocimiento previo sobre el fenómeno a representar. El modelador tiene algunas ideas o preconceptos y dependiendo de la cantidad de información y experiencia en el tema, se hará mas fácil o no su implementación durante las distintas fases.

En la gráfica el punto negro identifica el origen del proceso, la necesidad de representación de un aspecto de la realidad para su análisis. A partir de este surge una flecha que indica el sentido de avance y la secuencia de acciones de edición o modificación sobre el modelo. Sobre estas acciones, se presentan dos ciclos claramente diferenciados por la naturaleza de sus acciones, el ciclo de construcción y de producción del modelo. Al completar exitosamente estos ciclos en secuencia, se ha terminado una iteración dentro del ciclo de vida del modelo.

Durante el ciclo de construcción se realizan las distintas operaciones que construyen al modelo mediante aproximaciones sucesivas [Ciclos de ajuste]. Con la aplicación de estos ciclos de ajuste se obtiene una estructura estable e información sobre la representación del fenómeno con la cual se operará en el siguiente ciclo, donde se dará la producción o uso del modelo. En esta última etapa existe un ciclo interno de creación de escenarios o configuraciones particulares del modelo, que sirven al propósito de prueba o demostración de un concepto particular relacionado con el fenómeno representado en el modelo.

⁴⁵Esto es si tiene conocimientos sobre el fenómeno particular representado en el modelo.

Al finalizar cada iteración del ciclo completo de prototipado, se adiciona conocimiento sobre el fenómeno así como una refinación o expansión de las cualidades y capacidades representativas del modelo.

El término *continuo* presente en la denominación del proceso de construcción de prototipos-modelos hace alusión al hecho de la posibilidad de mejora a largo plazo de los aspectos tratados o en estudio. La secuencia total tiene un límite en lo práctico, se abandona su progreso, se llega a una estructura o modelo final. Pero la estructura obtenida depende del conocimiento que se tenga sobre el fenómeno, los formalismos utilizados, los artilugios técnicos y lingüísticos que se usen en su representación. El avance en cualquiera de estos aspectos hace factible una evolución en la expresión del fenómeno y por ende del modelo mismo, lo que hace que cada estructura producida sea solo una representación temporal en función de la forma actual de expresión de esa realidad particular⁴⁶.

En una dimensión distinta, también es posible relacionar esta temporalidad representativa con el hecho del aprendizaje individual. Cada experiencia de aprendizaje aporta elementos nuevos a su ejecutor y si estos conocimientos se plasman de manera formal y se comparten, se da beneficio a largo plazo para la comunidad. Mientras haya interesados en profundizar temáticas habrá siempre cabida a mejoras y ajustes en su representación, por lo tanto este es un proceso que en el menor de los casos se repite por un periodo desconocido de tiempo.

A.4.3.4. Visualización de oportunidades de mejora para la metodología y la herramienta software

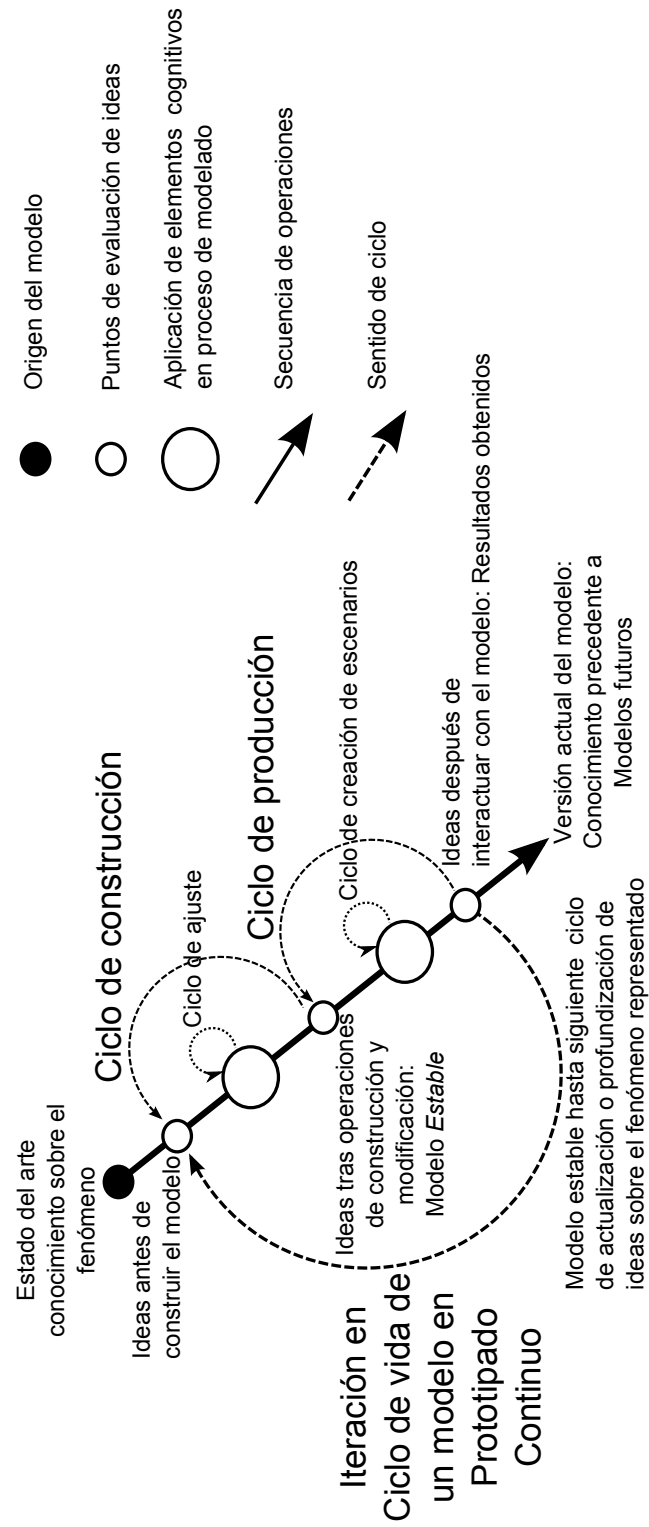
El paso entre ciclos internos dentro de una iteración del proceso de prototipado da oportunidades de evaluación sobre el conocimiento que tiene el usuario o modelador sobre el fenómeno a representar. Estos momentos o puntos de evaluación son: Antes de la construcción, Inmediatamente después de elaborar un formalismo estructural estable y posteriormente a su desarrollo aplicado en el medio de prueba o área de trabajo correspondiente. Realizar evaluaciones en cada uno de estos puntos reflejará no solo el conocimiento adquirido sobre la temática particular que propicia el proceso, también permite conocer si los útiles previstos en la metodología y el software aportan o demeritan a los elementos conceptuales sobre los que se apoya la construcción de conocimiento requerido para el proceso de modelado. Como resultado de esta evaluación se establecen observaciones, correcciones al material, recomendaciones y elementos de análisis que posteriormente sirven en la adecuación o evolución del software y la metodología. En última instancia esto hace posible que el útil tecnológico sea un mejor facilitador educativo o herramienta TIC.

A.4.3.5. Alcances colaterales

Al hacer visible las distintas etapas del proceso de modelado se hace posible dar a conocer al modelador la etapa en la que se encuentra y cual es su progreso. Lo ideal sería contar con las funcionalidades adecuadas en la metodología y el software para hacer captura y posteriormente dar interpretación a esta información, ya que si se comprende mejor el proceso de construcción del

⁴⁶La inercia, permanencia o resistencia al cambio-desarrollo de un modelo será posible solo si se mantiene inalterado el paradigma que lo sustenta.

Figura A.9: Proceso de modelado y prototipado continuo



modelo, que es una acción de expresión formal del pensamiento, se hace factible el apoyo dirigido hacia esa forma particular de trabajo mental. Esto es no solo mejorar la comprensión del tema dado sino también entender como se produce esto para aplicarlo luego en otras áreas, probablemente fuera del ámbito de uso del software y su metodología. Aplicar evaluaciones en estos instantes intermedios dentro de la secuencia de fases traería luces sobre la forma en la cual se da el desarrollo de la construcción de pensamiento en el modelador y sobre cómo es empleado el recurso educativo por parte del usuario.

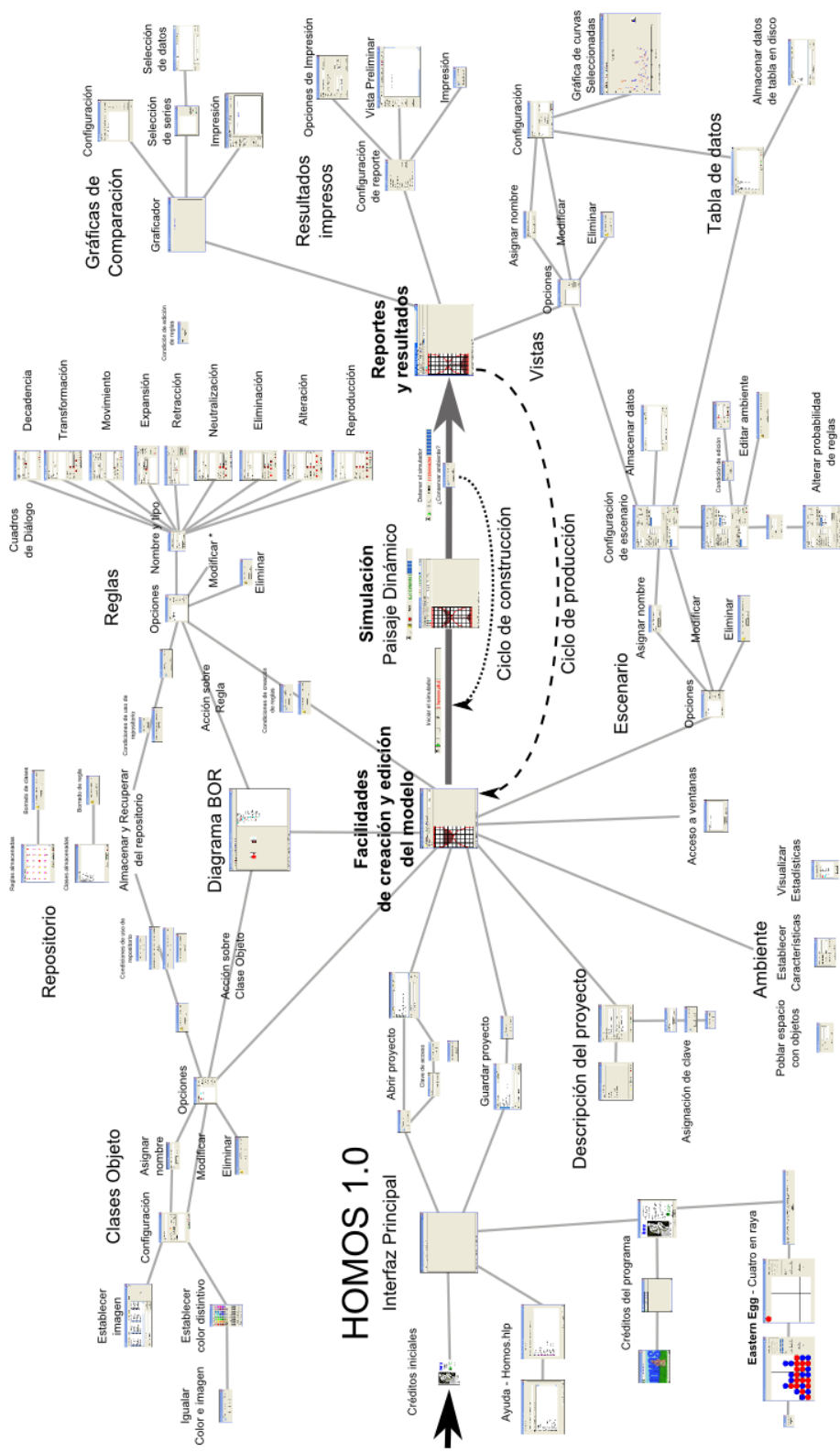
A.4.3.6. Guía de operaciones disponibles en interfaz de Homos 1.0

La guía de acciones disponibles de modelado según la interfaz gráfica de Homos 1.0 presentada en la figura A.10 en la página siguiente relaciona directamente lo disponible en la interfaz con la secuencia de modelado presentada en la sección A.4.3 en la página 157. La flecha negra centrada verticalmente en el extremo izquierdo del gráfico indica el punto de partida más común desde el cual se da inicio a la secuencia de operaciones con la herramienta.

En el diagrama, las líneas que unen a los diversos elementos expresan el acceso inmediato mediante botón u otro medio de enlace entre ventanas y cuadros de diálogo ofrecidos por la interfaz. la lectura del flujo de las operaciones debe hacerse desde el ecuador del diagrama hacia los polos, del centro hacia los extremos, ya que el acceso a otras secciones de la aplicación siempre parte desde la interfaz principal de la aplicación.

El diagrama integra la accesibilidad entre los distintos elementos componentes de la interfaz gráfica y el proceso de construcción de modelos. Las líneas punteadas o de ciclo identifican el regreso un estado de acciones previo o una repetición en la secuencia de pasos. La flecha gris oscura en medio del diagrama señala la secuencia normal de simulación: Se parte de un paisaje inicial preestablecido por el modelador, se opera el simulador avanzando mediante cálculos durante el número de pasos previstos en el escenario, se llega al paso final de simulación y se recuperan los datos o resultados.

Figura A.10: Guía de acciones disponibles en interfaz gráfica de Homos 1.0



Guía de acciones de la interfaz gráfica de Homos 1.0. Documento original en: ...\\imagenes\\homos_1.0\\intefaz_png\\interfazgrafica_edit7f.svg
Fuente: Autor:

A.4.4. Pertenencia de iconos en interfaz gráfica

Los iconos son metáforas gráficas⁴⁷, pictogramas. A pesar de su naturaleza subjetiva, es importante para la interfaz gráfica de la aplicación contar con una correspondencia práctica entre la representación usada y su asociación operativa. Se describen a continuación los elementos de imagen presentes en la interfaz gráfica de la herramienta.

A.4.4.1. Descripción de iconos usados en la interfaz gráfica

El análisis a realizar comprende la clasificación de las imágenes como tangibles, intangibles, crípticas y no relacionadas, describiendo en cada caso el uso dado a la imagen en la interfaz. Los iconos en cada categoría se listan alfabéticamente, se indica en cuales de ellos hay un *Hint*⁴⁸ asociado mediante un texto encerrado entre corchetes, P.ej: Icono:[*Hint*]. La representación gráfica de los elementos listados se encuentra en la figura A.11 en la página 168.

A.4.4.2. Referencias a elementos tangibles

Estos iconos hacen uso de representaciones de objetos físicos presentes en el mundo real con el propósito de facilitar su interpretación funcional.

Caja de herramientas: Representa el repositorio, un espacio donde se almacenan partes o secciones del modelo. Este icono se usa generalmente para acceder a utilidades y opciones de configuración de aplicaciones más que para representar un depósito o lugar de almacenamiento.

Calculadora: [Estadísticas de Objetos] Indica el acceso a las estadísticas de los objetos en el ambiente. Aunque denota la idea de cálculos, por su contenido estaría mejor representado por un símbolo como la sumatoria Σ , en semejanza con el uso dado a este símbolo gráfico en el programa Excel de Microsoft.

Carpeta amarilla abierta: [Abrir Proyecto] Apertura de proyectos en el directorio de trabajo, semejante al uso previsto en los exploradores de los sistemas operativos.

Cámara de video: [Escenarios] Señala el acceso a escenarios, usa una metáfora relacionada con la producción de cine.

Disquete: [Guardar Proyecto] Relacionado con el almacenamiento extraíble, señala la acción de guardar el proyecto.

El Pensador de Rodin: Sirve de isotipo⁴⁹ de Homos 1.0, representa el distintivo particular de la aplicación principal y la imagen del tipo de archivo asociado con el software en el

⁴⁷Ver "Usando iconos en el diseño de interfaces" por Eduardo Manchón en <http://www.ainda.info/iconos.html>

⁴⁸Cadena de texto que aparece cuando el usuario mueve el ratón sobre un control. Se usa para facilitar la interpretación del elemento señalado en la interfaz.

⁴⁹Marca gráfica con la cual se distingue a una corporación, institución o persona.

sistema operativo. Se hace una breve descripción de este icono en la sección A.4.4.8 en la página 169.

Engranajes: [Simular] Señala un mecanismo, da acceso a los controles de simulación.

Hoja en blanco “brillante”: [Nuevo Proyecto] Creación de un nuevo proyecto. Metáfora de algo nuevo, limpio.

Impresora: [Imprimir] Indica acción de producción de resultados en papel de los elementos del proyecto activo.

Llave: Relacionada con la acción de acceder o modificar el *password* o palabra clave del modelo.

Planeta tierra, Mundo: [Ambiente] Idea de lugar o espacio, trae a primer plano la ventana donde se implementa el ambiente de simulación.

Puerta: [Cerrar Proyecto] Su función es cerrar el proyecto actual, generalmente se usa en otros programas para representar la acción de fin de ejecución, salir de la aplicación.

A.4.4.3. Elementos intangibles o ideográficos

Estos iconos representan acciones, sustantivos abstractos o ideas. Se caracterizan porque pueden descomponerse gráficamente en imágenes más simples.

Diagrama de tres clases: [Diagrama] La imagen usada es similar a la representación del diagrama BOR implementado en la herramienta.

Documento en blanco, flecha roja y diskette: Indica la acción de guardar en la ventana tabla de datos.

Flecha roja de selección: [Indicador] Libera el puntero del ratón, desactiva la selección manual de clases en el ambiente.

Flecha roja entrando sobre cuadrícula: [Pegar] Pegar elementos en el ambiente.

Flecha roja saliendo sobre cuadrícula: [Copiar] Representa la acción de copiar el contenido de las celdas seleccionadas en el ambiente.

Mano borrando con un lápiz: Permite eliminar clases o reglas en el repositorio.

Valor de ángulo y flecha curva en sentido de giro antihorario: [Rotar Pegado *ángulo*°] Identifica la acción de pegar el contenido de la última selección de objetos en el ambiente según el ángulo seleccionado.

Dos rectángulos rojo azul y flecha conectora: [Pegar Espejo *Tal*] Acción de pegado con rotación del último contenido de celdas copiado. *Tal* identifica si la operación resultante es Vertical u Horizontal.

Series de datos, curvas en gráfica: [Vistas

Escenario Actual] Permite acceder a las curvas poblacionales a mostrar previstas en el escenario actual.

Tijera sobre malla: [Cortar] Almacenar temporalmente los objetos de las celdas previamente seleccionadas en el ambiente.

Los iconos mostrados en los botones aceptar y cancelar de los cuadros de diálogo son los predeterminados para estas funciones por Delphi 3.

A.4.4.4. Iconos crípticos

Para estos iconos su identificación y significado operativo no es inmediato, se asimila tras varias sesiones de uso. Generalmente identifican acciones intangibles y puede ser difícil encontrar un buen sustituto para estas imágenes, en ocasiones se convierten en nuevos patrones de reconocimiento. Su existencia sugiere una aproximación distinta para la representación gráfica del concepto.

Cuadrícula + figuras geométricas de colores + etiqueta azul cian: [Propiedades] Visualiza las propiedades del ambiente. Requiere del *hint* para su identificación.

Objetos geométricos distintos dentro de un círculo: [Clases - Objeto] Ilustra un diagrama de conjuntos, da acceso a la edición de clases objeto. Se asocia luego de observar el identificador de clases en el árbol de diagrama.

Objetos geométricos en ubicaciones diferentes y una flecha “relacionándolos”: [Reglas] Da acceso a la edición de reglas, es el icono más críptico. Se asocia luego de observar el identificador de reglas en el árbol de diagrama.

Figuras geométricas al rededor de una cuadrícula: [Insertar Objetos] Da acceso a insertar objetos. Se ven las figuras como objetos y la cuadrícula como idea del ambiente pero no hace énfasis en la acción de inserción.

Le siguen en crípticidad las opciones de rotación, copiado y pegado especial, su activación depende de la selección y corte de objetos en el ambiente.

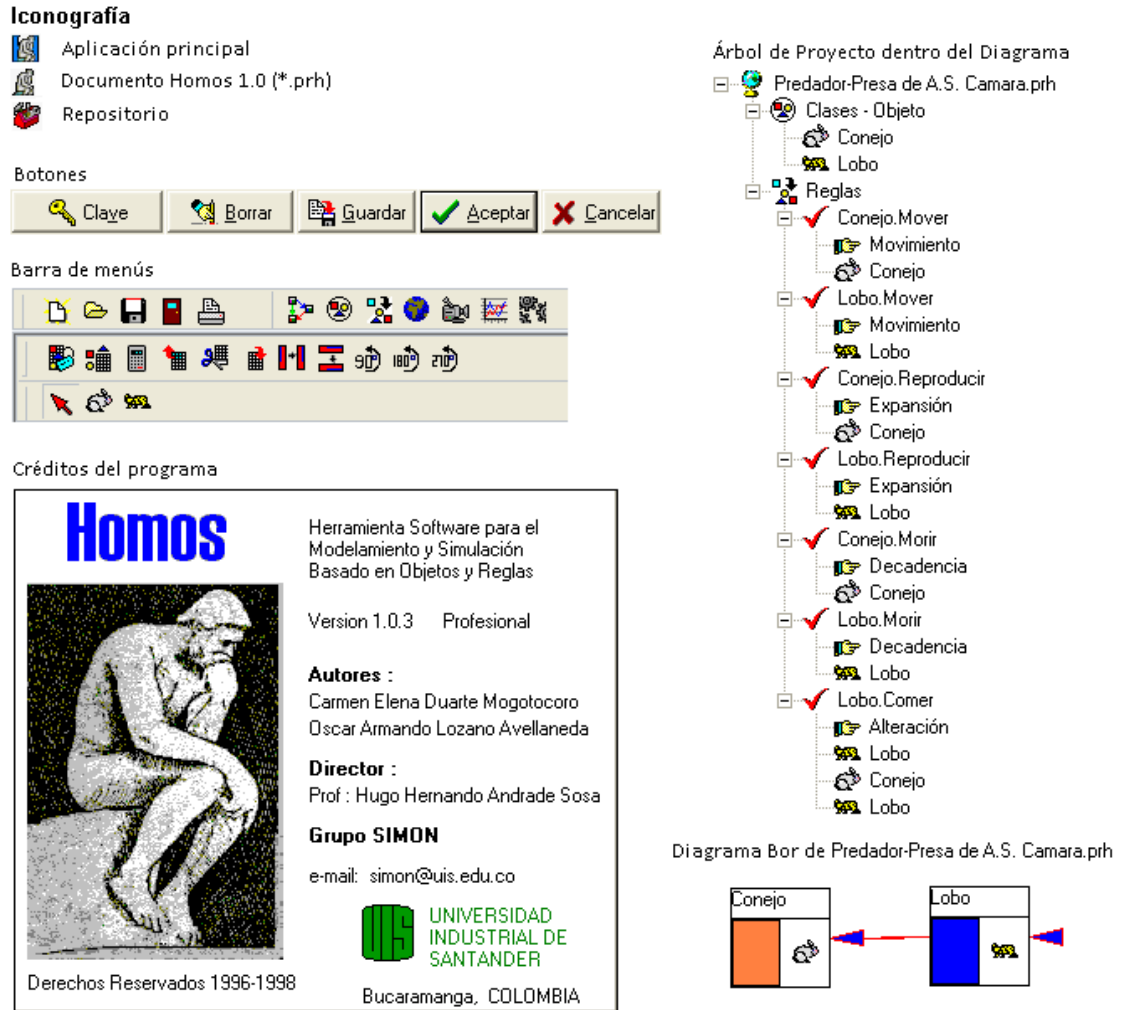
A.4.4.5. Iconos sin relación aparente

Estos iconos no tienen una correlación directa entre lo mostrado y su función en la interfaz. Su existencia sugiere el uso de otro tipo de representaciones gráficas.

Marca de validación: Presentes al inicio de la definición de gráfica de regla en el árbol de diagrama, indican la existencia individual de cada regla. Su presencia no esta relacionada con la interpretación de validaciones o calificaciones de algún tipo sobre la estructura o composición de las reglas de transición del modelo.

Mapamundi: Raíz del árbol de proyecto. Sería mejor usar la imagen creada para el representar el diagrama BOR, puesto que esta estructura se conforma de clases y reglas.

Figura A.11: Elementos de imagen en interfaz gráfica



A.4.4.6. Elementos de imagen en el diagrama BOR

El cuadrado dividido en 3 secciones usado para identificar las clases objeto en el diagrama BOR es una adaptación de la representación usada para la idea de clases en la teoría de objetos⁵⁰. La caracterización de las clases objeto en el diagrama se requiere de un nombre de clase, una imagen y color distintivo. Para las reglas se plasman líneas rectas de varios colores con o sin puntas de flecha que unen a los objetos. El color y dirección de las flechas representa el sentido de aplicación y el tipo

⁵⁰Para Coad y Yourdon, la clase se representa por un cuadrado dividido en tres secciones horizontales, En la primera sección se sitúa el nombre de la clase, en la segunda sus atributos y por último los servicios.

de regla utilizado según lo establecido en el documento de tesis de Homos 1.0. Todos estos elementos se disponen en un amplio espacio vacío dentro del diagrama BOR.

Se habla de con detenimiento de los componentes del diagrama BOR en el apartado A.7 en la página 272, se da un ejemplo de diagrama en la figura A.11 en la página anterior.

A.4.4.7. Iconos e imagen de representación de clases objeto

Las imágenes creadas por los usuarios para la identificación de las clases objeto en el ambiente se referirán generalmente a objetos físicos tangibles, P.ej: inanimados y animales, ya que la herramienta y la metodología hacen énfasis en el enfoque biológico de las reglas de transición y la capacidad del software de representar paisajes dinámicos. La herramienta incluye entre sus imágenes prediseñadas este tipo de motivos gráficos, como se ve en la Figura A.12.

Figura A.12: Imágenes en Recursos



A.4.4.8. Sobre el logo distintivo de la herramienta

El isotipo de Homos está inspirado en la obra El Pensador (*Le Penseur*) del escultor Francés François-Auguste-René Rodin⁵¹. Bautizada originalmente como “Dante Pensando” o “El Poeta” y junto a El Beso (*Le Baiser*) y la Danae (*La Danaïde*) pertenecen a una obra escultórica mayor, La “Puerta del Infierno” (1880-1917)⁵², un portal de dimensiones épicas basado en la Divina Comedia de Dante Alighieri. Se originó con el propósito de hacer parte del museo de artes decorativas de París, el cual nunca llegó a ser construido.

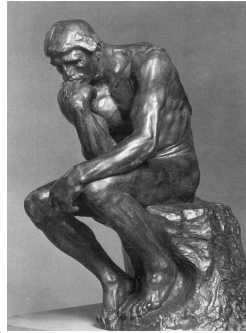
Emulando a Miguel Ángel, Rodin plasma poéticamente a Dante desnudo en reflexión profunda sobre su obra, el dolor que ha visto en el averno y las advertencias recibidas desde el cielo.

El pensador ha sido objeto de múltiples reproducciones en diversas versiones alrededor del mundo. El original se encuentra actualmente en el Museo Rodin de París. Ver Figura A.14 en la página siguiente.

⁵¹París, 12 de noviembre de 1840 - Meudon, 17 de noviembre de 1917.

⁵²Su biografía y obra en http://es.wikipedia.org/wiki/Auguste_Rodin, también en <http://www.epdlp.com/pintor.php?id=356>

Figura A.13: El Pensador de Rodin



El Pensador (1900)

Descripción: hecho en Bronce,
es una obra artísticamente desproporcionada.
Dimensiones: 1,98 × 1,29 × 1,34 metros.
Localización: Museo Rodin. París
Autor: Auguste Rodin.

Fuente: Tomado de <http://www.epdlp.com/cuadro.php?id=1081>

Figura A.14: Referencias directas al pensador en Homos 1.0

Créditos del programa



Derechos Reservados 1996-1998

Herramienta Software para el
Modelamiento y Simulación
Basado en Objetos y Reglas

Version 1.0.3 Profesional

Autores :
Carmen Elena Duarte Mogotocoro
Oscar Armando Lozano Avellaneda

Director :
Prof : Hugo Hernando Andrade Sosa

Grupo SIMON
e-mail: simon@uis.edu.co



UNIVERSIDAD
INDUSTRIAL DE
SANTANDER

Bucaramanga, COLOMBIA

Imagen final video Homos



Fuente: Autor.

Su significado: La asociación de esta obra con la herramienta es simple si relacionamos el proceso reflexivo del pensador con el necesario para construir modelos usando la herramienta. Es completo si vemos todo el cuadro, como en la puerta: Nosotros mediante la acción pensante, constructiva, salimos del pozo infinito de ideas inconexas que percibimos sobre un fenómeno a un modelo dinámico, que como consejero nos da duras advertencias y lecciones sobre nuestras acciones.

Otras simbologías relacionadas: Aunque no hay un consenso claro entre los tipos de logos y isotipos en otras herramientas del Grupo Simon, se está propiciando una tendencia hacia el uso de iconismos con motivos regionales, siendo un ejemplo de ello el símbolo precolombino usado como isotipo e icono principal de Evolución 3.5. Esta es una forma sencilla de promover el sentido de pertenencia nacional y la divulgación del patrimonio inmaterial colombiano. Es

posible que en el futuro se haga una adaptación del isotipo de Homos hacia un motivo regional relacionado con la idea del pensador.

A.4.5. Resultados obtenidos

Los elementos existentes en la interfaz corresponden con los enunciados por el trabajo de grado de Homos 1.0. Se hizo un repaso a los elementos que componen la interfaz y la iconografía disponible, en el proceso se encuentran elementos de cambio y mejora.

El analisis hace publica la forma en la cual se realiza el proceso de modelado en la herramienta software: El metodo presentado es independiente de los lineamientos establecidos por la MBOR, pero hace posible la aplicacion efectiva de la metodologia BOR con los utiles y funciones disponibles en la herramienta software. El metodo descrito es aplicable a cuaquier tipo de produccion de intangibles mejorables iterativamente, p.ej: Diagramas, modelos y demás construcciones relacionadas.

A continuación se deteminan algunos de los elementos de mejora mas destacados a realizar sobre la interfaz grafica de la herramienta software:

- Pautas de mejora:
 - El sistema debe impedir la simulacion de modelos vacios.
 - Manejar la seguridad de los modelos creados en un nivel superior, delegarla a aplicaciones especializadas en ese tipo de labores.
 - Se sugiere aumentar el tamaño del ambiente mas allá de los valores actualmente establecidos. Esto es con el proposito de dar cabida a grandes modelos.
 - Hay necesidad de reconfigurar o eliminar la funcionalidad del repositorio, que actualmente no tiene uso practico para el desarrollo de modelos.
 - Funcionalmente existe repeticion de cuadros de diálogo, especificamente en cuanto a la configuracion de reglas. El patron de configuración de las reglas no fue exitosamente aprovechado en la version actual del software.
- Mejora de funciones:
 - Implementar en el ambiente el zoom con valores decimales/flotantes.
 - Permitir la edición dinámica de las dimensiones del ambiente mediante el ratón.
 - Navegacion mediante teclado en diagrama seleccion de objeto y opciones de edicion de relaciones entre objetos en el diagrama mediante navegación y seleccion en el arbol de proyecto
 - Implementar el acceso a los elementos en el ambiente mediante teclado. Especialmente la ubicacion de objetos mediante controles de teclado.
 - Asociar a los escenarios con ambientes especificos, permitir que los escenarios tengan ambientes prediseñados.

- Integrar permanentemente la barra de simulación en la interfaz. No se determinó un claro propósito para la actual separación de este elemento del entorno gráfico
- Mejorar la secuencia de almacenamiento y recuperación de proyectos en la interfaz, pej: hacer que la petición de salvar el proyecto se active solo si se han realizado cambios en el modelo desde la última vez que se guardó, permitir la cancelación del proceso de creación de reglas de transición, entre otros.
- Se sugiere la inclusión de nuevos tipos de borde de simulación: Pej: , así como permitir la mezclas de varios tipos de bordes en un mismo ambiente de modelado.
- Presentar las estadísticas del modelo y de la simulación en una ventana específica fija dentro de la interfaz.
- Establecer un escenario por defecto automáticamente para cada simulación.
- Inclusión de nuevos tipos de gráficas producidas por los datos obtenidos por el simulador, pej: una gráfica de comparación de activa de poblaciones como la mostrada en el software Wator⁵³.
- Establecer una imagen más apropiada para los iconos identificados como cripticos y sin relación aparente presentes en la interfaz.
- Realizar un remplazo del tipo de imágenes principal usado en la aplicación. Pasar de iconos a otro tipo de formatos, especialmente de tipo vectorial. Se recomienda el formato SVG⁵⁴.

■ Proceso de modelado

- El modelo prosa no se nota dentro del proceso de modelado, es completamente opcional. se sugiere encontrar una forma para que se haga énfasis en uso dentro de la interfaz y en los reportes de resultados.
- Hacer de la documentación un paso activo dentro del proceso de modelado. Es posible pensar en formas de auto documentación automática, según el elemento y el momento de edición del modelo.
- Implementar una mejor representación de las clases-objeto, seguir los lineamientos establecidos en la nomenclatura para ese propósito. Los demás elementos componentes de diagrama se detalla en la sección A.7 en la página 272.
- Se sugiere la implementación del nombre automático de reglas, esto puede hacerse gracias a los nombres de clases y el tipo de reglas disponibles a usar en el modelo. Los nombres automáticos no reemplazan la forma actual de identificación, que se parece a establecer un alias para cada elemento.
- Algunas de las reglas permiten configuraciones que no se ajustan a su descripción funcional, Pej: neutralizaciones de iguales es retracción. Se propone que la interfaz pueda detectar los casos e informar al modelador de la configuración actual de la regla. dado el caso, cambiar el tipo de regla para obtener el tipo de comportamiento deseado.

⁵³Ver sección B.3.1 en la página 297

⁵⁴Ver www.w3.org/Graphics/SVG/

- Informar desde la interfaz grafica al modelador del momento actual de modelado en el que se encuentra según los elementos establecidos en la secuencia de operaciones de modelo presentada en la sección A.4.3 en la página 157.

A.5. Análisis de la interfaz de automatización OLE

Este apartado trata acerca de las capacidades particulares de la interfaz de automatización, el análisis pretende completar la documentación existente en el trabajo de grado de Homos 1.0. En semejanza con la estructura del análisis realizado en la interfaz gráfica en la sección A.4 en la página 146, se realiza un estudio en tres niveles:

- Se realiza un análisis de vacío funcional y se presenta una descripción de las funciones disponibles con el propósito de establecer de qué se dispone en la interfaz.
- Se establecen las características que permiten el modelado, se describe cómo modelar usando la interfaz.
- Se hace un estudio sobre cómo hacer uso efectivo de la interfaz, se propone cómo hacer un modelo en la práctica.

Los elementos de análisis enunciados se concretan en este documento mediante la incorporación de un método para la creación de modelos, instrucciones de construcción paso a paso de un modelo-script y un ejemplo práctico de modelo-script haciendo uso de interfaz de automatización.

A.5.1. Propósito de la interfaz

Esta interfaz permite el uso de las capacidades funcionales de Homos 1.0 en otras aplicaciones. Los programadores tienen a su disposición un conjunto de procedimientos, llamados a funciones y parámetros con los cuales crear modelos BOR y controlar vía automatización una instancia funcional del software.

A.5.2. Documentación sobre la interfaz

La implementación de la interfaz OLE está consignada en los archivos

`..\homos\bin\homos_tlb.pas`⁵⁵

y en la unidad

`..\homos\bin\automatizacion\usistema.pas`⁵⁶

del CdPS. La descripción original de las funciones y procedimientos disponibles así como sus parámetros se encuentra en el anexo 2, *Uso de automatización Ole con Homos* en el documento de trabajo de grado de Homos 1.0.

⁵⁵Definición de los métodos de la interfaz.

⁵⁶Definición del código de las funciones y procedimientos utilizados.

A.5.3. Instalación y desinstalación de la interfaz OLE en Homos 1.0

Para activar la capacidad OLE de Homos 1.0 luego de la instalación de la aplicación en su versión completa⁵⁷, es necesario registrar la interfaz OLE en el sistema con la siguiente instrucción en la línea de comandos:

```
homos.exe /regserver
```

Esto garantiza que se realiza la creación de las entradas en el registro necesarias para hacer uso efectivo del servidor OLE incluido en el archivo principal de la aplicación.⁵⁸

Una vez realizado el registro, estos son los distintos identificadores relacionados con la aplicación:

Tabla A.8: Identificadores CLSID de la Interfaz OLE

★ GUID = LIBID_Homos	{B9EF32C3-39C4-11D2-AA12-444553540000}
ISistema = interface(IDispatch)	{B9EF32C4-39C4-11D2-AA12-444553540000}
Class_Sistema	{B9EF32C5-39C4-11D2-AA12-444553540000}

GUID LIBID_Homos: Identificador global único de la interfaz de automatización OLE de Homos 1.0 en el sistema.

CLSID: Es un GUID de un objeto OLE.

Fuente: Autor.

Cuando sea necesaria la eliminación en el registro de todas las entradas relacionadas con esta interfaz se ejecuta en la línea de comandos:

```
homos.exe /unregserver
```

Posterior a esta operación, pueden realizarse los demás procesos de desinstalación de la herramienta, tal como se presentan en la sección A.1.4.3 en la página 102.

⁵⁷Ver Sección A.1.4.2 en la página 101

⁵⁸Ver las entradas de registro en la sección A.2.3 en la página 109.

A.5.4. Resumen de operaciones disponibles con la interfaz

Esta interfaz permite realizar las siguientes acciones:

- Crear un nuevo proyecto, abrir uno ya creado, guardar y cerrar el proyecto actual.
- Iniciar, pausar y terminar la simulación del proyecto actual.
- Crear clases objeto, especificar sus parámetros, determinar la cantidad de clases creadas y el número de objetos presentes en el ambiente por cada clase.
- Insertar instancias de una clase objeto cualquiera en el ambiente, estableciendo la cantidad, la ubicación y el tipo de ubicación deseada.
- Usar cualquiera de los 9 tipos de reglas presentes en la herramienta, asociarlas a una o más clases objeto, asignar su prioridad y probabilidad de ejecución. Admite la edición y la devolución por separado del valor de probabilidad de una regla específica.
- Determinar la cantidad de clases objeto por paso de simulación, reglas, escenarios y el número de instancias de cada clase presentes en el modelo durante la simulación.
- Crear escenarios, establecerles nombres distintivos y seleccionar uno de ellos como el escenario activo para la simulación.
- Establecer el tamaño en filas y columnas, el comportamiento del borde y el color de fondo del ambiente. Incorpora un procedimiento de redibujado, necesario para la actualización gráfica del ambiente cuando se realizan operaciones de inserción de objetos.
- Hacer visibles las ventanas de ambiente y diagrama BOR. por defecto, se da enfoque y se maximizan estas ventanas dentro del área de trabajo.
- Establecer la visibilidad del programa principal y una facilidad para la presentación interactiva de mensajes texto.

Tal y como está diseñada, la interfaz facilita la implementación funcional de un modelo en la herramienta, pero ofrece edición incompleta ya que no proporciona facilidades para eliminar los distintos elementos creados. Es necesario pasar a la interfaz gráfica para realizar estas operaciones. En el caso particular de las reglas de transición a pesar de que no sea posible eliminarlas por medio de la interfaz OLE, es posible configurarlas de tal forma que sea improbable su ejecución durante la simulación⁵⁹.

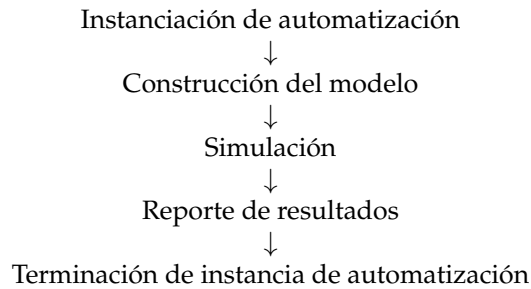
La interfaz promueve un diseño conservador del modelo: crear solo los elementos requeridos. Esta orientación es lo que hace innecesarias las funciones de eliminación de elementos dentro de cada tipo de elemento disponible. Crear modelos con esta interfaz presupone la existencia de una estructura de relaciones conocida con la cual implementar el modelo.

⁵⁹Establecer que la probabilidad de ejecución [%] de la regla sea igual a cero.

A.5.5. Etapas en la creación de un modelo mediante la interfaz OLE

El proceso de implementación de un modelo BOR mediante la interfaz OLE se compone de las siguientes etapas:

Figura A.15: Etapas de implementación de un modelo con la interfaz OLE



Fuente: Autor.

Etapas de construcción: Cubre las operaciones requeridas para la conformación de cada uno de los elementos que componen el modelo:

- Creación de clases objeto seguida de la creación de las reglas del modelo.
- Configuración del ambiente: determinación de dimensiones, establecimiento del comportamiento de borde, configuración de paisaje inicial de simulación.
- Creación de uno o más escenarios: identificación de cada escenario, duración de la simulación.

Estas acciones de edición del modelo pueden realizarse en cualquier orden, observando que las clases deben crearse siempre antes de ser usadas por las reglas de transición.

Etapas de simulación: Esta etapa requiere:

1. Un modelo BOR definido.
2. La selección de un escenario de simulación.
3. El llamado a las operaciones de manejo del simulador, mediante la construcción de uno o más ciclos de control con los cuales se hará efectiva la ejecución del modelo.

Reporte de resultados: Dependerá del propósito con el cual ha sido creado el modelo. Esta etapa generalmente estará constituida por la presentación de las gráficas o los valores numéricos que componen el comportamiento poblacional de las clases objeto obtenido durante la simulación.

La llamada inicial y el cierre de la interfaz por parte de la aplicación anfitriona son la primera y última tarea en el proceso de implementación del modelo. La figura A.15 permite establecer que la recuperación de los datos obtenidos durante la simulación es la parte más profunda de todo el proceso.

A.5.6. Organización de funciones disponibles según etapas de implementación del modelo

Con el propósito de facilitar el estudio y proporcionar elementos para posibles nuevas operaciones de la interfaz, las funciones que la componen se organizan en tablas que las agrupan según sea el aporte de cada una de ellas a las etapas principales de implementación del modelo: construcción y simulación. No se hace énfasis en la etapa de presentación de resultados ya que no existen extensiones funcionales⁶⁰ en la interfaz especializadas en estas labores. El reporte de datos estará generalmente en manos de la aplicación anfitriona⁶¹, la cual recolecta los datos obtenidos del proceso de simulación y los muestra de según las necesidades del modelador y las posibilidades de representación disponibles.

La organización de funciones según las etapas de construcción, simulación y reporte es virtual, no existe explícitamente dentro de la metodología o como característica funcional de la interfaz. En la práctica, esta división de etapas se realiza implícitamente durante la elaboración del modelo.

A.5.6.1. Análisis de vacío funcional mediante una tabla de contraste entre funciones disponibles y operaciones generales

Para realizar un ejercicio de análisis sobre las funciones disponibles de la interfaz, se establece la siguiente tabla de contraste⁶²:

Tabla A.9: Estructura de relaciones entre funciones en módulos disponibles vs operaciones específicas en etapa *tal*

Conjunto universal	Operaciones de contraste			
Módulos funcionales disponibles	O_1	O_2	$\dots$	O_m
M_1	$F_{1,1}$	$F_{1,2}$		$F_{1,m}$
M_2	$F_{2,1}$	$F_{2,2}$		$F_{2,m}$
$\vdots$			$\ddots$	
M_n	$F_{n,1}$	$F_{n,2}$		$F_{n,m}$

En cada celda, $F_{j,k}$ es el conjunto de funciones y procedimientos del módulo o bloque de funciones M_j que permiten realizar la operación O_k en la etapa *tal*. Las celdas vacías se traducen en ausencias funcionales.

Fuente: Autor.

Se comparan los elementos funcionales de cada módulo o bloque de funciones disponibles [filas] contra un conjunto de operaciones básicas comunes a todos los elementos listados a realizar la etapa correspondiente [columnas]. Las celdas internas de la tabla contienen por extensión los

⁶⁰La presentación de algunas ventanas de la aplicación y los mensajes modales de texto no son reemplazo adecuado para medios más elaborados de presentación de resultados.

⁶¹La que hace llamado a la interfaz. En este proyecto, los distintos ejemplos implementados en la sección E.4 en la página 588 hacen uso de un servidor web ejecutando un script hecho en PHP como aplicación anfitriona o *host*.

⁶²Se interpreta el término contraste como la comparación entre lo que hay y lo que se supone óptimo. La tabla permite establecer elementos de diferenciales, lo que hace falta para llegar a lo óptimo.

identificadores y una o más de las características principales de las funciones en la interfaz con las cuales se realizan las operación básicas establecidas para el contraste.

A partir de la estructura de la tabla se establece que:

- Determinar con la mayor claridad posible cuales son las operaciones básicas a realizar y cuales son los elementos componentes del modelo, se obtiene una mejor caracterización funcional del objeto en análisis.
- El límite teórico de operaciones de este sistema de organización funcional lo impone el propósito final de la interfaz⁶³ y la densidad deseada de métodos incluidos.
- El Incremento en el número de operaciones en una etapa solo es viable si estas ampliaciones están dirigidas en apoyo del *leitmotiv*⁶⁴ de la aplicación:
 - Muchas operaciones de tipos distintos: Variedad funcional, generalidad, necesidad de mejoras frecuentes y actualizaciones. Favorece la experiencia de buscar aplicaciones sustitutas en el mercado, en caso de no encontrar en el desarrollo actual la sinergia de intergración esperada o la característica particular deseada .
 - Muchas operaciones en especialización de una tarea: se acota el rango de aplicación del software, requiere la mejora constante de los algoritmos establecidos. Produce asociación a largo plazo entre el nombre y la actividad a la que esta dirigida el software.
- La existencia de celdas vacias en la tabla permite establecer la presencia de ausencias funcionales, respecto a las operaciones básicas consideradas en el análisis.

En el procedimiento de análisis a realizar se buscará hacer uso de operaciones que permitan la especialización operacional de la interfaz. El resultado del ejercicio de análisis sobre la interfaz usando el formato propuesto en la tabla A.9 en la página anterior se muestra en las tablas A.10 en la página 182 y A.11 en la página 183.

A.5.6.2. Construcción de las tablas de contraste funcional de la interfaz OLE

A continuación se describen las operaciones de contraste y los módulos de funciones establecidos para el análisis:

Descripción de los identificadores de operaciones básicas: Se refieren al reconocimiento de las distintas acciones posibles con los elementos componentes del modelo por parte de la aplicación.

- En la etapa de construcción se definen las siguientes operaciones básicas:

⁶³o la justificación de su existencia y la de sus operaciones.

⁶⁴Idea fija, caracterización repetitiva y fundamental de una cosa.

Creación: Involucra la generación de un elemento nuevo al modelo según el tipo de componente, representa la asignación de recursos, contiene tareas de configuración inicial, identificación, etc.

Apertura: Operación de recuperación y carga de un documento previamente almacenado en disco u otro medio de almacenamiento.

Edición / Modificación: Facilidad para la realización de cambios sobre los atributos o características distintivas de un elemento del modelo.

Almacenamiento: Operación de escritura de datos en un medio especializado para su conservación permanente.

Finalización / Eliminación: Terminación, recuperación de los recursos consumidos en la creación de ese elemento.

Selección: Elección de un elemento de entre otros del mismo tipo con algún propósito específico.

■ En la etapa de simulación se definen las siguientes operaciones básicas:

Ejecución: Realización por parte del motor de simulación o el núcleo principal de la aplicación de las acciones indicadas asociadas a un elemento del modelo.

Información sobre el elemento: Recuperación, extracción de uno o más datos relacionados con el elemento referenciado.

Presentación del elemento: Visualización en medio específico de las definiciones o contenido del elemento componente particular del modelo.

Las operaciones básicas que definen las tablas se eligieron pensando en aquellas acciones que describen el mejor ajuste posible a las actuales funciones contenidas en la interfaz. La definición de otras operaciones básicas diferentes en cada etapa permitiría teóricamente soportar nuevas extensiones funcionales en la interfaz.

Esta misma forma de análisis puede ser extendida a las funciones incluidas dentro del cuerpo principal de la aplicación y a otros aspectos de la aplicación.

Descripción de identificadores de filas módulos o elementos componentes del modelo: En

la primera fila de las tablas se listan los diversos elementos componentes de un modelo BOR. Según sus características particulares, será posible realizar sobre cada uno de ellos determinadas operaciones básicas.

Se incluye dentro de este listado a los Objetos en ambiente: es la parte del proceso de modelado en la que se realiza la configuración del paisaje de simulación, que comprende la ubicación espacial de instancias de clases objeto en el ambiente. En el caso particular de la interfaz se soporta la inserción de objetos en el ambiente, lo que facilita la creación del paisaje inicial de simulación.

La lectura de las tablas requiere el conocimiento de algunos aspectos básicos relacionados con la interfaz:

- Todas las funciones de la interfaz tienen acción dinámica: es posible realizarlas en cualquier momento durante el uso de la interfaz. De entre ellas se destacan aquellas que realizan operaciones de modificación o escritura de los parámetros principales del modelo. El uso inadecuado de estos métodos puede desde degradar completamente el paisaje dinámico presentado hasta interrumpir completamente la ejecución del modelo. Por esta razón, es conveniente no hacer uso libre de ellas durante esta etapa⁶⁵. En las tablas A.10 y A.11 se marcan con ✖ este tipo de métodos.
- Se hace configuración implícita del simulador al establecer los distintos parámetros de ambiente, objetos en ambiente y escenario. En esta versión de Homos está permitido iniciar la simulación de un modelo sin haber definido previamente clases objeto y reglas.
- La interfaz está diseñada para permitir la realización del modelo en caso de ausencia o imposibilidad de acceso a la interfaz gráfica de la aplicación.

⁶⁵Métodos de la interfaz que realizan una *acción dinámica negativa* durante la simulación.

Tabla A.10: Etapa de edición, métodos de construcción del modelo.

Interfaz OLE	Operaciones básicas sobre elementos del modelo en estado estático				
	Creación	Apertura	Edición Modificación	Almacenamiento	Finalización Eliminación
Proyecto	ProyectoNuevo	ProyectoAbrir *		ProyectoGuardar	ProyectoCerrar *
Clases-objeto	ClaseObjetoCrear				
Reglas	ReglaCrearTAL		ReglaAsignarProbabilidad		
Ambiente			ColorCeldas		
			TipoLimite *		
			NumeroFilas *		
			NumeroColumnas *		
<i>Paisaje:</i> Objetos en ambiente	ObjetosInsertar *				
Escenario	EscenarioCrear				EscenarioSeleccionar *
Simulador					
Diagrama BOR					
Gráficas y Vistas					
Resultados					
Medios de					
Interacción con el usuario					

TAL comprende los nombres de las nueve reglas de transición implementadas en Homos 1.0: Decadencia, Transformación, Movimiento, Expansión, Retracción, Eliminación, Neutralización, Alteración y Reproducción.

* Denota aquellos elementos problemáticos, mal documentados o con evidentes errores de funcionamiento.

* Resalta aquellos elementos que debido a la naturaleza de su acción, pueden realizar cambios en la estructura y datos del modelo durante la etapa de simulación.

Nótese que los espacios vacíos en la tabla pueden interpretarse como ausencias funcionales. Se destaca en este sentido los espacios blancos desde la fila Diagrama Bor y posteriores. La fila simulador se incluye por motivos de simetría con la tabla A.11.

Fuente: Autor.

Tabla A.11: Etapa de simulación: métodos de control del modelo.

Interfaz OLE	Operaciones sobre elementos del modelo en estado dinámico		
Elemento del modelo	Ejecución	Información sobre el elemento	Presentación del elemento
Proyecto			Visible*
Clases-objeto		ClaseObjeto(Cua1) CuantasClasesObjeto	
Reglas		Regla(Cua1) ReglaDarProbabilidad CuantasReglas	
Ambiente		ColorCeldas TipoLimite ✂ NumeroFilas ✂ NumeroColumnas ✂	VentanaAmbiente AmbienteActualizar Visible*
<i>Paisaje:</i> Objetos en ambiente		ObjetosDarCantidad	
Escenario		Escenario(Cua1) CuantosEscenarios	
Simulación	SimulacionIniciar SimulacionParar SimulacionTerminar		
Diagrama BOR			VentanaDiagrama
Gráficas y Vistas			
Resultados			
Medios de Interacción con el usuario			Mensaje

* Al hacer visible el proyecto, se hace visible simultáneamente el ambiente.

Fuente: Autor.

A.5.6.3. Resultados obtenidos del análisis de relaciones funcionales

Como resultado de esta organización de funciones, se destaca que:

- Con respecto a las operaciones básicas elegidas y los componentes considerados del modelo:
 - Los espacios vacíos en las tablas pueden interpretarse como ausencias funcionales, por ejemplo el almacenamiento separado de diagramas BOR, la creación de cierto tipo de resultados, la selección para presentación de gráficas y vistas, la obtención de ciertos datos del estado del simulador, la eliminación de clases objeto y reglas, la eliminación de objetos del paisaje, etc.
 - Algunas celdas representan combinaciones funcionales imposibles en la actual versión del

software: no es posible administrar⁶⁶simultáneamente varios ambientes, distintas sesiones de simulación, varios proyectos, múltiples diagramas BOR, etc.

- Las ausencias en la interfaz generalmente van en pares funcionales. Ya que no existe un método con el cual establecer la descripción general del proyecto, tampoco existe un método con el cual obtener o presentar dicha información. En este mismo sentido de ausencia funcional, no es posible presentar los detalles descriptivos de las clases y reglas presentes en el modelo.
 - Los diagramas BOR, las gráficas y vistas, los resultados obtenidos por la aplicación y la interacción con el usuario no son elementos relevantes para la actual implementación de la interfaz.
 - No existen facilidades en la interfaz para la documentación general del proyecto. Es posible establecer las descripciones de clases y reglas, pero no recuperarlas.
 - Solo el simulador posee elementos ejecutables en esta versión del software.
 - El único medio de interacción con el que cuenta la aplicación es la presentación modal de mensajes de texto.
-
- La división y organización de métodos presentados en la tabla no es fortuita, debería existir incorporada internamente dentro de la interfaz para que las funciones que ahora son posible causa de inestabilidad y fallas por el mal uso que se haga de ellas, pasen a ser funciones seguras y que realizan las operaciones indicadas en la etapa de creación del modelo que les corresponda.
 - La lista de operaciones básicas presentada cada tabla no es exhaustiva. Permite estructurar posibles procesos posteriores de ampliación y mejora de las funciones ofrecidas por la interfaz.
 - La lista de elementos componentes del modelo BOR puede refinarse / extenderse / recibir adiciones. Facilita una estructuración ordenada e incremental de los elementos previstos por la metodología y ofrecidos por el modelo⁶⁷.

A.5.7. Descripción de funciones y procedimientos disponibles

Los nombres de las funciones y procedimientos⁶⁸ de la interfaz se componen usando el siguiente formato:

[Nombre de elemento componente del modelo][Nombre de operación básica]

Los nombres de las funciones que instancian reglas de Transición se estructuran de la siguiente forma:

⁶⁶Crear, almacenar, eliminar, etc.

⁶⁷Existen elementos definidos en la metodología que no tienen equivalente o no están incorporados explícitamente en la versión software del modelo. Por ejemplo el modelo en lenguaje prosa y el modelo matemático particular del proyecto implementado.

⁶⁸La interfaz está compuesta de métodos, que pueden ser funciones o procedimientos. Se hará uso de estos términos según convenga a los objetivos de descripción de este documento.

ReglaCrear[Nombre de regla transición definida en la MBOR]

Los nombres de los parámetros y propiedades del modelo tienen la siguiente forma:

[Nombre de propiedad][Nombre de característica del elemento componente]

En los formatos presentados no hay separadores o espacio entre palabras, va con mayúscula la primera letra de cada parte del nombre del identificador.

Entre los nombres de parámetros se definen:

Nombre: Cadena que establece la ubicación de un directorio e identificador de archivo. Ejemplo: "C:\Program Files\UIS\Homos\Proyectos\reloj de arena.prh".

Cadena que identifica al elemento al que hace referencia. Ejemplo: "Conejo.Mover".

En general, las cadenas vacías y la repetición de identificadores en un mismo tipo de elemento componente se interpretan como un error y no son permitidos por la aplicación. Pej: Dos clases de igual nombre.

Imagen: Cadena que determina la ubicación del recurso gráfico *.bmp a usar como imagen representativa de la clase.

Descripción: Cadena que contiene información relativa a la caracterización en palabras del elemento al que esté asociado.

Color: Valor numérico en formato RGB(Rojo:[0-255],Verde:[0-255],Azul:[0-255])⁶⁹.

Cual: Valor entero que sirve de índice o ubicación dentro de la estructura interna que almacena un tipo específico de elemento componente del modelo.

Se evalúa en cada llamada para determinar si el valor referenciado se encuentra dentro del rango de índices válido de la estructura interna de la que hace referencia.

Paso: Valor entero comprendido entre cero y el número de divisiones del intervalo total de tiempo empleado en la simulación.

La organización de las funciones aquí presentada se fundamenta en la estructura presentada de las tablas A.10 en la página 182 y A.11 en la página 183.

Para aquellos métodos identificados con ~~o~~ en las tablas de relación, se hará evidente en su descripción la condición por la cual pueden afectar negativamente la simulación.

A.5.7.1. Manejo de un proyecto

En general estos métodos pertenecen a la etapa de construcción del modelo.

⁶⁹RGB, iniciales de Red, Green, Blue.

1. [1] Function ProyectoNuevo(const Nombre: WideString): Integer; safecall;

Crea un proyecto usando la ubicación e identificador de archivo especificados por Nombre, no permite la asignación de la Clave o contraseña de acceso del proyecto. En caso de error esta función siempre retorna cero. Posibles razones de falla al llamar esta función: referenciar un directorio que no existe o crear un archivo con extensión diferente a *.prh⁷⁰.

Solo se crea un proyecto por cada llamada exitosa de conexión con la interfaz OLE. Es posible crear un nuevo proyecto en la instancia de automatización en uso, siempre y cuando el proyecto anterior haya sido previamente cerrado.

2. [2] Procedure ProyectoAbrir(const Nombre, Clave: WideString); safecall;

Recupera un modelo previamente existente según los datos especificados por Nombre. Al abrir un proyecto homos mediante la interfaz, se tiene en cuenta si existe una Clave asignada. Esta debe ser pasada como parámetro para realizar la apertura efectiva del documento. Si no se ha asignado clave alguna al proyecto, se pasa una cadena vacía.

Hacer uso indebido de este procedimiento puede restablecer la totalidad del proyecto a valores distintos de los que han sido editados o configurados mediante el uso de la interfaz una sola operación, perjudicando la estructura lógica del modelo y sus resultados.

3. [3] Procedure ProyectoGuardar; safecall;

Realiza el almacenamiento de la información que compone el modelo mediante la sobreescritura del archivo que contiene el proyecto actualmente abierto.

4. [4] Procedure ProyectoCerrar(Guardar: WordBool); safecall;

Realiza el cierre del proyecto actual. El parámetro Guardar determina si se debe salvar o no el documento antes del cierre. Durante las experiencias de manejo con la interfaz se presentaron algunos inconvenientes con esta función, si se desea salvar el trabajo realizado se sugiere usar ProyectoGuardar antes de cerrar el proyecto. Este método actualiza internamente la lista de documentos recientes.

Hacer uso indebido de este procedimiento puede liberar antes de tiempo los recursos usados por el proyecto. Las referencias a objetos internos creados desaparecen, lo que produce violaciones de acceso y la caída de la aplicación.

A.5.7.2. Creación de Clases objeto

1. [5] procedure ClaseObjetoCrear(
const Nombre, Imagen: WideString; Color: Integer;
const Descripcion: WideString); safecall;

Método que crea las clases objeto del modelo. Si no se establece una imagen, se usa el recurso establecido por defecto, una imagen de franjas amarillo y rosado incluida en el CdPS en ...\\Cd Homos\\Homos\\Recursos\\Imagen Nueva 2.bmp.

⁷⁰Se produce una violación de acceso.

No existe límite práctico al número de clases objeto que pueden ser creados en un modelo.

A.5.7.3. Configuración del ambiente

1. [6] `property ColorCeldas: Integer read Get_ColorCeldas write Set_ColorCeldas;`
Usa a function `Get_ColorCeldas: Integer; safecall; & procedure Set_ColorCeldas(Value: Integer); safecall;`
Permite leer y establecer el color de fondo del ambiente.
2. [7] `property TipoLimite: Integer read Get_TipoLimite write Set_TipoLimite;`
Hace uso de function `Get_TipoLimite: Integer; safecall; & procedure Set_TipoLimite(Value: Integer); safecall;`
Permite establecer el tipo de límite del ambiente de simulación., el rango de datos válidos para este parámetro del modelo es: Cíclico = 1, Finito = 2 e Infinito = 3; valores fuera de este intervalo son ignorados.
Afecta el desarrollo del modelo modificando el comportamiento del borde del ambiente durante la simulación.
3. [8] `property NumeroFilas: Integer read Get_NumeroFilas write Set_NumeroFilas;`
Usa a function `Get_NumeroFilas: Integer; safecall; & procedure Set_NumeroFilas(Value: Integer); safecall;`
Permite leer y establecer el número de filas del ambiente.
4. [9] `property NumeroColumnas: Integer read Get_NumeroColumnas write Set_NumeroColumnas;`
Usa a function `Get_NumeroColumnas: Integer; safecall; & procedure Set_NumeroColumnas(Value: Integer); safecall;`
Permite leer y establecer el número de columnas del ambiente.

Hacer uso indebido de los métodos `NumeroFilas` y `NumeroColumnas` puede restablecer las dimensiones de la matriz que sirve de base al ambiente. El rango de valores permitido para el parámetro `Value` está entre 1 y 254, pero no se hace validación de esta condición en el código de las funciones. Se recomienda no realizar cambios a estos valores durante la simulación.

A.5.7.4. Configuración del paisaje

1. [10] `procedure ObjetosInsertar(
const ClaseObjeto, TipoUbicacion: WideString; Cuantos,
x, y: Integer); safecall;`
Introduce en el ambiente `Cuantos` [n objetos] de una `claseobjeto` en un `TipoUbicacion` específico en el ambiente.
Los datos válidos de posicionamiento de objetos para `TipoUbicacion` son:

“*Uniforme*”: Distribuidos a distancias iguales en el ambiente.

“*Aleatorio*”: En cualquier punto sobre el ambiente.

“*Agrupado*”: Cercanos a un punto específico establecido por los parámetros x y y . Una ubicación dentro del ambiente se refiere a una *posición física* dentro de la matriz de datos. x , y solo tienen significado en esta forma de ubicación, se recomienda hacer sus valores cero para los demás modos.

Este método no incorpora controles respecto a Cuantos elementos pueden ser introducidos, es una fuente potencial de error durante la simulación.

TipoUbicacion debería ser un valor numérico entero.

A.5.7.5. Creación de reglas

Todas las reglas de transición comparten las mismas características básicas:

- Un nombre único que la identifique claramente de las demás reglas contenidas en el modelo, se sugiere para esto la notación “Nombreobjeto.Acción.Subacción.Subsubacción...” para reforzar la idea de objetos y métodos en la herramienta.⁷¹
- Definición de los valores de probabilidad y prioridad que caracterizan a la regla.
- Lista de una o más clases objeto relacionadas según el tipo de regla.
- Una descripción lo más objetiva posible de la acción de regla.

La forma general de llamado del método de creación de las reglas es:

Reglacrear *TAL* (Nombre, Probabilidad, Prioridad, [Dirección,] ClasesObjeto, Descripción)

Donde:

Probabilidad Es un valor entero entre 0 y 100 que se asume como de porcentaje. Siempre es evaluado para determinar si se encuentra dentro del rango válido de valores.

Prioridad Es un valor entero entre 0 y 255 que resuelve los conflictos de aplicación de reglas durante el proceso de simulación, cuando varias reglas de transición son posibles simultáneamente. No se hacen validaciones en la interfaz sobre este valor. Para la regla Decadencia este su valor es fijo y es igual a cero, por esa razón no hace parte de su definición.

⁷¹Como se propone en la sección A.4.2.13 en la página 156.

Direccion Es un valor entero válido solo entre 1 y 255, donde la suma de los valores mostrados en la figura A.16 en la página siguiente representa las posibles orientaciones en el vecindario hacia las cuales aplica la acción de la regla. Por ejemplo, para configurar que en una regla de movimiento haya desplazamiento lateral en ambos sentidos del eje X, el parámetro **Direccion** debería ser igual a $132 = 128 + 8$. Nótese que cero no es un dato válido para este parámetro.

Figura A.16: Valores del parámetro **Direccion**

1	2	4
128		8
64	32	16

Los números en la matriz representan las identificaciones de las distintas ubicaciones posibles en el vecindario aledaño a la celda central en evaluación.

Fuente: Autor.

En cada caso, el parámetro **ClaseObjeto1** de las reglas de transición representa la clase primaria de evaluación, el sujeto principal de la acción descrita por la regla. No existe límite definido en la metodología o en la aplicación al número de reglas que pueden ser creadas en un modelo.

A.5.7.6. Reglas de transición implementadas en Homos 1.0

1. [11] procedure ReglaCrearDecadencia(
const Nombre: WideString;Probabilidad: Integer;
const ClaseObjeto,
Descripcion: WideString);safecall;
2. [12] procedure ReglaCrearTransformacion(
const Nombre: WideString;Probabilidad, Prioridad: Integer;
const ClaseObjeto1, ClaseObjeto2,
Descripcion: WideString); safecall;
3. [13] procedure ReglaCrearMovimiento(
const Nombre: WideString;Probabilidad, Prioridad, Direccion, NumeroCeldas: Integer;
const ClaseObjeto,
Descripcion: WideString); safecall;
4. [14] procedure ReglaCrearExpansion(
const Nombre: WideString;Probabilidad, Prioridad, Direccion: Integer;
const ClaseObjeto,
Descripcion: WideString); safecall;
5. [15] procedure ReglaCrearRetraccion(
const Nombre: WideString;Probabilidad, Prioridad, Direccion: Integer;
const ClaseObjeto,
Descripcion: WideString); safecall;
6. [16] procedure ReglaCrearNeutralizacion(
const Nombre: WideString;Probabilidad, Prioridad: Integer;
const ClaseObjeto1, ClaseObjeto2,
Descripcion: WideString); safecall;
7. [17] procedure ReglaCrearEliminacion(
const Nombre: WideString;Probabilidad, Prioridad: Integer;
const ClaseObjeto1, ClaseObjeto2,
Descripcion: WideString); safecall;
8. [18] procedure ReglaCrearAlteracion(
const Nombre: WideString;Probabilidad, Prioridad: Integer;
const ClaseObjeto1, ClaseObjeto2, ClaseObjeto3,
Descripcion: WideString); safecall;
9. [19] procedure ReglaCrearReproduccion(
const Nombre: WideString;Probabilidad, Prioridad: Integer;
const ClaseObjeto1, ClaseObjeto2, ClaseObjeto3,
Descripcion: WideString); safecall;

Las definiciones de parámetros de las reglas de transición presentadas pertenecen a las versiones 1.0.0.23 ... 1.0.0.50 de la herramienta software. En la sección E.3.2 en la página 586 se presentan la forma final de los parámetros para la última versión disponible del software, la 1.0.0.73.

A.5.7.7. Recuperación y modificación de información sobre reglas del modelo

1. [20] `function ReglaDarProbabilidad(const Nombre: WideString): Double; safecall;`

Retorna el valor de la probabilidad asignada a la regla identificada con Nombre.

2. [21] `procedure ReglaAsignarProbabilidad(const Nombre: WideString; Probabilidad: Double); safecall;`

Establece el valor de la probabilidad de ejecución de la regla referenciada por Nombre. Este método hace posible que se pueda implementar durante la simulación reglas de transición con valores de probabilidad variables, los cambios a estos valores se realizan mediante llamados regulares durante la etapa de simulación del modelo.

Para realizar un cambio exitoso del valor de probabilidad este debe estar entre 0 % y 100 %, en otro caso se ignora la petición. A diferencia del parámetro Probabilidad en cada una de los llamados de creación de reglas, esta función permite establecer valores de probabilidad tanto enteros como flotantes.

Este método debería ser una función que retorne un valor de error en caso de fallo en el cambio del valor de probabilidad, por ejemplo al hacer referencia a una regla que no existe en el modelo.

Estas funciones pueden inhibir o permitir la ejecución de una o más reglas de transición del modelo según el diseño previsto por el modelador. La consecuencia inmediata de su aplicación es la alteración del comportamiento global del modelo en el sentido específico dado por esas reglas.

A.5.7.8. Manejo de escenarios

1. [22] `procedure EscenarioCrear(const Nombre: WideString; NumeroPasos: Integer); safecall;`

Establece el nombre y la cantidad de pasos a usar de un escenario. El número de pasos puede ser cualquier valor entero, internamente no se discrimina si es positivo o no.

2. [23] `procedure EscenarioSeleccionar(Cual: Integer); safecall;`

Permite seleccionar el escenario identificado con el valor indicado en Cual. Esta debería ser una función que retorne un valor de error en caso de fallo en la selección del escenario.

Hacer uso indebido de esta función puede terminar la simulación si se elige dinámicamente un escenario inexistente.

A.5.7.9. Manejo del simulador

1. [24] procedure SimulacionIniciar(RetornarAmbiente: WordBool); safecall;

Metodo que inicia la simulación solo si existe un escenario previamente seleccionado y si ya se finalizaron los pasos de una simulación anterior. RetornarAmbiente establece si se debe recuperar la configuración del paisaje inicial del ambiente.

2. [25] procedure SimulacionParar; safecall;

Cada llamada a este método cambia el estado de ejecución del simulador: Detiene el avance si está ejecutándose, reinicia el simulador si se encuentra detenido.

3. [26] procedure SimulacionTerminar; safecall;

Termina el proceso de simulación activo. Requiere de un breve lapso de tiempo posterior a su ejecución para la normalización del estado interno del simulador. Se requiere terminar la ejecución de la simulación para realizar un cierre correcto del proyecto. Esta función vuelve el ambiente a su estado inicial si RetornarAmbiente es verdadero.

A.5.7.10. Funciones de visualización de elementos gráficos

1. [27] property Visible: WordBool read Get_Visible write Set_Visible;

Usa a function Get_Visible: WordBool; safecall; & procedure Set_Visible(Value: WordBool); safecall;

Hace visible todo el formulario principal de la aplicación, no solo el ambiente. Esta función hace que la interfaz no contemple mecanismos gráficos de interacción. hace posible una forma de control combinado entre manual y automatizado del modelo script.

2. [28] Procedure AmbienteActualizar; safecall;

Permite redibujar el ambiente, para reflejar los cambios realizados.

3. [29] procedure VentanaAmbiente; safecall;

4. [30] procedure VentanaDiagrama; safecall;

Los metodos VentanaAmbiente y VentanaDiagrama presentan dentro del espacio MDI interno del formulario principal de la aplicación las ventanas maximizadas del ambiente y el diagrama BOR. En cada llamado, se da enfoque a la ventana correspondiente.

A.5.7.11. Funciones de interacción con el usuario

1. [31] procedure Mensaje(const Texto: WideString); safecall;

Presenta el texto en un cuadro de diálogo modal que interrumpe la ejecución del simulador. Se requiere de la atención del usuario para continuar el proceso de simulación del modelo.

A.5.7.12. Manejo de referencias a elementos componentes del modelo

Las funciones de referencia listadas a continuación se retornan el identificador o nombre del elemento referenciado por el parámetro CUAL en cada tipo de elemento componente del modelo.

En caso de error⁷², las funciones retornan “@”.

1. [32] `function ClaseObjeto(Cual: Integer): WideString; safecall;`
2. [33] `function Regla(Cual: Integer): WideString; safecall;`
3. [34] `function Escenario(Cual: Integer): WideString; safecall;`

A.5.7.13. Recuperación de información sobre elementos del modelo

Estos métodos retornan la cantidad ítems presente de cada tipo de elemento componente del modelo.

1. [35] `property CuantasClasesObjeto: Integer read Get_CuantasClasesObjeto;`
`Usa a function Get_CuantasClasesobjeto: Integer; safecall;`
2. [36] `property CuantasReglas: Integer read Get_CuantasReglas;`
`Usa a function Get_CuantasReglas: Integer; safecall;`
3. [37] `property CuantosEscenarios: Integer read Get_CuantosEscenarios;`
`Usa a function Get_CuantosEscenarios: Integer; safecall;`
4. [38] `function ObjetosDarCantidad(Paso: Integer): OleVariant; safecall;`

Este método retorna por cada paso de simulación ya calculado, un vector de tipo Variant con la cantidad de objetos que hay por cada clase objeto presente en el modelo. Retorna -1 cuando se prueba un valor de Paso mayor que el número de pasos actualmente realizados por el simulador.

A partir de esta función se obtienen los resultados de simulación usados durante la etapa de presentación de resultados.

También hacen parte de este conjunto las funciones de referencia y los métodos `ReglaDarProbabilidad`, `ColorCeldas`, `TipoLimite`, `NumeroFilas` y `NumeroColumnas`; Estos últimos solo en modo de lectura. En general se sugiere no hacer uso directo de las funciones `Get_` y `Set_` existentes en la interfaz.

⁷²Por ejemplo, si `Cual` contiene un valor fuera del intervalo del índice interno de objetos

A.5.7.14. Otras funciones presentes en la interfaz

1. [39] `procedure Cerrar; safecall;`

Función no documentada en el manual, que presenta el conteo de instancias presentes del programa en la capa COM y termina la ejecución mediante un llamado al método interno `FHomos.close`. No realiza operaciones de descarga de elementos en memoria, ni realiza otro tipo de operaciones de liberación. Produce una violación de acceso cada vez que es ejecutada, es un método por renovar o remover de la interfaz.

A.5.8. Método de construcción de modelos script

Haciendo referencia al análisis previamente realizado y teniendo en cuenta las ideas presentadas en el anexo A.4.3 en la página 157, se ilustra en el figura A.17 un orden lógico de uso de las funciones disponibles en la interfaz para la construcción de modelos scripts de Homos 1.0.

La implementación real del proceso se lleva a cabo mediante el uso de un lenguaje de programación con capacidades de llamado a funciones OLE/COM/COM+. Esta forma de extensión programable de Homos permite implementar controles complejos de simulación y de presentación de resultados que no es posible realizar directamente con la herramienta software.

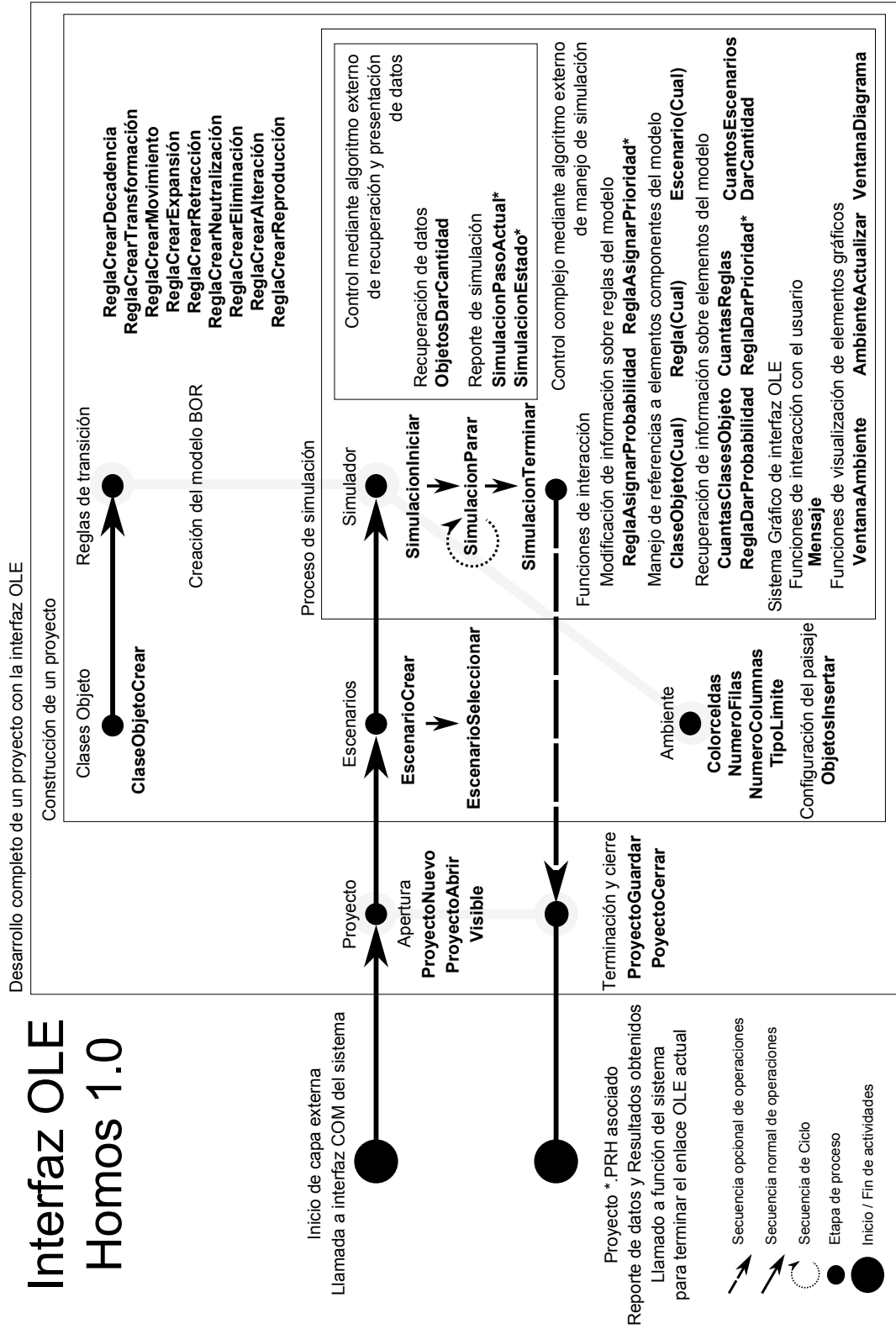
En este proyecto se eligió a PHP como el lenguaje de programación con el cual realizar las pruebas de uso de la automatización. Aunque los distintos comandos presentados en los scripts son específicos de PHP, tienen sus equivalentes funcionales en otros lenguajes que implementen llamadas a la interfaz COM.

A partir de las etapas de construcción de un modelo en la figura A.5.5 en la página 177, se establece el método de creación de modelos representado en figura A.17. El orden de llamados de las funciones está dado según la lógica necesaria para crear los distintos elementos del modelo y llevar a cabo su ejecución. El método tiene los siguientes pasos:

1. Instanciar la automatización y crear el archivo de proyecto.
2. Caracterizar los elementos del modelo BOR:
 - a) Creación del modelo de transiciones: Consiste en la creación de las clases del modelo y luego de las reglas que las relacionan.
 - b) Caracterización del ambiente.
 - c) Creación y configuración de los escenarios.
3. Configurar el simulador y realizar el ciclo de simulación.
4. Presentar los resultados. Este es un evento externo que no se integra directamente en la figura A.17.
5. Cierre del proyecto y terminación de la automatización.

Interfaz OLE Homos 1.0

Figura A.17: Funciones y etapas de creación de un modelo mediante la interfaz OLE



Al final del proceso quedan: el modelo script desarrollado + el archivo de proyecto *.prh + modelo BOR implementado y los datos generados por la simulación.

A.5.9. Detalles de implementación práctica de modelos script

A.5.9.1. Depuración de errores en scripts

En caso de error en cualquier etapa del proceso de ejecución de un modelo, se produce la interrupción inmediata del script y cierre de la instancia en uso de la automatización. Durante el proceso de construcción del modelo,

Es posible implementar desde PHP un control de errores en los métodos de tipo función⁷³ de la interfaz e interrumpir la ejecución del script en caso de error mediante el comando Die de PHP. El motor del lenguaje ejecuta la línea de código: si se produce falla, se hace efectiva la ejecución del comando Die relacionado a la instrucción mediante el operador booleano Or. El comando Die presenta en la consola el mensaje texto establecido como parámetro y termina la ejecución del script.

En los distintos scripts incorporados en este documento se hace más énfasis en lo requerido para la realización del modelo que en formas de detección y control de errores.

Script A.1: Control elemental de errores en scripts PHP

```
1 //si no es posible instanciar la interfaz , salir del script
2 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema"); //OK!
3 ...
4 //método sugerido de control de existencia de archivo
5 //si no es posible abrir el archivo , salir del script
6 $archivo="z:\directorio\un_proyecto.prh";
7 if(!file_exists($archivo)) { echo "no existe el archivo $el\n"; exit; }//control externo
8 //
9 //NO USAR ESTO! se produce una violación de acceso!
10 $h1->proyecnuevo($archivo) or die("Error al abrir $archivo")or die("falla al crear"); //es una
    funcion pero no retorna valores
11 $h1->proyectoabrir($archivo,"") or die("Error al abrir $archivo")or die("falla al abrir"); //es un
    procedimiento
12 ...
```

A.5.9.2. Estructura básica de un script

Se describe a continuación el código PHP con el cual se realizan las distintas operaciones de construcción del modelo script. Todo script que use esta interfaz requiere de los siguientes pasos básicos:

1. Hacer un llamado e instanciar una automatización OLE, comprobando el éxito de la operación.
2. Llenar la estructura recién instanciada. Esto se hace abriendo un modelo existente o creando uno desde cero.

⁷³Esta opción no es posible en métodos de tipo procedimiento, ya que no retornan ningún valor con el cual hacer comparaciones de estado de error. Se produce una violación de acceso y la caída de la automatización al intentar realizar esta operación de control sobre este tipo de métodos.

3. Establecer los distintos elementos del modelo, seguido de la definición del cuerpo principal de control del script. Dentro de las operaciones de control se incluye el ciclo de simulación principal y el de presentación de resultados.
4. Cerrar la instancia y terminar la ejecución del script

El paso 3 es el más complejo y se abordará con detalles en apartados posteriores. Se presentan a continuación porciones de código que realizan los pasos 1,2 y 4:

Script A.2: Creación y terminación de una instancia de automatización

```

1 //instanciación de una automatización OLE
2 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
3
4 //Creación de un modelo en la interfaz OLE
5 //Dos métodos posibles , solo una forma por proyecto:
6 //1) Crear un proyecto
7 $h1->proyectoNuevo('C:\Program Files\UIS\Homos\test_predador_presa.prh');
8 //ó
9 //2) Abrir un proyecto existente
10 $h1->proyectoAbrir('C:\Program Files\UIS\Homos\predador_presa.prh',''); //proyecto sin clave
11 ...
12 //grueso de código del script
13 ...
14 //4) Opciones de cierre del proyecto actual
15 //4a) Algo de paranoia
16 $h1.proyectoGuardar(); //guardar el proyecto antes de cerrar
17 $h1.proyectoCerrar(false); //cerrar la interfaz
18 //
19 //4b) Operación integrada
20 $h1.proyectoCerrar(true); //guardar el modelo y cerrar la interfaz , no recomendado
21
22 //Eliminación completa de referencia a automatización
23 $h1=null; //hacer nulo $h1
24 unset($h1); //borra la referencia a $h1

```

Con respecto al cierre del proyecto actual, se encontró durante la realización de algunas pruebas que el guardado automático al llamar a ProyectoCerrar(True) falla aleatoriamente. Por esta razón se incluye en el listado el cierre *paranoico* de la aplicación.

A.5.9.3. Control de simulación

El único requisito real para iniciar la simulación del modelo es crear y seleccionar al menos un escenario. Esta es la forma que ofrece la interfaz para asignar el número de pasos de simulación a realizar. En el listado A.3 se muestran tres de muchas formas posibles de control de la simulación mediante automatización, P.ej:

- Sin ciclo de control.
- Con un ciclo que se termina cuando se cumpla una determinada cantidad de interrupciones programadas con un valor de espera constante entre pausas.
- Con ciclo hasta el paso final de simulación previsto en el escenario actual.

En los distintos ejemplos descritos en este documento se hace uso de formas de control activa de la simulación del modelo.

Script A.3: Ejemplos de ciclos de control

```
1 FORMAS DE CONTROL DE SIMULACIÓN
2 a) Simulación con control de tiempo externo, sin ciclo
3 //configuración del escenario y del paso de simulación
4 $numpasos=15000; //se supone una larga ejecución, 0<=numpasos<=2147483647
5 $h1->escenariocrear("Escenario1",$numpasos1);
6 ...
7 $h1->simulacioniniciar(false);
8 sleep(120); //120 segundos = 120/60 = 2 minutos de simulación
9 $h1->simulacionterminar; //terminación explícita del ciclo de simulación
10 usleep(10000); //tiempo de normalización, 1/100 de segundo
11
12 b) Simulación por intervalos, presentación de mensajes intermedios
13 ...
14 $h1->simulacioniniciar(false);
15 $interrupciones = 5; //intervalos programados
16 $avance = 10; //tiempo de avance en segundos
17 for($qi=0;$qi<$interrupciones;$qi++) {
18     sleep($avance); //tiempo suficiente realizar algunas iteraciones
19     $h1->simulacionparar; //detiene la simulación
20     $h1->mensaje("se avanzaron $avance segundos"); //cuadro modal, interrumpe la ejecución de
        la interfaz y el script
21     $h1->simulacionparar; //reanuda la simulación
22 }
23 $h1->simulacionterminar; //terminación explícita del ciclo de simulación
24 usleep(10000); //tiempo de normalización
25
26 c) Simulación hasta el número de pasos previsto en el escenario
27 $numpasos=500;
28 $h1->escenariocrear("Escenario",$numpasos);
29 ...
30 echo "Ciclo de simulación<br>\n";
31 $pasofinal = $numpasos-1;
32 $ciclo=0;
33 $ultimodato = $h1->objetosdarcantidad($pasofinal);
34 $h1->simulacioniniciar(false);
35 while($ultimodato[0]==-1){ //simular hasta terminar
36     //pausa entre evaluaciones
37     usleep(10000); //1 seg = 1000000 microsegundos, 100 evaluaciones por segundo
38     //determinar si el simulador ya llegó al paso final
39     $ultimodato = $h1->objetosdarcantidad($pasofinal); //verificar el valor del último paso de
        simulación
40     $ciclo++;
41 }
42 echo "Se realizaron $ciclo ciclos<br>";
43 $h1->simulacionterminar; //terminación explícita del ciclo de simulación
44 usleep(10000); //tiempo de normalización
```

La última construcción presentada para el manejo del simulador presenta una solución a una ausencia importante dentro de las funciones implementadas en la interfaz OLE: Los diseñadores de Homos 1.0 no contemplaron una función que permitiera conocer directamente el valor del paso actual de simulación. Se requiere de este valor para la realización de formas de control de simulación más avanzadas, como se verá mas adelante en este documento.

La forma *c* de control de simulación presentada funciona ya que los valores poblacionales de las clases objeto en el Homos 1.0 son siempre positivos. La función `ObjetosDarCantidad` retorna un -1 cuando se evalúa un valor de paso de simulación que aun no ha sido calculado. Para este ciclo de control particular, se evalúa periódicamente si el valor almacenado en el último paso de simulación previsto es diferente de -1. Si este valor ha cambiado, entonces se ha alcanzado el paso final de simulación.

En el ejemplo presentado, se esperará hasta un $\frac{1}{100}$ de segundo por ciclo para determinar si se ha alcanzado el último paso previsto en la configuración del escenario.

El tiempo entre evaluaciones permite que durante un ciclo de evaluación puedan realizarse 0, 1, 2 o más pasos, según sea el tiempo requerido para completar un paso de simulación⁷⁴. Un control fino de la simulación, como por ejemplo, permitir el avance del simulador paso a paso o en un número fijo de pasos, mejoraría notablemente el manejo de la simulación mediante automatización.

A.5.9.4. Ciclo de presentación de resultados

La etapa de presentación de los resultados está a cargo de la aplicación que implementa la interfaz OLE. Los datos de simulación se recuperan en forma vectorial según el orden de creación de las clases-objeto presentes en el modelo. Homos retorna un vector de datos por cada paso de simulación realizado.

Script A.4: Ciclo de presentación de resultados

```
1 ...
2 echo "\nPresentación de resultados<br>";
3 echo "<table>\n<tr><td>Paso</td>";
4 $numclases=$h1->cuantasclasesobjeto;
5 for($i=0;$i<$numclases;$i++) echo "<td>".$h1->claseobjeto($i)."</td>";
6 echo "\n";
7 for($paso=0;$paso<$numpasos;$paso++){
8     $resultados = $h1->objetosdarcantidad($paso);
9     echo "<tr><td>$paso</td>\n";
10    foreach($resultados as $item) echo "<td>$item</td>\n";
11    echo "</tr>\n";
12 }
13 echo "</table>\n";
14 ...
```

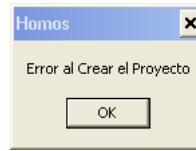
A.5.9.5. Uso de la interfaz y mensajes de error más comunes

Estos son algunos de los mensajes de error comunes que pueden presentarse durante la creación de modelos script con la interfaz OLE.

⁷⁴Gran número de reglas en el modelo y dimensiones amplias del ambiente configuran pasos de simulación lentos.

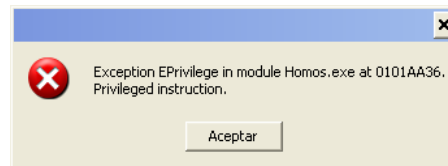
Figura A.18: Mensajes de error en uso de interfaz OLE

(a) Mensaje de error por fallo en la creación del proyecto



Fuente: Autor.

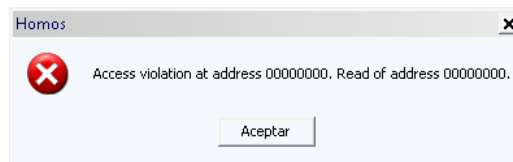
(b) Error por llamado a funciones get_ o set_ de la interfaz



Luego de este error se produce terminación inmediata de la simulación. Evitar realizar llamados a estas fuciones.

Fuente: Autor.

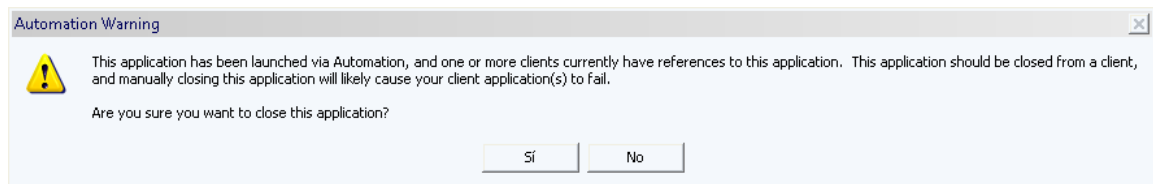
(c) Proceso de simulación no terminado, sin espera intermedia antes de cierre de automatización



Luego de este error se produce terminación inmediata de la simulación.

Fuente: Autor.

(d) Error por terminación externa de la instancia de automatización



En la figura A.18c, se muestra el caso cuando se desea cerrar la automatización sin dar tiempo suficiente para el terminación de la simulación. Este problema se soluciona adicionando una breve pausa justo después del fin de la simulación.

Fuente: Autor.

A.5.10. Ejemplo de implementación de un modelo script

En este modelo se implementa una dinámica de cambio de opinión. Las personas pueden elegir uno de los dos bandos disponibles y cualquier persona puede influenciar a otra, haciendo que el influenciado opine igual que el que lo influye. El modelo implementa 2 clases de objetos en un espacio de 10 celdas cuadradas. Se llena aleatoriamente todo el espacio con cantidades iguales de objetos para evitar definir otras reglas distintas a la acción de alteración del otro y para facilitar el cambio

de opinión por contacto inmediato. Al finalizar la ejecución se reportan los valores poblacionales obtenidos. En este script se hace uso de los elementos de definición y control previamente vistos.

Script A.5: Implementación de un modelo de voto simple

```

1 //proyecto de Ejemplo: modelo de voto simple
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = 'c:\phplets\OLE'; //directorio por defecto
4 $phplet_slf_ = "OLE_EJ_voto.php"; //nombre archivo
5 // $phplet_lib_ = ''; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 echo "Modelo del voto<br>"; //m1
9 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
10 $h1->proyectonuevo('C:\Program Files\UIS\Homos\OLE_EJ_voto.prh');
11 $h1->visible=false;
12 echo "Crear clases-objeto<br>";
13 $o1="V1";
14 $o2="V2";
15 $h1->claseobjetocrear($o1,"C:\Program Files\UIS\Homos\Recursos\Tortuga.bmp",hexdec(strrev("00FF00"))
16     ),"Votante 1");
17 $h1->claseobjetocrear($o2,"C:\Program Files\UIS\Homos\Recursos\Conejo.bmp",hexdec(strrev("000000"))
18     ),"Votante 2");
19 echo "Crear reglas<br>";
20 $r1="V1,V2:V1,V1";
21 $r2="V2,V1:V2,V2";
22 //reglas de cambio de opinión de los votantes, influencias entre votantes
23 $h1->reglacrearalteracion($r1,10,1,$o1,$o2,$o1,$r1, "Un votante de opinión V1 hace que un opositor
24     cambie de parecer, de V2 a V1");
25 $h1->reglacrearalteracion($r2,10,1,$o2,$o1,$o2,$r2, "Un votante de opinión V2 hace que un opositor
26     cambie de parecer, de V1 a V2");
27
28 echo "Configuración del ambiente<br>"; $ciclico=1; $finito=2; $infinito=3;
29 $nfilas=10;
30 $ncmnas=10;
31 $nceldas=$nfilas*$ncmnas;
32 $h1->numerofilas=$nfilas;
33 $h1->numerocolumnas=$ncmnas;
34 $h1->tipolimites=$ciclico;
35
36 echo "Crear Escenario<br>";
37 $numpasos=300; //duracion de la simulación
38 $h1->escenariocrear("Ambiente político normal",$numpasos);
39 $h1->escenarioseleccionar(0);
40 $h1->objetosinsertar($o1,"Aleatorio",$nceldas/2,0,0);
41 $h1->objetosinsertar($o2,"Aleatorio",$nceldas/2,0,0);
42 $h1->colorceldas=hexdec("EEEEEE");
43
44 echo "Actualizar el ambiente<br>";
45 $h1->ambienteactualizar;
46 // $h1->visible=true;
47
48 echo "\nCiclo de simulación<br>";
49 $ciclo=0;
50 $pasofinal=$numpasos-1;
51 $ultimodato = $h1->objetosdarcantidad($pasofinal);
52 $h1->simulacioniniciar(false);
53 while($ultimodato[0]==-1){ //simular hasta terminar
54     usleep(100000); //tiempo de espera entre evaluaciones
55     $ciclo++;
56     $ultimodato = $h1->objetosdarcantidad($pasofinal);
57 }
58
59 //finalización de simulaciones
60 $h1->simulacionterminar;
61 usleep(100000); //tiempo para cierre
62
63 echo "Se realizaron $ciclo ciclos de evaluación del estado de simulación del modelo 1<br>";
64
65 echo "\nPresentación de resultados del modelo 1<br>";
66 echo "<table>\n<tr><td>Paso</td>";
67 $numclases=$h1->cuantasclasesobjeto;
68 for($i=0;$i<$numclases;$i++) echo "<td>".$h1->claseobjeto($i). "</td>";
69 echo "\n";
70 for($paso=0;$paso<$numpasos;$paso++){
71     $resultados = $h1->objetosdarcantidad($paso);
72     echo "<tr><td>$paso</td>\n";

```

```

69         foreach($resultados as $item) echo "<td>$item</td>\n";
70         echo"</tr>\n";
71     }
72     echo"</table>\n";
73
74     echo "Cierre del enlace y terminación del script<br>\n";
75     $h1->proyectoguardar; //llamada explícita al proceso de guardado del modelo
76     $h1->proyectocerrar(false); //cerrar y salir
77
78     $h1 = null;
79     unset($h1);

```


A.5.11. Extensión de capacidades de modelado mediante programación

En cada listado de programa incluido en este apartado solo se resaltar  la secci n de c digo que establece la extensi n funcional. El listado completo de los scripts presentados en esta secci n se incluye en la secci n E.3 en la p gina 584. Se ofrece con la interfaz de automatizaci n un medio que hace posible ampliar las posibilidades de representaci n y uso de la herramienta software. Se presentan en este apartado ejemplos de cambio en el comportamiento de la simulaci n por defecto mediante programaci n en el script que define el modelo y un ejemplo de simulaci n simult nea de dos modelos.

A.5.11.1. Implementaci n una regla de transici n con probabilidad din mica y pulso de regeneraci n poblacional

El c digo completo del script se encuentra en la secci n E.16 en la p gina 594.

Script A.6: Ejemplo de modelo con pulso

```
1 echo "Actualizar el ambiente<br>";
2 $h1->ambienteactualizar;
3 $h1->visible=true;
4
5 $actualizar_regla=0; //variable de control de pulso de regeneracion
6
7 echo "Ciclo de simulaci n<br>";
8 $pasofinal=$numpasos-1;
9 $ciclo=0;
10 $pospaso=0;
11 $ultimodato = $h1->objetosdarcantidad($pasofinal);
12 while($ultimodato[0]==-1){ //simular hasta terminar
13     if ($ciclo==0) { //iniciar/continuar la simulaci n
14         $h1->simulacioniniciar(false);
15     } else $h1->simulacionparar;
16     $ciclo++;
17 //tiempo de simulaci n por ciclo , pausa entre evaluaciones
18     usleep(62500); //1/16 seg | 1 seg = 1000000 microsegundos
19     $h1->simulacionparar; //detener
20
21 //determinar el paso actual
22     for($j=$pospaso;$j<$pasofinal;$j++){
23         $vec=$h1->objetosdarcantidad($j); //obtener los valores de paso indicados
24         if ($vec[0]==-1) break;
25     }
26     $pospaso=$j; //ultimo indice conocido
27
28 //realizar la operaciones con los datos de simulacion obtenidos
29     if ($j>0){
30         $actualdato = $h1->objetosdarcantidad($j-1); //obtener los datos del paso actual
31         if ($actualdato[0]==0){
32             $h1->reglaasignarprobabilidad($r4,100); //pulso de descristalizacion
33             $actualizar_regla=1;
34             $contador_solidificacion++;
35         }
36     } else
37         if($actualizar_regla){
```

A.5.11.2. Implementación de una regla de transición con probabilidad dinámica y retardo en cambio de información

El código completo del script se encuentra en la sección E.17 en la página 600.

Script A.7: Ejemplo de modelo con retardo

```
1
2 $h1->simulacioniniciar(false);
3 while($ultimodato[0]==-1){ //simular hasta llegar al paso final
4     $h1->simulacionparar; //pausa a la simulacion actual
5
6 //determinar el paso actual en ml
7     for($j=$pospaso;$j<$pasofinal;$j++){
8         $vec=$h1->objetosdarcantidad($j); //obtener los valores de paso indicados
9         if ($vec[0]==-1) break;
10    }
11    $pospaso=$j; //Último indice de paso conocido en h1
12
13    if ($j>0){
14        $actualdato = $h1->objetosdarcantidad($j-1); //obtener los datos del paso actual
15 //si se ha definido un valor de retardo entonces usar el codigo para ese proposito
16        if($rett>0){
17            if($ciclo<$rett+1){
18 //la cantidad de objetos en el ambiente es constante en este modelo, no hay eliminaciones
19                $retardo[] = $actualdato[0]/$nceldas; //llenar el vector de retardo
20            } else {
21                if($reti>$rett) $reti=0; //posicionamiento ciclico dentro de
                //retardo;
22 //probabilidad de alteracion en funcion de la poblacion contraria
23                $h1->reglaasignarprobabilidad($r1,1-$retardo[$reti]);
24                $h1->reglaasignarprobabilidad($r2, $retardo[$reti]);
25 //actualizacion del valor actual de poblacion
26                $retardo[$reti]=$actualdato[0]/$nceldas;
27 //puntero de avance en la tabla de retardo
28                $reti++;
29            }
30        }
31    }
32    $ciclo++;
33    $ultimodato = $h1->objetosdarcantidad($pasofinal);
34    $h1->simulacionparar; //continuar la simulacion
35 }
36 //vector con los ultimos valores de retardo calculados
37 print_r($retardo);
38
39 //finalización de simulaciones
40 $h1->simulacionterminar;
41 usleep(100000); //tiempo para cierre
42
43 echo "Se realizaron $ciclo ciclos de evaluación del estado de simulación del modelo 1<br>";
44
45 echo "\nPresentación de resultados del modelo 1<br>";
46 echo "<table>\n<tr><td>Paso</td>";
47 $numclases=$h1->cuantasclasesobjeto;
48 for($i=0;$i<$numclases;$i++) echo "<td>". $h1->claseobjeto($i). "</td>";
49 echo "\n";
50 for($paso=0;$paso<$numpasos;$paso++){
51     $resultados = $h1->objetosdarcantidad($paso);
52     echo "<tr><td>$paso</td>\n";
53     foreach($resultados as $item) echo "<td>$item</td>\n";
54     echo "</tr>\n";
55 }
```

A.5.11.3. Ejecución simultánea / paralela de dos o más modelos en un mismo script

Además de conocer los elementos que configuran cada proyecto, para controlar dos o más modelos se requiere:

- Una instancia de automatización por cada proyecto a manejar. Pueden crearse tantas instancias de Homos como sean necesarias. Las instancias de automatización no comparten datos entre ellas, así que el paso de datos entre los distintos modelos creados se realiza como una capa externa a nivel del script. Se da un ejemplo de creación de instancias en el listado A.2.
- Conocer el paso de simulación actual de cada modelo. Para conocer el valor del paso de simulación actual en un modelo, se realiza un ciclo de búsqueda secuencial desde paso = 0 hasta que se encuentre que el primer valor de paso en revisión no haya sido alcanzado por el simulador, cuando `ObjetosDarCantidad` retorna -1. Una vez que este paso ha sido encontrado, el ciclo de búsqueda termina: se hace que el valor encontrado sea el siguiente punto de inicio de búsqueda. Se mantienen estas actualizaciones por cada ciclo de control de simulación hasta que se encuentra el valor del paso final de simulación. Esta búsqueda secuencial es un paliativo para la ausencia de una función que permita determinar el valor actual del paso de simulación del modelo a controlar.

En los listados A.8 en la página 205 y A.8 en la página 205 se asume que se han realizado dos llamadas a la interfaz de automatización donde `$H1` y `$H2` son las referencias individuales a cada modelo.

El código completo del script se encuentra en el anexo E.18 en la página 612.

Script A.8: Definición de dos modelos en un mismo script

```

1 echo "Modelo 1:bacterias en una probeta<br>"; //m1
2 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
3 $h1->proyecnuevo('C:\Program Files\UIS\Homos\OLE_m1_bacterias.prh');
4 $h1->visible=false;
5 echo "Crear clases-objeto del modelo 1<br>";
6 $obj1="bacteria";
7 $h1->claseobjetocrear($obj1,"C:\Program Files\UIS\Homos\Recursos\ladrillo.bmp",hexdec(strrev("
  FF0000"))),"Bacteria");
8 echo "Crear reglas<br>";
9 $r1="Movimiento";
10 $r2="Reproducirse";
11 $r3="Morir";
12 $heb=50; //hostilidad entre bacterias
13 $h1->reglacrearmovimiento($r1,10 ,1,255,2,$obj1,"Movimiento de la bacteria");
14 $h1->reglacrearexpansion ($r2,0 ,1,255 ,,$obj1,"Orden de aumento de la cantidad de bacterias");
15 $h1->reglacrearretraccion($r3,$heb,1,255 ,,$obj1,"Orden de reducción de la cantidad de bacterias");
16 echo "Configuración del ambiente<br>"; $ciclico=1; $finito=2; $infinito=3;
17 $h1->numerofilas=50;
18 $h1->numerocolumnas=50;
19 $h1->tipolimite=$ciclico;
20 echo "Crear Escenario<br>";
21 $numpasos1=250;
22 $h1->escenariocrear("Recipiente Normal",$numpasos1);
23 $h1->escenarioseleccionar(0);
24 $h1->objetosinsertar($obj1,"Aleatorio",5,0,0);
25 $h1->colorceldas=hexdec("000000");
26 //echo "Actualizar el ambiente<br>"; $h1->ambienteactualizar; $h1->visible=true;
27
28 echo "\nModelo 2:factor ambiental sobre bacterias<br>"; //m2
29 $h2 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
30 $h2->proyecnuevo('C:\Program Files\UIS\Homos\OLE_m2_factor_ambiental.prh');
31 $h2->visible=false;
32 echo "Crear clases-objeto del modelo 2<br>";
33 $og1="Estado 1";
34 $og2="Estado 2";
35 $h2->claseobjetocrear($og1,"C:\Program Files\UIS\Homos\Recursos\agua.bmp",hexdec(strrev("0000FF")),
  "Estado 1");
36 $h2->claseobjetocrear($og2,"C:\Program Files\UIS\Homos\Recursos\ladrillo.bmp",hexdec(strrev("00FF00
  00")), "Estado 2");
37 echo "Crear reglas<br>";
38 //adaptación de un modelo de dinámica del voto
39 $rg1="E1->E2";

```

```

40 $rg2="E2->E1";
41 $rg3="E1.Movimiento";
42 $rg4="E2.Movimiento";
43 //previsión para la generación del objetos en el estado contrario , por si alguno desaparece durante
    la simulación
44 $rg5="E1.Generar";
45 $rg6="E2.Generar";
46 $h2->reglacrearalteracion ($rg1,10 ,1, $og2,$og1,$og2,"Paso de E1 a E2");
47 $h2->reglacrearalteracion ($rg2,10 ,1, $og1,$og2,$og1,"Paso de E2 a E1");
48 $h2->reglacrearmovimiento ($rg3,100,2,255,2,$og1, "Movimiento del objeto E1");
49 $h2->reglacrearmovimiento ($rg4,100,2,255,2,$og2, "Movimiento del objeto E2");
50 $h2->reglacreartransformacion($rg5,1 ,1, $og2,$og1, "Generar un E1 de E2");
51 $h2->reglacreartransformacion($rg6,1 ,1, $og1,$og2, "Generar un E2 de E1");
52 echo "Configuración del ambiente<br>";
53 $h2->numerofilas=15;
54 $h2->numerocolumnas=15;
55 $h2->tipolimite=$ciclico;
56 echo "Crear Escenario<br>";
57 $factor=8;//multiplicador del tiempo de simulación de m2 + exceso por imprevistos
58 $numpasos2=$numpasos1*$factor; //se establece un valor mayor que el definido en la simulación
    destino ml
59 $h2->escenariocrear("Recipiente",$numpasos2);
60 $h2->escenarioseleccionar(0);
61 $h2->escenarioseleccionar(0);
62 $h2->objetosinsertar($og1,"Aleatorio",50,0,0);
63 $h2->objetosinsertar($og2,"Aleatorio",50,0,0);
64 $h2->colorceldas=hexdec("EEEEEE");
65 //echo "Actualizar el ambiente<br>"; $h2->ambienteactualizar; $h2->visible=true;
66
67 echo "\nCiclo de simulación<br>";
68 $pasofinal1=$numpasos1-1;
69 $pasofinal2=$numpasos2-1;
70 $ciclo=0;
71 $pospasos1=0;
72 $pospasos2=0;

```

- Sincronizar el avance de los modelos: Aunque dos modelos tengan igual estructura y configuración es probable que difieran plenamente en la serie de valores aleatorios que usan internamente. Esto hace que se presenten diferencias en la duración del cálculo de un paso de simulación, haciendo que un modelo sea mas rápido que otro. Esta es la razón por la cual hay necesidad de sincronizar el avance entre modelos.

Para realizar el proceso de avance en el ciclo de control de dos o más modelos se dan los siguientes pasos: se inicia la simulación de cada instancia, se da un breve tiempo de ejecución y luego se detienen. posteriormente se hace uso de la función `ObjetosDarCantidad` para determinar el paso de simulación actual en cada modelo. De esta forma es posible implementar un control simultáneo de hasta de n modelos en un script. Se muestra en el listado A.9 en la página 206 un ejemplo de esta forma de control para el ejemplo del listado A.8.

El código completo del script se encuentra en el anexo E.18 en la página 612.

Script A.9: Ciclo de control principal y simulación síncrona de dos modelos

```

1
2 //avance sincronizado en ambos modelos
3 while($ultimodato[0]==-1){ //simular hasta terminar
4     if ($ciclo==0) { //iniciar/continuar la simulación
5         $h2->simulacioniniciar(false);
6         $h2->simulacionparar; //detener h2
7         $h1->simulacioniniciar(false);
8         $h1->simulacionparar; //detener h1
9     } else {
10         $h2->simulacionparar; //iniciar h2
11         $h2->simulacionparar; //detener h2

```

```

12             $h1->simulacionparar; //iniciar h1
13             $h1->simulacionparar; //detener h1
14         }
15         $ciclo++;
16
17         //determinar el paso actual en m1
18         for($j=$pospaso1;$j<$pasofinal1;$j++){
19             $vec=$h1->objetosdarcantidad($j); //obtener los valores de paso indicados
20             if ($vec[0]==-1) break;
21         }
22         $pospaso1=$j; //Último índice de paso conocido en h1
23
24         //determinar el paso actual en m2
25         for($k=$pospaso2;$k<$pasofinal2;$k++){
26             $vec=$h2->objetosdarcantidad($k); //obtener los valores de paso indicados
27             if ($vec[0]==-1) break;
28         }
29         $pospaso2=$k; //Último índice de paso conocido en h2
30
31         //realizar la operaciones con los datos de simulación obtenidos
32         if ($k>0){
33             $actualdato = $h2->objetosdarcantidad($k-1); //obtener los datos del paso actual
34         //alterar la probabilidad de reproducción de las bacterias en función de la población de E1 del
           modelo 2
35             $h1->reglaasignarprobabilidad($r2,$actualdato[0]);
36         }
37         //bloque de operaciones sobre m1
38         if ($j>0){
39             $actualdato = $h1->objetosdarcantidad($j-1); //obtener los datos del paso actual
40             $dato=$actualdato[0];
41         //definición de condiciones de aplicación del pulso $heb en el modelo 1
42             if ($dato>=400){
43                 if($actualizar_regla==0){
44                     $h1->reglaasignarprobabilidad($r1,100); //movilidad máxima
45                     $h1->reglaasignarprobabilidad($r3,100); //hostilidad máxima
46                     $actualizar_regla=1;
47                     $suceso[]="Hostilidad máxima en $j";
48                 }
49             }
50             else
51                 if($actualizar_regla){
52                     $h1->reglaasignarprobabilidad($r1,10 ); //movilidad normal
53                     $h1->reglaasignarprobabilidad($r3,$heb); //normalización de $heb
54                     $actualizar_regla=0;
55                     $suceso[]="Hostilidad normal en $j";
56                 }
57             }
58         $ultimodato = $h1->objetosdarcantidad($pasofinal1);
59     }
60     echo "Se realizaron $ciclo ciclos de evaluación del modelo 1<br>";
61     if(count($suceso)>0){
62         echo "Sucesos de simulación:<br>";
63         foreach($suceso as $item) echo "$item<br>";
64     }
65
66     //finalización de simulaciones
67     $h1->simulacionterminar;
68     $h2->simulacionterminar;
69     usleep(100000); //tiempo para cierre
70
71     echo "\nPresentación de resultados del modelo 1<br>";
72     echo "<table>\n<tr><td>Paso</td>";
73     $numclases=$h1->cuantascclasesobjeto;
74     for($i=0;$i<$numclases;$i++) echo "<td>".$h1->claseobjeto($i)."</td>";
75     echo "\n";
76     for($paso=0;$paso<$numpasos1;$paso++){

```

A.5.12. Resultados del análisis

A pesar de los vacíos presentes en las tablas de contraste del análisis funcional se determina que las funciones establecidas en la actual versión de la interfaz son suficientes para el propósito de construcción de modelos y que la automatización cumple con el objetivo para la cual que fue construida. El autor reconoce la lamentable pérdida de trabajo y experiencias causada por haber mantenido en reserva a la interfaz de automatización durante tantos años.

El análisis realizado sirvió como fundamento para los productos descriptivos incorporados. Se completó con éxito la documentación sobre la interfaz de automatización, integrando los útiles presentes con un método de uso y un ejemplo de realización práctica de modelos-script. Ahora es posible realizar modelos-script con información idónea sobre la interfaz de automatización de la herramienta software.

A.6. Análisis a la Metodología BOR

p122 dancy asimetria epistemologica

A.6.1. Fundamentos de la Metodología de modelamiento basada en objetos y reglas MBOR

La tesis de Homos 1.0 definió la metodología de modelamiento basada en objetos y reglas MBOR con el propósito de “[definir] la dinámica del fenómeno mediante la identificación de los objetos, sus comportamientos e interacciones expresadas por reglas, el espacio y la distribución de los objetos en el mismo.”⁷⁵. Los conceptos sobre los cuales está construida la metodología pertenecen al campo de los autómatas celulares, la orientación a objetos y el proceso propio de modelado y simulación. Estos tres elementos confieren características particulares a la metodología, establecen las normas con las cuales se realizan la representación de transiciones, el orden de operaciones de modelado e imponen límites a lo que es posible representar en la herramienta software. La suma total de los elementos considerados hace plausible toda la estructura metodológica sobre la que se fundamenta la herramienta software.

A.6.1.1. Aporte de los autómatas celulares

Toma de los autómatas celulares los siguientes conceptos:

- La idea de espacio, su discretización y división en áreas homogéneas o celdas.
- La idea del tiempo, su discretización y división en intervalos finitos o pasos de simulación.
- Define el uso de estados simples en las celdas, establece que cada celda puede representar solo uno de varios estados finitos.
- El concepto de vecindad⁷⁶, que determina con cuales celdas es posible realizar interacciones.
- La idea de transición con la cual se establece que el estado de cada celda cambia obedeciendo a determinadas instrucciones y condiciones conocidas como reglas de transición o simplemente las reglas. Se supone que son locales al espacio donde se produce el fenómeno, algo ajeno a la propia célula.
- La idea de condiciones de limite o comportamiento de borde que definen lo que le sucede al contenido de las celdas mas allá del horizonte espacial de simulación previsto.
- La actualización de los estados en las celdas se aplica unisono, se evaluan todas en cada paso de simulación.

La premisa de los autómatas celulares es representar al universo mediante un sistema dinámico de estados discretos.

⁷⁵Ver introducción En la página 5 de la tesis de Homos 1.0

⁷⁶las celdas alrededor de una celda

A.6.1.2. Aporte de teoría de objetos

Toma de la tecnología orientada a objetos los siguientes conceptos teóricos⁷⁷

Objeto: Elemento tangible o intangible del cual es posible emitir un concepto.

Clase: Agrupación de objetos que comparten iguales características.

De los autómatas y la teoría de objetos se adaptan algunos conceptos,

Propiedad: Característica específica de una clase.

Servicio: Acción que realiza un objeto para satisfacer requerimientos propios o ajenos.

Método: La forma como una clase se comporta, los pasos que ejecuta para prestar un servicio.

Mensaje: Solicitud entre objetos para realizar servicios.

Estado: Condición en la que se encuentra un objeto.

Evento: Un suceso en el tiempo.

Encapsulación: Mecanismo que oculta los detalles de la implementación de un objeto.

Herencia: Mecanismo que permite replicar propiedades y servicios entre jerarquías de clases.

Polimorfismo: Homonimia de métodos de funcionalidad distinta.

Composición: Construcción de un objeto a partir de otros más básicos.

La mayoría de estos elementos son solo referentes teóricos, solo hace uso intensivo de

La tesis de homós 1.0 altera algunos de los elementos listados para su propio beneficio:

- Establece el uso de las clases objeto, son aquellas clases que pueden ser instanciadas y los objetos como las instancias de esas clases objeto. Ignora metódicamente a las clases abstractas.
- Define el ambiente como el espacio donde se recrea la dinámica del fenómeno. Se diferencia del espacio de los autómatas porque cada celda contendrá un objeto en vez de una colección de estados finitos.
- Establece un único tipo de vecindad para la implementación del autómata. La vecindad de 9 celdas, con 8 direcciones posibles a partir de un punto central. Esta forma de vecindad es fácil de implementar como una matriz, es popular y de amplia aceptación⁷⁸.

⁷⁷Para todo el tema de orientación a objetos, el marco de referencia conceptual es el modelo de objetos que incluye cuatro elementos fundamentales: abstracción, encapsulamiento, modularidad y jerarquía; así como tres elementos secundarios: tipos, concurrencia y persistencia.

⁷⁸Que viene desde el origen mismo de los autómatas celulares con el autómata autoreproductor de von Newman.

- Acorde con los cambios en el ambiente, se definen las reglas de transición. Operan en la vecindad de la celda en evaluación sobre objetos, en lugar de estados discretos como en los autómatas celulares. Están fundadas en principios biológicos y usan la comparación espacial de estados provista por los autómatas celulares. Estos sistemas de reglas de transición son equivalentes en su funcionamiento y propósito ya que ambas aplican a un mismo elemento, si se observa que los objetos con los que se trata en cada celda pueden reducirse a múltiples estados diferentes pero finitos. No hacen uso de la configuración exacta de los objetos en el espacio, solo se preocupan de que existan dichos objetos / estados en la vecindad próxima a la celda en evaluación.
- Iguala las reglas de transición del mundo de los autómatas con los métodos de clase de la teoría de objetos, ellas establecen el comportamiento particular de una clase objeto y sus descendientes.
- Hace probabilísticas a las reglas de transición de los autómatas celulares mediante la implementación de los conceptos de probabilidad y prioridad de ejecución. La probabilidad establece la frecuencia con la cual es posible que se ejecute una regla, la prioridad permite resolver conflictos de simultaneidad al momento de elegir una regla de entre un conjunto de reglas igualmente aplicables.

El proceso de modelado establece la forma en la cual se hace posible la conversión de las observaciones de un fenómeno dado en un modelo basado en objetos y reglas. Esta consta de las siguientes fases:

Adquisición de la información: Es un estudio analítico del fenómeno, permite identificar los elementos que lo caracterizan e intervienen en su desarrollo. Es una etapa libre, depende de los métodos de análisis utilizados y de la naturaleza misma del fenómeno a estudiar.

Implementación: Es la etapa del modelado encargada de la construcción del modelo. Está definida siguiendo la siguiente secuencia de fases:

Desarrollo del modelo en lenguaje prosa: Se elabora una descripción textual (enunciado) del fenómeno resultado del análisis preliminar. Para que sea útil al proceso de modelado esta debe:

- Incluir elementos descriptivos como P.Ej: hacer referencias al ambiente donde se desarrolla el fenómeno.
- Identificar los elementos que intervienen y sus acciones.
- También debe incluir los alcances y limitaciones presentes del modelo a plantear, esto último sirve de marco para las hipótesis a las que hubiera lugar, luego de implementado el modelo.

Descripción de los objetos y sus acciones: Esta etapa se resume en la búsqueda de los sustantivos y verbos contenidos en el modelo en lenguaje prosa. De este

análisis se obtiene que sustantivos serán las clases objeto, que verbos serán métodos para las clases encontradas.⁷⁹

Conversion al lenguaje de objetos y reglas: Los objetos encontrados de la etapa anterior se convierten en clases - objeto, luego se buscan / adaptan las reglas de transicion que satisfagan las acciones desarrolladas por los objetos. Los elementos que caracterizan a los parametros de control de las reglas, probabilidad y prioridad de ejecucion, se deducen de los datos presentes en el modelo prosa.

Implementacion del modelo en la herramienta: En esta etapa se crea efectivamente el modelo en la herramienta. Las fortalezas y debilidades funcionales de este proceso pueden observarse en el analisis de interfases y el analisis funcional presentados en este documento.

Simulacion: Esta etapa comprende la obtencion de resultados a partir de los elementos establecidos y las configuraciones presentes en el modelo. Presenta la evolucion del modelo en el tiempo graficamente: en el ambiente mediante la vision de la interaccion de diversas instancias de clase previstas y cualitativamente mediante las curvas de comportamiento en las vistas. A partir de ella puede evaluarse si el modelo desarrollado se ajusta o no al fenomeno representado. Siempre es posible volver a una etapa anterior y realizar los cambios necesarios para mejorar el ajuste o las condiciones de simulacion del modelo.

Ajuste de sensibilidad y parametros: Es la etapa final de la metodologia, asume que se ha encontrado un modelo candidato⁸⁰ a representar el fenomeno en estudio, luego de las posibles iteraciones de ajuste realizadas sobre el proyecto. Corresponde a la etapa productiva de la metodologia, permite responder a preguntas del tipo ¿que pasaría si...? en el modelo.

Estos elementos juntos constituyen la metodologia MBOR

existe una independencia entre la adquisicion de la informacion sobre el fenomeno y el desarrollo del fenomeno.

informacion sobre el fenomeno

empirico, experimental, estadistico

la descripcion del fenomeno depende de la experiencia del modelador?

los productos de la metodologia son el modelo prosa, el modelo Bor y su diagrama, la representacion espacial y temporal del fenomeno en el ambiente de simulacion, las curvas de comportamiento. estas ultimas permiten que el ajuste de sensibilidad y parametros tenga sentido

la calidad de los productos obtenidos dependerá de

⁷⁹Para Coad y Yourdon, La primera tarea del método es identificar las clases y los objetos. Los objetos existen en el ámbito del problema y son instancias de clases. En cambio, las clases son abstracciones de objetos.

⁸⁰Que se ajusta en cuanto a su comportamiento y estructura con los datos conocidos del fenomeno en estudio. Esta consideracion puede llegar a ser subjetiva si no se cuenta con un estudio apropiado del fenomeno.

la habilidad del modelador en el uso de la herramienta

capacidades de analisis y sintesis

en un apartado posterior se hablará mas extensamente sobre el diagrama bor

la obtencion del diagrama es un subproducto del proceso de creacion de clases objetos y reglas.

De todas las distintas actividades que componen el proceso de modelado,

la conversion al lenguaje de objetos y reglas es la mas

no se incluye en esta tabla la etapa de adquisicion de informacion ya que es específica del fenómeno y el area de estudio que la aborde.

Durante el manejo de la aplicacion no existe una division clara entre etapas, su formalismo es teorico, el modelador puede reconocer en que etapa del modelado se encuentra de acuerdo a las actividades realizadas.

implementacion de la metodologia en la herramienta

En la practica solo las ideas de objeto, clase objeto y reglas-metodos estan implementadas, las demas hacen parte del cuerpo teorico de la metodologia pero no tienen extension practica en la herramienta. Esto se corresponde con la vision orientada a objetos⁸¹ planteada en la tesis de Homos 1.0. Aunque no se soporta nativamente el polimorfismo, ya que las cadenas de identificacion de reglas deben ser distintas entre si, es posible emular esta caracteristica usando la notacion "Nombre de la clase objeto.función de la regla", esto es, puede repetirse el texto que define la funcion de una regla siempre y cuando se aplique en clases objeto distintas. Ejemplo: mismo tipo de regla, aplicacion a diferentes metodos y clases objetos, *mismo* nombre = "c1.mover", "c2.mover".

Las reglas en realidad no hacen parte de las clases objeto. Las reglas establecen asociaciones entre las clases y determina el estado anterior y el siguiente a su aplicacion junto con las condiciones de operacion. hay una independencia clara entre estos elementos, las reglas no dependen de las condiciones locales del fenomeno como en los automatas celulares, dependen de la funcionalidad observada y traducida por el modelador del fenomeno en estudio.

mejora a la MBOR

orientacion a objetos segun coad y yourdon => Orientado a objetos = Objetos + Clasificación + Herencia + Comunicación.

no hace uso practico de la notacion matematica prevista por los automatas celulares.

el gran problema con esta notacion es que teoricamente hablando no se sabe que es lo que hacen las reglas de transicion implementadas. se entiende en ellas sus resultados pero no existe un formalismo que explique su funcionamiento. recordando que no existe una notacion escrita para el modelo entonces...

⁸¹Segun Coad -Yourdon, Orientado a objetos = Objetos + Clasificación + Herencia + Comunicación.

Tabla A.12: Integración de elementos componentes de la MBOR

Requerimientos para la aplicación efectiva de la MBOR					Experiencia requerida
Etapas	¿Etapas integradas en la herramienta?	¿Descripción de proceso documentado?	Formalismo teórico	Formalismo operativo	
Desarrollo del modelo en lenguaje prosa	No	Si, mediante un ejemplo	Si, en la tecnología orientada a objetos	■ Usa los datos obtenidos de la adquisición de información como fuente de datos. ■ Identifica los elementos característicos del fenómeno según la orientación a objetos.	■ Comprensión básica del fenómeno. ■ Redacción, análisis y síntesis de textos. ■ Análisis sintáctico del idioma. ■ Conocimiento sobre tecnología orientada a objetos.
Conversión al lenguaje de objetos y reglas	No	Si, mediante un ejemplo.	No	Es un proceso empírico. Supone la comparación del comportamiento de las acciones reales con las definidas por las reglas de transición ofrecidas por la herramienta.	Experiencia de modelado
Implementación del modelo	Si	Si, seguir el manual de usuario	Si, ver tesis de Homos		Ninguna
Simulación	Si	Si, seguir el manual de usuario, manejo intuitivo.	No, es un proceso iterativo intuitivo	■ Determinar en que punto el modelo está listo para ejecuciones de pruebas, es independiente de las pruebas de concepto preliminares. ■ Establecer los escenarios de prueba ■ Manejar el simulador: iniciar, detener, terminar. ■ Realizar comparaciones de los resultados obtenidos con los esperados, determinar si hay necesidad de ajustes.	Ninguna
Ajuste de sensibilidad y parámetros	Si	Si, depende del objetivo e implementación del modelo.	No por la Mbzor, es particular a la naturaleza del fenómeno a modelar		Ninguna

Fuente: Autor.

Tabla A.13: Integración entre elementos componentes de la MBOR

Requerimientos para la aplicación efectiva de la MBOR					
Etapa	¿Etapa integrada en la herramienta?	¿Descripción de proceso documentado?	Formalismo teórico	Formalismo operativo	Experiencia requerida
Desarrollo del modelo en lenguaje prosa	No	Si, mediante un ejemplo	Si, en la tecnología orientada a objetos	■ Usa los datos obtenidos de la adquisición de información como fuente de datos. ■ Identifica los elementos característicos del fenómeno según la orientación a objetos.	■ Comprensión básica del fenómeno. ■ Redacción, análisis y síntesis de textos. ■ Análisis sintáctico del idioma. ■ Conocimiento sobre tecnología orientada a objetos.
Conversión al lenguaje de objetos y reglas	No	Si, mediante un ejemplo.	No	Es un proceso empírico. Supone la comparación del comportamiento de las acciones reales con las definidas por las reglas de transición ofrecidas por la herramienta.	Experiencia de modelado
Implementación del modelo	Si	Si, seguir el manual de usuario	Si, ver tesis de Homos		Ninguna
Simulación	Si	Si, seguir el manual de usuario, manejo intuitivo.	No, es un proceso iterativo intuitivo	■ Determinar en qué punto el modelo está listo para ejecuciones de prueba, es independiente de las pruebas de concepto preliminares. ■ Establecer los escenarios de prueba ■ Manejar el simulador: iniciar, detener, terminar. ■ Realizar comparaciones de los resultados obtenidos con los esperados, determinar si hay necesidad de ajustes.	Ninguna
Ajuste de sensibilidad y parámetros	Si	Si, depende del objetivo e implementación del modelo.	No por la Mbor, es particular a la naturaleza del fenómeno a modelar		Ninguna

El proceso de modelado establece la forma en la cual se hace posible la conversion de las observaciones de un fenomeno dado en un modelo basado en objetos y reglas. Esta consta de las siguientes fases:

adquisicion de la informacion

implementacion

lenguaje prosa

descripcion de los objetos y sus acciones

conversion al lenguaje de objetos y reglas

implementacion en el modelo

simulacion

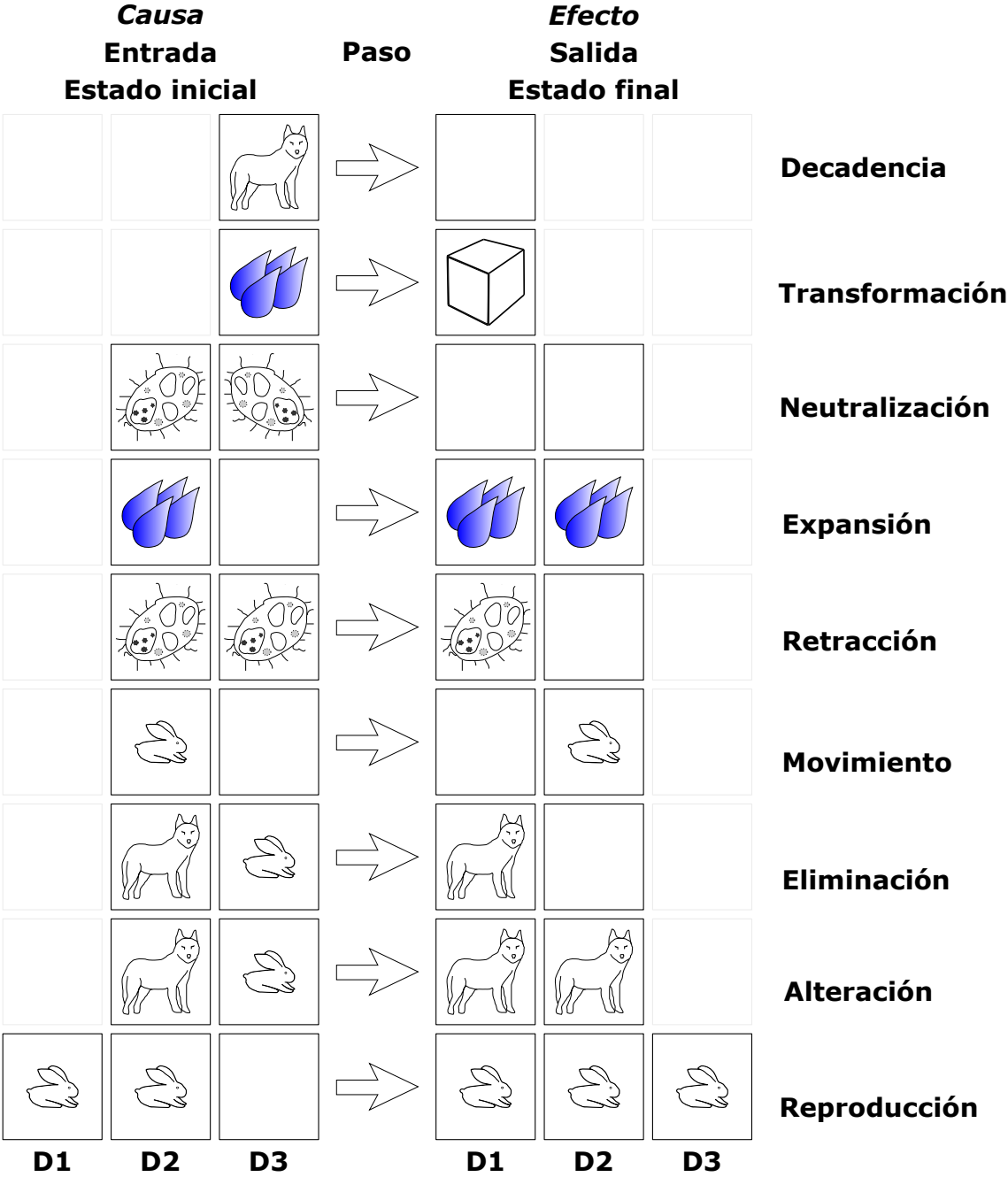
ajuste de sensibilidad y parametros

manejo del tiempo de simulacion

REGLAS DE TRANSICION EN LA METODOLOGIA DE MODELAMIENTO BASADA EN OBJETOS Y REGLAS

apendice A. metodologia de modelamiento basada en objetos y reglas, páginas 73-76 del manual de usuario de Homos 1.0

Figura A.19: Estructura y representación gráfica de reglas de transición en la MBOR



Fuente: Autor.

la metodología de modelamiento como tal no define el modelo basado en objetos y reglas, solo define y establece los elementos que componen la metodología en general.

define clase-objeto, ambiente y cada una de las reglas de transición,

el algoritmo basico de simulacion

establece claramente las etapas de modelamiento de con la metodologia y los utiles requeridos en cada etapa. propone una notacion grafica para las clases-objeto y algunas reglas de transición. establece un modelo matematico, define de forma incompleta los estados de las transiciones implementadas

por ende, los elementos que no hacen parte de la lista no se definieron precisamente en la metodologia, se deducen de la implementacion en la herramienta o de lo expresado en otros apartados de la tesis o el manual de usuario.

. este se deduce del diagrama bor,

si luego de elegida la regla esta no se aplica sobre la vecindad en evaluacion, entonces ocurre idempotencia de todas las celdas pertenecientes a esa vecindad, estas permanecen inalteradas.

esto corresponde a la tabla de estados generados descrito en la tesis de homos.

revision de descripcion de salidas

$H,I,-:H,I,J \mid H,I,-$

lo que describe una regla en notacion de lenguaje de transicion

situacion inicial de la regla

situacion final

si la situacion inicial no corresponde con la configuracion de elementos presentes de la vecindad en evaluacion, no es posible continuar con la ejecucion de la regla de transicion, la vecindad que queda como está.

si la situacion inicial se da pero los valores de probabilidad de ejecucion o su prioridad no son suficientes, entonces no es posible continuar con la ejecucion de la regla, la vecindad que queda como está.

si la situacion inicial corresponde con el vecindario en evaluacion y si los valores de probabilidad y prioridad son los correctos, entonces es posible ejecutar la regla, el vecindario se configura segun la situacion final prevista para la regla en curso.

A.6.2. proceso de modelamiento

que puede y que no puede ser modelado en homos?

componentes espacial y temporal, principalmente sistemas biologicos

cualquier sistema donde la identificacion de los objetos, sus comporamientos e interacciones sea intuitiva.... ???

Tabla A.14: descripcion

Descripcion de los estados generados por las reglas de transicion definidas en homos 1.0 ^a			
Regla	Siguiente estado		Salida
movimiento	vacio		el objeto se coloca en la celda seleccionada
expansion	el mismo objeto		un nuevo objeto de la clase objeto definida en la regla se coloca en la
retraccion	el mismo objeto		elimina el objeto de la clase objeto de la regla, ubicado en la celda sel
decadencia	vacio		no genera salida
reproduccion	el mismo objeto		un nuevo objeto de la clase objeto definida segun la regla se coloca en
alteracion	el objeto de la clase objeto en la cual se altera	el mismo objeto	el objeto de la celda seleccionada de transforma a la clase objeto de la
neutralizacion	vacio		la celda elegida se convierte en vacia
transformacion	un objeto de la clase objeto en la cual se transforma		no genera salida
eliminacion	vacio	el mismo objeto	elimina el objeto de la clase objeto de la regla, ubicado en la celda sel

^a Adaptacion de las tablas 13 y 14 la tesis de Homos 1.0

A.6.2.1. hhh

lenguaje prosa:

descripciones de ambientes posibles

elementos que intervienen y sus acciones o interacciones

alcances y limitaciones del modelo

descripcion de objetos:

objetos activos, sujetos los que hacen algo

ocupan espacio fisico

acciones, verbos

limitaciones actuales en verbos

lenguaje de objetos y reglas

intuitivo

toma desiciones sin soporte conceptual? pero ajustadas a lo que se percibe??

<http://es.wikipedia.org/wiki/Intuici3n>

<http://www.nodo50.org/sindpitagoras/Cientifico.htm>

la idea no es eliminar el aspecto intuitivo de la herramienta,

es estudiar como lo que se intuye sobre un fenomeno facilita la construccion de conocimiento

mas reglas intuitivas?

soporte a la conceptual a la intuitividad para construir conocimiento de largo plazo?

concepto de probabilidades y prioridades de ejecucion

las reglas estan controladas por la probabilidad de ejecucion y la prioridad de ejecucion

el concepto de probabilidad aplicado en las reglas, hace no determinista su ejecucion, depende de la aleatoriedad del momento el que una regla en particular se aplique o no. Las probabilidades de ejecuci3n se miden con respecto al tiempo, representado en la herramienta como el numero de pasos a usar en la simulacion. Tienen especial significado de los valores extremos de la probabilidad de ejecucion, ya que representan determinismos:

- 0 % asigna la total imposibilidad de ejecucion, para cualquier tiempo de simulacion
- 100 % asigna la total posibilidad de ejecucion para cualquier tiempo de simulacion.

Los demas valores representan la posibilidad de que una regla se ejecute durante la simulacion segun lo especificado.

que sea posible la aplicacion de una regla no significa que se ejecute. la aleatoriedad presente en la herramienta se encarga de dar respuesta a este fenomeno. los objetos no piensan/deciden!

El concepto de prioridad, permite dar importancia a una regla sobre otras igualmente elegibles o con una misma probabilidad de ejecucion.

La prioridad cero representa el valor maximo, una forma de determinismo en cuanto al orden de importancia de las cosas, la regla que no tiene apelaciones tal como en es el caso de la regla de decadencia.

la prioridad mas baja es 255, representa el valor de importancia mas bajo.

probabilidad cero y 100

sistema de prioridades

resaltar una clase sobre otra por su importancia relativa

proporcion del tiempo en la herramienta

avance por deltas de tiempo, pasos de simulacion

construccion de nuevas reglas de transicion—————

notacion, explicacion basica de su funcionamiento

A.6.3. Limitaciones presentes en la MBOR

A.6.4. Protonotación y análisis de transiciones

A.6.5. Sub elementos dentro del proceso de modelamiento

representar modelos que requieran el espacio?

identificacion de la clase objeto con

un nombre e imagen representativos

facilitar la representacion espacial enambiente

la eleccion del color, puede estar influenciada por la naturaleza del objeto a represenar, pero en general es arbitraria

propositos distintivos. ver colores distintivos, usos aplicaciones del color

ciclo de repeticion en simulacion

evaluacion de todas las celdas en ambiente, aplicacion de reglas segun corresponda

ta:tb

hacer $t_a = t_b$

continuar hasta finalizar el numero de pasos de simulacion o terminar/ abortar el proceso

A.6.5.1. notas

el libre albedrio es dependiente de la situacion que le precede. asi como de la experiencia acumulada sobre comportamientos observados previamente

A.6.5.2. Limitaciones / beneficios presentes

Esta beneficia a la metodologia al permitir un sistema de simulacion con las siguientes características:

- Es posible extenderla a representaciones de múltiples dimensiones (no solo hasta 3D).
- No usa metodos numericos para la obtencion de soluciones
- No presenta problemas de truncamiento o redondeo durante los procesos de calculo.
- Aborda problemas que estan fuera del alcance de otras disciplinas, mediante formulaciones simples que producen resultados de complejidad variable. incluye el espacio
- es dependiente del orden de ejecucion de los elementos que la componen. leves cambios el orden de ejecucion configuran cambios en el comportamiento/desarrollo final observado Ef. mariposa
- implica una forma de representacion causa efecto mas amplia, cercana al lenguaje natural??

La limita en las siguientes formas:

- las ecuaciones discretas / resultados obtenidos son dificiles de estudiar analiticamente, generalmente no se reducen a funciones conocidas o continuas
- Es dependiente de factores de escala del ambiente.
- generalmente es irreversible

factores de irreversibilidad de los sistemas

efecto mariposa en sistemas

si una cosa se explica en terminos de otra y esta se comporta de una cierta forma, por semejanza una valida la otra en el aspecto que esta representa??? trasnsitividad de modelos???

no hace uso de

implicaciones de incluir el espacio en la simulacion

transformacion - cambio NOTA

de estado interno

de clase

autoclasificación - función informativa

Nomenclatura y diagrama Bor

clases abstractas no soportadas

como estas cubren el espacio formando un mosaico, una geometría particular o una teselación.

permite establecer la cantidad necesaria de clases objeto y las reglas asociadas a ellos para la creación del modelo.

para realizar un modelo en homós es necesario que el fenómeno a modelar pueda descomponerse en reglas simples y que estas reglas puedan ser representadas o encontrar un equivalente entre los tipos de reglas disponibles en la herramienta. aunque la herramienta es de propósito general, encuentra en los modelos poblacionales

características de los modelos creables en la herramienta

debido a la naturaleza de las reglas implementadas y por las estructuras que es posible crear en la herramienta

toma de la teoría de objetos:

el concepto de objeto, como un elemento abstracto, particular y unitario que realiza interacciones en el ambiente, siendo la clase objeto de ese objeto el tipo al que pertenece.

los métodos, son las acciones o funciones de esos objetos, ellos se ejecutan mediante los eventos, que es la oportunidad que encuentra el simulador de aplicar una regla. como puede verse, las reglas equivalen en homós a los métodos.

Falencias Presentes

simulación de polimorfismo mediante la homonimia de reglas cuando se usa una formalidad en la escritura de nombres. es más visible como un aspecto funcional, al aplicar un mismo tipo de regla (igual configuración de parámetros y clases objeto asociadas) sobre diferentes clases objeto.

no existe el concepto de herencia

no se ha implementado el concepto de atributos en las clases y por ende las demás características del manejo de dicha capacidad.

ni los atributos

los escenarios permiten la configuración de las condiciones de simulación para un ambiente especificado

establece un ambiente de simulación en el cual se realiza la interacción de las diferentes instancias de las clases objeto del modelo

la aleatoriedad de las reglas mediante el estado de su determinismo,

quienes permiten la evolución determinista de la simulación.

la acción de las reglas sobre las clases objeto es de inspiración biológica. estas reglas pueden clasificarse

accion espacial

aquellas que tienen componente de movimiento en el espacio

accion poblacional

Todas las reglas tienen en comun los siguientes parametros:

- Nombre de la clase objeto a la que se aplica la regla o *primera clase*.
- La probabilidad de ejecución, comprendida entre los valores 0 % y 100 %.
- la prioridad de aplicación de la regla, comprendida entre los valores de 0 y 255.
- Un campo de texto que sirve para dar una breve descripción sobre la utilidad de la regla.

La regla de decadencia esta completamente satisfecha con estos tres parámetros.

Las reglas que tienen componente en el espacio adicionan a las anteriores:

- Direccion en el espacio: conjunto casillas de marcado que establecen las direcciones de movimiento.

Este parametro hace parte de la definicion de las reglas de Expansión, Retracción y Movimiento, adicionando en esta última el número de celdas a desplazarse.

Las reglas que tienen componente de influencia sobre otras clases adicionan a las anteriores:

- La clase objetivo o *segunda clase*: es la clase con la cual se tiene interaccion y sobre la que se aplican los efectos de la regla

Hace parte de las reglas de Alteración, Eliminación, Neutralización, Transformación y Reproducción.

La regla de reproducción incluye un parámetro adicional llamada *clase resultante* que representa a una tercera clase, efecto de la aplicacion de dicha regla.

A.6.5.3. Consecuencias de la ausencia de una forma de notación formal para la expresión de reglas de transición

Las reglas presentadas en [picture] y en la tesis de homos 1.0 [1.3 mbor]

en [] no se hizo evidente la regla de eliminación presentada en [].

la tesis de homos 1.0 consideró adicionalmente el espacio como elemento grafico necesario para la presentacion de la regla de transicion, algo no contemplado originalmente en []

se requiere de un medio de notacion de transiciones flexible

facilite la comprension de la accion presentada en la regla

A.6.5.3.1. Requisitos de una notación de transición considerar el espacio, haciendo referencias explícitas a su uso

considerar los elementos que intervienen durante la transición

considerar el tiempo

elementos descriptivos

fácil de construir

que permita manipulaciones posteriores

A.6.5.3.2. Un prototipo de notación para Homos 1.0 Conocido el hecho de que no se dispone de una notación escrita para la representación de las reglas de transición definidas en la MBOR

distintas de la descripción narrativa y la notación de autómatas,

se propone la siguiente notación para la representación de la acción de las reglas en Homos 1.0

componentes de notación

descripción de una regla:

Se compone de la descripción de la configuración en el ambiente de una clase objeto y sus vecinos en un instante inicial y en el instante siguiente.

puede leerse como la descripción de una causa, la configuración inicial de esas clases objeto, y un efecto o la descripción de la configuración final a la que llegaron esas clases objeto.

situación anterior y la situación posterior

de uno o más objetos en el ambiente descritos por los siguientes símbolos:

las reglas describen parejas

una regla describe una pareja de términos cuyo propósito es servir de
símbolos de notación: Causa:Efecto => Situación1:Situación2 => Configuración inicial:Configuración
final => Evento_en_Ti:Evento_en_Ti+1

definición de una regla esta

una regla se define mediante la declaración de la situación inicial y final.

la situación es la configuración de los terminales

necesarios para que pueda

situación

los identificadores de clases objeto que aparecen en la situación inicial son las clases principales de la
regla

evento inicial y evento final

descripcion de una regla:

una regla esta compuesta por 2 partes fundamentales:

Situacion inicial: Es la configuracion de instancias de clases objeto que debe presentarse para hacer posible la ejecucion de la regla, se entiende como un evento que se presenta dentro del ambiente durante el tiempo de simulacion.

Situacion final: Es la configuracion de terminales que representa el resultado de la aplicacion de la regla. Es la configuracion final de las terminales establecidas en la situacion inicial.

en la notacion, las situaciones inicial y final de una regla estan separadas por el simbolo de transicion ":".

Las reglas se describen mediante las relaciones de identificadores de clases objeto y celdas vacias.

en la practica, el simulador busca en el ambiente las instancias de clase que correspondan a las clases objeto referenciadas por la regla en evaluacion.

Se definen como terminales los nombres de las clases objeto (clobjetos) y las instancias de las clases objeto (inobjetos) presentes en el modelo junto con las referencias a celdas vacias.

ñl

en esta notacion la ubicacion de un identificador en la configuracion es importante,

ya que cambiar un elemento de ubicacion implica una transformacion

Claseobjeto1,Claseobjeto2,Claseobjeto3...: Claseobjeto1,Claseobjeto2,Claseobjeto3...

Listado de los identificadores de instancias de clases objeto presentes antes de la aplicacion de la regla (configuracion inicial) y despues de la aplicacion de la regla (resultado, efecto de la regla). estas descripciones estan separadas por el simbolo de transicion. ejemplo, H,J:H,J. si se ubican 3 clases en la configuracion inicial de la regla, es necesario definir 3 clases en la configuracion final de dicha regla.

en esta notacion no se hace enfasis en la ubicacion de las celdas vecinas a la celda principal, ya que el simulador en su funcionamiento busca en todas las posibles ubicaciones hasta encontrar

, representa la eliminacion del objeto si existe uno representado en esa posicion de la lista de clases presente en la configuracion

A.6.5.3.3. Elementos de notación constructores de reglas

Esta es la definicion de los simbolos usados en el protolenguaje Homos:

A.6.5.3.4. Directivas de construccion de reglas presentes en Homos 1.0 contruccion de reglas

{Situacion inicial : Situacion final}

donde una Situacion es una lista de identificadores de clases u elementos de modelado necesarios para describir una regla.

Simbolo	Uso
,	Separador de los terminales presentes en la definicion de las reglas
H,I,J,K...	Identificadores de terminales presentes en el modelo
_	Representa a una celda vacia o espacio libre en ambiente
:	simbolo de transicion o cambio de paso de simulacion, de un tiempo anterior al siguiente, t0:t1
(x,y)	Expresa la ubicacion de un terminal en el ambiente de simulacion, permite especificar textualmente
%	indicador del porcentaje de ejecucion de la regla. ejemplo 100 % o cien por ciento de probabilidades
\$	indicador de la prioridad de ejecucion de la regla. ejemplo \$1 o prioridad = 1
{...}	indica el bloque de instrucciones que describen una regla

Tabla A.15: Simbolos del protolenguaje Homos

Principio de conservación de terminales: La cantidad de terminos involucrados en la configuracion inicial de una regla es igual a la cantidad de terminos presentes en la configuracion final de dicha regla.

{Lista de Identificadores : Lista de resultados de transiciones de los Identificadores}

{IdentificadorA : Resultado del primer identificador en lista = IdentificadorA}

{IdentificadorA, IdentificadorB : Resultado del primer identificador en lista = IdentificadorA, resultado del segundo identificador en lista = IdentificadorB}

{IdA, IdB, ..., IdN : Resultado del primer identificador en lista = IdA, resultado del segundo identificador en lista = IdB, ..., resultado del segundo identificador en lista = IdN}

Ejemplo: Si para establecer la configuracion inicial de una regla se requieren de dos clases objeto (dos terminos), entonces la configuracion final contendrá tambien dos terminos, que pueden ser las mismas clases objeto referenciadas u otros elementos definidos en el lenguaje.

prioridad de evaluacion de los teminos que definen una situacion inicial de una regla.

Estas directivas se asemejan en su funcionamiento a los elementos encontrados en los leguajes de programacion para definir sus gramaticas.

esencia de terminales

cada espacio terminal en la situacion inicial de la regla tiene su equivalente en la situacion final de esa regla

cada espacio terminal en la situacion inicial de la regla tiene su contraparte en la situacion final de esa regla

principio de composicion de primitivas

la composicion de primitivas es conmutativa con respecto al orden de los terminos, no importa el orden de aplicacion, el resultado es el mismo.

$$\{\text{IdentificadorA}, \text{IdentificadorB} : \text{ResultadodeA}, \text{ResultadodeB}\} = \{\text{IdentificadorB}, \text{IdentificadorA} : \text{ResultadodeB}, \text{ResultadodeA}\}$$

se altera la prioridad con la cual

el orden de evaluacion se altera, haciendo prioritario al identificadorA en el primer caso y en el segundo al identificadorB

Para acercar esta notacion con lo establecido en la MBOR, el identificadorA será la clase principal a la cual esta relacionada la regla de transicion descrita y los demas identificadores las clases restantes necesarias para la plena descripcion de la regla de transicion.

mas allá de estos elementos

una operacion "simple" por cada paso

A.6.5.3.5. principio de cerradura algebraica!

A.6.5.3.6. potencia expresiva / descriptiva del lenguaje

A.6.5.4. elementos internos y externos de evaluación de reglas??

requisitos espaciales, requisitos de clase, mixtos

A.6.5.4.1. Elementos de construccion de primitivas Los supuestos que soportan a las primitivas de transicion son:

Identificacion: Abstraccion necesaria para separar una cosa de otras, categorizacion, clasificacion, si se puede dar nombre a una cosa esta se hace inteligible, manejable. Se identifican en esta notacion las clases objeto, las instancias de esas clases, el vacio en el espacio, cambios particulares en el estado de los objetos entre otros elementos representables en el modelo.

Cambio en el tiempo: Describe observacion temporal de los sucesos que acontecen a un objeto. Es una narrativa de esos acontecimientos.

Relacion causa-efecto: Corresponde a la descripcion de un estado inicial y de un estado posterior separados por instantes de tiempo establecidos. Supone la idea de que existe una continuidad - relacion entre dichas descripciones. Establecer los elementos que componen una relacion de causa y efecto depende de varios factores:

La unidad de medida usada para establecer dicho cambio: ya que en instantes de tiempo diferentes se presentan nuevos estados describibles / inteligibles o no con el lenguaje usado. Para efectos del modelado se usan aquellos representables por una notación. Para la MBOR serán aquellos que puedan representarse mediante las primitivas y sus construcciones.

La descripción que se hace del fenómeno: de que es inicial y que es final para esa situación particular, de que elementos la componen, sus relaciones. El sentido de avance en el tiempo generalmente establece estas pautas pero no es el único elemento que pudiera contemplarse para determinar este orden. Para los propósitos de simulación del modelado en la MBOR el tiempo es uno de sus fundamentos y establece estas descripciones.

Finitud en el alcance descriptivo: Permite que la relación pueda simularse usando cada segmento presente en la división de espacio tiempo prevista por los autómatas celulares. Predice la existencia de primitivas de alcance infinito, no soportadas por la notación propuesta, así como por la MBOR.

La unicidad en la continuidad por simplificación: suponemos, deseamos, una causa, un efecto. Esta descripción rechaza el hecho de que leves variaciones en la causa implican un cambio total en el efecto⁸², no contemplar esas variaciones denuncia un desconocimiento en la forma en que funciona esa relación en particular. Ignorar estos leves cambios puede explicar el porque una *descripción de causa* pueda producir distintos efectos.

- Contemplar la influencia de variaciones en una causa hace que estas se particularicen, induciendo a la identificación de nuevas causas. Cada nueva descripción de causa tendrá así una descripción de efecto(s) más directa.
- En la notación se describen las fases de estas relaciones mediante la identificación de los estados de los actores que intervienen ??????
- Es una situación no deseada para los propósitos de su modelado y su formalismo en un lenguaje.
- En humanos puede representarse mediante abstracciones probabilísticas, la determinación de estos valores inicialmente serán empíricos, progresivamente se harán estadísticos y en el mejor de los casos determinísticos.

causa1:efecto1 <- descripción deseada, simplificada de una relación de este tipo

relación

causa?:efectos?

causa:efecto1 ¿? % | efecto2 ¿? % | ... | efectoN ¿? %

causa:efecto1 10 % | efecto2 20 % | ... | efectoN 30? %

causa:efecto1 cada 5 | efecto2 cada 10 | ... | efectoN solo una vez

sumatoria de porcentaje de efectos = 100 %

múltiples causas para un efecto

causa1 | causa2 | causa3 | ... | causaN:efecto

⁸²Efecto mariposa

múltiples causas para un múltiples efectos, requiere más detalle en análisis, denuncia composición de causas, falta de independencia.

causa1 | causa2 | causa3 | ... | causaN:efecto1 | efecto1 | efecto2 | ... | efectoM.

prioridades de ejecución

probabilísticas

se desconoce el mecanismo interno, se proponen valores que permiten predecir la ocurrencia del elemento al que caracteriza

calificador

correspondencia estadística

nunca, pocas veces, a veces, casi siempre, siempre

los rangos de valores los define el modelador respaldado en la observación empírica o sistemática del fenómeno

determinísticas

se conoce el mecanismo, es fijo, no dan lugar a dudas sobre su ocurrencia, tiene una probabilidad fija de ocurrencia y un componente de secuencia que es lo que las hace determinables

su probabilidad no depende del tiempo total de simulación, depende de la organización de pasos interna

los mecanismos automáticos, diseñados para que sucedan por etapas generalmente exhiben este comportamiento. depende de situaciones ajenas el que se altere esta condición

estos valores no son describibles por medios estadísticos, los definen mal, reducen su valor, cuando es posible asignar un valor este es difícil de calcular

la regularidad destruye el aspecto probabilístico

puede ser regular o no

ejecución solo una vez durante toda la simulación

determinada cantidad de veces en intervalos regulares

ocurre siempre en cada paso de simulación

A.6.5.4.2. tablas de combinaciones y de permutaciones de primitivas expresión de conjunto de acciones posibles / expresiones en un sistema de combinaciones de primitivas dado

combinaciones resalta el orden de evaluación de las reglas de izquierda a derecha

a la menor falta interrumpir la evaluación de la regla en curso

permutaciones cuando es necesario hacer una evaluación de términos sin importar el orden, se asume que todos los términos tienen igual prioridad por lo tanto se hace irrelevante el orden de aparición de términos

se hace con respecto a las primitivas que componen a la regla

variaciones de reglas =

reglas distintas entre si en al menos por un termino

tipos distintos de reglas

expresiones distintas

numero total de posibles descripciones de reglas = n^k

n = cantidad de simbolos disponibles representables en el lenguaje

k = numero de terminos que componen una situacion, numero de primitivas involucradas en la realizacion de la regla

cantidad total de tipos de reglas distintas formadas por K primitivas

define una combinatoria con repeticion:

combinatoria de $(n + p - 1, p) - 1$

n = numero de primitivas disponibles en el lenguaje

p = numero de primitivas aplicadas simultaneamente

primitivas

binarias, trinarias, cuaternarias quinternarias.. etc

para el vecindario de 9 celdas, se tiene que:

descripcion de 9 terminos por celda

reglas por coincidencia de interaccion

reglas por accion sobre elementos internos

reglas combinadas

la cantidad de terminos en las reglas es arbitrario

entre mas compleja mas dificil e improbable es su ejecucion??

probabilidades calculables de acuerdo a condiciones del espacio y geometria subyacente

geom de 3 != geom 4 ...

limites de representacion de reglas

enumeracion finita de atributos de reglas

sumatoria de atributos

la cantidad de primitivas a evaluar dependerá del vecindario

depende de la geometria del espacio instalado en el modelo

no cubre repeticion de reglas (reuso)

ni las diferentes interpretaciones a una primitiva por causa de los elementos con los que ella opera
por ejemplo $\_H$, $H:I$ mejor ejemplo

geometrias con vertices distintintos

numero maximo de super tipos distintos =

obtencion de tablas de combinaciones

primitiva inexistente inexpressable: denotada por U , es la definicion de un elemento que no hace parte del lenguaje, existe con fines de separacion de las distintas combinaciones.

se refiere al conjunto de primitivas desconocidas que no estan representadas por la herramienta???

no pueden ser representables

en un sentido mas practico es una no primitiva, como un nop

vacio de primitivas

las primitivas en las tablas estan identificadas por una letra del abecedario.

en general: $(3 + k - 1, k)$

A.6.5.5. limites de representación

usa una metodologia de caja negra en cuanto a la descripcion de los objetos, no representa la mecanica interna de las transiciones dentro de los objetos, ni las explica, se funda en lo que se observa

hace caja blanca para relaciones entre objetos, las describe conforme a lo observado, permite una posible explicacion conforme a como se ve que funcionan las cosas.

no representa infinidades, interacciones locales, parciales

a medida que se aumenta el conjunto de operadores posibles se hace complejo el lenguaje. mas memoria, mas recursos, mas tiempo de cómputo

el objetivo de la simplificacion original se pierde.

representa abstracciones en varios niveles

presenta variedad de opciones de representacion

presenta comportamiento dinamico en varios niveles con naturalidad

facilita la mecanizacion de procesos, permite realizar predicciones conceptuales?

la notacion de lenguaje describe combinaciones de reglas que no tienen equivalente conocido en el mundo fisico. las relaciones son artificiales, son desconocidas? representan otros sistemas fisicos??

Aunque existan muchas reglas aplicables posibles, solo se aplica una regla de transición a la vez por cada celda en evaluación.

aplicar varias reglas solo si son de dimensiones diferentes

actualizar un estado interno y cambiar ubicacion en el espacio..

A.6.5.6. relaciones primordiales de un sistema dinámico

Elemento divino : lo incomputable, godël?

Idempotencia

Cambio

—

_P

_C

_PP

_PC

_CC

PPP

PPC

PCC

CCC

—

—

_P

_C

_PP

_PC

_CC

PPP

PPC

PCC

CCC

—

??? notoignoto, desconocido total

??P

??C

?PP

?PC naturaleza de las descripciones, contienen los tres elementos

?CC

PPP totalmente permanente, imposible

PPC

PCC

CCC determinista, imposible???

estados (?,P,C)

elemento permanece P

elemento cambia C

elemento en estado de transición desconocido ?, D

toda regla de transición contempla en su interior estados de transición nulos y primitivos de transición desconocidos

la sumatoria de transiciones nulas

combinaciones = dimensiones conocidas, + desconocidas contempladas contenidas, necesarias sobre el fenómeno

—

—

A.6.5.7. Génesis de un sistema para la representación de cambios de estado

En un modelo cualquiera, las reglas de transición describen el cambio de estado de objetos a los que hacen referencia.

la ausencia de un método con el cual

Vistas las necesidades de un formalismo que permita

describir y facilitar la génesis de nuevos tipos de reglas de transición, se hace necesario establecer una forma de representación de cambios de estado con la cual

A partir de la idea del reconocimiento de un estado⁸³ se propone una notación que permita construir representaciones de cambio de estado.

Para hacer posible la identificación de un estado, este debe ser conocido de alguna forma, luego existen estados conocidos y desconocidos. Estos últimos generalmente son aquellos

¿Un estado puede ser conocido o no,

partimos desde su reconocimiento inicial

⁸³Circunstancia, modo de ser, situación de una persona o cosa

Tabla A.16: Conocimiento previo sobre un estado de cambio

Identificador de Estado	Descripción
?	Estado desconocido, no reconocido
<i>Identificador</i>	Estado reconocido, inteligible

desconocido: de lo que no se tiene idea, no conocido

no reconocido: estado existente, distinguible por ciertos caracteres, identificable pero no definido formalmente

El elemento fundamental de este sistema es el hecho de reconocer o no un estado:

Para estudiar los estados en general, estos pueden deben estar reconocidos o no.

plante

como se identifica un estado?

Un estado puede no ser identificable si nos es completamente desconocido⁸⁴ o simplemente no ser reconocido⁸⁵,

se establece entonces una gran división al inicio de la organización de la representación de los estados, haciendo una primera distinción entre aquellos que conocemos de otros ignorados para el momento actual de conocimiento:

?: Representa el conjunto de todos aquellos estados que no pueden ser descritos mediante sistema de representación alguno, así como todos aquellos estados que aunque representables no hacen parte de ninguna notación o conjunto de descripciones actuales conocidas por el lenguaje en el que se implementa. También incluye todos los elementos desconocidos futuros y todos los ininteligibles posibles que puedan existir.

Identificador: Representa un elemento del conjunto de todos los estados que han sido descritos y que tienen representación simbólica inteligible actual o futura en el lenguaje elegido. De ellos se tiene alguna referencia, están fundados en el conocimiento empírico de las cosas o por medio de formalismos en algún sistema de conocimiento previamente existente.

—Mediante esta convención, es posible representar distintos estados. el cambio de estados es posible mediante la operación de transición:

—

Transición: acción de cambio

No transición: no acción de cambio.

—

⁸⁴Sin referencias previas, No tenemos idea de su existencia.

⁸⁵distinguible de otros por ciertos rasgos o caracteres.

al reconocer nuevos estados, ampliamos el conjunto de expresiones posibles en el sistema de representación.

—

ya que los distintos cambios se realizan con referencia al tiempo

—

Transición: Acción pasar de un estado a otro. A partir de esta definición se establece la siguiente notación de cambio de estado o notación de transición:

Identificador1:Identificador2

Donde:

- Identificador 1 y 2 son representantes de estados, sin hacer distinción de si estos son iguales o distintos, si son conocidos o no.
- Identificador1 representa el estado inicial e Identificador2 representa el estado final de la transición, los dos puntos [:] son un símbolo que indica la separación de cambio entre los estados relacionados.
- El orden del cambio de estados en la notación se realiza de izquierda a derecha, siendo el estado inicial el estado que debe acontecer para que posteriormente sea realice el cambio al estado final descrito en la transición.

—

Los componentes de una descripción de transición establecen tres categorías distintas del tipo o forma de cambio

—

cambio de estado de un momento a otro

conocimiento o no del estado de transición

si se conoce el estado, describirlo en términos inteligibles

para construir un sistema de esta naturaleza, partimos de la idea de representación de un estado de cambio mediante un identificador, en este caso, usando un símbolo del abecedario

el siguiente paso es la consideración de los posibles cambios a representar, en este caso de la descripción de los cambios de estado más simples posibles:

Para los propósitos de este estudio, a partir de este punto se ignorarán las transiciones que impliquen estados desconocidos.

Ya que no aportan una información distinta al hecho de que se desconoce sobre ellos,

Tabla A.17: Cambios de estado fundamentales

Identificador de transición	Notación	Descripción
?	? : ?	Transición desconocida
P	A : A	Transición de permanencia
C	A : B	Transición de Cambio

A,B son estados / clases / objetos previamente definidos

Tabla A.18: Transiciones entre estados ?, A

Identificador de transición	Notación funcional	Notación de Transición	Descripción de Transición
C	C(?, ?)	? : ?	Entre estados desconocidos distintos
P	P(?)	? : ?	Entre estados desconocidos iguales
C	C(?, A)	? : A	De un estado desconocido al estado A
C	C(A, ?)	A : ?	Del estado A a un estado desconocido
P	P(A)	A : A	De permanencia en A

A es un estado previamente definido

? denota uno cualquiera de muchos estados desconocidos

Tabla A.19: Transiciones entre estados ?, A, B

Identificador de transición		Notación	Descripción de Transición
C	C(?, ?)	? : ?	Entre estados desconocidos distintos
P	P(?)	? : ?	Entre estados desconocidos iguales
C	C(?, A)	? : A	De un estado desconocido al estado A
C	C(?, B)	? : B	De un estado desconocido al estado B
C	C(A, ?)	A : ?	Del estado A a un estado desconocido
P	P(A)	A : A	De permanencia en A
C	C(A, B)	A : B	De cambio, de A a B
C	C(B, ?)	B : ?	Del estado B a un estado desconocido
C	C(B, A)	B : A	De cambio, de B a A
P	P(B)	B : B	De permanencia en B

A,B son estados previamente definidos o conocidos

? denota uno cualquiera de muchos estados desconocidos

Las diferentes transiciones descritas pueden expresar la combinación de N estados distintos, pero no pueden manipular más de una pareja de cambios por cada oportunidad de realizar una transición.

Para soportar la simultaneidad de transición de estados, se propone la siguiente extensión a la notación de transición ya presentada:

Identificador1,IdentificadorA:Identificador2,IdentificadorB

Tabla A.20: Transiciones entre estados ?, A, B

Identificador de transición		Notación	Descripción de Transición
C	$C(? , ?)$	$? : ?$	Entre estados desconocidos distintos
P	$P(?)$	$? : ?$	Entre estados desconocidos iguales
C	$C(? , A)$	$? : A$	De un estado desconocido al estado A
C	$C(? , B)$	$? : B$	De un estado desconocido al estado B
C	$C(A , ?)$	$A : ?$	Del estado A a un estado desconocido
P	$P(A)$	$A : A$	De permanencia en A
C	$C(A , B)$	$A : B$	De cambio, de A a B
C	$C(B , ?)$	$B : ?$	Del estado B a un estado desconocido
C	$C(B , A)$	$B : A$	De cambio, de B a A
P	$P(B)$	$B : B$	De permanencia en B

A,B son estados previamente definidos o conocidos

? denota uno cualquiera de muchos estados desconocidos

que se traduce en

Transición Identificador1:Identificador2 simultáneamente ejecutada con
IdentificadorA:IdentificadorB

por extensión:

$Id1, IdA, \dots, IdN : Id2, IdB, \dots, IdM$

Transición $Id1 : Id2$ simultáneamente con $IdA : IdB$ simultáneamente con ... simultáneamente con
 $IdN : IdM$

gráficamente:

donde

Esta extensión de la notación aporta los siguientes elementos conceptuales a la descripción de cambios estado:

asdsadf

asfasdf

Para propósitos de este estudio, nos interesan las transiciones de las cuales conocemos sus estados componentes, esto es aquellas de permanencia y cambio.

Tambien es conocido que la transicion entre estados desconocidos

sistema de cambios

Tabla A.21: Sistema con dos transiciones simultáneas entre estados ?, A, B, C

Identificador de transición		Notación	Descripción de Transición
C	$C(? , ?)$	$? : ?$	Entre estados desconocidos distintos
P	$P(?)$	$? : ?$	Entre estados desconocidos iguales
C	$C(? , A)$	$? : A$	De un estado desconocido al estado A
C	$C(? , B)$	$? : B$	De un estado desconocido al estado B
C	$C(A , ?)$	$A : ?$	Del estado A a un estado desconocido
P	$P(A)$	$A : A$	De permanencia en A
C	$C(A , B)$	$A : B$	De cambio, de A a B
C	$C(B , ?)$	$B : ?$	Del estado B a un estado desconocido
C	$C(B , A)$	$B : A$	De cambio, de B a A
P	$P(B)$	$B : B$	De permanencia en B

A,B son estados previamente definidos o conocidos

? denota uno cualquiera de muchos estados desconocidos

paso de definicion de estados a manejo de objetos

cada letra representa una descripcion fundamental de cambio. se describen tres formas iniciales posibles:

descripciones fundamentales de cambio

D : transicion desconocida

P :transicion conocida, en la que se permanece en el mismo estado

C :transicion conocida, en la que se cambia el estado

Por cada descripcion de cambio existen una o mas transiciones desconocidas asociadas.

por definición y sentido común, la transicion desconocida no tiene respresentacion simbolica en el simulador? lenguaje a construir mediante las sistema de

el termino transicion implica que cada estado de cambio esta conformado por un paso de un momento del tiempo a otro.

para la notacion, al realizarse una transicion se configura un suceso en el tiempo y el espacio.

DD:composicion de transiciones desconocidas

DP:composicion de transicion con un elemento desconocido y permanencia

DC:composicion de transicion con un elemento desconocido y cambio

Tabla A.22: Sistema con dos transiciones simultáneas entre estados $?$, A, B, C

[illegible]

A,B son estados previamente definidos o conocidos

? denota uno cualquiera de muchos estados desconocidos

PP:composicion de transiciones desconocidas

PC:composicion de transiciones desconocidas

CC:composicion de transiciones desconocidas

—

DDD

DDP

DDC

DPP

DPC

DCC

PPP

PPC

PCC

CCC

—

en general pueden describirse relaciones de transicion mediante la agregacion de descripciones de transiciones:

E1E2E3E4E5... En

donde E representa uno de los cambios descritos de transicion

composicion de funciones mediante composicion de transiciones

—

Toda regla contiene elementos desconocidos que la componen

el numero de elementos desconocidos es infinito, pero no impiden la descripción de su funcionamiento, en un grado inteligible

mejora en la descripción = eliminacion de elementos desconocidos, aumento de dimensiones sobre objeto de estudio, complejidad++. paso a nuevos elementos desconocidos

A.6.5.7.1. Primitivas de construccion de reglas en Homos 1.0 orden de primitivas

complejidad adicional

	tiempo	complejidad

idempotencia, creacion, terminacion

todos los automatas celulares usan estos principios basicos

transformacion = creacion y terminacion de una o mas particularidades fuera del tiempo de simulacion

individualiza comportamientos, permite describir fenomenos

no es posible describir exactamente el momento del cambio ni la interaccion final en ese instante del tiempo!?

idempotencia: permanencia en el espacio. no explica porque permanece, solo establece que es asi

creacion: no explica razones, establece diferencia poblacional positiva

terminacion:

cambio o transformacion no explica como se produce el fenomeno en si mismo. habla de que existe ese fenomeno y que puede compararse con una accion semejante

las primitivas no explican como se produce el fenomeno en si mismo. hablan de que existe ese fenomeno, que es representable y que puede compararse con una accion semejante a la descrita por la regla.

para una explicacion se requiere descomponer las primitivas en otros elementos, pero esto es algo materia de investigacion.?

Durante el analisis de funcionamiento de las reglas de transicion en Homos 1.0, se encontró que estas pueden ser descompuestas en una serie de reglas fundamentales o primitivas. Una primitiva es una regla que no puede ser descompuesta en otras mas simples, no depende de otras para su construccion y definicion.

A continuacion se listan las primitivas encontradas y su notacion en protolenguaje homos:

Instanciacion: $\_ : H$

Representa la creacion de la instancia de una clase objeto a partir de una celda o espacio vacio, es una operacion implicita presente en el funcionamiento del simulador. Es empleada por las reglas que generan nuevas instancias de clases objeto en el ambiente durante la simulacion y por el desarrollador cuando crea el paisaje dinamico inicial de simulacion en el ambiente.

Idempotencia: $H : H$ (persistencia)

Representa el paso inalterado del estado interno de una instancia de clase objeto en el tiempo. Es implicita, justifica los intervalos de tiempo durante los cuales no se realiza la aplicacion de reglas de transicion sobre una instancia de clase objeto durante la simulacion. Por ejemplo, esta regla sirve para representar la idempotencia de una celda vacia $\{ \_ : \_ \}$, estaba vacia un paso antes y sigue estando vacia despues.

Representa la inalteracion en el estado interno de una instancia de clase objeto durante un paso de simulacion. Es implicita, justifica los intervalos de tiempo durante los cuales no se realiza la aplicacion

de reglas de transición sobre una instancia de clase objeto durante la simulación. Por ejemplo, esta regla sirve para representar la idempotencia de una celda vacía $\{ \_ : \_ \}$, estaba vacía un paso antes y sigue estando vacía después.

Representa el paso inalterado del estado interno de un inobjeto en el tiempo. Es implícita, justifica los intervalos de tiempo durante los cuales no se realiza la aplicación de reglas de transición sobre una instancia de clase objeto durante la simulación. Por ejemplo, esta regla sirve para representar la idempotencia de una celda vacía $\{ \_ : \_ \}$, estaba vacía un paso antes y sigue estando vacía después.

Representa el paso sin cambios de inalterado del estado interno de un inobjeto en el tiempo. Es implícita, justifica los intervalos de tiempo durante los cuales no se realiza la aplicación de reglas de transición sobre una instancia de clase objeto durante la simulación. Por ejemplo, esta regla sirve para representar la idempotencia de una celda vacía $\{ \_ : \_ \}$, estaba vacía un paso antes y sigue estando vacía después.

Representa el paso inalterado del estado interno de una instancia de clase objeto en el tiempo. Es implícita, justifica los intervalos de tiempo durante los cuales no se realiza la aplicación de reglas de transición sobre una instancia de clase objeto durante la simulación. Por ejemplo, esta regla sirve para representar la idempotencia de una celda vacía $\{ \_ : \_ \}$, estaba vacía un paso antes y sigue estando vacía después.

Cambio o Transformación H:I

cambio es la composición de creación y eliminación instantáneas

es una primitiva porque no es una operación definible en términos de las anteriores

el componente instantáneo hace la diferencia

ya que ese intervalo de tiempo es el mínimo definido por la aplicación

implica una subdivisión del tiempo no establecida en el marco temporal

$\{H: \_ \} \{ \_ : \_ \}$ pasos separados, no se hace notar que el cambio es en un mismo espacio

$\{H: \_ : \_ \}$ reuso de espacio pero en 2 tiempos

define 2 pasos diferentes

esta se da en un paso

Representa el cambio de una instancia de clase objeto por otra. Es explícita, implica la eliminación de H y la posterior instanciación de I en el siguiente paso de simulación. Es la descripción básica de las reglas que involucran cambio y alteración de clases.

Sirve también para expresar auto referencias si se asume que H e I son una misma instancia de clase objeto en la que la diferencia está expresada en el cambio de los valores que definen su estado interno. el estado H de la instancia de clase referenciada es cambiado a un estado I definido del mismo objeto.

H en y al instante posterior cambia en la clase objeto I.

Sirve también para expresar reglas de auto referencia si se asume que H e I son una misma instancia de clase objeto que difieren en los valores presentes en su estado interno. El estado H de la instancia

de clase referenciada es cambiado a un estado I definido del mismo objeto. un ejemplo de esto es la regla de movimiento, que altera el valor de su ubicacion en el espacio cuando es ejecutada.

esta primitiva sirve a dos propositos cambio completo de una instancia de clase por otra, cambio en el estado interno de una instancia.

H en y al instante posterior cambia en la clase objeto I.

Sirve tambien para expresar auto referencias si se asume que H e I son una misma instancia de clase objeto, diferenciadas unicamente por los valores presentes en su estado interno. El estado H de la instancia de clase referenciada es cambiado a un estado I definido del mismo objeto.

H en y al instante posterior cambia en la clase objeto I.

La instancia de clase objeto H existe y al instante posterior cambia en la clase objeto I.

esta primitiva se corresponde claramente con el planteamiento de cambio de estados presente en los automatas, asi como con la idea de cambio de clasificacion presente en la teoria de objetos

paso de Estado1 a Estado2

el objeto cambia de de un tipo a otro

cambio como una idempotencia con alteracion en el estado interno

idempotencia por que es el mismo objeto en el tiempo, pero que cambia uno o mas de sus parametros internos, sin que esto induzca a una reclasificacion de su género

oruga hambrienta:oruga no hambrienta

cambio como una terminacion e instanciacion instantanea de clases diferentes

Aunque es el mismo objeto, sus cambios internos son tan grandes que inducen a una reclasificacion de su género

oruga:mariposa

Terminacion o Decadencia H:_

La instancia de clase objeto H existe y al instante posterior desaparece, se reemplaza por una celda vacia. Es la descripcion basica de las reglas que involucran eliminaciones

Entre las primitivas presentadas, se encuentran dos deficiones de reglas ya implementadas en homos. las demas reglas son resultado de sus combinaciones.

La combinacion de estas primitivas da origen a las reglas implementadas en esta version del software. Con esta notacion es posible observar que es posible obtener nuevos tipos de reglas de transicion

La configuracion de la ubicacion de las celdas en el espacio es irrelevante en esta notacion para todas las reglas excepto para la de movimiento, se describió asi para emular el funcionamiento interno del simulador: este busca las clases presentes alrededor de la celda en evaluacion y luego determina si existe alguna regla que involucre a esas clases, sin importar su ubicacion exacta con respecto a la celda en evaluacion. posterior a esto se aplica las probabilidades y prioridades propias de cada regla,

realiza comparaciones y determina cual de esas reglas es la que finalmente será aplicada al vecindario en evaluacion.

excepto para la de movimiento

La descripcion de la ubicacion de las instancias de clases objeto en las celdas esta referida a la celda en evaluacion

usando coordenadas cartesianas

la celda elegida esta siempre en (0,0)

movimiento

descripcion de las reglas de transicion en esta notacion

Decadencia: existe una clase objeto en el instante anterior y al siguiente dicho objeto desaparece

Movimiento

Expansion

Retraccion

Alteracion

Eliminacion

Neutralizacion

Trasformacion

Reproduccion

+E Libera espacio

-E Ocupa espacio

N movimiento hasta en N celdas

Pr = cualquier valor entre 0 y 255

ubicacion: cantidad de cambios en el ambiente producidos por la aplicacion de la regla

poblacional: magnitud del cambio poblacional inducido por la aplicacion de la regla

Espacial: presenta el cambio del estado de las celdas en la cantidad de celdas liberadas u ocupadas por la aplicacion de

Decadencia: existe una clase objeto en el instante anterior y al siguiente dicho objeto desaparece

Movimiento

Expansion

Retraccion

Alteracion

Tabla A.23: Analisis de reglas de transicion mediante primitivas

Accion de las reglas							
Regla	Notacion	Ubicación	Poblacional	Espacial	Distintas	Notacion	Prioridad
Instanciacion	C	0	+1	-E	1	H:_	Pr
Idempotencia	I	0	0	0	1	H:H	Pr
Cambio clase	C	0	-1+1=0	-E+E=0	2	H:J	Pr
Cambio estado	C	0	0	0	1	Hi:Hj	Pr
Terminacion	T	0	-1	+E	1	H:_	Pr
Decadencia		0	-1	+E	1	H:_ \$0	Fija = 0
Movimiento		1	0	-E+E=0	1	H(f1,c1):H(f2,c2)	Pr
						H, _:_,H	
Expansion		0	+1	-E	1	H, _:H,H	Pr
Retraccion		0	-1	+E	1	H,H:H, _	Pr
Alteracion		0	-1 +1 = 0	0	3	H,I:H,J	Pr
Eliminacion		0	-1	+E	2	H,I:H, _	Pr
Neutralizacion		0	-2	+2E	2	H,I:_, _	Pr
Trasformacion		0	-1 +1 = 0	0	2	H:I	Pr
Reproduccion		0	+1	-E	3	H,I, _:H,I,J	Pr

Tabla A.24: Construcción de reglas de transición mediante primitivas

Descomposicion de Reglas de transicion en primitivas			
Regla	Compuesta	Primitivas	Notacion
Decadencia	No	Terminacion	H:_
Expansion	Si	Idempotencia e instanciacion	H, _:H,H
Retracción	Si	Idempotencia y terminacion	H,H:H, _
Alteración	Si	Idempotencia y cambio	H,I:H,J
Eliminación	Si	Idempotencia y terminacion	H,I:H, _
Neutralización	Si	Doble terminacion	H,I:_, _
Trasformación	No	Cambio (de clasificación)	H:I
Reproducción	Si	Doble idempotencia e instanciacion	H,I, _:H,I,J
Movimiento	No	Cambio (de ubicacion)	H(x1,y1):H(x2,y2)

Eliminacion:

Neutralizacion

Trasformacion

Reproduccion

todas las reglas excepto la de eliminacion

estudios de antonio S. Camara

ver modificaciones!

Tabla A.25: Ejemplo de nuevas reglas de transicion

Ejemplo de descripción de nuevas reglas a partir de la propuesta de notación de reglas con extenciones						
Regla	Ubicación	Poblacional	Espacial	Clases Req	Notación	Prioridad
Fisión de H en I y J	0	-1+1+1=+2	-2E	3	H,_,_:_:,I,J	Pr
Alteración múltiple	0	-1-1+1+1=0	0	4	H,I:J,K	Pr
Brote si hay 3 espacios libres	0	+1	-E	1	_,_,_:_:,H,_	Pr
Fusión de H,I,J en K	0	-1-1-1+1=+2	+2E	4	H,I,J:_:,K,_	Pr
Expansión binaria	0	+2	-2E	1	H,_,_:_:H,H,H	Pr
Neutralización múltiple	0	-3	+3E	3	H,I,J:_:,_,_	Pr
Repulsión en eje x	1	0	0	2	H,I:H,I.x++	Pr
“Rotación” derecha de 3 clases	3	0	0	3	H,I,J:J,H,I	Pr
“Intercambio”	2	0	0	2	H,I:I,H	Pr
“Traslado” a un espacio libre	1	0	0	1	H,_,_:_:,H	Pr

agrupacion, familias, familiaridad de reglas

accion

En la regla repulsión, la descripción de cambio en las coordenadas de I es un ejemplo de una posible forma notación para la descripción de ese evento.

Las reglas rotación, intercambio y traslado, van entre comillas ya que la forma de la descripción presentada permite múltiples interpretaciones. Se requiere de un medio de representación en la notación para hacer más precisa su descripción funcional.

Si se establece que puede hacerse referencia explicita al objeto contenido en la situacion inicial de la transicion, entonces se hace la siguiente equivalencia:

$$Id1, IdA, \dots, IdN: Id2, IdB, \dots, IdM$$

$$\&1, \&2, \dots, \&N : 1\&, 2\&, \dots, N\&$$

&numero, donde numero hace referencia al primer estado indicado por la regla de transicion

hace referencia a la ubicacion en

& indica referencia, la ubicacion del numero determina el lado del cual se obtiene la ubicacion de la referencia.

&2: el segundo elemento descrito en la seccion inicial de la transición

3& : el tercer elemento descrito en la seccion final de la transición

por el sentido de ejecucion de izquierda a derecha de la transición, Solo las referencias del lado izquierdo de la transición representan idempotencia por referencia.

***las referencias en sentido contrario se implementan por razones de simetria referencial, !!!!

idempotencia por referencia:

$Id1:1$ y $1:Id1$ son equivalentes

la referencias al igual que las parejas en una transicion, estan en relación 1 a 1

se propone que el uso de referencias se

referencias a una instancia de objeto &

referencias al tipo de clase de un objeto mediante #, solo para reducir el tamaño las expresiones descritas

el manejo de referencias en ambos operadores es el mismo

se requieren para esta extension de la notación, la en caso del manejo de referencias

En la regla atracción, se usa una función de evaluación de distancia entre H e I y una función de reubicación para I acorde con el propósito de la regla de transición. Esta es una de varias formas posibles de notación para la descripción de ese evento.

la bilocacion viola el principio de conservacion de cambio de estado que fundamenta la notación de transición

—

en el espacio, aumentando o disminuyendo la cantidad y ubicacion de celdas disponibles

poblacional aumentando o disminuyendo la cantidad de instancias de una clase objeto particular

la cantidad de clases objeto necesarias para definir la regla

todo incremento en la influencia poblacional implica

una mayor area de accion en el ambiente/espacio

—————

notacion de reglas y reglas de simulacion graficas [R]

en la notacion de las reglas graficas de simulacion propuestas en [r] se observan

ausencia especifica de notacion para el espacio: establece implicitamente las necesidades de espacio en la descripcion de las reglas, en la notacion presentada solo hace enfasis en los cambios sobre los objetos

transfomacion: describe una especie de transformacion alimentada : $B,A:D,-$ y para ser mas exactos propone $B,A,-:B,-B,$

habla de inicios de division, paso de celula en estado normal a estado de division celular (sin resultado final del proceso, dos o mas celulas resultado de la division)

Las reglas de no interaccion [no contact] y no cambio presentadas en [R] son resultado de las consecuencias funcionales de la idempotencia entre transiciones, por esta razón no se hace enfasis en ellas en este documento.

mas importantes reglas de transicion faltantes

Tabla A.26: Exploración de nuevas reglas de transición

Exploración de nuevas reglas a partir de la propuesta de notación de transiciones						
Regla	Ubicación	Poblacional	Espacial	Clases Req	Notación	
Atracción en eje x	x-1	0	0	2	Distancia_x(H,I)>1:H,I.acercar_en_x_a(&1)	
Transformaciones H en J, I en H y J en I	0	0	0	3	H,I,J:J,H,I	
Transformaciones H en I y I en H	0	0	0	2	H,I:I,H	
Terminación y creación de H	0	0	0	1	H,:-,H	
Rotación derecha de 3 clases	3	0	0	3	H,I,J:&3,&1,&2	
Intercambio	2	0	0	2	H,I:&2,&1	
Traslado a un espacio libre	1	0	0	1	H,:-,&1	
Terminación de H en presencia de J	0	-H	+E	2	H,2&:_,J	
J ocupa el espacio vario tras la terminación de H	1	-H	+E	2	H,J:&2,-	
Expresión en referencias sin sentido para el simulador					1&,2&:&2,&1	
Expresión no válida, bilocación de H					H,J:&1,&1	

atraccion [y repulsion]

herencia

Importantes en el sentido de su aplicacion biologica, aunque no se hacen explicaciones adicionales en este documento del porque de su importancia trascendente.

Paso adelante en reglas de transición:

Reglas de herencia dinámica

al heredar un atributo, se heredan tambien las distintas funciones relacionadas con la modificación de su valor.

reglas de contenencia

creacion de estados compuestos, objetos compuestos

los ejemplos son solo ideas de posibles representaciones construibles con la notación propuesta.

La flexibilidad de la notación permite predecir un alto nivel de complejidad al momento de su implementación práctica. Las capacidades enunciadas no son definitivas, cambios mayores son posibles en su forma final de presentación y funcionalidades disponibles.

A.6.5.7.2. ideas los objetos, la cantidad de relaciones de los objetos en el espacio determina la geometria

la geometria se establece en el espacio, los objetos se restringen por ella

reglas dependientes del espacio de las clases ... dependen de los conceptos definidos!

construccion de modelos mediante ideas simples.....!

modelo complejo explicado en terminos simples

todo sistema dinamico esta compuesto de relaciones de idempotencia, creacion y terminacion de sus elementos

el orden y la forma en que estos procesos se desarrollan caracteriza / diferencia a cada uno

transformacion establece cambios en la representacion interna/ externa

la interaccion de comportamientos simples constituye la complejidad del comportamiento total del sistema

las combinaciones de los comportamientos

orden de complejidad de los sistemas dinamicos a partir de la interaccion

magnitud de complejidad

elementos que intervienen

Tabla A.27: Exploración de nuevas reglas de transición a partir de la protonotación

Ejemplo de descripción de nuevas reglas a partir de la propuesta de representación/notación de reglas					
Regla	Notación	Ubicación	Poblacional	Espacial	Clases Req
Si H contiene I extraerlo en espacio vacío	H.contiene(I),_:H,H.extraer_2n(I)		H.contiene.I-	-E	2
Integrar en H a I y J	H,I,J:H.integrar(&2) H.integrar(&3),-,-	2	H.integra+=2	+2E	3
En presencia de I, H contiene a J	H,I,J:H.contener(&3),I,-	1	H.contenido++	+E	3
H almacena a I si almacén de H está vacío	H.almacen={},I:H.almacenar(&2),-		H.almacen++	+E	2
Si existe un I en el almacén de H, eliminar uno de ellos	H.almacen.existe_un(I):H.almacen.eliminar_d_un(I)				2
Heredar atributo color de H en I si I no tiene este atributo	Existe(H.color),Noexiste(I.color):H,I.heredar(&1.color)				2
Copiar en I color el valor del atributo color de H	H,I:H,I.color=&1.color	0	0	0	2
si existe I o J, eliminar H y reemplazar por el objeto que cumpla la condición.	H,solo_uno_de(I,J):&2,-	1	-1	+E	3 tipos, pero solo 2 requeridas
					Pr

comportamientos observados
espacio y geometria de interaccion
dados dos sistemas cual es mas complejo?
simplificacion de complejidades en sistemas dinamicos
significado de la complejidad de un sistema, tratabilidad
cual es la influencia de aumentar la cantidad de relaciones posibles entre objetos
hay o no diferencias entre espacios de 3 a 9? el espacio influye en estos elementos? como?
movilidad?
espacios con vertices diferentes? dimension de conexion promedio.
paralelismo implicito!!!!!!!

A.6.5.7.3. degradacion de representaciones transitividad de representaciones desde un nivel basico a uno complejo

en cada etapa se pierde algo de realidad

“transitividad con perdidas”

elemento indefinible no expresable en terminos objetivos lo que esta fuera de ser expresable, el siguiente nivel de gödel

primitiva: elemento definible mas basico, el ladrillo de construccion de conocimiento de la MBOR

regla de transicion: elemento compuesto de primitivas, obtenido por construccion de elementos mas basicos (primitivas)

metafora de regla de transicion: simil propuesto por el modelador de la realidad analizada

descripcion del elemento encontrado en analisis: resultado del analisis realizado al fenomeno encontrado

descomposicion del fenomeno, categorizacion

elemento en la realidad fenomeno tal cual es en la realidad, incomprensible en su totalidad sin una descomposicion previa.

desde la complejidad total intrinseca de la realidad a la simplicidad necesaria para la comprension del fenomeno

las primitiva tienen componente operacional e informativo

informativo permite establecer las cualidades del fenomeno

operacional permite establecer la forma en que se hace una cosa

descripcion de la forma operativa

¿que?

descomposicion

clases objeto, elementos

¿porque?

descripcion

componente causal de las primitivas

descripcion de atributos de clases

¿para que?

primitivas

reglas

¿como?

primitivas

reglas

modelo

¿cuando?, ¿donde?

simulacion

reglas

por composición de primitivas

orden de evaluación

por relacion explicita de causa efecto

cualquier pareja / combinacion causa-efecto no representable con las primitivas actuales representa una nueva primitiva nuevo tipo de primitiva

complejidad de regla = sumatoria de comparaciones, número de primitivas empleadas

si solo un objeto por celda = cantidad maxima de relaciones = vecindad de celda

si varios objetos por celda = numero ilimitado de relaciones, reducidas a un entero finito

definiciones

$I(a) = a:a$

$D(a) = a:_$

Tabla A.28: expresiones en el lenguaje?

Clases involucradas	regla	Primitivas	variación de orden de composición	Notación en composición de Primitivas(Clobjeto)	Proto Notación	
1	Idempotencia	I		I(a)	a:a	
1	Decadencia	D		D(a)	a:_	
2	Transfomación	T		T(a,b)	a:b	
1	Movimiento				a, _: _a	
1	Expansión	I,C	C,I	I(a),C(_a)	a, _:a,a	
1	Retracción	I,D	D,I	I(a),D(a)	a,a:a, _	
2	Neutralización	D,D		D(a, _),D(b, _)	a,b: _ , _	
2	Eliminación	I,D	D,I	I(a),D(b, _)	a,b:a, _	I
3	Alteración	I,T	T,I	I(a),T(b,c)	a,b:a,c	I I I
3	Reproducción	I,I,C	I,C,I C,I,I	I(a),I(b),C(c)	a,b, _:a,b,c	

$T(a,b) = a:b$; degenera en I si $a=b$, $T(a,a) = I(a)$

$C(a) = \_ :a$

C y D degeneran en I si $a=_ \Rightarrow \_ :\_$ idempotencia de vacío

orden irrelevante

idempotencia no tiene inversa

$I(a)$ no tiene inversa, la inversa es si misma? módulo

importa el orden de operandos

$T(a,b)$ inversa = $T(b,a)$, inversas reciprocas

$D(a)$ inversa de $C(a)$, inversa no reciproca

casos triviales

entradas = salidas, idempotencias

A.6.5.8. limites del protolenguaje

los limites ...

innato a todo lenguaje la capacidad de representacion

este protolenguaje representa

representacion de abstracciones, no hay disponibles primitivas especificas para ellas

A.6.5.9. Caracteristicas fortalezas metodologicas de la MBOR

usando como trasfondo la degradacion de representaciones obtenemos que ...

discernir primitivas nuevas permite construir reglas mas metaforizables - convertibles asimilables, trasladables a metáforas conocidas

la experiencia del modelador es importante.

modelos de realizados por niños nos acercarian a una vision de la construccion mas primitiva posible

comprension del mundo segun su capacidad, mirada a la construccion de pensamiento en el cerebro en construccion

no es posible presentar , alcanzar representaciones para todas las primitivas posibles, estas a su vez pueden descomponerse en superprimitivas??

descomponer el tiempo en subelementos abstracto

la experiencia de modelado es relevante

el conocimiento en el area en estudio es relevante

construccion de modelos e igualacion de estos con fenomenos externos (ingenieria inversa?)
construccion teorica de conocimiento

A.6.5.9.1. limitaciones de la mbor

A.6.5.9.1.1. fallas inducidas falencias de analisis preliminar / adquisicion de la informacion se notan en el modelado

falencias en el desarrollo del lenguaje prosa se notan

relaciones erradas entre objetos y clases

ambito del problema, alcances, limitaciones

determinacion de sustantivos, sujetos de accion y verbos

conversion de sustantivos a objetos y acciones a verbos

conversion a objetos y reglas

objetos no representables? cuales?

reglas no representables? cuales?

implementacion en la herramienta

medida del tiempo erroneas, manejo de dimensiones?

A.6.5.9.1.2. fallas internas que pasa si no es posible convertir un verbo?

necesidad de obviar reglas porque la herramienta no tiene como soportarlas???!!!

si se ve como manzana, huele como manzana, sabe a manzana es una manzana??? no necesariamente se establece un modelo y se obtienen resultados, es lo que se queria representar? verificacion y validacion intuitivas, de comparacion de datos.

ver <http://www.informs-sim.org/wsc06papers/006.pdf>

book "Simulation Modeling and Analysis"

que puedo representar conceptualmente que no pueda ser implementado en la herramienta???

analisis, comprension de curvas de comportamiento

ausencia de conjunto de valores aleatorios repetibles para realizar comparaciones, en situacion de misma "eventualidad"

conocimiento obtenido de modelado solo mediante curvas de comportamiento. no se hacen otros analisis sobre el modelo creado.

detalles descriptivos de relaciones.

el modelo/formalismo matematico no es util para realizar evaluaciones u obtener conclusiones sobre el modelo, no hace parte de los reportes de la herramienta.

la nomenclatura define elementos innecesarios, no se usa en la creacion de reportes.

A.6.5.9.1.3. fortalezas descripcion grafica de eventos en ambiente

descripcion de relaciones entre objetos mediante reglas

analisis de sensibilidad mediante escenarios multiples

A.6.5.10. elementos eliminados

reglas de no cambio y no contacto descritas en

A.6.5.11. sobre la metodologia

En este aspecto, para Homos la MBOR cubre el proceso de creación de modelos, que se resume en definir los elementos necesarios para la realización del proceso de simulación y el como pasar de la descripción y datos del fenomeno a un modelo computacional (formal) que reproduzca su comportamiento y con el cual hacer posible la experimentación. Aqui el interes está dirigido indistintamente al usuario final o el usuario modelador de la aplicación.

Pero en el plano del aula de clase (La aplicación practica del PS con enfoque constructivista) están ausentes en este esquema las implicaciones propias de los roles docente y alumno y como ellos hacen uso eficiente de la herramienta. Incluir estos perfiles requerirá establecer un marco metodologico mas amplio. "...[se] hace necesario el diseño de un ambiente educativo que integre docentes, alumnos, material educativo y formulación de objetivos y contenidos de la clase."

Hasta el presente no se ha establecido:

- Una metodología que defina los requisitos, la forma de construcción de materiales - modelo destinados al aula de clase.
- El rol del educador y los aprendices con respecto a la herramienta o mas exactamente:
- Una metodología de trabajo que facilite la introducción de la herramienta en el aula de clase y dentro de ella, y
- Una metodología de trabajo que integre la herramienta al proceso curricular. Actualmente se llevan acabo en el grupo Simon labores de investigación destinadas a establecer estos marcos teoricos.

A.6.5.12. aplicaciones de la metodologia

objetos como organismos vivientes

comportamientos inspirados en analogias biologicas

no se ofrece una guia para la adaptacion hacia la mbor.

Sobre el esfuerzo cognitivo necesario requerido por la aplicación Estos son los elementos considerados dentro de lo congintivo que son requeridos para el manejo de la aplicacion.

Control Psicomotriz: coodinación ojo mano mediante el uso del raton requerido por la aplicación.

Comprensión / Interpretación: Requerida para la construcción de los modelos y la aplicación en ellos de cambios que representen sus características durante la fase de experimentación.

Comparación / Relación: Requerida para la interpretación y diferenciación de las diferentes curvas producidas durante la fase de experimentación. Análisis / Síntesis: requerida en la fase inicial de construcción de los modelos, y durante su construcción en la herramienta.

Exploración / Experimentación: Se hace presente durante la fase de ajuste de los modelos y su explotación, refleja los supuestos iniciales del modelador y como estos evolucionan en el tiempo, es la funcionalidad principal del software.

Observación / Valoración: necesaria durante la fase previa del proceso de modelado, de donde se obtienen las características preliminares del modelo. Promovida en el proceso de análisis de resultados obtenidos durante la simulación.

Expresión - verbal, escrita, gráfica... / Crear : Se presenta el modelo gráficamente y pero no se incluye dentro de esta categoría debido a falencias en cuanto a documentación de los modelos y del proceso de construcción de los mismos.

Reflexión metacognitiva: Necesaria en la fase de construcción de los modelos, incluye los supuestos iniciales del modelador. Es también un producto del proceso de creación de modelos, representa uno de los valores agregados principales del software, se incentiva durante la comparación activa del comportamiento real del fenómeno y la virtual del modelo, y al usar el modelo creado como herramienta en la creación de posibles proyecciones a futuro del fenómeno.

Razonamiento - deductivo, inductivo, crítico: Necesario durante el proceso de construcción de los modelos en la herramienta. Estos elementos están presentes pero no son parte activa del trabajo en la presente versión del software:

Resolución de problemas: El software no tiene una orientación específica en cuanto a la solución o resolución de problemas. Por otro lado, dentro del proceso de creación del modelo se requieren de este tipo de habilidades, especialmente en la fase de ajuste.

Memorización / Evocación: Los conceptos manejados por la MBOR son mínimos, esta exigencia es más bien particular y depende en gran medida de la complejidad del modelo implementado. Pensamiento divergente / Imaginación: No se usan en la herramienta otros enfoques para la creación de modelos, solo se cuenta con la MBOR para este proceso.

Cálculo : Aunque hay generación de datos y presentación de tablas y gráficas, no se incluyen en la herramienta funciones dedicadas al tratamiento numérico de los datos, originado esto en la naturaleza inicialmente no matemática del software. Elementos específicos de la herramienta
Condiciones necesarias para que un fenómeno pueda ser representable en forma de objetos y reglas
Las características básicas de tales fenómenos son: Que sea posible identificar elementos (objetos) y que estos participen activamente en el fenómeno. Que los objetos encontrados realicen acciones (reglas) y que estas sean representables en la herramienta por equivalencia directa o mediante sustitución por medio de combinaciones de ellas.

Sobre el pensamiento sistémico y la naturaleza constructivista de la herramienta De acuerdo con las formas básicas de Pensamiento Sistémico , el software apoya los procesos educativos constructivistas en cada uno de sus aspectos:

Tabla A.29: Representación de Formas de pensamiento en HOMOS

Forma de pensamiento	descripción	Representado en Homos
Estructural	Se reconoce causalidad entre diversos elementos de un fenómeno – visión como sistema	Causalidad debida a la forma, los objetos estan relacionados por reglas, formalizada en el diagrama Bor.
Dinámico	Identifica patrones de comportamiento que rigen la dinámica de los sistemas	Patrones identificables como las curvas obtenidas de las simulaciones.
Genérico	Identifica similitudes entre fenómenos de naturaleza diferente	Modelos de origen diferente e igual estructura se comportaran igual. Permite la construcción de modelos que comparar.
Operacional	Hace referencia a cómo es matemáticamente que funcionan las cosas y cómo están planteadas las ecuaciones para permitir la comprensión del modelo	Aunque Homos no hace referencia explicita a ecuaciones o expresiones matematicas, estas se describen en terminos de las relaciones entre los objetos y reglas.
Científico	Cuantifica variables. Propone y prueba hipótesis mediante experimentos	Los valores presentados en las vistas y las gráficas de comparación durante la simulación exponen las diferencias entre los experimentos realizados según se realizen cambios en los parametros usados durante el proceso de ejecución del modelo.
Cíclico	Identifica la relación entre estructura del modelo y el comportamiento observado en el fenómeno.	La estructura define el comportamiento . Mediante los escenarios es posible observar las diferencias en el comportamiento del modelo de acuerdo a los cambios realizados en los parametros durante la simulación.
Continuo	Aprecia y explica los fenómenos como resultado de interdependencias continuas y no como hechos aislados.	El usuario mediante la observación de la evolución espacio-temporal del fenomeno y la visión de las curvas de comportamiento obtenidas, encuentra en la estructura de las relaciones (las reglas) la explicación a los desarrollos observados.

Mediante los elementos citados, Homos tiene los fundamentos necesarios para servir a su proposito como herramienta constructivista, al apoyar positivamente el proceso de aprendizaje. también debe notarse que la idea de sistema esta omnipresente en Homos, revelandose tácita o explicitamente en la mente del desarrollador al crear un proyecto, cuando contempla las posibilidades frente al tema elegido y en los usuarios cuando ejecutando el modelo observan la totalidad a la que tienen acceso y como esta les permite explorar sin riesgo.

Sobre el enfoque pedagogico constructivista

Su lema es “el aprendizaje centrado en el aprendiz”. Es un proceso basado en la auto-construcción de conocimientos, apoyado por el profesor quien aporta la guía y experiencia necesarias para una exitoso ambiente de aprendizaje en el tema a profundizar. En este esquema el estudiante es responsable de su aprendizaje, estableciendo el ritmo y el orden de las actividades, promoviendo el contraste de conceptos e hipótesis entre sus compañeros. En este aspecto, Homos facilita el proceso mediante las labores de creación de modelos y contraste de hipótesis.

Sobre la relación existente entre la Mbor y la dinámica de sistemas

Puede verse en la introducción de este documento el origen de la herramienta y el proceso de ideas de las cuales surge Homos. En forma puntual y resumida, en la siguiente tabla se establecen algunos puntos de contacto y diferencias entre la DS y la MBOR:

el modelo mas simple creable en homos esta definido por una clase y un conjunto de reglas que defina su comportamiento

por las reglas existentes los cambios que pueden esperarse en dicho modelo serian en las dimensiones cambios espaciales

reubicación dinámica de recursos, limitados por la disponibilidad de espacios en el ambiente para crecer

si la clase objeto tiene movimiento entonces esta puede reubicarse dinámicamente alterando la densidad de esa clase en el ambiente y favoreciendo su número

el movimiento en una clase puede favorecer el crecimiento al permitir que se encuentren espacios vacíos dentro del ambiente donde multiplicarse así como desfavorecer esta misma acción en otra clase.

las relaciones son inversamente proporcionales, a mayor cantidad de instancias presentes en el ambiente

se reduce l

aumenta la probabilidad de interacciones con otras clases

ciclos de realimentación

positivos y negativos

	ambiente	

, menor es la capacidad de expansión

cambio poblacional

reglas aplicadas a una unica clase hacen

la creacion de interacciones poblacionales entre clases aumenta la complejidad del modelo.
Dependiendo de si las interacciones son positivas o negativas aparecen sistemas de ecuaciones no lineales

la interaccion de reglas simples construyen sistemas complejos

las reglas que implican cambios internos tienen componente de idempotencia

estados como clases

	Giro de Ciclo		total poblacional	Finito y Ciclico
Regla	Poblacion	Condiciones de aplicación		Espacio
Idempotencia	Neutro	Marco espacio temporal	indiferente	Neutro
Creacion	Positivo	Espacio libre		Negativo
Cambio	¿?		neutro	
Cambio			indiferente	
Terminacion	Negativo	Existencial	negativo	Postivo
Decadencia	Negativo	Existencial	negativo	Positivo
Movimiento		Espacio libre	indiferente	
Expansion	Positivo	Espacio libre	positivo	Negativo
Retraccion	Negativo	Objetos de la misma clase	negativo	Positivo
Alteracion	Bivalente sobre alterado	# de alterables	neutro	Nulo
Eliminacion	Negativo	# de eliminables	negativo	Positivo
Neutralizacion	Negativo	# de neutralizables	negativo	Positivo
Trasformacion	Bivalente	Existencial	neutro	Nulo
Reproduccion	Positivo	Complemento y espacio libre	positivo	Negativo

densidad = poblacion / area

Suponiendo que la clase tiene instancias en el ambiente y que tiene la regla analizada tiene probabilidad de ejecucion > 0 , se listan las otras condiciones necesarias para su ejecucion.

a mayor cantidad de condiciones de aplicacion, menor la probabilidad de ejecucion

el aporte de la dimension del espacio en el modelo es la de reducir la probabilidad de interaccion de las instancias de clases objeto en el ambiente, agrega necesidades existenciales adicionales a las condiciones propias de ejecucion de las reglas

limite finito

limite ciclico

limite infinito

los cambios al espacio no se tienen en cuenta!

el numero de clases objeto presentes en el modelo que actuen como niveles indican el grado de la ecuacion diferencial subyacente

sobre las formas estructurales subyacentes en los modelos

modelado de un objeto en homos

una clase objeto y multiples reglas

varias clases objeto y multiples reglas

diagrama bor lectura de las estructuras graficas

se propone la siguiente notacion para la estrucutra grafica

a usanza de la estructura de las reglas segun el protolenguaje

un inobjeto cambia a otro por la accion de tercer inobjeto

transicion

reproduccion

neutralizacion

Las relaciones de las clases objeto nos recuerdan las estructuras de los grafos, aspecto en el cual Homos hace presente sus fundamentos en automatas.

Las relaciones las construyen las reglas que unen a los objetos, existen figuras fijas o patrones en las cuales se establecen estas relaciones.

Al crear un modelo en Homos las clases objeto estaran en forma combinada o simple, en una o mas de los siguientes tipos de formas:

Por el orden de sus relaciones

Estructura Simple o sin relaciones entre clases:

Las clases objeto presentes no tienen reglas que los relacionen entre si. Ejemplo: la representación una partícula en suspensión con movimiento aleatorio.

Secuencias finitas de objetos, (un rosario o líneas de objetos).

Un tipo de objeto es eliminado para dar paso a otro, en secuencia. Ejemplo: la metamorfosis de la mariposa, pasando sucesivamente por huevo, oruga, crisalida y mariposa. Cada paso esta relacionado por reglas que implican cambio, como transformación, alteración o reproducción.

Bifucación, Elección multiple o estados multiples simultaneos (arboles de seleccion) La clase objeto puede convertirse solo en una de varias clases posibles cuando se produzca el evento establecido por una regla determinada. Una clase objeto dentro del ramal sera elegible o no deacuerdo con las prioridades de las reglas y sus probabilidades para la clase objeto. Ejemplo: Una partícula de arena luego de ser golpeada por otra puede pasar a uno de varios estados. El movimiento posterior al impacto depende del angulo con el cual se haya producido el choque: puede quedarse quieta, moverse con la mitad de la energia recibida o moverse con el 100 % de la energia recibida en el impacto

en una dirección determinada. Para el modelo del ejemplo, estos son los 3 estados simultáneos posibles luego del evento que los desencadena.

Secuencias cerradas: Son las estructuras en las cuales se regresa al objeto original que inició la serie, formando un ciclo cerrado unidireccional. Ejemplos: los cambios de estado regulares de un semáforo, de verde al amarillo y luego al rojo sucesivamente; La oscilación de una partícula en 2 o más estados recurrentes; el modelado de una población de individuos en la dinámica del voto: cada individuo está de acuerdo con una de varias opciones para elegir, e influyen aleatoriamente en la elección de sus vecinos. Tal y como está descrito, si no se rompe el orden de la cadena, el ciclo de cambios continuará indefinidamente. Las secuencias cerradas finitas son posibles al insertar en la lista una clase objeto que contemple la decadencia entre sus reglas de comportamiento o mecanismos de bifurcación a clases con esta característica.

Clases objeto como indicadores de estado, usando secuencias bidireccionales o multidireccionales

Las clases objeto pueden pasar a la clase siguiente o devolverse a la anterior según sean los parámetros del modelo. Uno de sus posibles usos es la representación del aumento gradual de la dificultad o facilidad para cambiar el estado de la clase objeto. Ejemplo: la representación de la compresibilidad del aire alrededor de un obstáculo. La densidad representada estará en función de la aglutinación de partículas de un mismo estado alrededor del obstáculo. A mayor concentración, mayor cantidad de interacciones aumentando la probabilidad de pasar al siguiente nivel, hasta llegar a la representación máxima disponible de densidad en el modelo. Sin aglomeraciones, las partículas difícilmente podrían cambiar de estado, retornando rápidamente y manteniéndose en el estado inicial.

Redes de objetos

Pueden presentarse matrices multidimensionales, grafos, redes de objetos, en los cuales la tendencia de interacción será de todos contra todos. Ejemplo: representación de una cadena trófica o un sistema ecológico.

Por sus relaciones de influencia

Aunque la MBOR no hace referencia explícita a los conceptos de manejo de material y de información, el modelador puede hacer una implementación de estas ideas en la herramienta:

Influencia de un objeto sobre otra influencia poblacional

Una o más reglas relacionan diversas clases objeto, donde al menos uno de ellos influye positiva o negativamente sobre la población del otro. Ejemplo: los lobos comen conejos, reduciendo su población. Aquí las clases objeto Lobo y conejo son objetos únicos sin etapas o fases.

Influencia sobre sus estados

Una o más reglas relacionan diversas clases objeto, donde al menos uno de ellos influye positiva o negativamente sobre una de las etapas, estados o fases del otro. Ejemplo: los tiburones comen peces adultos, reduciendo su población. Los peces están representados por varias etapas: huevos, alevinos, adultos. Los tiburones influyen sobre la población de peces solo durante su etapa adulta.

Aspectos pedagogicos

Este apartado trata sobre los fundamentos conceptuales sobre los que Homos esta fundado y como Homos apoya el proceso de aprendizaje.

Sobre la tematica, el area y la materia de aplicacion

Homos es un simulador de proposito general, con el es posible:

Experimentar, es su función principal, luego de la creación o el uso de un modelo determinado, le permite al usuario probar hipotesis (representadas como cambios en los parametros)sobre el modelo y asi corroborar o refutar los paradigmas planteados en el modelo

el entrenamiento de destrezas, al mejorar las habilidades cognitivas directamente relacionadas con la tematica del modelo asi como en el modelado basado en objetos y reglas.

Instruir, al mejorar el conocimiento en un tema determinado a traves de una aproximación grafica y dinamica del fenomeno representado y teorica mediante la manipulación sus parametros basicos. se induce por esta via en la profundización de los conceptos planteados en el modelo. también sirve como herramienta de introducción en el universo del pensamiento sistémico, de la observación dinamica del fenomeno representado y la conciencia sobre la influencia de las decisiones en la evolución un sistema.

Motivar, mediante la experimentación a la observación del comportamiento de los fenomenos en la naturaleza y el estudio de los fenomenos que en ella se presentan.

Explorar: mediante la observación de los cambios ante las modificaciones o alteraciones al modelo actualmente en ejecución.

también puede convertirse en una opción de sano entretenimiento, ofreciendo reto intelectual a sus usuarios en la búsqueda de la perfección y mejora de los modelos creados con la herramienta.

A.6.5.12.1. Limitaciones de representación y modelado presentes en Homos 1.0 A continuación se presentan algunos elementos que por experiencia, por representar una innovación y por estar presentes en otras herramientas de servicio similar merecen ser contempladas como faltantes, elementos de mejora para la siguiente versión de homos.

[descomponer la mbor en sus elementos fundamentales y explicar las distintas ausencias, donde es posible mejorar a la metodología]

el proceso de modelado no es lineal

prototipo de modelo bor 1

ciclo de simulacion:

ejecucion del modelo

comparacion con respecto al comportamiento esperado

edicion sobre el modelo = correccion total por diferencias significativas con el comportamiento esperado prototipo de modelo bor i

edición sobre el modelo = ajustes menores en configuración de elementos prototipo de modelo bor i

fin de ciclo de simulación

prototipo final = modelo bor n

elaboración de distintos escenarios= el escenario de desarrollo + escenarios adicionales

presentación de resultados en cada caso

la duración del ciclo final depende de muchos factores, pero los más importantes en este proceso son el nivel de conocimiento sobre el fenómeno a representar y la experiencia de uso con la herramienta.

El ajuste al comportamiento esperado es un proceso convergente. Se alcanza a partir del punto en el cual cuando el modelador solo realiza ajustes de configuración en los distintos parámetros y se inicia la creación de otros escenarios de prueba del modelo.

La premisa de la orientación a objetos es la comprensión del mundo mediante su modelamiento como entes independientes y complejos (objetos) que se comportan, interactúan y cambian de estado.

describen las implicancias del espacio

Para la MBOR, los autómatas celulares establecen el tiempo y el espacio de la metodología, determinan la forma en la cual se realizarán las evaluaciones dentro del proceso de simulación. La orientación a objetos trae la complejidad de la representación de los elementos presentes en los fenómenos y su transformación-abstracción como objetos, su organización y jerarquización como clases y subclases, su caracterización particular mediante propiedades y métodos.

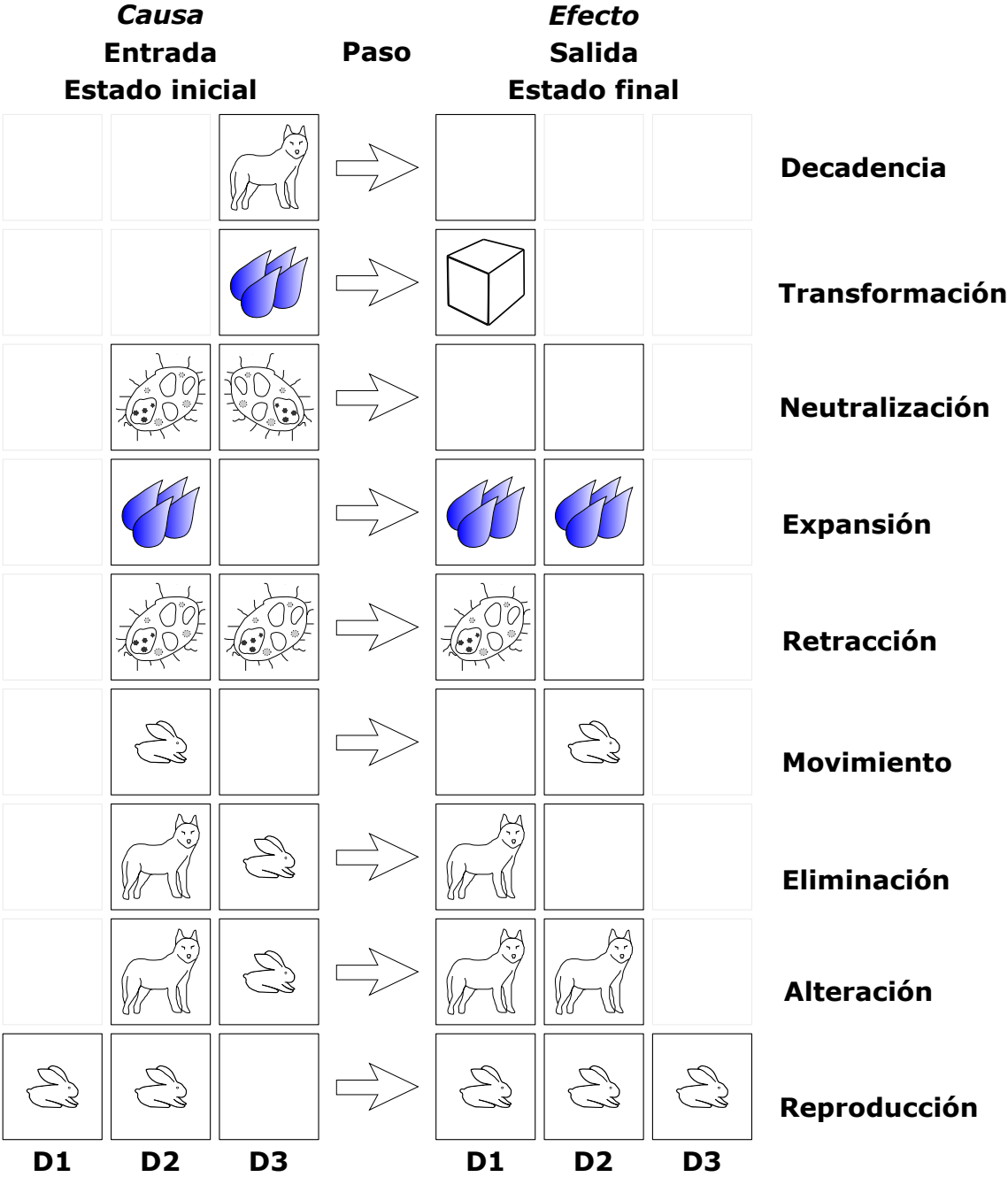
teoría de objetos: modelamiento en espacio mediante punto en física hace irrelevante la forma y dimensiones del objeto para el espacio, solo requiere que este exista y ocupe un lugar

inclinación al modelado de objetos tangibles

definición de eventos

fortalece el cambio de estado de los autómatas mediante la representación de los elementos en el ambiente mediante objetos.?

Figura A.20: Estructura y representación gráfica de reglas de transición en la MBOR



Fuente: Autor.

A.6.6. Resultados del análisis

a partir de los elementos encontrados durante el análisis podemos afirmar que:

Dinamica de Sistemas	Metodología basada en objetos y reglas
Permiten la formalización de conocimiento mediante la creación de modelos	
Es un cuerpo de disciplinas teoricas	Es una herramienta teorica para la dinamica de sistemas
Requieren poder computacional para usar y validar los modelos creados	
Cuenta con lenguaje descriptivo propio matematico, Usa funciones y ecuaciones diferenciales como fundamento basico	Usa reglas de producción, teoria de objetos y una variación de computación usando automatas celulares
Descripción del modelo basada en ecuaciones diferenciales	Usa descripciones texto del problema, Describe objetos y sus relaciones mediante reglas, caracterizadas por parametros especificos y justificadas por metáforas simples de inspiración biologica
Tienen lenguajes gráficos simples	
Manejo simplificado mediante el lenguaje grafico, requiere conocimientos basicos de calculo para abordar su descripción matematica	El lenguaje grafico es simple, aunque no tiene notación formal (un lenguaje particular de la aplicación), su descripción de objetos y reglas es facilmente asimilable, sin necesidad de profundizar en su aspecto matematico.
Tiene bien definido el concepto de patrones, tal como puede verse en los road maps	Por generar estructuras, la presencia de patrones esta garantizada, aunque se han realizado pocos estudios al respecto .
Su lenguaje grafico cuenta con Diagramas de flujo – nivel y diagramas de influencia	Usa diagramas basados en objetos y reglas
Manejan bucles o ciclos de realimentación	
Explicito, como parte de los diagramas de influencia y flujo nivel	Implicito: de describe mediante el diagrama Bor una estructura de relaciones, la dimensión temporal permite la construcción de ciclos
Usa niveles como acumuladores, que son abstractos y generales	Usa valores poblacionales, explicitos y particularizables en el ambiente.
Hace distinciones entre flujos de material y de informacion	No hace distinciones explicitas en ese aspecto, aunque generalmente asume el manejo flujos de materia. Del modelador depende establecer el significado de las relaciones en el modelo.
Comtepla el uso de retardos ya sea de material o de informacion	No se maneja ese concepto
El proceso de simulación es iterado, la ilustración del comportamiento del modelo mediante curvas en el plano	
Usa variables Continuas	Usa variables Discretas
No incluye al espacio como parte activa de sus supuestos de modelado	Usa al espacio como parte activa de la simulación y de la representación del modelo
Fundada a principios de 1960 por Jay Forrester en el MIT Sloan School of Management	Nacido en 1998; iniciativa del grupo Simon, UIS
Ampliamente difundido, usado en la toma de decisiones, soporte teorico de aplicaciones de empresas software de todo el mundo con pequeñas variantes	267 Localmente difundido, Homos es unico en su especie

Tabla A.31: Tabla comparativa entre Dinámica de sistemas y MBOR

Tabla A.32: Algunas limitaciones presentes en Homos 1.0

Descripción	Implementación en la herramienta
<i>Superposicion de objetos en el espacio</i> La vaca sobre el pasto: Se busca representar los patrones de crecimiento del pasto en un terreno determinado completamente verde. Esto incluye a una vaca de los alrededores que gusta de pastar libremente en ese campo. . .	En Homos 1.0 se asume que todos los objetos estan en un mismo plano y que cada celda del ambiente puede contener solo un objeto a la vez. La vaca y el pasto no estan simultaneamente en una misma celda del ambiente. Con la herramienta actual, la vaca pueda dar un paseo solo si está alimentándose del pasto más próximo, abriéndose paso por el terreno.
<i>Contenencia de objetos por un medio</i> El pez en el agua: Un pez oceánico decide arriesgadamente nadar por aguas costeras cerca de una isla remota. El agua y la tierra hacen contacto en la playa, zona límite hasta donde el pez puede llegar. . .	Dependiendo de las habilidades del modelador, podría suceder durante la simulación que el pez “nade” fuera del agua. En Homos no existe el concepto de contenencia: solo es posible para el pez nadar dentro el agua. Es factible representar algo como esto mediante la creación de objetos adicionales en el ambiente que limiten el movimiento y les sirvan de bordes, sacrificando simultáneamente memoria y velocidad de ejecución en el proceso.
<i>Agrupación de clases objeto</i> El objeto mariposa: las mariposas son seres vivos que pasan por 4 etapas durante su desarrollo: huevo, oruga, crisálida y mariposa.	Para efectos de modelado, se determinan las diferentes etapas del objeto con sus reglas basicas y se implementan una a una. No existe en homos una clase (o super clase) en la cual se agrupen las diversas etapas de un mismo objeto, ni el soporte para la asociación de clases del tipo A es un B, A pertenece a B, entre otras funciones.
<i>Manejo de propiedades particulares de la clase objeto</i> Niveles de bienestar: En la empresa X, se contemplan 3 niveles de bienestar por cada empleado que representan respectivamente: su estado economico, afectivo y emocional. En cada uno, el empleado puede estar bajo, medio y alto.	No existen otras propiedades asignables en homos mas que las provistas para cada regla. No es posible crear nuevos parametros a las reglas existentes ni crear nuevas reglas.
<i>Manejo de propiedades continuas en la clase objeto</i> Dinero del banco: En promedio, el banco gana 5 centavos Por cada transacción exitosamente realizada. . .	Aunque homos está fundado en operaciones discretas, la ausencia de elementos de tipo nivel (acumuladores) y el manejo de variables numericas para el control y registro de operaciones limita la aplicación de homos a un conjunto restringido de representaciones, generalmente de tipo biologico.
<i>Manejo de influencias multiples sobre un objeto</i> Presión de grupo: Cuando existan mas de 3 celulas muertas alrededor, la celula en evaluación muere.	Aunque en homos una celda pueda relacionarse hasta con 8 vecinos, en homos las relaciones entre objetos son uno a uno, el ejemplo propuesto no es representable.
<i>Inclusión de reglas fundadas en otras disciplinas</i> Repulsión y atracción electrica: Se tienen en un recipiente una serie de moléculas eléctricamente cargadas. Se desea ver el efecto de la atracción y repulsión eléctrica.	Las reglas de transición biológicas usadas en Homos pueden representar difusamente fenómenos de este tipo.
<i>Interacción de particulas sin contacto</i> Transmisión inalámbrica: Una torre de transmisión envia datos a todas las antenas de la region. . .	Igual que el caso anterior, no existen algoritmos implementados que emulen o implementen este tipo de funcionalidades.
<i>Manejo de herencias</i> Clase ave, gallina y canario: Se define una clase principal ave y de ella descienden las clases gallina y canario. . .	Existen implementadas la idea de clases objeto e instancias, pero no hay mecanismos para la realización efectiva de operaciones de herencia en la herramienta.
<i>Química y Conservación de la materia</i> Moléculas: reaccion den molculas de Hidrogeno y Iodo*****	La forma en la que se relacionan estas moléculas no es realizable por las actuales reglas de transición previstas en homos 1.0.

Tabla A.34: Algunas limitaciones presentes en Homos 1.0

Descripción	Implementación en la herramienta
<i>Superposición de objetos en el espacio</i> La vaca sobre el pasto: Se busca representar los patrones de crecimiento del pasto en un terreno determinado completamente verde. Esto incluye a una vaca de los alrededores que gusta de pastar libremente en ese campo...	En Homos 1.0 se asume que todos los objetos están en un mismo plano y que cada celda del ambiente puede contener solo un objeto a la vez. La vaca y el pasto no están simultáneamente en una misma celda del ambiente. Con la herramienta actual, la vaca puede dar un paseo solo si está alimentándose del pasto más próximo, abriéndose paso por el terreno.
<i>Contenencia de objetos por un medio</i> El pez en el agua: Un pez oceánico decide arriesgadamente nadar por aguas costeras cerca de una isla remota. El agua y la tierra hacen contacto en la playa, zona límite hasta donde el pez puede llegar...	Dependiendo de las habilidades del modelador, podría suceder durante la simulación que el pez <i>nade</i> fuera del agua. En Homos no existe el concepto de contenencia: solo es posible para el pez nadar dentro del agua. Es factible representar algo como esto mediante la creación de objetos adicionales en el ambiente que limiten el movimiento y les sirvan de bordes, sacrificando simultáneamente memoria y velocidad de ejecución en el proceso.
<i>Agrupación de clases objeto</i> El objeto mariposa: las mariposas son seres vivos que pasan por 4 etapas durante su desarrollo: huevo, oruga, crisálida y mariposa.	Para efectos de modelado, se determinan las diferentes etapas del objeto con sus reglas básicas y se implementan una a una. No existe en homos una clase (o super clase) en la cual se agrupen las diversas etapas de un mismo objeto, ni el soporte para la asociación de clases del tipo A es un B, A pertenece a B, entre otras funciones.
<i>Manejo de propiedades particulares de la clase objeto</i> Niveles de bienestar: En la empresa X, se contemplan 3 niveles de bienestar por cada empleado que representan respectivamente: su estado económico, afectivo y emocional. En cada uno, el empleado puede estar bajo, medio y alto.	No existen otras propiedades asignables en homos mas que las provistas para cada regla. No es posible crear nuevos parámetros a las reglas existentes ni crear nuevas reglas.
<i>Manejo de propiedades continuas en la clase objeto</i> Dinero del banco: En promedio, el banco gana 5 centavos Por cada transacción exitosamente realizada...	Aunque homos está fundado en operaciones discretas, la ausencia de elementos de tipo nivel (acumuladores) y el manejo de variables numericas para el control y registro de operaciones limita la aplicación de homos a un conjunto restringido de representaciones, generalmente de tipo biológico.
<i>Manejo de influencias múltiples sobre un objeto</i> Presión de grupo: Cuando existan mas de 3 celulas muertas alrededor, la célula en evaluación muere.	Aunque en homos una celda pueda relacionarse hasta con 8 vecinos, en homos las relaciones entre objetos son uno a uno, el ejemplo propuesto no es representable.
<i>Inclusión de reglas fundadas en otras disciplinas</i> Repulsión y atracción eléctrica: Se tienen en un recipiente una serie de moléculas eléctricamente cargadas. Se desea ver el efecto de la atracción y repulsión eléctrica.	Las reglas de transición biológicas usadas en Homos pueden representar difusamente fenómenos de este tipo.
<i>Interacción de partículas sin contacto</i> Transmisión inalámbrica: Una torre de transmisión envía datos a todas las antenas de la región...	Igual que el caso anterior, no existen algoritmos implementados que emulen o implementen este tipo de funcionalidades.
<i>Manejo de herencias</i> Clase ave, gallina y canario: Se define una clase principal ave y de ella descienden las clases gallina y canario...	Existen implementadas la idea de clases objeto e instancias, pero no hay mecanismos para la realización efectiva de operaciones de herencia en la herramienta.
<i>Química y Conservación de la materia</i> Moléculas: reaccion den moléculas de Hidrogeno y Iodo*****	La forma en la que se relacionan estas moléculas no es realizable por las actuales reglas de transición previstas en homos 1.0.
<i>Velocidad de simulación e "inteligencia" de los objetos</i> El Laberinto: En un espacio de 100x100 celdas se establece un laberinto hecho con ladrillos que será resuelto por un ratón..	El modelo propuesto es completamente realizable y funcional, excepto que su rendimiento durante la simulación no es el esperado, Establecer objetos como activos o inertes para la simulación mejoraría enormemente su desempeño. Cabe anotar que la inteligencia en las decisiones de este raton no será mejor que la simple elección al azar de sus movimientos, los objetos no cuentan con memoria o comportamientos programables.

Tabla A.36: Algunas limitaciones presentes en Homos 1.0

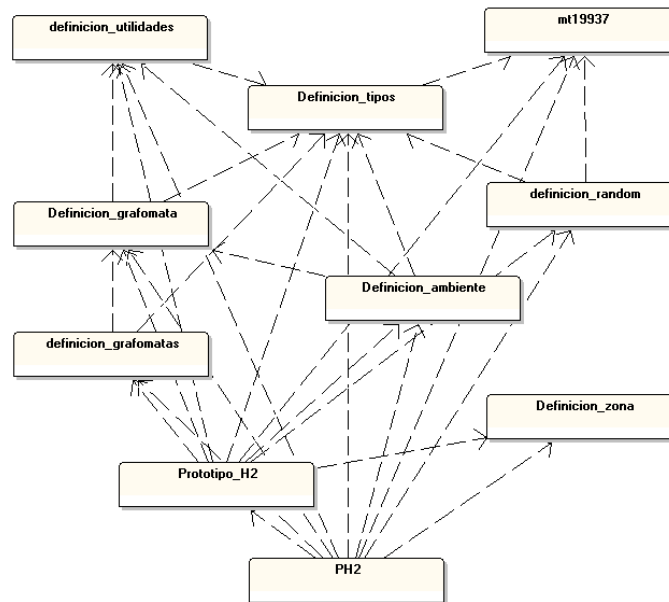
Descripción	Implementación en la herramienta
<i>Superposición de objetos en el espacio</i> La vaca sobre el pasto: Se busca representar los patrones de crecimiento del pasto en un terreno determinado completamente verde. Esto incluye a una vaca de los alrededores que gusta de pastar libremente en ese campo...	En Homos 1.0 se asume que todos los objetos están en un mismo plano y que cada celda del ambiente puede contener solo un objeto a la vez. La vaca y el pasto no están simultáneamente en una misma celda del ambiente. Con la herramienta actual, la vaca puede dar un paseo solo si está alimentándose del pasto más próximo, abriéndose paso por el terreno.
<i>Contenencia de objetos por un medio</i> El pez en el agua: Un pez oceánico decide arriesgadamente nadar por aguas costeras cerca de una isla remota. El agua y la tierra hacen contacto en la playa, zona límite hasta donde el pez puede llegar...	Dependiendo de las habilidades del modelador, podría suceder durante la simulación que el pez <i>nade</i> fuera del agua. En Homos no existe el concepto de contenencia: solo es posible para el pez nadar dentro del agua. Es factible representar algo como esto mediante la creación de objetos adicionales en el ambiente que limiten el movimiento y les sirvan de bordes, sacrificando simultáneamente memoria y velocidad de ejecución en el proceso.
<i>Agrupación de clases objeto</i> El objeto mariposa: las mariposas son seres vivos que pasan por 4 etapas durante su desarrollo: huevo, oruga, crisálida y mariposa.	Para efectos de modelado, se determinan las diferentes etapas del objeto con sus reglas básicas y se implementan una a una. No existe en homos una clase (o super clase) en la cual se agrupen las diversas etapas de un mismo objeto, ni el soporte para la asociación de clases del tipo A es un B, A pertenece a B, entre otras funciones.
<i>Manejo de propiedades particulares de la clase objeto</i> Niveles de bienestar: En la empresa X, se contemplan 3 niveles de bienestar por cada empleado que representan respectivamente: su estado económico, afectivo y emocional. En cada uno, el empleado puede estar bajo, medio y alto.	No existen otras propiedades asignables en homos mas que las provistas para cada regla. No es posible crear nuevos parámetros a las reglas existentes ni crear nuevas reglas.
<i>Manejo de propiedades continuas en la clase objeto</i> Dinero del banco: En promedio, el banco gana 5 centavos Por cada transacción exitosamente realizada...	Aunque homos está fundado en operaciones discretas, la ausencia de elementos de tipo nivel (acumuladores) y el manejo de variables numericas para el control y registro de operaciones limita la aplicación de homos a un conjunto restringido de representaciones, generalmente de tipo biológico.
<i>Manejo de influencias múltiples sobre un objeto</i> Presión de grupo: Cuando existan mas de 3 celulas muertas alrededor, la célula en evaluación muere.	Aunque en homos una celda pueda relacionarse hasta con 8 vecinos, en homos las relaciones entre objetos son uno a uno, el ejemplo propuesto no es representable.
<i>Inclusión de reglas fundadas en otras disciplinas</i> Repulsión y atracción eléctrica: Se tienen en un recipiente una serie de moléculas eléctricamente cargadas. Se desea ver el efecto de la atracción y repulsión eléctrica.	Las reglas de transición biológicas usadas en Homos pueden representar difusamente fenómenos de este tipo.
<i>Interacción de partículas sin contacto</i> Transmisión inalámbrica: Una torre de transmisión envía datos a todas las antenas de la región...	Igual que el caso anterior, no existen algoritmos implementados que emulen o implementen este tipo de funcionalidades.
<i>Manejo de herencias</i> Clase ave, gallina y canario: Se define una clase principal ave y de ella descienden las clases gallina y canario...	Existen implementadas la idea de clases objeto e instancias, pero no hay mecanismos para la realización efectiva de operaciones de herencia en la herramienta.
<i>Química y Conservación de la materia</i> Moléculas: reaccion den moleculas de Hidrogeno y Iodo*****	La forma en la que se relacionan estas moléculas no es realizable por las actuales reglas de transición previstas en homos 1.0.
<i>Velocidad de simulación e "inteligencia" de los objetos</i> El Laberinto: En un espacio de 100x100 celdas se establece un laberinto hecho con ladrillos que será resuelto por un ratón..	El modelo propuesto es completamente realizable y funcional, excepto que su rendimiento durante la simulación no es el esperado, Establecer objetos como activos o inertes para la simulación mejoraría enormemente su desempeño. Cabe anotar que la inteligencia en las decisiones de este raton no será mejor que la simple elección al azar de sus movimientos, los objetos no cuentan con memoria o comportamientos programables.

Tabla A.38: Integración entre elementos componentes de la MBOR

Requerimientos para la aplicación efectiva de la MBOR					
Etapa	¿Etapa integrada en la herramienta?	¿Descripción de proceso documentado?	Formalismo teórico	Formalismo operativo	Experiencia requerida
Desarrollo del modelo en lenguaje prosa	No	Si, mediante un ejemplo	Si, en la tecnología orientada a objetos	■ Usa los datos obtenidos de la adquisición de información como fuente de datos. ■ Identifica los elementos característicos del fenómeno según la orientación a objetos.	■ Comprensión básica del fenómeno. ■ Redacción, análisis y síntesis de textos. ■ Análisis sintáctico del idioma. ■ Conocimiento sobre tecnología orientada a objetos.
Conversión al lenguaje de objetos y reglas	No	Si, mediante un ejemplo.	No	Es un proceso empírico. Supone la comparación del comportamiento de las acciones reales con las definidas por las reglas de transición ofrecidas por la herramienta.	Experiencia de modelado
Implementación del modelo	Si	Si, seguir el manual de usuario	Si, ver tesis de Homos		Ninguna
Simulación	Si	Si, seguir el manual de usuario, manejo intuitivo.	No, es un proceso iterativo intuitivo	■ Determinar en qué punto el modelo está listo para ejecuciones de prueba, es independiente de las pruebas de concepto preliminares. ■ Establecer los escenarios de prueba ■ Manejar el simulador: iniciar, detener, terminar. ■ Realizar comparaciones de los resultados obtenidos con los esperados, determinar si hay necesidad de ajustes.	Ninguna
Ajuste de sensibilidad y parámetros	Si	Si, depende del objetivo e implementación del modelo.	No por la MBOR, es particular a la naturaleza del fenómeno a modelar		Ninguna

A.7. Análisis al diagrama BOR

Figura A.21: Componentes del diagrama BOR



Las flechas indican las dependencias inmediatas entre módulos.

Fuente: Autor.

En este apartado de análisis se cubren los aspectos relacionados con la composición, la forma y la funcionalidad teórica de diagrama BOR. La justificación para su inclusión corresponde con la importancia que tienen las expresiones gráficas para representar información y facilitar la comprensión de las relaciones presentes, hacer visible lo que compone una regla y cómo está construido el modelo de transiciones implementado en la herramienta software.

En general, un diagrama es una representación simbólica de información en forma bidimensional que corresponde con alguna técnica de visualización específica. El propósito del diagrama BOR o de proyecto es hacer visible la definición del modelo mediante la presentación gráfica de las clases objetos y las reglas definidas en el mismo. La técnica utilizada para su construcción corresponde con las pautas establecidas para ello en la MBOR; estas convenciones están influenciadas a su vez en ideas obtenidas de la notación gráfica establecida en la teoría orientada a objetos de Coad - Yourdon descrita en [CY90b, CY90a].

A.7.1. Elementos gráficos de la notación

Los elementos de notación del diagrama BOR se encuentran descritos en la tesis de Homos 1.0, en la ayuda de Homos y en el manual de usuario. Se muestra la representación gráfica usada en las

distintas versiones disponibles en la tabla A.39 en la página 277. La diferencia entre versiones consiste en cambios gráficos en el uso de tipos de líneas y mejoras en las descripciones, los colores y los identificadores de clase se mantienen.

Según las expresiones encontradas en la nomenclatura de la tesis de Homos 1.0, los elementos que componen dicha representación gráfica son:

- Líneas simples y dobles para representar relaciones entre las clases que componen la descripción gráfica.
- Cuadrados de esquinas redondeadas como identificadores de clase.
- Cuadrados o rectángulos como identificadores de instancias de clase.
- Uso de los datos de nombre, probabilidad y prioridad para caracterizar las reglas de transición.
- Se relacionan hasta tres clases distintas, según el tipo de regla a representar.

En la notación actual se establecen tres tipos de relaciones básicas: A afecta B, A coexiste con B, A se transforma en B. En la representación gráfica, Las puntas de flecha de las líneas indican el sentido de aplicación y acción de la regla. El tipo de línea, simple o doble, se usa para diferenciar gráficamente la acción de la regla; el significado en el diagrama se deduce de su aplicación final.

Una *primitiva de diagrama* es un elemento estructural básico cuya descripción gráfica es independiente de otras presentes en la notación. No tiene una explicación compositiva en términos de las demás primitivas presentes, está construida directamente mediante elementos gráficos fundamentales⁸⁶. Su definición es arbitraria, depende de la armonía entre los elementos usados y los propósitos descriptivos del diagrama. En el diagrama se hace uso de tres primitivas gráficas: la estabilidad (como una línea sin puntas de flecha que relaciona un par de clases), la transformación (una línea apuntando en el sentido de cambio entre clases) y la eliminación (línea doble que indica la clase a eliminar).

En la notación gráfica del diagrama BOR se presentan incongruencias semánticas en lo que corresponde al uso de las primitivas de diagrama disponibles. Haciendo uso de estas primitivas y suponiendo tres clases distintas A, B y C, se describen en la tabla A.39 en la página 277 los elementos que apoyan esta afirmación.

Con respecto a la cromatismo de las reglas, los colores sirven para diferenciar el tipo de relación o transición entre clases. En la metodología no se hacen explícitas las razones por las cuales se estableció cierto color para representar a un tipo de transición específico. Se presume que estos obedecen a una selección arbitraria / particular realizada por los desarrolladores del proyecto. Se observa una tendencia de uso de colores oscuros para representar reglas que realizan operaciones de eliminación.

A.7.2. Observaciones generales

La implementación actual del diagrama BOR va construyéndose automáticamente a medida que avanza el proceso de creación del modelo en la herramienta, permitiendo la reubicación manual de los elementos gráficos en el espacio de diagrama.

⁸⁶Puntos, líneas, curvas, figuras geométricas y demás elementos primarios de construcción de gráficos

En esta versión del software no es posible saber cuántas reglas han sido descartadas durante el desarrollo del del modelo, su orden de aparición y en general otro tipo de estadísticas que permitan determinar la evolución del proceso del modelado. Esto permitiría una mejor comprensión de los procesos de creación de modelos en la herramienta, también haría posible detectar debilidades y fortalezas de las distintas funcionalidades presentes en la herramienta para el propósito de la creación y edición de modelos BOR.

A.7.2.1. Deficiencias en la implementación gráfica del diagrama BOR

- Aunque el nombre, el valor de probabilidad y prioridad de la regla son elementos caracterizadores de las reglas en la notación, no se encuentran implementados en la versión final del diagrama BOR disponible en la herramienta. Dentro del código fuente existente relacionado con la aplicación se encuentra parte de un ejemplo funcional de esta forma de representación que lamentablemente no se integró en el algoritmo final de dibujo de los diagramas BOR.
- Cuando una clase se relaciona consigo misma (hay una autoreferencia de clase), el algoritmo de dibujo muestra una punta de flecha apuntando hacia esa clase e indicando la existencia de dicha relación. Cuando una clase tiene más de una de estas formas de relación, no es posible interpretar correctamente el diagrama BOR: se hace necesario recurrir al árbol de reglas para determinar el tipo de regla aplicado. Esto también sucede cuando una clase es referenciada por varias reglas; el sistema de identificación por colores falla debido a la tendencia del algoritmo de dibujo a la superposición de las puntas de flecha y las líneas que representan las relaciones. La notación de autoreferencias como tal, no está contemplada dentro de las definiciones de nomenclatura propuestas por la MBOR.
- La figura A.23 presenta un ejemplo de falsa simplicidad del diagrama. Esta es causada por la superposición de conectores y los supuestos internos con los cuales se implementó el algoritmo de dibujo del diagrama.
- Las puntas de flecha de las líneas conectoras usadas en el diagrama son siempre azules. Estos elementos gráficos serían de mayor utilidad si fueran del color establecido para la regla transición a la que pertenecen o por si por ejemplo indicaran el color de la clase objeto a la que apuntan.
- El actual espacio de trabajo del diagrama BOR está pensado para representar gráficamente modelos pequeños. Teóricamente es posible albergar gran cantidad de clases-objeto y una cantidad indeterminada de relaciones entre ellos, pero al presentarlos gráficamente el orden y las relaciones entre las clases- objeto que el diagrama pretende representar se ofusca fácilmente.
- Hay limitaciones prácticas en la edición del diagrama BOR implementado en la herramienta. Pej: No es posible en esta versión del software diseñar el modelo directamente desde la ventana de presentación del diagrama BOR.

A.7.2.2. Inconsistencias y ausencias en notación del diagrama BOR

- La representación gráfica de las reglas eliminación y transformación es la misma en las notaciones #1 y #2. Ver la tabla A.39 en la página 277.
- No se hace uso estándar de la línea doble en la representación gráfica: Si este tipo de línea indica la eliminación de la clase a la que apunta tal y como sugiere la notación #1 para las reglas de eliminación y neutralización, su uso sería incorrecto para la representación gráfica de la regla alteración. Para esta regla, se hace ver que las clases B y C son eliminadas, cosa que no concuerda con su planteamiento teórico.
- Es notorio que las reglas Decadencia, Movimiento, Expansión y Retracción no tengan representación gráfica en la nomenclatura establecida en la MBOR y consecuentemente en el diagrama BOR. La regla Movimiento tiene características particulares que la hacen difícil de representar si solo se dispone de los elementos de notación gráfica previstos por la MBOR. En la figura A.24 se muestra un ejemplo de diagrama incompleto: hay 22 reglas de transición en el modelo presentado, pero el diagrama solo revela 14 de las reglas existentes. Se hace notar que la mayoría son de Alteración (12 relaciones en rojo) y las demás son de Transformación (2 en azul). No se muestran las relaciones de Expansión (4 reglas) y Movimiento (4 reglas) que también hacen parte del modelo.
- La representación final de una clase objeto en la herramienta es óptima y la mínima necesaria: incluye el nombre, el color y la imagen que representa a la clase en el modelo. La integración gráfica propuesta en la nomenclatura para este elemento es útil para determinar de un vistazo las reglas que pertenecen o están asociadas a determinada clase, pero no resulta conveniente en la construcción gráfica del diagrama. En la interfaz gráfica de Homos se ofrecen cuadros de diálogo particulares para cada tipo de elemento del modelo.
- No se incluye dentro de la notación la idea de destino final de un elemento al ser eliminado, como por ejemplo el usado en el diagrama de Forrester mediante la nube o pozo infinito.
- En general, el diagrama BOR no es suficiente para la interpretación de un modelo. Hay necesidad de recurrir al árbol de proyecto para comprender la estructura del modelo implementado en la herramienta.

A.7.3. Resultados del análisis

Como resultado de este análisis y de los distintos hallazgos realizados, se concluye que:

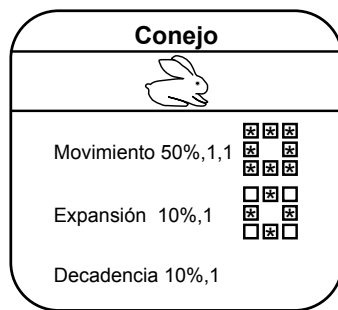
- Los diagramas BOR son difíciles de leer: se requieren mejoras tanto en la notación prevista en la metodología como en el algoritmo de dibujo implementado en la herramienta software para hacer uso práctico de la representación gráfica.
- La representación del diagrama BOR es incompleta e insuficiente, no es posible expresar un modelo de transiciones en su totalidad usando solo la notación gráfica prevista por la metodología BOR.

- La nomenclatura ofrecida no facilita la adición o extensión de nuevas expresiones gráficas de relaciones de transición basadas en las formas ya establecidas.
- Se requiere encontrar una forma de notación o expresión gráfica que permita representar todas las reglas de transición disponibles y si es posible, que esta misma notación permita la definición de nuevas formas de transición en formato escrito, gráfico o ambos. La ausencia de descripciones escritas de los modelos impide en la práctica el desarrollo de nuevos tipos de análisis.

En la version actual del software se sugieren cambios en la nomenclatura en lo que corresponde a la representacion grafica de transiciones con las reglas disponibles y el remplazo total de la forma actual de representacion de diagramas de transicion. Pensando a largo plazo, se sugiere la refundacion completa de la nomenclatura. Al hacerlo se espera una nueva forma de diagramas BOR.

Figura A.22: Notación gráfica de una clase objeto

(a) Representación ideal en nomenclatura



(b) Representación en la herramienta software



Tabla A.39: Semántica gráfica de reglas en la forma final de representación en el diagrama BOR

Regla	Notación gráfica #1	Notación gráfica #2	Denota gráficamente	Análisis gráfico	Color de línea
Eliminación			A elimina B. Relación <i>fuerte</i> de A sobre B	Es una primitiva de diagrama. Por su definición se establece la acción que realiza la regla, pero gráficamente no se nota la eliminación de B.	Morado
Transformación			A se transforma en B. Relación <i>débil</i> entre A y B. Énfasis en el sentido del cambio.	Es una primitiva de diagrama. Coincidencia entre notaciones 1 y 2. Sugiere la transición de un estado A a un B.	Azul
Alteración			A hace que B se convierta en C.	Cada relación está expresada en términos de eliminación según la notación gráfica 1. La notación 2 no aporta más datos. La representación usada es confusa, sugiere que B y C se eliminan o que C se crea a partir de la relación de eliminación entre A y B.	Rojo
Neutralización			Eliminación mutua de A y B. Doble relación, un evento simultáneo	Usa bidireccionalmente la notación de eliminación. Hace énfasis en la idea de que la línea doble representa eliminación,	Negro
Reproducción			Coexistencia fija de A y B en la vecindad, C se crea como resultado de la interacción.	Mezcla un elemento de estabilidad entre clases con una transformación que nace de la estabilidad. Está descrita como una transformación, lo que se transforma es la <i>relación</i> entre A y B en la clase C.	Verde

Notación gráfica #1: Recreada de la establecida en *Nomenclatura*, en la Sección 1.3.5 de la tesis de Homos 1.0, Pág 23-24.

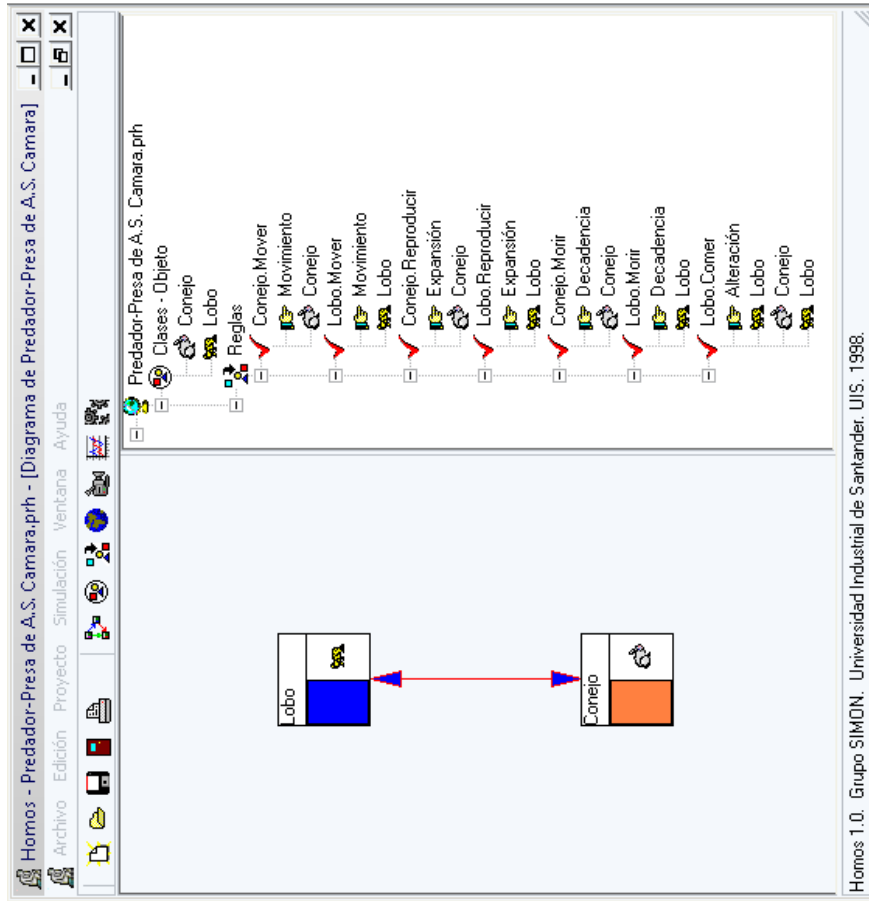
Notación gráfica #2: Recreada de la presentada en *Diagramador: Representación de las reglas en el diagrama* en el Capítulo 10, Pág 62 del manual de usuario de Homos 1.0

Las letras A,B,C representan clases, reemplazan el texto presente en cada elemento.
Obsérvese la ausencia de notación gráfica para las reglas Decadencia, Expansión, Retracción y Movimiento. La notación gráfica #2 es la que está actualmente implementada en Homos 1.0.

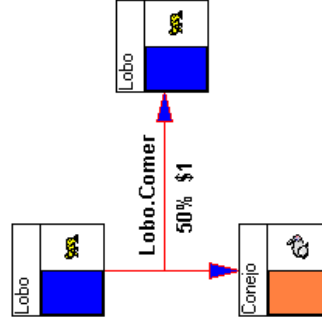
En este proyecto se establece que una relación es fuerte si implica un cambio existencial de la clase. Una relación entre clases es débil si representa un cambio de tipo de clase.

Fuente: autor.

Figura A.23: Ejemplo 1 de diagrama BOR



Correcta representación de la regla de alteración según la nomenclatura



Ejemplo de problemas de visualización en el algoritmo de dibujo del diagrama BOR. Las reglas de tipo Decadencia, Expansión y Movimiento están excluidas por defecto del diagrama. El algoritmo de dibujo existente hace una representación simplificada de la regla alteración.
Fuente: Autor.

[illegible]

279

Apéndice B

Elementos de evaluación del Software Homos 1.0

Introducción

Al hablar de evaluar, lo que se realiza es un "*Acto de juzgar o inferir juicios atribuir o negar cualidades, medir la eficacia de un método o los resultados de una actividad; por tanto, se puede decir que la evaluación es un juicio de valor, un proceso sistemático, una medida de algo*"[Cer00]. Aunque contrario al sentido común, en este proyecto no se hará énfasis en el elemento de medida propio de cada una de las distintas evaluaciones propuestas. No estamos interesados en fijar un número para calificar al software. Lo que se hará con las evaluaciones es usarlas como un patrón o guía acerca de lo que otros encuentran valioso (valorable) en un software. Al aplicarlo a la herramienta software, se determinan las distintas falencias, ausencias, debilidades, así como las fortalezas a las que hubiera lugar según el sentido particular de cada evaluación.

En este proyecto se plantean tres formas de evaluación. Se realiza una evaluación interna con dos formatos de un mismo origen, uno contemporáneo al lanzamiento de la herramienta y otro algo más reciente. La idea es observar las diferencias producidas por el cambio de paradigma de evaluación.

Luego se plantea una evaluación con criterio externo, en este caso se hace uso de los criterios de evaluación de herramientas de simulación propuestos por [NPH98]. La idea de aplicar esta evaluación se centra en ubicar al software con respecto a otros del mismo tipo mediante un estándar internacional.

Para finalizar se realiza una comparación entre varias aplicaciones de propósito similar al de la herramienta software. Esto es para determinar lo que otros han hecho, y de allí destacar lo que pueda servir de aporte a la herramienta software.

Los elementos encontrados en las evaluaciones servirán al propósito de generación de ideas necesario para dar soporte a las innovaciones a realizar en la herramienta software.

B.1. Evaluación mediante criterio interno

El grupo Simon evalúa internamente las aplicaciones producidas mediante un formato de evaluación de software educativo. Los programas producidos por el grupo pretenden ser parte activa de las TIC, específicamente software dirigido a las aulas y de uso en la educación. Se implementan dos versiones de la evaluación de software educativo de Pere Marquès [Gra98, Gra01].

B.1.1. Primer formato interno de evaluación de software

La primera evaluación corresponde con lo establecido en [Gra98]. Se lista esta información por su valor histórico e informativo. La evaluación destaca las bondades del software en el momento de su lanzamiento. Se presenta el cuerpo del formato de evaluación en las tablas B.7 en la página 287 a B.14 en la página 290.

el formato se describe brevemente así:

consta de una descripción del software, evalúa la funcionalidad del software mediante elementos de eficacia, facilidad de uso y acondicionabilidad. Tiene en cuenta elementos técnicos y estéticos como la calidad de elementos visuales y contenidos presentados, la interacción con el usuario y la aplicación de la tecnología disponible como medio de representación. Luego, destaca los elementos pedagógicos de motivación, adecuación a los usuarios, potencialidad de los recursos disponibles, el fomento a la iniciativa, la actualidad de enfoque pedagógico y la documentación ofrecida. Al final se establecen indicadores de el esfuerzo cognitivo requerido para hacer uso de software y se ofrece un espacio para las observaciones sobre ventajas, problemas, lo destacable y la opinión del evaluador.

B.1.1.1. Resultado de la evaluación

La valoración global del software es correcta o aceptable para su uso. Pese a algunos elementos negativos presentes, es una aplicación viable para mejoras y desarrollos posteriores. Como ya se sabe estas mejoras no se realizaron (aun), la siguiente evaluación revela lo que causó la demora en la aplicación de cambios.

Hay poca versatilidad aunque la herramienta permite el logro de la observación de la dinámica de fenómenos. Hace correcto uso de sus capacidades técnicas y estéticas (para ese momento, 1998) y tiene debilidades en cuanto a la potencialidad de recursos y de documentación (esto con respecto a lo que puede ser adjuntado a los modelos). Al hacer uso de la herramienta software se producen evidentes manifestaciones de iniciativa y el autoaprendizaje en la construcción de modelos.

Tabla B.1: Primera evaluación interna 1998, Descripción del software

Formato de catalogación y evaluación multimedia ©PERE MARQUÈS GRAELLS 1998

Nombre del programa:

Homos.

Versión:

1.0.0.23 (1998) en español.

Autores:

Carmen Elena Duarte Mogotocoro y Oscar Armando Lozano Avellaneda.

Fabricante:

Grupo SIMON de investigaciones [GSI] – Universidad Industrial de Santander [UIS].

Licencia:

Gratuito y de libre distribución para uso Académico. Es propiedad intelectual de la UIS, del GSI y sus autores.

Temática:

Herramienta de Simulación de propósito general, con fortalezas en la representación de sistemas dinámicos poblacionales.

Área:

Ciencias básicas

Materia:

Ciencias naturales

Objetivo General:

Facilitar los procesos de construcción del pensamiento en el área de ciencias, sus cimientos están soportados en el pensamiento sistémico, la pedagogía constructivista y con inspiración en la dinámica de sistemas, los autómatas celulares y la teoría de objetos.

Contenidos que se tratan – *Hechos, procedimientos, actitudes...*

El tema educativo es dependiente del contenido científico del modelo plasmado en la herramienta. Permite la creación y edición de modelos, además de permitir la experimentación en ellos mediante la variación de sus parámetros particulares. Presenta curvas de comportamiento y evolución en vivo del sistema implementado.

Destinatarios – *Características, etapa educativa, edad, conocimientos necesarios...*

Educación básica, media y vocacional, profesionales, modeladores en dinámica de sistemas, con o sin experiencia en modelado con objetos y reglas. Conocimientos previos mínimos, con una corta curva de aprendizaje. Dirigido por igual a principiantes y expertos en el uso de computadores y sistemas Windows.

Breve Descripción:

Homos es un simulador y modelador de propósito general que apoya los procesos de construcción del pensamiento usando como fundamento la metodología de objetos y reglas y la dinámica de sistemas. Facilita experimentación y la vivencia práctica del conocimiento científico plasmado en los modelos creados con la herramienta.

Requisitos técnicos – *Hardware y software mínimo, periféricos necesarios...*

CPU Pentium de 100 MHz, 16 MB de Memoria RAM, Espacio libre en Disco Duro de 10 MB, Monitor SVGA 800x600, Windows 95 o superior, el Ratón es imprescindible, la impresora es opcional.

Tabla B.2: Primera evaluación interna, Aspectos funcionales del software

Aspectos funcionales				
Escala de valoración - <i>Baja, Correcta, Alta, Excelente</i>	B	C	A	E
Eficacia - <i>Puede facilitar el logro de los objetivos que pretende</i>			✓	
Facilidad de uso e instalación - <i>Entorno amable</i>		✓		
Versatilidad - <i>Ajustable, Modificable, Niveles de dificultad, Evaluación e informes</i>	✓			

Fuente: PERE MARQUÈS GRAELLS 1998.

Tabla B.3: Primera evaluación interna, Aspectos técnicos y estéticos

Aspectos técnicos y estéticos				
Escala de valoración - <i>Baja, Correcta, Alta, Excelente</i>	B	C	A	E
Calidad del entorno audiovisual - <i>Pantallas...</i>		✓		
Calidad en los contenidos - <i>Texto, audiovisual...</i>		✓		
Navegación e interacción		✓		
Originalidad y uso de tecnología avanzada				✓

Fuente: PERE MARQUÈS GRAELLS 1998.

Tabla B.4: Primera evaluación interna, Aspectos pedagógicos

Aspectos pedagógicos				
Escala de valoración - <i>Baja, Correcta, Alta, Excelente</i>	B	C	A	E
Capacidad de motivación			✓	
Adecuación a los usuarios - <i>Contenidos, Actividades, Entorno...</i>		✓		
Potencialidad de los recursos didácticos - <i>Actividades, Organizadores, preguntas, tutoriales...</i>	✓			
Fomento de iniciativa y auto aprendizaje				✓
Enfoque pedagógico actual			✓	
Documentación - (Si tiene...)	✓			

Fuente: PERE MARQUÈS GRAELLS 1998.

Tabla B.5: Primera evaluación interna, Esfuerzo cognitivo y observaciones
Esfuerzo cognitivo que exigen sus actividades - Selección múltiple

Control Psicomotriz	✓	Observación / Valoración	✓
Memorización / Evocación		Razonamiento - <i>Deductivo, Inductivo, Crítico</i>	✓
Comprensión / Interpretación	✓	Pensamiento divergente / Imaginación	✓
Comparación / Relación	✓	Resolución de Problemas	
Análisis / Síntesis	✓	Expresión - <i>Verbal, Escrita, Gráfica / Crear</i>	✓
Cálculo		Reflexión metacognitiva	✓
Exploración / Experimentación	✓		

Fuente: PERE MARQUÈS GRAELLS 1998.

Tabla B.6: Primera evaluación interna, Observaciones encontradas
Observaciones

Ventajas que reporta frente a otros medios:	Permite el modelado basado en objetos y reglas, facilita la experimentación y la construcción de conocimiento con esa metodología
Problemas e inconvenientes:	Existen fallas en el software por corregir. Son necesarias algunas mejoras en sus facilidades que permitan un mejor uso de sus capacidades.
Aspectos a destacar:	Modelado simple, intuitivo, cercano al lenguaje natural. Presenta la dinámica del fenómeno gráficamente mediante paisajes dinámicos y curvas de comportamiento.
Impresión personal:	¿Me ha gustado? Si [✓] No [] ¿Lo recomendaría? Si [✓] No []

Fuente: PERE MARQUÈS GRAELLS 1998.

B.1.2. Segundo formato interno de evaluación de software

La segunda evaluación corresponde con lo establecido en [Gra01]. Se lista esta información por su valor histórico e informativo. la evaluación destaca una pérdida de las bondades que presentaba el software en el momento de su lanzamiento. Se presenta el cuerpo del formato de evaluación en las tablas B.7 en la página 287 a B.14 en la página 290. Con respecto al resultado de evaluación anterior, Existen diferencias en cuanto a como se percibe el software. La revolución técnica y las mejoras multimedia hacen sentir su ausencia en estos resultados.

El formato consta de los siguientes elementos: Una descripción de software en sus elementos mas externos, seguida de establecer funcionalidad en terminos de la utilidad del software. Se mide lo pedagogico segun los utiles disponibles en la herramienta para facilitar el proceso de aprendizaje y se analiza el elemento tecnico y estetico con lineamientos propios de la interaccion humano computadora o HCI. Se enfatiza el uso que hace la herramienta de los recursos didacticos disponibles segun las necesidades de expresiones graficas y se establece una medida del esfuerzo cognitivo necesario para hacer uso de la aplicacion. Al final se hace una valoracion global y se da un espacio para establecer las observaciones a las que haya lugar, en semejanza con el formato anteriormente presentado.

B.1.2.1. Resultado de la evaluación

La herramienta apoya e libre descubrimiento mediante asociacion de ideas relacionadas con diferencias entre estabilidad y cambio entre objetos. Las especificaciones de hardware requeridas es mucho mayor que lo planteado en el trabajo de grado de Homos 1.0, sobretodo si se ejecutan modelos por largos periodos de tiempo.

La herramienta es eficaz en cuanto a sus capacidades de modelado y de atraer la atencion mediante lo dinamico de la ejecucion de los modelos, da credito a sus autores y no tiene ningun tipo de publicidad.

La herramienta no cuenta con elementos relacionados a la integracion mediante medios de comunicacion distintos del correo electronico a los autores y las ayudas establecidas en el software. No se dispone de guias o elementos de redireccion o enfoque de contenidos y no se hace enfasis en el trabajo cooperativo. La aplicacion hace uso del modelo prosa como introduccion, se presentan en forma ordenada las transiciones mediante el arbol de proyecto.

Con las descripciones planteadas, se nota una baja en la valoracion global de la aplicacion, A partir del 2002 en adelante, la aplicacion pierde parte de su atractivo multimedia.

Hay que recordar que estas evaluaciones son para software educativo propiamente dicho. ¿Porque aplicárselas a Homos? El software tiene una componente visual dinámica y puede hacerse uso del programa para comunicar ideas, esa es su componente educativa. Al no contar con muchos de los elementos propios de los software educativos como un material fijo o tema principal definido, elementos graficos adecuados al material a presentar y no hacer uso de presentacion ordenada / jeraquizada de informacion, termina siendo evaluado con claras connontaciones negativas. Estos elementos pueden ser establecidos con una capa externa que le permita al software adicionar

informacion a los modelos. En el software esto se implementa mediante con Micromundos, es algo que ya existe y en dos versiones distintas. Lo que queda de estas evaluaciones es: integrar facilidades para la creacion de materiales educativos directo en la herramienta software. Hacer esto haria posible el deseo de ver a la herramienta como verdadero software educativo.

En conclusion, el software deja mucho que desear. Se encuentra tecnologicamente atrasado en varios aspectos, no se han realizado las mejoras multimedia y otras requeridas para hacer de la experiencia de uso con la herramienta algo mas ameno. La valoracion del esfuerzo cognitivo y las observaciones realizadas son las mismas en los dos formatos de evaluacion.

Tabla B.7: Segunda evaluación interna: Identificación del entorno, descripción del software

Formato de catalogación y evaluación multimedia©PERE MARQUÈS GRAELS -UAB/2001

Título del programa: Homos 1.0, Versión 1.0.0.23 (1998) en español e inglés.

Autores: Carmen Elena Duarte Mogotocoro y Oscar Armando Lozano Avellaneda.

Email de contacto: oscar.lozano@copex.com.co

Colección/Editorial: Grupo SIMON de investigaciones [GSI] – Universidad Industrial de Santander [UIS].

URL: http://www.uis.edu.co/portal/investigacion/grupos/simon_uis/simon.html

Email de contacto: simon@uis.edu.co **Teléfono:** +57 7 6344000 Ext 2681

Licencia: Gratuito y de libre distribución para uso Académico. Es propiedad intelectual de la UIS, del GSI y sus autores.

¿Material de libre acceso?: Si, descarga desde el sitio web.

¿Incluye publicidad? No.

Temática: Ciencias básicas, ciencias naturales.

Objetivos formativos: Facilitar los procesos de construcción del pensamiento en el área de ciencias, sus cimientos están soportados en el pensamiento sistémico, la pedagogía constructivista y con inspiración en la dinámica de sistemas, los autómatas celulares y la teoría de objetos.

Contenidos que se tratan: *Hechos, conceptos, procedimientos, actitudes...* El tema educativo es dependiente del contenido científico del modelo plasmado en la herramienta. Permite la creación y edición de modelos, además de permitir la experimentación en ellos mediante la variación de sus parámetros particulares mediante el uso de una metodología propia, la MBOR. Presenta curvas de comportamiento y evolución en tiempo real o paisaje dinámico del modelo implementado.

Destinatarios: *Características, etapa educativa, edad, conocimientos necesarios...* Educación básica, media y vocacional, profesionales, modeladores en dinámica de sistemas, con o sin experiencia en modelado con objetos y reglas. Conocimientos previos mínimos, con una corta curva de aprendizaje. Dirigido por igual a principiantes y expertos en el uso de computadores y sistemas Windows.

Tipología: Preguntas y ejercicios, Unidad didáctica tutorial, Base de datos, Libro, Simulador / Aventura, Juego / Taller creativo, Herramienta para procesar datos.

Estrategia didáctica: Enseñanza dirigida, Exploración guiada, Libre descubrimiento

Función: Ejercitar Habilidades, Instruir, Informar, Motivar, Explorar, Entretener, Experimentar/Resolver Problemas, Crear/Expresarse, Evaluar, Procesar datos

Mapa de navegación y breve descripción de actividades: Homos es un simulador y modelador de propósito general que apoya los procesos de construcción del pensamiento usando como fundamento la metodología de objetos y reglas y la dinámica de sistemas. Facilita experimentación y la vivencia práctica de la construcción de conocimiento científico plasmado en los modelos creados con la herramienta.

Valores que potencia o presenta: Pensamiento sistémico, analítico y crítico, Holismo.

Documentación: Ninguna, Manual, Guía didáctica, En papel, En Cd, On-line.

Servicios On-line: Ninguno, Sólo consultas, Teleformación, Por internet.

Requisitos técnicos: Pc, Mac, Teléfono Wap, Impresora, Sonido, Cd, Dvd, Internet.

Otros: *Hardware y Software* CPU Pentium de 100 MHz, 16 MB de Memoria RAM, Espacio libre en Disco Duro de 10 MB, Monitor SVGA 800x600, Windows 95 o superior, el Ratón es imprescindible, la impresora es opcional.

Tabla B.8: Segunda evaluación interna, Aspectos funcionales del software, utilidad

Aspectos funcionales				
Escala de valoración - Baja, Correcta, Alta, Excelente	B	C	A	E
Eficacia didáctica - <i>Facilita el logro de objetivos propuestos</i>				✓
Relevancia de los aprendizajes , contenidos			✓	
Facilidad de uso - <i>Entorno amable</i>			✓	
Facilidad de instalación - <i>Programas y complementos</i>	✓			
Versatilidad didáctica - <i>Modificable, niveles, ajustes, informes...</i>	✓			
Carácter Multilingüe		✓		
Múltiples enlaces externos - <i>Si el material es Online</i>	✓			
Canales de comunicación bidireccional	✓			
Documentación , guía didáctica o de estudio	✓			
Servicios de apoyo on-line	✓			
Créditos , fecha de actualización, autores, patrocinadores...				✓
Ausencia de publicidad				✓

Fuente: PERE MARQUÈS GRAELLS 2001.

Tabla B.9: Segunda evaluación interna, Aspectos pedagógicos

Aspectos pedagógicos				
Escala de valoración - Baja, Correcta, Alta, Excelente	B	C	A	E
Especificación de los objetivos que se pretenden			✓	
Capacidad de motivación , atractivo, interés				✓
Adecuación a los destinatarios de los contenidos, actividades		✓		
Adecuación a los usuarios	✓			
Recursos para buscar y procesar datos	✓			
Potencialidad de recursos didácticos - <i>Síntesis, resumen...</i>	✓			
Carácter completo , proporciona todo lo necesario para aprender			✓	
Tutorización y evaluación preguntas, refuerzos	✓			
Enfoque aplicativo / creativo de las actividades			✓	
Fomento del autoaprendizaje , la iniciativa, toma de decisiones			✓	
Facilita el trabajo cooperativo	✓			

Fuente: PERE MARQUÈS GRAELLS 2001.

Tabla B.10: Segunda evaluación interna, Aspectos técnicos y estéticos

Aspectos técnicos y estéticos				
Escala de valoración - <i>Baja, Correcta, Alta, Excelente</i>	B	C	A	E
Calidad del entorno audiovisual - <i>Pantallas...</i>		✓		
Elementos multimedia: Calidad, cantidad	✓			
Calidad y estructuración de contenidos	✓			
Estructura y navegación <i>por las actividades</i>	✓			
Hipertextos descriptivos y actualizados	✓			
Interacción <i>diálogos, entrada de datos, análisis respuestas</i>		✓		
Ejecución fiable, velocidad de acceso adecuada		✓		
Originalidad y uso de tecnología avanzada			✓	

Fuente: PERE MARQUÈS GRAELLS 2001.

Tabla B.11: Segunda evaluación interna, Recursos didácticos

Recursos didácticos que utiliza - <i>Selección múltiple</i>			
Introducción	✓	Ejercicios de aplicación	
Organizadores previos		Ejemplos	✓
Esquemas, cuadros sinópticos	✓	Resúmenes/síntesis	
Gráficos	✓	Actividades de autoevaluación	
Imágenes	✓	Mapas conceptuales	
Preguntas			

Fuente: PERE MARQUÈS GRAELLS 2001.

Tabla B.12: Segunda evaluación interna, Esfuerzo cognitivo
Esfuerzo cognitivo que exigen sus actividades - *Selección múltiple*

Control Psicomotriz	✓	Razonamiento	✓
		<i>Deductivo, Inductivo, Crítico</i>	
Memorización / Evocación		Pensamiento divergente / Imaginación	✓
Comprensión / Interpretación	✓	Planificar / Organizar / Evaluar	
Comparación / Relación	✓	Hacer hipótesis / Resolver problemas	✓
Análisis / Síntesis	✓	Exploración / Valoración	✓
Cálculo		Expresión / Crear	✓
		<i>Verbal, Escrita, Gráfica</i>	
Buscar / Valorar la información	✓	Reflexión metacognitiva	✓

Fuente: PERE MARQUÈS GRAELLS 2001.

Tabla B.13: Segunda evaluación interna, Valoración global

Valoración global				
Escala de valoración - <i>Baja, Correcta, Alta, Excelente</i>	B	C	A	E
Calidad Técnica		✓		
Potencialidad Didáctica <i>efectiva</i>	✓			
Funcionalidad, Utilidad		✓		

Fuente: PERE MARQUÈS GRAELLS 2001.

Tabla B.14: Segunda evaluación interna, Observaciones encontradas

Observaciones	
Ventajas que reporta frente a otros medios:	Permite el modelado basado en objetos y reglas, facilita la construcción de conocimiento y la experimentación con esa metodología.
Problemas e inconvenientes:	Existen errores y fallas en el software por corregir. Es necesaria la implementación de algunas mejoras en sus facilidades para hacer mejor uso de sus capacidades, por ejemplo desarrollo de notación escrita para de modelos. Largo periodo de inactividad en cuanto al desarrollo de la aplicación. Necesidad de Actualización o renovación del software.
Aspectos a destacar:	Modelado gráfico, simple, intuitivo, cercano al lenguaje natural. Presenta la dinámica del fenómeno gráficamente mediante paisajes dinámicos y curvas de comportamiento.
Impresión personal	¿Me ha gustado? Si [✓] No [] ¿Lo recomendaría? Si [✓] No []

Fuente: PERE MARQUÈS GRAELLS 2001.

B.2. Evaluación mediante criterio externo

Mediante esta evaluación se determina como está el software con respecto a estándares internacionales. La evaluación se dice externa ya que se usan para su desarrollo criterios ajenos a los propios o particulares del grupo Simon. Para esta evaluación se usan los criterios de evaluación de software de simulación de NIKOUKARAN, PAUL y HUPLIC [NPH98]. Estos constan de tres niveles bien definidos:

Se caracteriza al software mediante la experiencia del fabricante¹, la interacción con los usuarios y los elementos técnicos del software.

En el cuerpo de evaluación el elemento más complejo es el relacionado con el software. En esta sección se evalúan las capacidades del software para manejar el modelo y configurar las entradas de datos, la ejecución de la simulación, la realización de pruebas y la eficiencia de procesos, las capacidades de animación y las salidas o reportes disponibles.

Al aplicar la evaluación se obtienen dos tipos de aportes para el análisis: se determina lo que existe en el software y como se compara con cada criterio aplicable según lo común a otras herramientas del mismo tipo y permite discriminar las posibles ausencias, aquello que no está disponible o no tiene similar en la actual versión del software.

B.2.1. Resultados obtenidos

El criterio internacional hace saber que existen muchos elementos técnicos con los que no se cuenta en este momento y que caracterizan a otros softwares más conocidos, p.ej: el soporte a usuarios y sitio web propio, control fino de la simulación, uso de varios tipos imágenes en sus representaciones gráficas, controles de prueba y registro de la simulación, extensiones programables, etc.

El programa cuenta con una documentación regular y un soporte deficiente. Implementa solo una forma de distribución estadística (distribución uniforme), está implementado solo para plataformas Windows, no cuenta con librerías o módulos reusables, no permite la fusión de modelos pero sí hace buen uso de los elementos de construcción de modelos disponibles y cuenta con interfaz programable que permite la creación de modelos-script².

A pesar que cuenta con la interfaz de automatización, en términos de la evaluación de aspectos de codificación considerados por los criterios internacionales es insuficiente. Hay elementos en contra, p.ej: No está bien visto iniciar la simulación en vacío.

No cuenta con suficientes opciones de ejecución como por ejemplo el control paso a paso de la simulación. La aplicación no integra elementos de validación o verificación de los modelos producidos. Debido a su forma interna, la aplicación no requiere de etapa de precalentamiento y se destaca positivamente el sistema de configuración de resultados impresos.

¹Su pedigree, capacidad técnica, trayectoria.

²Ver interfaz de automatización en la sección A.5 en la página 174.

Tabla B.15: Criterios de evaluación: Fabricante

Criterios para la evaluación de software de simulación NIKOUKARAN, PAUL y HUPLIC. 1998		Aporte	
		¿?	P
Fabricante		✓	B
Categoría	Historia del vendedor	✓	E
	Otros productos ofrecidos	✓	E
	Software	✓	D
	Edad	✓	R
	Cobertura	✓	R
Referencias		✓	R
Documentación	Manuales, tutoriales, ejemplos, índices, tarjetas de referencia, solución de problemas, introducción a la simulación, fundamentos estadísticos	✓	R
Soporte	Entrenamiento, mantenimiento técnico, asesorías, actualizaciones (conversión de archivos), línea telefónica gratuita, boletín informativo, reunión de grupos de usuarios, debates en internet	✓	D
Actividades previas a la compra	Disco de demostración, pruebas gratuitas, demostraciones en sitio	✓	B

Valores de puntuación establecidos por el autor: **Excelente**, **Bueno**, **Regular**, **Deficiente**, **Inexistente**
Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

Tabla B.16: Criterios de evaluación: Usuario

Criterios para la evaluación de software de simulación NIKOUKARAN, PAUL y HUPLIC. 1998		Aporte	
		¿?	P
Usuario		✓	E
Tipo de simulación	<u>D</u> iscreta, <u>C</u> ontinua, <u>A</u> mbas	✓	D
Orientación	Aplicación específica	✓	B
	Aplicación general	✓	B
Hardware	<u>P</u> c, <u>M</u> ini, <u>S</u> ervidor, <u>E</u> stación de trabajo	✓	P
	Portabilidad	✗	
Dispositivo de seguridad		✓	D
Sistema operativo	Compatibilidad	✗	
	<u>D</u> OS, <u>U</u> NIX, <u>O</u> S/2, <u>W</u> indows, <u>O</u> tro	✓	W
Versión para red		✗	
Aspecto financiero	Precio, requerimientos, mantenimiento, entrenamiento, descuentos...	✓	E

Valores de puntuación establecidos por el autor: **Excelente**, **Bueno**, **Regular**, **Deficiente**, **Inexistente**
Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

Tabla B.17: Evaluación de software - Modelo y entradas

Criterios para la evaluación de software de simulación					Aporte		
NIKOUKARAN, PAUL y HUPLIC. 1998					?	P	
Software					✓	R	
Modelo y entradas	Librerías de módulos reusables				✗		
	Construcción de modelos	Herramientas	Elementos	Código, menús, gráficos	✓	E	
			Medios	Ratón, teclado, pantalla táctil	✓	E	
	Fusión de modelos				✗		
	Asistencia	Modularidad			✓	D	
			Cuadros de diálogo		✓	E	
			Alertas - <i>Prompting</i>		✓	R	
	Modelado jerárquico				✓	B	
	Lógica formal				✗		
	Enrutamiento condicional				✗		
	Distribuciones estadísticas	Estándar, ajuste, definidas por el usuario			✓	D	
		Generadores de números aleatorios			✓	D	
	Aspectos de codificación	Programación	Herramientas		✓	R	
			Velocidad de compilación		✗		
		Enlace a otros lenguajes				✗	
		Acceso al código fuente				✗	
		Generador de programa				✗	
		Valores globales				✗	
		Funciones	Incorporadas		✗		
			Definidas por el usuario		✗		
		Atributos				✗	
		Manejo de colas - <i>Queuing policies</i>	Fifo, lifo			✗	
	Entradas	Características	Rechazo de entradas ilegales		✓	B	
			Entradas múltiples		✗		
		Modo	Interactiva, procesamiento por lotes, colección de datos automática, archivos		✓	R	

Valores de puntuación establecidos por el autor: Excelente, Bueno, Regular, Deficiente, Inexistente
Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

Tabla B.19: Evaluación de software - Ejecución

Criterios para la evaluación de software de simulación - Software NIKOUKARAN, PAUL y HUPLIC. 1998		Aporte	
		?	P
Ejecución	Control de velocidad	X	
	Ejecuciones múltiples	X	
	Ejecución de procesos por lotes automáticos	X	
	Periodo de precalentamiento	X	
	Capacidades de reinicio	✓	B
	Inicio en estado no vacío	✓	I
	Paralelismo	X	
	Modelos ejecutables	X	

Valores de puntuación establecidos por el autor: Excelente, Bueno, Regular, Deficiente, Inexistente
Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

Tabla B.20: Evaluación de software - Pruebas & eficiencia

Criterios para la evaluación de software de simulación - Software			Aporte	
NIKOUKARAN, PAUL y HUPLIC. 1998			?	P
Pruebas & eficiencia	Rastreo - <i>Tracing</i>		X	
	Capturas - <i>Snapshots</i>		X	
	Función paso a paso - <i>Step function</i>		X	
	Validación & verificación	Pruebas de validación del modelo, chequeos de completitud, preanalizadores, ayuda en línea, tutoriales en línea, chequeo de errores en línea, calidad de los mensajes de error, depurador interactivo, chequeos lógicos	X	
	Reversa temporal - <i>Backward clock</i>		X	
	Interacción		X	
	Capacidades multitarea		X	
	Puntos de interrupción		X	
	Generador de modelos conceptuales		X	
	Límites	Tamaño del modelo, número de elementos, número de iconos, etc..	✓	B
	Características en pantalla	Directorios, variables, atributos, estados, funciones, eventos	✓	R

Valores de puntuación establecidos por el autor: Excelente, Bueno, Regular, Deficiente, Inexistente
Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

Tabla B.21: Evaluación de software - Animación

Criterios para la evaluación de software de simulación - Software					Aporte	
NIKOUKARAN, PAUL y HUPLIC. 1998					?	P
Animación	Integridad	Parte del paquete			✓	E
		Como agregado			✗	
	Iconos	Librería	Usuario	Interface: CAD, bitmap, control de medios	✗	
				Editores	✓	B
		Estándares	Número & calidad		✓	E
		Características	3d, color, cambio de colores, cambio de tamaño	✓	B	
	Simulando - <i>Running</i>	Características	Control de velocidad, movimiento, encendido/apagado, zoom/pan, rotaciones, cambio de iconos	✓	B	
		Modos	Concurrentes	✓	E	
			Posteriores a la simulación	✓	B	
	Disposición de elementos en pantalla - <i>Screen layout</i>	Imprimible			✓	E
		Editores de pantalla			✗	
		Pantallas virtuales			✗	
		Pantallas múltiples			✓	B
Desarrollo	Elementos	Código, menú, gráficos		✓	E	
	Medios	Ratón, teclado, pantalla táctil, escaner, <i>traceball</i> , tableta digitalizadora		✓	E	

Valores de puntuación establecidos por el autor: **Excelente**, **Bueno**, **Regular**, **Deficiente**, **Inexistente**
Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

Tabla B.22: Evaluación de software - Salidas

Criterios para la evaluación de software de simulación - Software NIKOUKARAN, PAUL y HUPLIC. 1998			Aporte	
			?	P
Salidas	Entregas	Archivos (periodicidad), acceso a los archivos	X	
		Registros permanentes - <i>Hardcopy</i> : impresora, plotter, etc..	✓	E
		Interfaces a otros software	X	
	Reportes	Estándar	✓	E
		Personalizados	✓	E
	Base de datos	Almacenamiento	X	
		Recuperación	X	
		Manipulación	X	
	Integración	Hoja de cálculo, administrador de base de datos, paquetes estadísticos, procesadores de texto	✓	B
	Análisis	Intervalos de confianza, bondad de ajuste, series de tiempo, prueba de hipótesis, de interpretación, de autocorrelación, diseño de experimentos, de optimización, regresiones, espectral	X	
	Gráficos de negocios	Gráficos estáticos: histogramas, gráficos de barras, gráficos de líneas, etc. . .	✓	B
		Gráficos dinámicos	✓	B

Valores de puntuación establecidos por el autor: Excelente, Bueno, Regular, Deficiente, Inexistente

Fuente: NIKOUKARAN, PAUL y HUPLIC 1998.

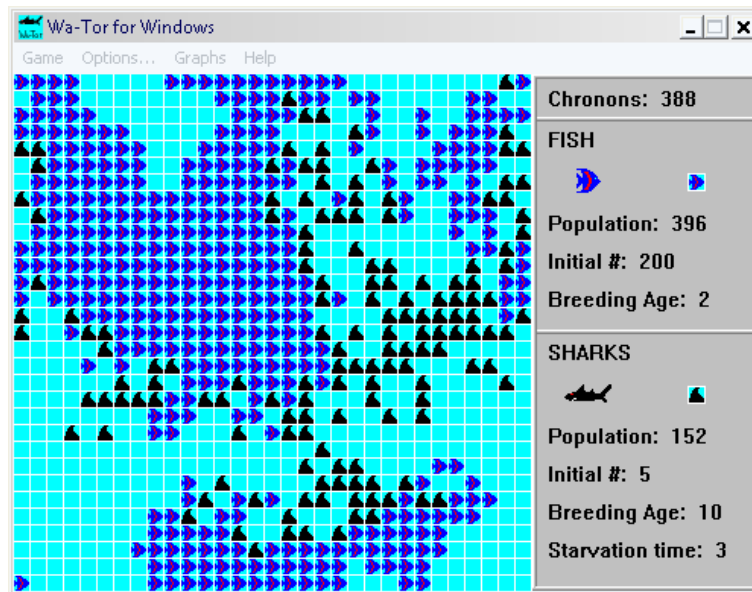
B.3. Comparación con respecto a programas similares

Este apartado se dedica a comparar las funcionalidades existentes entre Homos 1.0 y dos aplicaciones de propósito similar. La idea de la comparación es apropiar para Homos lo que sea más conveniente con respecto a la mejora de sus funciones y capacidades. Para los propósitos de comparación se usan elementos de caracterización que existan simultáneamente en los programas a comparar.

B.3.1. Wator

Es un juego de simulación ecológica en el cual se recrea sobre un toroide una dinámica poblacional entre peces y tiburones. Los patrones poblacionales obtenidos coinciden con las ecuaciones de Lotka-Volterra, mejor conocidas como el modelo depredador-presa[May73]. El programa se basa en una idea de A.K. Dewdney publicada en la columna *Computer Recreation* de la revista *Scientific American* de 1984[Dew84].

Figura B.1: Ambiente de Wator



Fuente: Captura de pantalla por parte del Autor.

Seria bueno para Homos contar con los tipos de curvas presentes en Wator, hacer uso de la presentación dinámica de los valores poblacionales y lograr un sitio web propio. No se descarta la idea de generar algún tipo de beneficio económico, pero se duda acerca de su conversión a un software tipo *Shareware*. Wator ha inspirado a otros a implementar y recrear su dinámica ecológica³.

³Algunas páginas conocidas (11/11/2011):
<http://www.fluxury.com/a5x/wator1.htm>
<http://cs.nyu.edu/~hain/wator/>
http://alife.fusebox.com/planet_wator.html
<http://www.leinweb.com/snackbar/wator/>

Homos aún no ha alcanzado tal nivel de popularidad.

Tabla B.23: Comparación de características compartidas entre Wator y Homos 1.0

Característica	Wator	Homos 1.0
Cantidad de celdas en ambiente	Fija: 31×31	Variable: hasta 255×255
Modelo usado	Modelo predador presa fijo, configurable.	Editor y simulador de modelos.
Curvas reportadas	Poblacionales, Comparacion entre poblaciones, Edad de los objetos en simulación.	Solo curvas poblacionales.
Visualización de estadísticas	Visualización dinámica de estadísticas integradas a ambiente.	Presentacion modal de valores poblacionales inmediatos.
Escenarios y configuración	Modelo y datos escenario integrados	Creacion y administración de escenarios.
Presentacion de datos de simulación	Gráficas y archivo de texto	Reporte gráfico, tablas de datos y archivo texto.
Costo	Shareware	Gratis

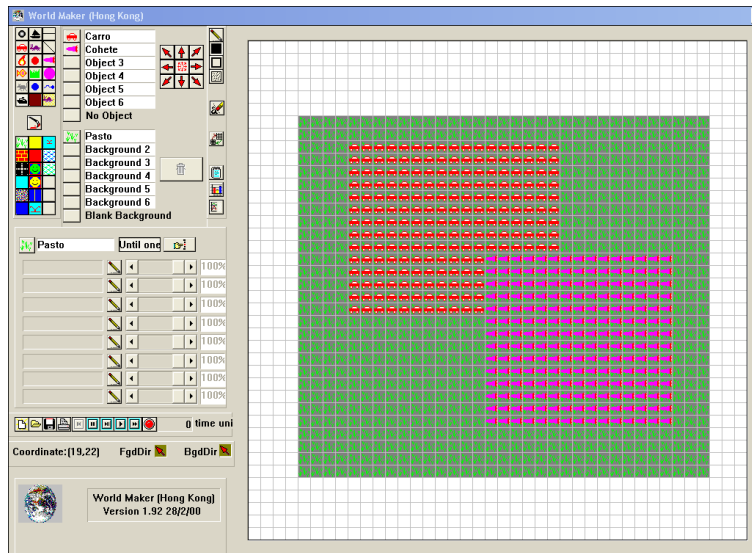
Wator también se ofrece gratis en versión demo. Ver <http://www.kovcomp.co.uk/wator/index.html>

Fuente: Autor.

B.3.2. WorldMaker

WorldMaker es un ambiente icónico de modelado con orientación a autómatas celulares desarrollado originalmente por Richard Boohan, Jon Ogborn y Simon Wrigh [Boo92] con una segunda version y una actualizacion realizada por Nancy Law[LT98b, LLLT00]⁴.

Figura B.2: Ambiente en WorldMaker



Fuente: Captura de pantalla por parte del Autor.

En WorldMaker hace uso de objetos mediante representaciones iconicas sobre un espacio llamado la malla del mundo. Los objetos interactuan entre si mediante reglas definidas por el usuario. Al igual que Homos, WorldMaker no tiene formalismo escrito y usa una representacion visual para el armado de las reglas. Los objetos estan limitados en sus interacciones al vecino más inmediato. Una descripción del uso educativo de WorldMaker se encuentra en [Dim02, pág. 107]

El elemento de Red de WorldMaker seria una adición bienvenida a Homos 1.0. Homos 1.0 supera a WorldMaker en cuanto a prestaciones de espacio de simulación. WorldMaker hace un mejor uso de recursos, permite usar muchas mas reglas⁵ que Homos y ya cuenta con una segunda version de software. Se observa en el sitio web de WorldMaker que el desarrollo del proyecto se encuentra inactivo desde el julio de 2001.

⁴WorldMaker(HK) y WorldMaker 2.0.

⁵9 reglas en Homos vs casi 200 formas distintas en WorldMaker.

Tabla B.24: Comparación de características compartidas entre WorldMaker 2.0 y Homos 1.0

Característica	WorldMaker	Homos 1.0
Versiones	WorldMaker(HK) y 2.0	1.0
Fundamento de transiciones	Autómata celular en vecindario de 9 celdas (V. de Moore)	
Representación de objetos en ambiente	Mediante iconos	
Ubicación de objetos	Los objetos pueden estar en primer plano o en el fondo	Los objetos están a un mismo nivel
Celdas en ambiente	Entre 20×20 y 100×100	Hasta 255×255
Valores de probabilidad	0 %-100 %,Valores enteros	0 %-100 %,Valores enteros y decimales
Objetos por regla	Relacion 1 a 1 entre objetos	Hasta 3 objetos por regla
Objetos	Hasta 6 objetos distintos por plano + no objeto y cualquier objeto	Todos los que sean necesarios
Representación de reglas	Mediante lenguaje gráfico	
Pasos de simulación	No configurable. Solo terminación voluntaria	Intervalo de pasos y terminación voluntaria
Coordenada inicial	(0,0) Arriba a la izquierda	
Plataforma de desarrollo	Java + Delphi 4 sobre Windows	Delphi 3 sobre Windows
Velocidad de simulación	Excelente, uso balanceado de CPU	Consumo de CPU no controlable
Interfaces adicionales	Conexión por red entre instancias y con chat	Interfaz OLE
Formulario principal	Interfaz propia, centralizada, no redimensionable	Interfaz de aplicación estandar tipo windows, menús, con MDI, redimensionable.
Escenarios	Escenario único	Múltiples escenarios solo uno activo
Resultados de simulación	Gráfica de comportamiento Descripción escrita de reglas del modelo	Diagrama BOR, Datos en archivos de texto y Gráficas de comportamiento

Fuente: Autor. WorldMaker se caracterizó a partir de manual de WorldMaker y la bibliografía disponible[CIT01].

B.4. Resultado de evaluaciones

El balance general de las distintas evaluaciones es negativo. Como enunciábamos al comienzo de esta sección el interes no está centrado en un puntaje o calificación obtenible, estamos interesados en las

diferencias que se revelan al aplicar cada sistema de evaluación.

Con respecto a los formatos internos, se afirma que se requiere mayor énfasis en lo multimedia y en la mejora de la caracterización de modelos en la herramienta para fortalecer su faceta como software educativo. El criterio internacional ubica a la herramienta software como una aplicación en etapas iniciales de desarrollo y tiene razón, Homos aun hoy es solo un prototipo. El criterio internacional ofrece luces acerca de elementos que pueden ser tenidos en cuenta para ser realizados en una siguiente versión de la herramienta software.

La comparación entre aplicaciones es provechosa, permite hacer uso y aprender de los errores cometidos en otros proyectos. Cada programa revisado aporta al conjunto de elementos con los cuales realizar innovación en la herramienta software.

B.5. ¿Cuáles son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?

A partir de las evaluaciones previamente realizadas en este apartado, se afirma que la herramienta software ha sido evaluada como software educativo y como software de simulación.

la integración de los criterios cubiertos por las evaluaciones es la siguiente:

los formatos de evaluación coinciden

en realizar una descripción del software con respecto a elementos generales

, cada una con la tendencia propia que caracteriza a la evaluación.

se evaluar aspectos funcionales, técnicos

y estéticos

pedagógicos

recursos didácticos

esfuerzo cognitivo

valoración global

observaciones

B.6. Criterios de mínimos de evaluación de Homos 1.0

A partir de las evaluaciones previamente realizadas en este apartado, se afirma que la herramienta software ha sido evaluada como software educativo y como software de simulación.

Los criterios con los cuales se evalúa la herramienta software corresponden los son:

la herramienta software como

Con el propósito de encontrar elementos suficientes que permitieran un proceso de innovación saludable de la herramienta software, se realizaron todos los elementos de análisis y evaluación que componen este proyecto de grado⁶. Esto no se había planeado así desde el inicio. Originalmente se pensaba que a partir de un conjunto reducido de elementos podría ser posible obtener novedades suficientes

Es lógico pensar que el conjunto reducido de elementos está inmerso dentro del total de acciones realizadas. cuáles son entonces estos elementos, al encontrar los elementos podemos entonces establecer evaluar con precisión lo que da valor, importancia o transcendencia al objeto de estudio.

⁶Ver el cuerpo de análisis en la sección A.

había supuesto un conjunto reducido de

En este apartado se da respuesta a la pregunta de investigación,

Diccionario de la lengua española © 2005 Espasa-Calpe:

Criterio: Norma, regla o pauta para conocer la verdad o la falsedad de una cosa.

Juicio para discernir, clasificar o relacionar una cosa

ese no es un buen criterio de clasificación.

Capacidad o facultad que se tiene para comprender algo o formar una opinión:

me baso en mis propios criterios.

Evaluación: Cálculo, valoración de una cosa:

mínimo,ma:

adj. sup. irreg. de pequeño.

Que es lo más pequeño dentro de su especie:

rendimiento mínimo.

adj. y s. [Religioso o religiosa] de la Orden de San Francisco de Paula.

amer. Pusilánime, cobarde:

no seas mínimo y decídetete.

m. Límite inferior o extremo a que se puede reducir una cosa:

está al mínimo de su capacidad.

f. Temperatura mínima alcanzada en una zona:

Castilla-León suele dar las mínimas más bajas de España.

mínimo común múltiplo mat. El menor de los múltiplos comunes a varios números.

como mínimo loc. adv. Como poco:

eso costará como mínimo cien mil pesetas.

lo más mínimo loc. adv. Nada en absoluto:

no me gusta lo más mínimo cómo te comportas.

evaluación de los beneficios, de los conocimientos.

análisis

m. Distinción y separación de las partes de un todo hasta llegar a conocer sus principios, elementos, etc.:

el análisis es la segunda regla del método cartesiano.

Estudio minucioso de una obra, de un escrito o de cualquier otro objeto de estudio intelectual:

análisis de un poema.

gram. Examen de los componentes del discurso y de sus respectivas propiedades y funciones:

análisis gramatical, sintáctico.

med. Examen cualitativo y cuantitativo de ciertos componentes o sustancias del organismo según métodos especializados, con un fin diagnóstico:

análisis clínico.

inform. Estudio, mediante técnicas informáticas, de los límites, características y posibles soluciones de un problema al que se aplica un tratamiento por ordenador:

análisis informático.

No varía en pl.

B.6.1. ¿Cuáles son los criterios mínimos necesarios que deben ser considerados para la evaluación de un software de simulación como Homos 1.0?

Con el propósito de encontrar elementos suficientes que permitieran un proceso de innovación saludable de la herramienta software, se realizaron todos los elementos de análisis y evaluación que componen este proyecto de grado.

Al iniciar el trabajo de grado, se supuso que lo mejor y más sensato era que exist

el conjunto de criterios mínimos está presente dentro del proceso de análisis y evaluación realizado

Se presenta una justificación a la respuesta dada en este documento desde la misma labor de análisis realizada

lo que se ha encontrado hasta este momento de la investigación es que no hay un conjunto tal de criterios mínimos. Este trabajo requirió de todos y cada uno de los elementos realizados para configurar los elementos que posteriormente se convirtieron en innovación para la herramienta software.

Los pasos realizados en esta tesis configuran un conjunto de criterios de evaluación no mínimos, sino suficientes para el proceso de innovación y mejora planteado por este proyecto.

la evaluación es algo subjetivo

Evaluar es valorar, es el "acto de juzgar o inferir juicios atribuir o negar cualidades, medir la eficacia de un método o los resultados de una actividad; por tanto, se puede decir que la evaluación es un juicio de valor, un proceso sistemático, una medida de algo" (Cerdeña, 2000)

CERDA, H. La evaluación como experiencia total. Bogotá: Editorial Magisterio. 2000, 302 págs.

Un criterio es un “medio para juzgar”. En nuestro caso, serían los medios que nos permitan juzgar el ambiente de aprendizaje, en los sentidos que establecen los objetivos del proyecto. De manera que se puede aceptar la definición de De Miguel et al.(1994):

“criterio es un objetivo de calidad respecto al cual puede ser valorado un fenómeno o un aspecto concreto de la realidad”

. Se trata, pues, de un término genérico y poco específico.

De Miguel, M. et al. (1994). Evaluación para la calidad de los Institutos de Educación Secundaria. Madrid, Editorial Escuela Española.

[Evaluación de ambientes] de aprendizaje Miguel Ángel González Castañón Universidad EAFIT
Línea I+D Informática Educativa Departamento de Ingeniería de Sistemas

los criterios no se usan en el sentido de dar calificación, se usan para obtener los elementos ausentes o deficientes en la herramienta actual con respecto a lo que caracteriza a otros softwares similares

ver calificaciones en [an evaluation]

inicialmente el proyecto inicio con la idea de obtener un conjunto minimo de criterios de evaluacion de software

no es posible determinar el conjunto de elementos minimos. esto es asi ya que se realiza mejor si se cuenta con la mayor cantidad de datos posibles

el proceso de inv

teniendo en cuenta que el analisis es tan profundo como sea el proposito a lograr si se quieren cambios profundos se hace un analisis acorde

analisis superficial lo que se ve desde afuera, sin contacto con el codigo, experiencias

analisis funcional basico comportamiento de la herramienta, limitaciones, recursos

analisis funcional completo acceso a codigo y estructuras, conocer que existe y como se hizo

analisis de fundamentos es la capa mas interna, todas las demas obedecen a esta ya que existen para hacer realidad / implementar lo que se plantea en este nivel fundamental.

el conjunto de elementos no corresponde con algo minimo, mas bien constituye en su lugar se ofrece un conjunto de criterios suficientes para la evaluacion de software la descripcion anterior es la forma general el desarrollo de los apartados esta realizado pensando en la situacion particular de h1

suficiente bastante , adecuado para cubrir lo necesario

minimo que es lo mas pequeno dentro de su especie, limite inferior o extremo a que se puede reducir una cosa

los criterios minimos buscados [se encuentran en el / estan contenidos dentro de el] conjunto suficiente ofrecido

la búsqueda de elementos de mejora e innovación que motiva el proyecto requiere de insumos. la variedad y cantidad de insumos ofrecidos influye en el impacto variedad cantidad calidad de las novedades planteables

El conjunto de criterios suficientes ofrecidos son los distintos elementos contemplados en el análisis realizado

elementos destacados de análisis: Análisis de elementos de funcionamiento y comportamiento de las distintas interfaces existentes Definición de la forma de uso de cada interfaz disponible El acceso al código fuente análisis de metodología que fundamenta análisis de los medios de representación

la aplicación de los criterios establecidos debe ser realizada por

en general los criterios establecidos

las características del personal idóneo para la aplicación de la evaluación son: personal independiente de la fuente propósitos de mejora con conceptos de pensamiento sistémico: crítico, preguntón, expositor espíritu crítico no destructivo programador / desarrollador usuario avanzado de la herramienta

evaluar:

analizar una cosa para determinar su valor

criterio de evaluación de tal

norma para determinar el valor, importancia, trascendencia de tal

criterio mínimo de evaluación de tal

norma mínima para determinar el valor, importancia, trascendencia de tal

actualmente,

a través de todo el proceso de análisis realizado durante este proyecto se ha ido construyendo un todo multifacético. tiene un todo de estudio. se han usado múltiples enfoques para hacer visibles las distintas facetas que componen al software. las facetas mostradas se hicieron visibles gracias a los múltiples enfoques realizados. la idea es,

B.6.2. ¿Que es evaluar?

Enciclopédicamente hablando, de evaluar es :

- Analizar una cosa para determinar su valor, importancia o trascendencia

evaluar

tr. Valorar.

- Estimar el valor [de las cosas] no materiales.

Diccionario Enciclopédico Vox 1. © 2009 Larousse Editorial, S.L.
evaluar

- v t evaluar [eβa'luar] estimar el valor de algo o alguien

Evaluó la posibilidad de viajar.

B.6.3. ¿Que evaluar de Homos 1.0?

porque estos elementos y no otros??

puntos neurálgicos, donde duele, lo que realmente se necesita??

B.6.3.1. Validez de la pregunta de investigación y planteamiento de una respuesta satisfactoria soportada por el trabajo realizado

conjunto mínimo de criterios de evaluación

particularización de evaluaciones, dependientes de la temática de evaluación

evaluación totalizante, sistémica:

consideración de elementos relevantes

evaluación de un software como Homos comprende estos elementos

no hay conjunto de criterios mínimos

estudio sistémico requiere una

consideración de elementos más importantes

abierta a todo elemento que permita describir aspectos de la realidad (perspectivismo)

conocer para poder decir sobre

evaluación depende de la orientación

lo sistémico es holista, comprende integración de elementos y su dinámica

no es posible evaluar sin un contexto claro (fondo)

elementos clave, aspectos de evaluación relacionados con la herramienta software

proyección de evaluación

los elementos determinados en este apartado sirven de fundamento para la evaluación de futuras versiones de la herramienta software

evaluar:

Analizar una cosa para determinar su valor, importancia o trascendencia

evaluar

tr. Valorar.

Estimar el valor [de las cosas] no materiales.

Diccionario Enciclopédico Vox 1. © 2009 Larousse Editorial, S.L.

evaluar

v t evaluar [eβa'luar] estimar el valor de algo o alguien

Evaluó la posibilidad de viajar.

—

[http://www.farq.edu.uy/estructura/unidades_de_gestion/uap/matevalaprend/Juan %20Manuel %20Alvarez %](http://www.farq.edu.uy/estructura/unidades_de_gestion/uap/matevalaprend/Juan%20Manuel%20Alvarez%20)

<http://www.eduteka.org/pdfdir/ListaChequeo1.pdf>

Apéndice C

Grafómatas: Autómatas de movimiento

Introducción

Existen necesidades básicas de modelado no cubiertas en Homos 1.0. Una de ellas es contar con geometrias de vecindarios distintas al tradicional mosaico de celdas cuadradas. En este capítulo se describe a los grafomatas, un útil geométrico para la representación generalizada de divisiones regulares del espacio. Su objetivo es permitir la incorporación de múltiples sistemas de división del espacio en el ambiente de simulación de futuras versiones de la herramienta software Homos.

Inicialmente se establece el origen y las razones que condujeron al desarrollo del concepto de grafomata. A partir de un proceso de análisis se determinan las construcciones derivadas que es posible realizar con los grafomatas. Desde los resultados obtenidos se formulan elementos teórico-prácticos suficientes para la generación efectiva de los algoritmos requeridos. Lo anterior se complementa con la inclusión de ejemplos en los cuales se recrean varios tipos de teselados periódicos. Todo el trabajo realizado sustenta la demostración funcional del concepto en el prototipo Homos 2.0.

Al final del documento se establecen elementos de continuidad con la esperanza de motivar a otros a profundizar en el tema, implementar utilidades y realizar trabajo teórico-práctico relacionado con el uso aplicado de los grafomatas.

C.1. Origen y descripción básica

Entre las recomendaciones realizadas por los autores de Homos 1.0, esta la inclusión en el software y su metodología de nuevas formas de división del espacio¹. El actual proyecto hizo eco de esta recomendación en particular y ya puesta en marcha la investigación, se dedicaron recursos y tiempo a resolver el problema: ¿Cómo implementar nuevas geometrías o divisiones del espacio en la herramienta software Homos?. Este anexo es la respuesta a esta pregunta, pero antes de explicar que es lo que sirve de respuesta, se hará un recuento breve sobre los útiles actuales para la representación del espacio usado en los autómatas celulares.

En busca del tipo de representación del espacio a incluir en el software, durante la revisión bibliográfica se encontró que existe una múltiple variedad de tipos de vecindarios, hay un amplio surtido de formas de división del espacio. A parte de las mallas cuadradas y las geometrías hexagonales, triangulares, pentagonales [Bay94, Bay05] además de otras formas con polígonos regulares [Bay09], se está haciendo uso de teselados Voronoi, triangulaciones Delaunay y espacios con divisiones irregulares [FH01, BB10a], espacios de geometría dinámica [DH00] y teselaciones no periódicas para la implementación y desarrollo de modelos de autómatas celulares. De todas las formas disponibles, se determinó que las estructuras que presentan la mayor flexibilidad y cobertura de representaciones del espacio son las descritas por las teselaciones periódicas. Las mallas fundadas en polígonos regulares como la malla cuadrada están incluidas por defecto en este conjunto. Una vez establecida la forma de divisiones del espacio a implementar, llegó el momento de construir los útiles y algoritmos necesarios para hacer uso del tipo de representación elegido en el prototipo software. A pesar de la amplia variedad de algoritmos y el cúmulo de teoría descriptiva sobre teselados, sorprende lo poco que se encuentra acerca de métodos de construcción práctica de vecindarios basados en teselados periódicos de polígonos regulares o irregulares [KZ01].

De la búsqueda realizada se abstraerá que no hay un método único para representar el espacio, cada uno tiene ventajas y desventajas. No existe un medio estándar o *framework* para la división del espacio para uso común en los autómatas celulares y se hacen necesarias las implementaciones particulares para cada forma de geometría, especialmente de teselados con polígonos regulares. Si se precisa de una forma específica, la idea es que hay que re-implementarla según los requisitos particulares más que pensar en reusar un método común.

La búsqueda de medios para la realización práctica de teselados regulares se hizo infructuosa, hasta el punto de impulsar la creación de un método propio para la recreación de los teselados. A la estructura obtenida que sirve al propósito de reconstrucción de teselaciones regulares se la bautizó como grafómata². El cuerpo de este anexo se dedica a explicar y a mostrar lo que es un grafómata y como hacer uso práctico del concepto.

Al avanzar el desarrollo del proyecto, se encontró que existe un aporte teórico en el cual los grafomatas son una versión aplicada a superficies infinitas del concepto de *matroides graficas*

¹Tesis de Homos 1.0, Cap. 5 Alcances del proyecto y recomendaciones para futuros desarrollos, página 102.

²Es la unión del término grafo (γράφος: grabar, trazar, escribir) y (-ματος: que persigue, relativo al espíritu), que literalmente significa trazado esencial, espíritu de trazado o escritura. Se propone se haga uso del término en forma similar al de la palabra autómatas (αὐτόματος: espontáneo, que se mueve por sí mismo).

y *cograficas*³. Los matroides son estructuras que generalizan las propiedades de independencia, combinan el algebra lineal con la teoria de grafos. Introdución y fundamentos sobre el tema en [Whi32, Oxl92, BVS⁺93, Tru98]. Se espera que en proximas iteraciones de desarrollo se determinen las similitudes y el soporte teorico adicional que los matroides pueden brindar al concepto aplicado de grafomatas.

C.1.1. Definición de grafómata

Aunque una teselacion periodica contenga infinitos patrones de repeticion, existe al menos una forma basica (minima) que condensa lo invariable de cuerpo de repeticiones presente en el teselado. Esta forma mínima es un grafomata. Un grafomata es un pseudo-grafo dirigido que acepta bucles en sus vertices. Es una estructura compuesta de vertices y aristas tal que presenta una forma de conexion cerrada entre sus vertices. Hay un numero finito de vertices a los cuales hacer paso, cada uno esta configurado de cierta forma y siempre permite el paso hacia el mismo tipo de vertices.

En un teselado, los vertices son los puntos o ubicaciones en el espacio donde se unen las aristas o trazos que conforman los distintos patrones geometricos. Este proyecto denomina *puntos duales* al centro geometrico de los poligonos delimitados por los vertices y aristas que componen el grafomata. Los puntos duales son en realidad los vertices de la estructura del grafomata dual, cada poligono del teselado tiene uno y la conexión entre los distintos puntos duales existentes da forma al grafomata dual y por consiguiente al teselado dual.

El conjunto de aristas de un vertice establece el vecindario inmediato de ese vertice. Existen tres tipos de aristas en un grafomata:

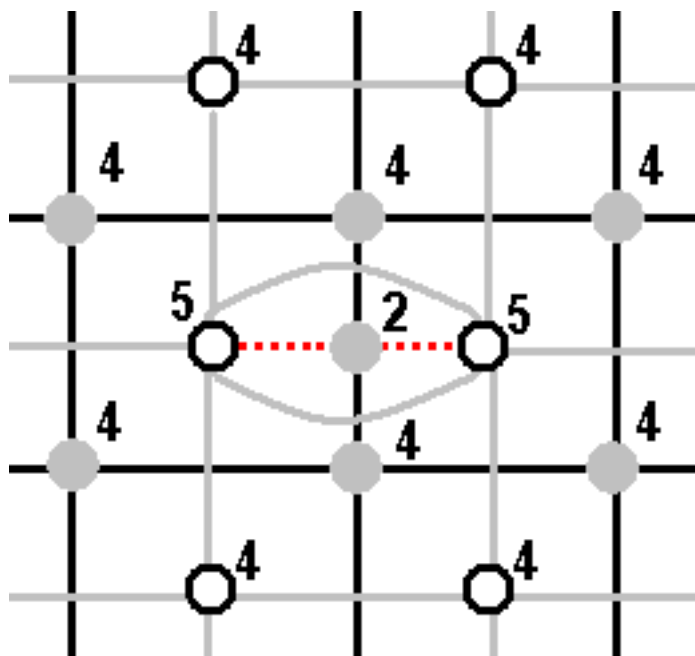
Aristas estacionarias, autoreferentes o bucle: Son aquellas aristas que apuntan hacia el vertice que les da origen. Permiten simular la permanencia en un vértice en el sistema. Todos los vertices de los grafomatas implementados en este proyecto tienen una arista estacionaria, notada como arista cero o a_0 . Estas aristas hacen del grafomata un pseudografo.

Aristas estructurales: Estas son las aristas establecen el paso de un vertice a otro y establecen los limites entre poligonos en el teselado. Estas son las aristas usadas para la busqueda de vertices y la determinacion de los dibujgrafos y duales. Las aristas estructurales hacen del grafomata un grafo planar. Este tipo de grafos tiene grafos duales. Hay tres o mas de estas aristas estructurales por cada vertice existente en el grafomata. Ver figura C.1 en la página siguiente.

Aristas conectivas, conectoras o de conexión: Se caracterizan porque generalmente se cruzan con las demas aristas estructurales del sistema. Permiten dar saltos entre vertices que no estan conectados directamente o entre vertices distantes. Estas se usan extensivamente en los vecindarios de Moore implementados y son las que establecen la conectividad particular de cada vecindario. En una estructura dada pueden existir o no aristas de conectividad, no hay un número limite para la cantidad de aristas conectoras a usar en un grafomata. Estas aristas hacen que el grafomata ya sea un multígrafo.

³Ver [Jho09] . Para esta tesis se interpretan en terminos de grafómata base y dual

Figura C.1: Razon de uso de 3 o mas aristas por vértice



Se muestra en la figura un conteo del total de aristas en cada vertice resaltado. Los puntos grises son vertices del sistema base, los circulos blancos son puntos duales. Al conectar las aristas que rodean al vertice de dos aristas se generan *dígonos*. El punto dual de un dígono generalmente está justo sobre el vertice de dos aristas que lo produce, causa repeticion de aristas estructurales e inserta poligonos de area cero en la estructura.

Fuente: Autor.

Una arista es *poda* si conduce a otro vertice. Lo contrario es la *antipoda*, que es la que viene desde otro vertice. La poda y antipoda tienen la misma magnitud y ocupan el mismo espacio, se diferencian en su sentido de aplicacion. Solo las aristas estructurales y las de conexion tienen antipodas.

En un grafomata puede darse la repeticion de cualquiera de los tipos de aristas establecidos. Al momento de uso de la estructura, la repeticion hace que aumente la probabilidad de elegir a la arista mas redundante. No se aconseja implementar redundancias, en los ejemplos de estructuras implementados en este proyecto no hay tales repeticiones, cada arista denota una relacion unica entre vertices.

En cada vertice deben existir 3 o mas aristas. Esto es con el proposito de evitar digonos en la estructura. Un dígono es un polígono de dos lados curvilíneos. Si los lados se hacen rectos se convierte en un polígono de dos lineas. El area interna de ese digono es cero.

La regularidad de los sistemas a representar permite establecer ejes de repetición, un par no lineal de estos ejes establece la malla de repeticion del sistema. La medida del paso entre ejes lo establece la distancia entre vertices de un mismo tipo presente en la estructura del teselado.

C.2. Grafómatas y construcciones relacionadas

Un grafomata permite un acceso ordenado a la estructura de vertices que componen un teselado. A partir de un grafomata podemos:

- Acceder al conjunto de vertices o nube de puntos que caracterizan al teselado en el espacio.
- Determinar el dibugrafo o trazado minimo del patron que rige la estructura de teselado.
- Establecer la malla regular o malla de repeticion de dibugrafo subyacente en la estructura de teselado.
- Con el dibugrafo y la malla regular podemos reconstruir el teselado del cual se extrajo originalmente el grafomata.
- Tambien es posible calcular mediante un algoritmo la estructura dual de un grafomata dado. Del grafomata dual tambien se puede obtener su dibugrafo y malla regular haciendo posible recrear el teselado dual.

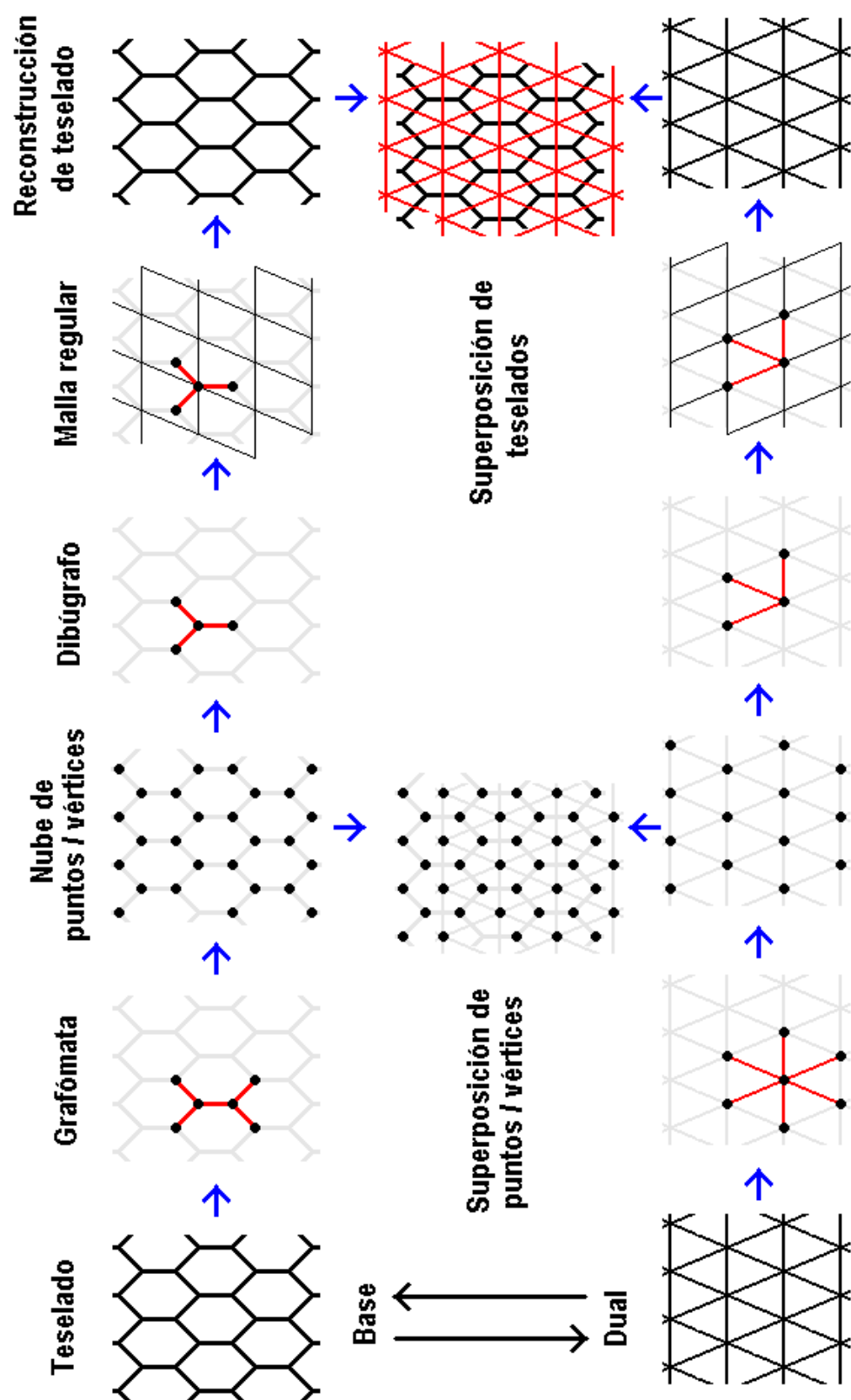
En síntesis, con un grafomata es posible realizar tres de operaciones básicas: moverse sobre la estructura de vertices del teselado, determinar la estructura dual asociada y reconstruir el teselado que le da origen. En la figura C.2 se muestran lado a lado dos ejemplos de teselados y las distintas construcciones obtenidas de los grafomatas particulares de cada estructura. Un ejemplo de aplicacion inmediato de los grafomatas es la superposición o mezcla de teselados.

El proceso de produccion de las diferentes construcciones relacionadas con los grafomatas se muestra en la figura C.2. En este proyecto se realizó:

- Una forma de implementacion manual de los grafomatas a partir de la nube de puntos que define un teselado. En adelante a estas estructuras se las denomina como grafomatas base.
- Se ofrecen metodos para la determinacion del dibugrafo y la malla regular.
- Se establece un algoritmo de teselado en funcion del dibugrafo y la malla de repeticion propias de un grafomata.
- Se establece un conjunto de pasos y algoritmos para la construccion automatica del grafomata dual.
- Se detalla la forma en la cual es posible recorrer la estructura de un teselado mediante el grafomata asociado.
- Se implementó un objeto en pascal/Delphi que permite la realizacion practica de todos los elementos anteriormente descritos.

En el resto de este apartado se describen: Los pasos necesarios para la construccion manual de un grafomata, la determinación del dibugrafo asociado, el proceso de construccion de la estructura dual y una relacion de las funciones componentes del objeto Tgrafomata implementado. El proceso de teselado se presenta por separado en la sección C.6.

Figura C.2: Grafoformas y construcciones relacionadas



Fuente: Autor.

¿Construcción Automática?

Espacio → **Teselado origen** → **Grafómata base**

Construcción manual

Formas de división del espacio → **Teselado periódico**
 Grafo planar infinito + teselas formadas por polígonos regulares e irregulares

Construcción Automática

Dibúgrafo + malla regular → **Ordenar de aristas** → **Determinar antipodas** → **Calcular puntos duales** → **Construir el dual del grafómata** → **Algoritmo de teselado** → **Enrutamiento en el espacio mediante grafómata + Reconstrucción del teselado**

Teselado dual → **Grafómata dual** → **Nube de puntos de teselación base**

C.2.1. Construcción manual de un grafomata

Para codificar manualmente un grafomata a partir de una estructura de teselado conocida se necesita que el teselado a representar:

- Sea una estructura planar⁴ y bidimensional. Es el tipo de formas de teselado manejado en este proyecto.
- Debe estar definido por un patron unico que se repite por toda su extension.
- Este compuesto por vertices con al menos tres aristas estructurales en cada uno.

En este proyecto, la parte no automatizada del proceso es la que corresponde con el paso de la nube de puntos del teselado a la forma basica de la estructura de grafomata. El paso de estructura de teselado a grafomata se realiza en los siguientes pasos:

1. Se establece el teselado a procesar.
2. Se determinan las distintas formas de vertices presentes en el teselado.
3. Se filtran y agrupan en una unidad o patron los vertices obtenidos.
4. Se establece la configuración de los enlaces entre los vertices.

Una vez construida la capa estructural del grafomata, se hace uso de las distintas funciones implementadas en el objeto Tgrafomata previsto en este proyecto para el armado final de la estructura y el calculo del dual asociado.

Al crear un grafomata se sugiere tener en cuenta las siguientes condiciones:

- Evitar la repetición de aristas estructurales, los algoritmos implementados en Tgrafomata suponen la conexión mas simple entre vertices.
- Hacer uso de valores enteros para la ubicación del destino de cada arista. La magnitud de las aristas del sistema está en función del numero 120, un numero magico o valor arbitrario que funciona para todas las estructuras implementadas.
- Las funciones implementadas en Tgrafomata solo procesan la capa de aristas estructurales del grafomata. Al crear la estructura, siempre procesar la capa de aristas estructurales con la función `construir_grafomata` antes de adjuntar en la estructura la capa de aristas conectivas⁵.

En la seccion C.4 en la página 337 se presentan ejemplos de implementación de grafomatas desde cero haciendo uso del objeto Tgrafomata implementado en el prototipo Homos 2.0. Tgrafomata está implementado en Pascal/Delphi 7.0.

⁴o grafo plano.

⁵Ver la sección C.3 en la página 330.

C.2.2. Configuración de un dibujgrafo

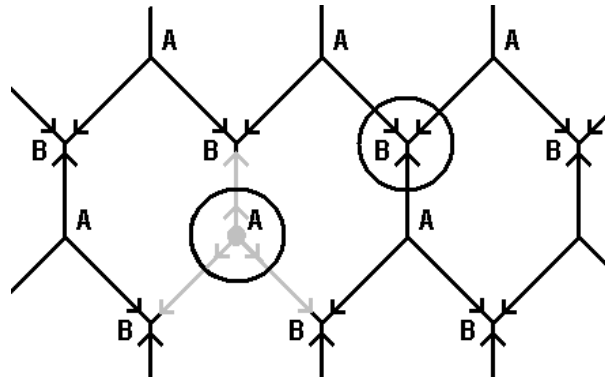
Aunque un grafomata es una abstracción por comprensión del patrón de teselado, no es práctica su utilización directa al momento de la construcción gráfica del teselado. Al usarlo, se repiten en el espacio de dibujo muchas de sus aristas. Para resolver esto simplificamos el grafomata mediante una selección de un conjunto mínimo de aristas requeridas para la representación gráfica del teselado. Esta selección de trazos es el dibujgrafo, un subgrafo de un grafomata. Las características de un dibujgrafo son:

- Las aristas que lo componen no se repiten o solapan.
- Su estructura no tiene aristas estacionarias ni de conectivas, todas las aristas componentes de la estructura salen de cada vértice, son podas de cada vértice.
- No existe una única forma de dibujgrafo. La forma final de la estructura depende de la manera como se elijan las aristas en cada vértice del grafomata. Las operaciones de ordenamiento realizadas sobre las aristas durante la creación del grafomata determinan la forma final de la representación gráfica. Una selección distinta de vértices y aristas produce una forma diferente.
- Todos los dibujgrafos sin importar el orden de selección de vértices, tienen la misma cantidad de aristas constituyentes.

El objetivo del dibujgrafo es reducir la cantidad de operaciones de dibujo. El dibujgrafo como tal está integrado dentro de la propia estructura del grafomata mediante un bit en cada arista. Para determinar un dibujgrafo se realiza el siguiente procedimiento: Para cada vértice presente en el grafomata se listan todas las aristas y se determina si el bit de dibujgrafo de la antipoda de la arista actual está activado. Si no está activo, se establece que la arista actual hace parte del dibujgrafo.

Se muestra gráficamente el proceso en la figura C.4. La creación efectiva del dibujgrafo se realiza en la función `calcular_antipodas`, se presenta en la sección C.2.3.2 en la página 320.

Figura C.4: Determinación del dibúgrafo de un grafómata



Tomando un vertice cualquiera se revisan sus conexiones. Del vertice A salen aristas y del B solo recibe aristas. Una arista es poda si se origina en el vertice actual. La antipoda es una arista de igual magnitud pero de sentido contrario, que enlaza un vertice externo con el vertice actual. Solo las aristas que salen de cada vertice se eligen como parte del dibugrafo. El dibugrafo de la malla presentada se muestra en la sección C.4.3 en la página 341.

Fuente: Autor.

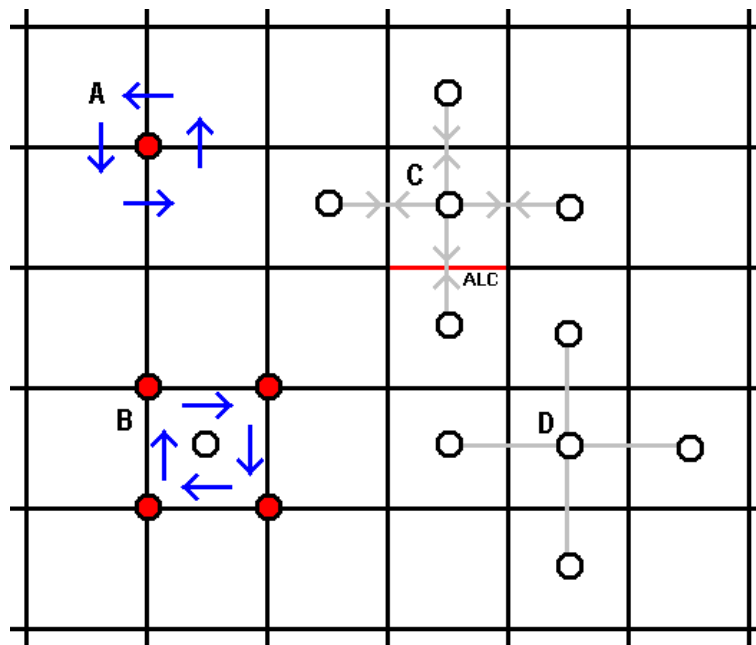
C.2.3. Construcción del dual de un grafomata

A partir de un grafomata base, se realizan las siguientes operaciones para obtener el grafomata dual:

- Se ordenan las aristas en cada vertice segun un sentido de giro establecido.
- Se determinan de las antipodas de todas las aristas estructurales del sistema.
- Se determinan de los vertices duales de cada uno de los poligonos adjuntos a la estructura del grafomata.
- Se construye el grafomata dual mediante la conexcion entre los vertices duales obtenidos.

En la figura C.5 se resume graficamente los distintos pasos establecidos para la determinacion del dual de la malla 4.4.4.4. El resto de este apartado trata con los detalles de la construccion del dual de un grafomata.

Figura C.5: Proceso simplificado de construcción de estructuras duales



Los puntos rojos son vertices del sistema base, los puntos blancos son vertices duales. Se muestran en la figura los procesos de:
A - Ordenamiento de aristas en vértices del grafomata.
B - Determinacion del punto o vertice dual de los poligonos delimitado por los vertices y aristas del sistema.
C - Conexion de vertices duales mediante la coincidencia de la arista lateral complementaria [ALC].
En D se observa la estructura de grafomata dual obtenido.

Fuente: Autor.

C.2.3.1. Ordenamiento de aristas

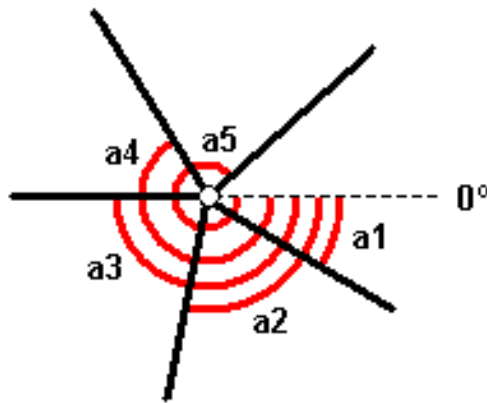
La figura C.6 da un ejemplo grafico de la forma como se ordenan las aristas en los vertices de grafomata. El proceso de construccion del grafomata no se establece ningun tipo de orden especifico

sobre las aristas; hacerlo concientemente durante el armado puede complicar un poco el proceso de construccion.

El ordenamiento se requiere para facilitar los posteriores procesos de iteracion sobre la estructura, particularmente en el recorrido del perimetro de los poligonos asociados a un vertice. Solo se procesan las aristas estructurales presentes en el sistema y se ordenan con respecto al eje coordenado x . Esto se hace determinando el angulo de inclinacion de cada arista y se reubicandola en la estructura segun este valor.

Al realizar el ordenamiento se garantiza que cada vertice, las aristas que lo componen se encuentran en una secuencia de giro predeterminada. La eleccion del sentido de giro es arbitrario, lo importante es la uniformidad de la aplicacion del sentido de giro a toda la estructura. En el prototipo el sentido de giro por defecto es hacia la izquierda, el proceso se implementa mediante el ordenamiento por mezcla o *mergesort*.

Figura C.6: Forma de ordenamiento de aristas en cada vértice del sistema



El vertice de la figura tiene cinco aristas que se ordenan segun la apertura angular con respecto al eje coordenado x . Se realiza este ordenamiento con el proposito de facilitar la construccion de los poligonos que hacen parte del grafomata. El ordenamiento de las aristas en la figura es horario.

Fuente: Autor.

C.2.3.2. Determinación de las antipodas del sistema

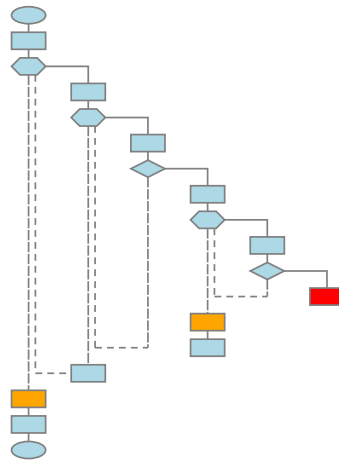
Este proceso evalua todas las aristas estructurales del grafomata en busqueda de su contrapartes o antipodas. La operacion se realiza ya que se requiere para determinar de los vertices y aristas que delimitan a un poligono dentro de la estructura.

Al construir el grafomata, la parte de la estructura de datos que configura las relaciones de podalidad entre aristas no se ha establecido. El procedimiento que completa los datos es el siguiente: Se itera sobre cada vertice y sobre cada arista del sistema, se detecta si ya se ha obtenido la antipoda de la arista en proceso. Si no se ha determinado, se busca dentro de las aristas del vertice destino de la arista

actual cual es la antipoda correspondiente. Se establecen los valores de antipodalidad y se continua el proceso hasta completar la estructura.

La figura C.7 muestra la estructura proceso. Determinar las antipodas permite establecer la integridad de la estructura y la forma de sus conexiones. La funcion realiza simultaneamente la determinacion de dibujografo. El sentido del ordenamiento realizado al momento de construir el grafomata influye sobre la forma final del dibujografo. El codigo fuente de la funcion se muestra en el listado C.2 en la página 321 .

Figura C.7: Determinacion de antípodas del sistema



El algoritmo recorre cada vertice y cada arista, si a la arista actual no se le ha determinado la antipoda entonces se busca su antagonista en el vertice destino. Durante el recorrido por las aristas del sistema, se configura la estructura del dibujografo. Detalles acerca de los diagramas de flujo se presentan en la sección E.7 en la página 654.

Fuente: Autor.

Fuente en Pascal C.2: Enrutamiento: Paso entre vértices del grafomata

```

1 function Tgrafomata.calcular_antipodas:estructura_grafomata;
2 var
3   h,i,j,k,l,n,m:integer;
4   ik:~tpoint; //ubicaciones de posicion a comparar entre enlaces
5   aristas_revisadas:integer;
6   test:tpoint;
7 begin
8   with definicion do
9   begin
10    j:=high(vertice);
11    aristas_revisadas:=0;
12    for i:=0 to j do //evaluar cada vertice
13    begin
14      with vertice[i] do
15      begin
16        l:=high(arista);
17        arista[0].enlace.antipoda:=0; //la arista identidad o estacionaria no tiene antipoda, luego
18        antipoda = 0 se entiende como no calculada
19        radio:=0; //radio de vertice = 0
20        for k:=1 to l do //evaluar las aristas no estacionarias del vertice
21        begin
22          with arista[k] do
23          begin
24            with direccion.punto do inc(radio,round(sqrt(sqr(x)+sqr(y)))); //bloque para calcular el
25            valor del radio interno medio del vertice actual
26            if enlace.antipoda=0 then //si no se ha evaluado la antipoda
27            begin

```

```

26         h:=enlace.vertice;
27         n:=high(vertice[h].arista);
28         ik:=@direccion.punto; //ubicacion en el vertice[i].arista[k]
29         for m:=1 to n do //existe una arista en algun vertice del grafomata que cumpla con la
            condicion establecida?
30             begin
31                 test:=sumar(vertice[h].arista[m].direccion.punto,ik^);
32                 if comparemem(@test,@cero,sizeof(tpoint)) then break; //verificar la neutralizacion
                    de las aristas antipodas
33             end;
34 //si m>n entonces el vertice[i].arista[k] apunta a un vertice[h] con el cual no tiene arista en
            comun = error en la construccion del grafomata
35             assert(m<=n,'Problema de conectividad en la arista '+IntToStr(m)+' del v rtice '+
                IntToStr(h)+' de la construcci n '+nombre);
36             enlace.antipoda:=m; // aristas antipodas
37             dibugrafo:=true; // esta arista hace parte del dibugrafo
38             vertice[h].arista[m].enlace.antipoda:=k;
39             inc(aristas_revisadas,2); //se realizan cambios por pares en este algoritmo
40         end;
41     end;
42 end;
43 radio:=round(radio/(2*1)); //v[i].radio = valor entero del alcance promedio de este vertice
44 end;
45 end;
46 //info.total_aristas es un valor par. cada arista tiene su contraparte o antipoda
47 assert(aristas_revisadas=info.total_aristas,'No se encontraron algunas antipodas para las aristas
    componentes de la estructura '+nombre);
48 result:=vertice;
49 end;
50 end;

```

C.2.3.3. C lculo de los vertices duales

En este paso, se determinan las ubicaciones de los vertices duales del sistema. El proceso consiste en determinar para cada par vertice-arista existente el perimetro del poligono al cual se delimita. Por cada poligono revisado se calcula su centro o lo que es lo mismo, el vertice dual correspondiente.

Para realizar el algoritmo se requiere del ordenamiento de las aristas estructurales en cada vertice. Esto hace posible saber cual es la arista anterior y siguiente a la arista actual. La visita al perimetro del poligono asociado al vertice origen del recorrido se realiza de esta forma:

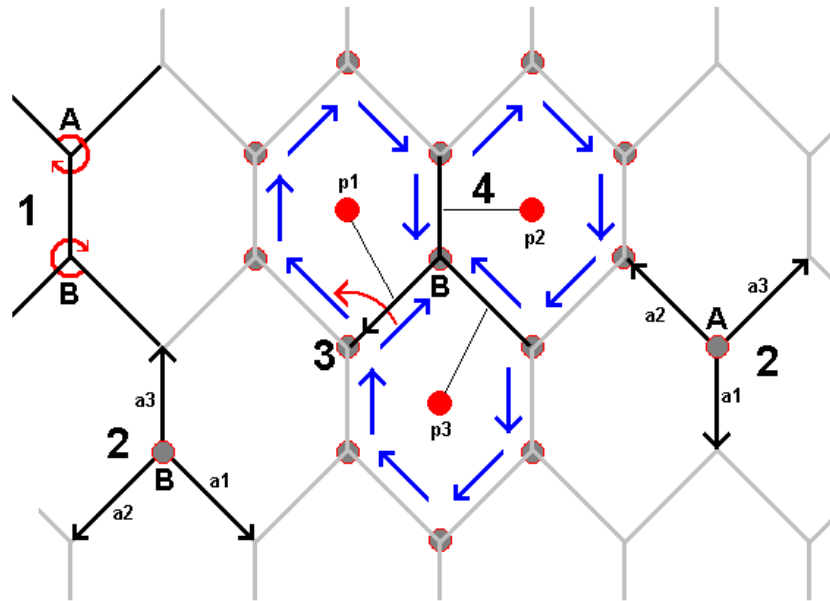
- Se elige la primera arista del recorrido, esta lleva al siguiente vertice del perimetro.
- En el siguiente vertice se determina la antipoda a la arista de llegada, se usa la antipoda obtenida para determinar la arista que le es anterior en secuencia en el vertice actual.
- La arista obtenida lleva al nuevo siguiente vertice y asi sucesivamente, hasta volver al vertice origen del recorrido.
- En cada llegada a un vertice se actualiza una sumatoria de las ubicaciones de los vertices visitados, el promedio de las ubicaciones ser  el lugar del vertice dual dentro del poligono revisado.

Al final del proceso se almacena en cada arista del sistema base el dato del punto dual asociado. Solo las aristas que pertenecen a un mismo poligono tienen igual valor de vertice dual. Los demas tipos de aristas no sirven para la construccion de duales. las aristas estacionarias no delimitan superficie

alguna y las aristas conectivas al cruzarse con otras, no definen un lado común que permita el calculo de un punto central sobre una superficie.

Si existe algun problema de conexcion no detectado durante la determinacion de las antipodas, es posible que el recorrido de perimetros no sea finito. El exceso de ciclos en el armado de un poligono indica un error de configuracion de la estructura o en la determinacion de las antipodas. En cualquiera de estos casos el algoritmo detecta el problema e interrumpe la ejecucion.

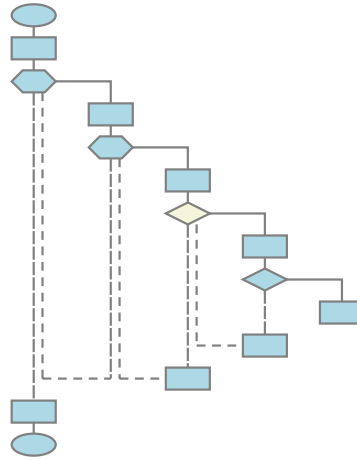
Figura C.8: Calculo de vertices duales



En 1 se muestra el sentido de orden de las aristas del cada vertice del sistema. En 2 se identifican las aristas en cada vertice segun el ordenamiento. En 3 se observa el proceso de paso al siguiente vertice del perimetro del poligono. En 4 se ilustra la relación entre vertices duales y las aristas del vertice B. Cada arista tiene un punto dual asociado, el orden en las aristas garantiza que sean vertices duales distintos. Para el ejemplo: (p_1, a_2) , (p_2, a_3) y (p_3, a_1) .

Fuente: Autor.

Figura C.9: Proceso de calculo de vertices duales



El algoritmo visita cada vertice y arista disponible, en el ciclo **repetir-hasta** mas interno realiza el recorrido del perimetro del poligono.

Detalles acerca de los diagramas de flujo se presentan en la sección E.7 en la página 654.

Fuente: Autor.

Fuente en Pascal C.3: Enrutamiento: Paso entre vértices del grafomata

```

1 function Tgrafomata.calcular_dual:estructura_grafomata;
2 var
3   i,j,k,l:integer;
4   cv,wj,hi:integer;
5   tk:arista_grafomata;
6   pi:tpoint;
7 begin
8   j:=high(definicion.vertice);
9   for i:=0 to j do //evaluar cada vertice
10    begin
11      with definicion.vertice[i] do
12        begin
13          l:=high(arista);
14          for k:=1 to l do //evaluar todas las aristas no estacionarias del vertice
15            begin
16              tk:=arista[0];
17              cv:=0; //conteo de aristas/vertices visitadas por el algoritmo
18              with arista[k].direccion do
19                begin
20                  dual.x:=0;
21                  dual.y:=0;
22                  hi:=k; //arista inicial para el recorrido
23                  repeat
24                    tk:=calcular_arista(hi,tk); //avance al siguiente punto en el recorrido
25                    //sumatoria para hallar el punto central
26                    dual.x:=dual.x+tk.direccion.punto.x;
27                    dual.y:=dual.y+tk.direccion.punto.y;
28                    wj:=high(definicion.vertice[tk.enlace.vertice].arista);
29                    hi:=tk.enlace.antipoda+1; //sentido de busqueda horario
30                    if hi>wj then hi:=1; //1 es la arista siguiente a la ultima arista
31                    inc(cv); //paredes o lados del poligono cuyo centro es el valor del punto en calculo
32                    assert(cv<=info.total_aristas,'Exceso de ciclos al buscar un punto dual en la
33                      estructura '+nombre);
34                  until (tk.direccion.punto.x=0)and(tk.direccion.punto.y=0); //cuando en la vuelta se
35                end
36              //control al incremento de vertices visitados... en caso de exceso verificar la construccion del
37              grafomata
38              //para crear el dual:
39              //registrar el vertice-arista usado para calcular el promedio del punto dual actual
40              //este punto dual será un vertice del grafomata dual
41              //es necesario un proceso de simplificacion / reducion de vertices iguales mediante el camino
42              establecido por el dibugrafo
43            end
44          end
45        end
46      end
47    end
48  until (tk.direccion.punto.x=0)and(tk.direccion.punto.y=0); //cuando en la vuelta se

```

```

        encuentre el inicio entonces se ha terminado el viaje
40      dual.x:=round(dual.x/cv); //dual = promedio de los puntos geometricos de las ubicaciones
        de los vertices del recorrido
41      dual.y:=round(dual.y/cv);
42      pi:=dual; //test
43      pi:=pi;
44    end;
45  end;
46 end;
47 end;
48 result:=definicion.vertice;
49 end;

```

C.2.3.4. Proceso de armado de la estructura dual

La construccion del dual se realiza en tres etapas:

- Se crea un vertice dual por cada arista de cada vertice que pertenezca al dibugrafo del grafomata base, en cada vertice se crean tantas aristas como paredes tenga el poligono del cual es centro. En cada arista se determina cual es su ALC. Esta informacion posteriormente hará posible la conexion entre vertices.
- Se filtran los vertices duales producidos, se eliminan los repetidos, se ordenan las aristas de los vertices.
- Posteriormente se conectan los vertices sobrevivientes usando la configuracion de las conexiones ALC establecidas en las aristas, se configura el dibugrafo y la configuracion de antipodas de todas las aristas de la estructura dual.

Como se observa en todo el proceso, para la construccion del dual solo se requiere de la informacion proporcionada por la estructura de datos del grafomata base. Se espera fortalecer la integracion de las funciones presentadas en proximas iteraciones del prototipo y mejorar algunos de los procesos realizados, especialmente la construccion final de la estructura dual.

Fuente en Pascal C.4: Enrutamiento: Paso entre vértices del grafomata

```

1 function Tgrafomata.construir_dual:Tgrafomata;
2 var
3   i,j,k,l,m,n:integer;
4   il,l1,m1,nl:integer;
5   cv,wj,hi:integer;
6   tj,tk,tw:arista_grafomata;
7   pa:tpoint;
8   terminar_busqueda:boolean;
9   p_arista:array of ^arista_grafomata;
10 begin
11   result:=nil;
12   Gmdual:=Tgrafomata.create;
13   Gmdual.Gmdual:=nil; //no se encadenan los grafómatas entre si
14   Gmdual.nombre:='Dual - '+self.nombre;
15 //etapa de recorrido de grafomata y creacion de vertices del grafomata dual
16   j:=high(definicion.vertice);
17   for i:=0 to j do //evaluar cada vertice
18     begin
19       with definicion.vertice[i] do
20         begin
21           l:=high(arista);
22           for k:=1 to l do //evaluar las aristas del vertice excepto la arista estacionaria
23             begin

```

```

24     if not arista[k].dibugrafo then continue; //contemplar solo las aristas que componen el
25     dibujrafo
26     Gmdual.agregar_vertice; //agregar un vertice dual por cada arista del vertice en
27     evaluacion
28     tj:=arista[0]; //ubicacion del vertice
29     cv:=0; //conteo de aristas/vertices visitadas por el algoritmo
30     wj:=high(arista);
31     with arista[k].direccion do
32     begin
33         hi:=k; //arista inicial para el recorrido
34         repeat
35             tk:=calcular_arista(hi,tj); //avance a la siguiente arista en el recorrido
36
37             hi:=hi+1; //busqueda del dual de esta arista
38             if hi>wj then hi:=1; //1 es la arista siguiente a la ultima arista
39             tw:=calcular_arista(hi,tj); //ubicacion de la arista siguiente a la arista presente en
40             este vertice
41
42             pa:=sumar(tw.direccion.dual,reflejar_xy(tk.direccion.dual)); //punto, ubicacion de
43             extremo final de la arista actual
44             GmDual.agregar_arista(0,pa); //se inician con enlace al vertice dual cero, ya que se
45             desconoce en este instante su estructura final
46
47             //este dato realiza la magia de la conexion ALC = antipoda original = Arista Lateral
48             Complementaria
49             Gmdual.p_arista_reciente.direccion.dual.x:=tk.enlace.origen; //datos de ALC
50             Gmdual.p_arista_reciente.direccion.dual.y:=tk.enlace.vertice; //datos de ALC
51
52             wj:=high(definicion.vertice[tk.enlace.vertice].arista);
53             hi:=tk.enlace.antipoda+1; //sentido de busqueda horario
54             if hi>wj then hi:=1; //1 es la arista siguiente a la ultima arista
55             inc(cv);
56             assert(cv<=info.total_aristas,'Exceso de ciclos, problema al buscar punto dual');
57             tj:=tk;
58
59             until (tk.direccion.punto.x=0) and (tk.direccion.punto.y=0); //cuando en la vuelta se
60             encuentre el inicio entonces se ha terminado el viaje
61
62             n1:=high(GmDual.definicion.vertice);
63             l1:=high(GmDual.definicion.vertice[n1].arista); //determina el tamaño del vector patron
64             comparar
65             GmDual.ordenar_aristas_vertice(n1); //ordenacion de las aristas del vertice n1 para
66             facilitar la comparacion
67
68             if n1>0 then //si hay mas de un vertice
69             begin
70                 i1:=0; //comparando desde el primer vertice del nuevo sistema
71                 while i1<n1 do
72                 begin
73                     if l1=high(GmDual.definicion.vertice[i1].arista) then //si la cantidad de aristas
74                     corresponde entonces comparar los vertices i1 y n1
75                     begin
76                         for m1:=1 to l1 do //comparacion de aristas excluyendo la arista estacionaria
77                         if not
78                             (comparemem(@GmDual.definicion.vertice[i1].arista[m1].direccion.punto,
79                             @GmDual.definicion.vertice[n1].arista[m1].direccion.punto,sizeof(
80                             tpoint)) and //comparar dimensiones de aristas
81                             comparemem(@GmDual.definicion.vertice[i1].arista[m1].direccion.dual,
82                             @GmDual.definicion.vertice[n1].arista[m1].direccion.dual,sizeof(
83                             tpoint))) then break; //comparar vertices destino
84
85                         if m1>l1 then
86                         begin
87                             setlength(GmDual.definicion.vertice,n1); //puedo eliminar el vertice n1 porque
88                             es igual a un vertice i1
89                             s:=s+'*';
90                             break;
91                         end;
92                     end;
93                     inc(i1);
94                 end;
95             end;
96         end;
97     end;
98 //
99     end;
100     end;
101     end;
102     end;
103 //nota:

```

```

88 //en hexagono, aparecen como resultado intermedio 2 vertices que tienen la misma estructura pero
    que tienen diferentes posiciones de punto medio.
89 //el algoritmo elimina uno de estos vertices
90
91 {
92 hasta este punto del algoritmo, se han hallado los vertices del dibugrafo que construye el
    grafomata dual y se han establecido
93 las distintas ubicaciones en el espacio a las que se dirigen las aristas.
94 falta establecer cual es el vertice al que esta dirigido cada arista y su antipoda
95 se desconoce si el algoritmo implementado es general a toda estructura de grafomata, funciona con
    los ejemplos implementados
96 }
97
98 m:=0;
99 p_arista:=nil;
100 with GmDual.definicion do
101 begin
102     j:=high(vertices); //determinar el numero final de vertices del grafomata dual
103     for i:=0 to j do
104         m:=m+high(vertices[i].arista); //determinar el numero final de aristas en este vertice del
            grafomata dual
105
106     setlength(p_arista,m); //crear el vector de aristas a procesar
107     n:=m-1;
108
109     m:=0;
110     for i:=0 to j do
111         begin
112             l:=high(vertices[i].arista);
113             for k:=1 to l do
114                 begin
115                     p_arista[m]:=addr(vertices[i].arista[k]); //listar las aristas
116                     inc(m);
117                 end;
118             end;
119
120 //para los casos probados hasta ahora para este algoritmo, para realizar las conexiones de pares de
    vertices es mejor revisar primero los demas nodos y luego en si mismo
121 //se realiza una comparacion burbuja para este fin
122 //este algoritmo es mejorable si se desarrolla una forma mas practica de realizar la conexion de
    las aristas, ojala sin la necesidad de realizar ordenamientos
123 //se desconoce como realizar y garantizar que un vertice deba estar unido especificamente a otro.
    este algoritmo lo hace por el orden de aparicion de las aristas y la
124 //suposicion de que es necesario probar primero las aristas de otros vertices y luego las del
    vertice en evaluacion
125
126     m:=n;
127     while m>0 do
128         begin
129             l:=m;
130             for i:=m downto 1 do //no se incluye la primera arista del listado
131                 begin //el elemento 0 se compara con los demas
132
133                     terminar_búsqueda:=(p_arista[0].direccion.punto.x + p_arista[i].direccion.punto.x = 0) and (
                        p_arista[0].direccion.punto.y + p_arista[i].direccion.punto.y = 0);
134                     //buscar la pareja que tenga el mismo enlace lateral
135                     if j>0 then //si existe mas de un vértice que explorar
136
137 //condicion original
138                     terminar_búsqueda:=terminar_búsqueda and (p_arista[0].direccion.dual.x=p_arista[i].direccion.
                        dual.y) and (p_arista[0].direccion.dual.y=p_arista[i].direccion.dual.x);
139
140                     if terminar_búsqueda then
141                         begin
142                             p_arista[0].dibugrafo:=true;
143                             //determinación del vértice destino y la antípoda de cada arista par encontrada
144                             p_arista[0].enlace.vertice :=p_arista[i].enlace.origen;
145                             p_arista[0].enlace.antipoda:=p_arista[i].enlace.id;
146                             p_arista[i].enlace.vertice :=p_arista[0].enlace.origen;
147                             p_arista[i].enlace.antipoda:=p_arista[0].enlace.id;
148                             p_arista[0]:=nil;
149                             p_arista[i]:=nil;
150                             m:=m-2;
151                             break;
152                         end;
153                     end;
154                     if m>=1 then

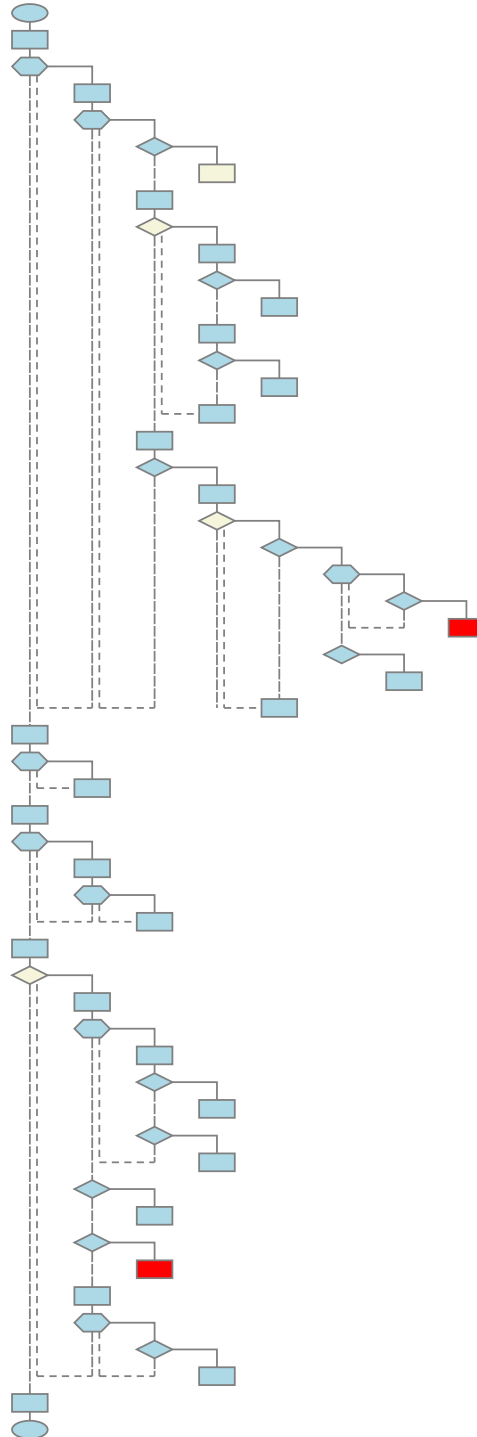
```

```

155     begin
156         m:=0;
157         p_arista:=nil;
158         assert(m=0,'Hay un problema con la estructura de este grafómata, no se encontró una arista
           par/complementaria a la arista[0] actual');
159         exit; //hay un problema con la estructura de este grafómata, no se encontró una arista par/
           complementaria a la arista[0] actual
160     end;
161     if m<0 then break;
162     k:=0;
163     for i:=0 to 1 do //se eliminan los valores nil que existan en la lista , sin alterar el orden
           de aparición de las aristas
164     begin
165         if p_arista[i]<>nil then
166             begin
167                 p_arista[k]:=p_arista[i];
168                 inc(k);
169             end;
170         end;
171     end;
172     p_arista:=nil;
173 end;
174 result:=Gmdual;
175 end;

```

Figura C.10: Construcción de la estructura dual a partir de vertices duales



En el ciclo doble inicial, el primer ciclo interno se encarga de crear los vertices en la estructura del dual, el segundo ciclo filtra los vertices repetidos producidos. Los ciclos intermedios construyen una lista de todas las aristas que componen la estructura y el ciclo final se encarga del empalme entre vertices y aristas. El empalme se hace mediante la búsqueda de parejas con el mismo ALC, determinado durante la parte inicial del proceso.

Detalles acerca de los diagramas de flujo se presentan en la sección E.7 en la página 654.

Fuente: Autor.

C.3. Selección de funciones del objeto Tgrafomata

Se listan a continuación algunas funciones relacionadas con la construcción y uso de un grafomata. No se listan todas las funciones disponibles porque algunas son herramientas de desarrollo y otras existen solo como pruebas de concepto. Se prevee una mejor organización e integración funcional de las funciones listadas en próximas iteraciones de desarrollo del prototipo.

C.3.1. Funciones útiles en la construcción manual de grafómatas

Estas funciones no están integradas al objeto Tgrafomata.

- `function rotar(angulo:precision_angulo;punto:tpoint):tpoint;`
Rotación de una arista centrada en el origen según el ángulo especificado, ángulo es flotante y está en grados.
- `function rotar90d(punto:tpoint):tpoint;`
Rota la arista con respecto al origen 90 grados a la derecha o rotación horaria.
- `function rotar90i(punto:tpoint):tpoint;`
Rota la arista con respecto al origen 90 grados a la izquierda o rotación antihoraria.
- `function reflejar_y(punto:Tpoint):tpoint;`
Refleja la arista con respecto al eje y .
- `function reflejar_x(punto:Tpoint):tpoint;`
Refleja la arista con respecto al eje x .
- `function reflejar_xy(punto:Tpoint):tpoint;`
Refleja la arista simultáneamente con respecto a los ejes x y y .
- `function sumar(p1,p2:tpoint):tpoint;`
Realiza la suma vectorial de dos puntos centrados en el origen.
- `procedure ordenar_aristas(arista:aristas_grafomata);`
Ordenamiento de las aristas de
- `function ubicar_vertices_dibugrafo(xc,yc,v:integer;g:Tgrafomata):puntos_xy;`
Calcula las ubicaciones de los vértices del dibugrafo con respecto a una ubicación x_c, y_c dada.
- `function angulo_arista(p_arista:Tpoint):precision_angulo;`
Determina el valor del ángulo de una arista.

C.3.2. Funciones de construccion de un grafomatas

A partir de este punto las funciones listadas hacen parte del objeto Tgrafomata. En cada nuevo vertice del sistema se agrega automaticamente una arista estacionaria. Observese que en la construccion de un grafomata lo importante es establecer correctamente las conexiones entre vertices, la arista que va hacia un vertice tiene otra igual y opuesta en el vertice destino, la bidireccionalidad es necesaria para simplificar la eleccion de rutas o caminos desde cada vertice.

- `function agregar_vertice:vertice_grafomata;`
Anexa un nuevo vertice a la estructura del grafómata.
- `function agregar_arista(
vertice_destino:integer;punto_destino:tpoint):arista_grafomata;`
Anexa una nueva arista en vertice_destino. La arista se dirige a punto_destino.
- `function agregar_reflexion_vertice(
vertice_patron,vertice_destino:integer;
reflexion:tipo_reflejo):vertice_grafomata;`
- Se anexa en vertice_destino los datos de una arista previamente existente en vertice_patron, aplicando sobre ella una operacion de reflexion disponible. Las operaciones de reflexion disponibles son:
`tipo_reflejo = copiar_valores,reflejar_en_x,reflejar_en_y,
reflejar_en_xy,rotar_90d,rotar_90i`
- `function agregar_reflexion_arista(
vertice_patron,arista_patron,vertice_destino:integer;
reflexion:tipo_reflejo):arista_grafomata;`
Se agrega en vertice_destino una copia de la arista_patron del vertice_patron. Se aplica sobre la arista una operacion de reflexion.
- `function agregar_rotacion_arista(
vertice_patron,arista_patron,vertice_destino:integer;
angulo:integer):arista_grafomata;`
Se agrega en vertice_destino una copia la arista_patron del vertice_patron. Se aplica sobre la arista una rotacion de *angulo* grados.
- `function punto_arista(este_vertice,esta_arista:integer):tpoint;`
Permite recuperar la ubicacion a la que apunta una arista en un vertice determinado.

C.3.3. Funciones relacionadas con la construccion del grafomata dual

- `function ordenar_aristas_grafomata:estructura_grafomata;`

Realiza una ordenacion de aristas en todos los vertices del grafomata. El sentido de ordenamiento por defecto es antihorario.

- `function calcular_antipodas:estructura_grafomata;`

Determina y actualiza automaticamente las antipodas de todas las aristas disponibles del sistema. La funcion falla si encuentra cualquier tipo de imperfeccion estructural, P.ej: aristas sin antipoda en el sistema, conteo impar de aristas, vertices sin aristas, etc. El código fuente de la funcion se muestra en el listado C.2 en la página 321.

- `function calcular_dual:estructura_grafomata;`

Funcion que determina los puntos o vertices duales que produce la estructura del grafomata. Esta funcion es sensible a problemas de truncamiento y redondeo. El código fuente de la funcion se muestra en el listado C.3 en la página 324.

- `function construir_dual:Tgrafomata;`

Construye el dual de un grafomata a partir de los vertices duales previamente calculados. En esta version del software solo hay conexion desde el sistema base al dual, el dual apunta a nil. Se evita encadenar ciclicamente a los grafomatas o calcular duales de duales. Ver sección C.7.3 en la página 433. El código fuente de la funcion se muestra en el listado C.4 en la página 325.

- `function max_min_aristas_grafomata:max_min_aristas;`

Establece algunos valores de control relacionados con la estructura particular del grafomata.

- `procedure construir_grafomata;`

Esta funcion se realiza inmediatamente despues de haber establecido la estructura principal de un grafómata. Usa a las demas funciones presentadas en este apartado en una secuencia especifica para completar los datos de la estructura del grafomata y realizar la creacion efectiva del dual. Se presenta su codigo fuente en el listado C.5.

Fuente en Pascal C.5: Enrutamiento: Paso entre vértices del grafomata

```
1 //este procedimiento resume las operaciones que se necesitan para completar el grafomata creado:
2 //ordena las aristas , calcula las antipodas de las aristas , los puntos de los vertices duales y
  luego construye el grafomata dual.
3 //un fallo en cualquier etapa de este proceso implica una estructura de grafomata base incompleta o
  incorrectamente construida.
4 procedure Tgrafomata.construir_grafomata;
5 begin
6   max_min_aristas_grafomata; //determina algunos valores caracteristicos del grafomata actual
7   ordenar_aristas_grafomata; //correcciones en el orden de las aristas en cada vertice , se requiere
  antes de realizar cualquier otra operacion sobre los datos del grafomata
8   calcular_antipodas; //determina y establece la informacion de los enlaces de las aristas en cada
  vertice del grafomata
9   calcular_dual; //encuentra los valores de los puntos-vertices duales
10  construir_dual; //construye el grafomata dual
11 end;
```

Se construyó adicionalmente el procedimiento `descripcion_grafomata`, que permite exportar la estructura de los grafomatas definidos como archivos `*.SVG`⁶. También recolecta datos sobre propiedades particulares de las estructuras procesadas y las reporta en un archivo de texto. Esta función está desactivada por defecto, sirve solo a propósitos de procesos de desarrollo de código.

C.3.4. Funciones de uso del grafomata

- `function calcular_arista( camino:integer;nodo:arista_grafomata):arista_grafomata;`

A partir de una ubicación válida en el teselado, permite obtener la ubicación y la identificación del siguiente vértice apuntado por la arista indicada en `camino`. Esta función es la que permite la realización del movimiento de los objetos a través de los caminos establecidos por el teselado base. El código fuente de esta función muestra en el listado C.6 en la página 334.

- `function ubicar_vertice(ag:arista_grafomata;vd:integer):aristas_grafomata;`

Retorna una lista con las ubicaciones más cercanas a `ag` del vértice `vd`, usando para la búsqueda solo las aristas estructurales del sistema. Por defecto el algoritmo de búsqueda profundiza en la estructura.

C.3.5. Funciones de búsqueda de vértices en el espacio de teselado

- `function camino_metropolitano( punto:Tpoint;nodo:arista_grafomata):arista_grafomata;`
`function camino_euclidiano( punto:Tpoint;nodo:arista_grafomata):arista_grafomata;`

Estas funciones permiten determinar a partir de una ubicación conocida almacenada en `nodo`, cuál es el vértice más cercano al punto dado usando la distancia euclidiana o metropolitana según corresponda. `punto` es una ubicación cualquiera en el espacio. Se avanza un vértice por cada llamado a la función. En algunos teselados da el caso de la interrupción en el avance de la búsqueda o puntos muertos, esto se debe a la naturaleza de los enlaces entre vértices.

- `function ubicar_punto(punto:Tpoint;metodo:byte=0):arista_grafomata;`

Encuentra la ubicación del vértice dentro del teselado más próximo al punto según el método de búsqueda elegido. Para la búsqueda usa la distancia euclidiana por defecto. Los valores del parámetro `metodo` son: 0 para distancia euclidiana y 1 para distancia metropolitana.

- `function camino_puntos( var camino:puntos_xy;punto:Tpoint;metodo:byte=0):arista_grafomata;`

Encuentra la ubicación del vértice dentro del teselado más próximo al punto según el método de búsqueda elegido. Retorna la lista de los vértices usados durante el recorrido. Para la búsqueda usa la distancia euclidiana por defecto.

⁶Con esta función se crearon los patrones de grafómatas y dibúgrafos presentados en la sección C.4 en la página 337.

C.3.6. Movimiento sobre el espacio teselado

Una vez establecido el grafomata se requiere saber como usarlo. Para pasar de un vertice a otro dentro del teselado usando el grafomata asociado se requiere:

- Conocer la ubicacion en el espacio del vértice actual. Cada vertice se localiza en un lugar especifico de la nube de puntos del teselado.
- Determinar a cual de los vertices del grafomata pertenece el vertice actual.
- Elegir una de las aristas existentes en el vertice actual. La cantidad de vertices siguientes depende del total de aristas disponibles en el vertice actual.
- Actualizar la ubicacion en el espacio y la identificacion del vertice hacia el cual se realiza el paso segun la arista elegida.

Si se conoce el grafomata del teselado, la ubicacion de al menos un vertice en ese espacio y la identificacion del vertice del grafomata sobre el cual se esta ubicado, ya se tiene acceso a la nube de puntos / vertices que configura el patron de teselado. La estructura del grafomata encausa las reubicaciones dentro de la segmentacion del espacio establecida por el teselado. La operacion de reubicacion implementada se presenta en el listado C.6.

Fuente en Pascal C.6: Paso entre vértices del grafomata

```
1 //sistema de direccionamiento para el cálculo de ubicaciones en teselación mediante su grafómata
2 //la información de dual y enlace va implicita en la primera asignación
3 //enrutamiento dentro del grafómata, elección de arista de movimiento siguiente arista[n>0]
4 //cada vértice del sistema tiene una arista estacionaria = arista[0]
5 function Tgrafomata.calcular_arista(camino:integer;nodo:arista_grafomata):arista_grafomata;
6 var
7 ubicacion:arista_grafomata;
8 begin
9   ubicacion:=definicion.vertice[nodo.enlace.vertice].arista[camino]; //siguiente vértice
10  ubicacion.direccion.punto.x:=nodo.direccion.punto.x+ubicacion.direccion.punto.x; //reubicación
11  ubicacion.direccion.punto.y:=nodo.direccion.punto.y+ubicacion.direccion.punto.y;
12  result:=ubicacion;
13 end;
```

C.3.7. Ejemplo de uso del objeto Tgrafomata

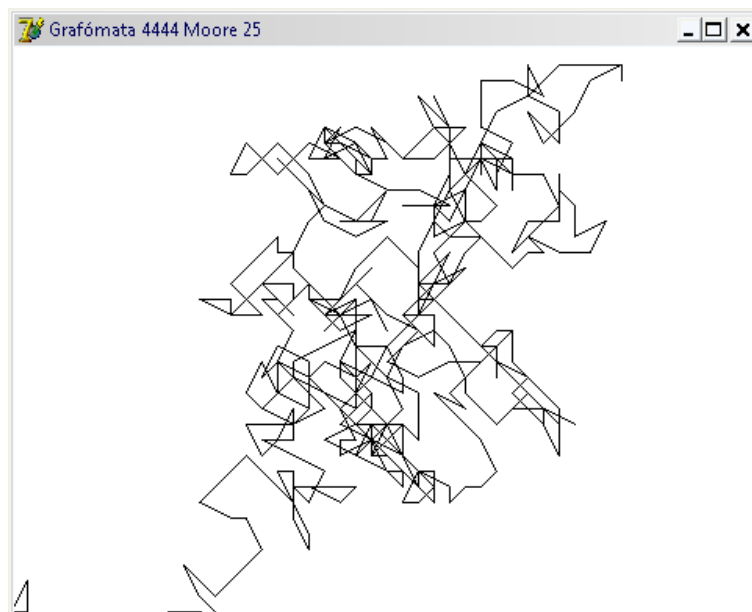
En el listado C.7 se presenta una prueba basica de uso del objeto Tgrafomata implementado en el prototipo PH2.exe. El ejemplo esta implementado en pascal, compilado con Delphi 7.0. Para recrearlo se necesita:

- Copiar en el directorio de proyecto las unidades definicion_grafomata.pas, definicion_grafomatas.pas, definicion_tipos.pas y mt19937.pas. mt19937 se incluye ya que es requerido por definicion_tipos.
- Incluir
en la seccion uses del formulario las unidades definicion_grafomata, definicion_grafomatas, definicion_tipos.
- Insertar en el formulario un objeto Timage, Asociar en el metodo OnClick del Timage la funcion OnClick mostrada en el listado C.7.

Al hacer click sobre el area del Timage, desde el centro de la ventana se muestra un camino de 500 pasos formado por la eleccion aleatoria de una arista del ultimo vertice visitado. El camino presentado pertenece a un grafomata elegido aleatoriamente. La función calcular_arista se presenta en el listado C.6 en la página 334. seleccionar_grafomata da acceso a la libreria de estructuras disponibles implementadas en la unidad definicion_grafomatas.

El codigo fuente del ejemplo y el ejecutable producido se encuentran en la carpeta contenido\anexos\Prototipo Homos 2.0\uso_grafomata.

Figura C.11: Ejecución de ejemplo de uso de Tgrafomata



Fuente: Autor.

Fuente en Pascal C.7: Ejemplo mínimo de uso de grafómatas

```
1 //actualizar la imagen en cada click
2 //Ejemplo mínimo de uso de grafómata
3 //se elige un grafómata
4 //se establece una ubicación válida dentro del teselado
5 //se calcula la ubicación de los demás vértices usando la función calcular_arista
6 //la dirección de movimiento se establece eligiendo una arista del vértice actual
7 procedure TForm1.OnClick(Sender: TObject);
8 var
9   g:Tgrafomata;
10  a:arista_grafomata;
11  x:integer;
12  p,m:Tpoint;
13 begin
14   //iniciar el canvas y establecer variables
15   image1.Canvas.brush.color:=clwhite;
16   image1.Canvas.Fillrect(image1.Canvas.ClipRect); //limpiar canvas
17   randomize; //renovar valores de aleatoriedad
18   m:=point(image1.Width div 2, image1.Height div 2); //punto medio del canvas
19   x:=0; //iniciar el contador de iteraciones
20
21   //g:=seleccionar_grafomata(random(27)).dual; //0..26 selección de duales
22   g:=seleccionar_grafomata(random(27)); //0..26 selección de bases
23   a:=g.estructura.vertice[0].arista[0]; //origen del teselado
24
25   form1.Caption:='Grafómata '+g.nombre;
26   image1.Canvas.PenPos:=sumar(a.direccion.punto,m); //MoveTo, ubicación inicial de dibujo
27   repeat
28
29     //determinar siguiente ubicación
30     a:=g.calcular_arista(random(high(g.estructura.vertice[ a.enlace.vertice ].arista)),a);
31
32     p:=a.direccion.punto; //ubicación de siguiente vértice
33     p:=point(p.X div 12,p.Y div 12); //escalado de la imagen, tamaño de arista = número mágico =
        120
34     p:=sumar(p,m); //reubicación respecto al centro de la imagen
35
36     image1.Canvas.LineTo(p.x,p.y); //dibuja y actualiza la ubicación actual
37     inc(x);
38     until x>=500;
39 end;
```

C.4. Estructuras implementadas

En esta sección se presentan todos los grafomatas implementados en el prototipo software Homos 2.0. Los grafomatas base o iniciales han sido construidos manualmente a partir de teselados de poligonos regulares, no regulares y la estructura funcional de los vecindarios de Moore. A partir de estos grafomatas iniciales se determinaron los sistemas duales.

Con el proposito de hacer explicita la forma y función de los grafomatas disponibles se presenta para cada uno:

- Una tabla que compara la estructura de vertices y aristas del grafomata y su dibugrafo asociado.
- Un patrón gráfico del grafomata y su dibugrafo en el cual se identifican los vertices internos en negro y los vertices hacia donde se dirigen las aristas en azul. Se evitó numerar las aristas para simplificar la representación grafica.
- El codigo fuente de la función que construye al grafomata base. No hay codigo fuente disponible para las estructuras duales ni para los distintos dibugrafos ya que estos son construidos automaticamente a partir de los grafomatas base disponibles.

El dual de los vecindarios de Moore es el grafomata de la malla 4.4.4.4. Aunque se implementan seis de estos vecindarios en el prototipo, solo se ilustran los tres primeros por razones prácticas. Al final de la seccion se incluye el codigo fuente de la funcion que genera tales estructuras.

La tabla C.1 define el formato usado para la presentacion de la tabla estructural que acompaña cada grafomata presentado. Esta tabla tiene las siguientes características:

- Para diferenciar la función del identificador de vértice dentro de la tabla, se escribe en mayúscula el nombre del vértice y en minúscula el identificador del vértice destino en cada arista dentro de la estructura.
- La arista estacionaria $a0$ se dirige al vértice origen y es antipoda de si misma o no tiene antipoda. Las demas aristas $a1...an$ son estructurales o conectivas, se dirigen a un vertice cualquiera del sistema y tienen una antipoda diferente de si misma. Los grafómatas que implementan los vecindarios de Moore incluyen además las aristas de conectividad al final de cada listado.
- Las aristas presentes en cada estructura reportan: Su identificacion particular en el vertice origen, la identificacion del vertice destino, la antipoda o la arista contraria presente el vertice destino y la ubicacion relativa entre los vertices destino y origen.
- Las ubicaciones de las aristas se calculan usando un *número mágico*, cuyo valor es 120^7 . Se eligió este numero ya que demostró reducir la cantidad de errores de redondeo que se presentan durante la construccion de los duales de los grafomatas. Este numero es algo provisional, se espera hacer obsoleto el uso de numeros magicos en proximas iteraciones del prototipo.

⁷sus divisores son: 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60 y 120.

- Los angulos entre las aristas pueden determinarse a partir de las ubicaciones dadas. Los valores obtenidos generalmente serán solo aproximaciones, esto se debe al uso de valores enteros para las ubicaciones.
- En cada caso y cuando fue posible se organizó el orden de los vertices en cada estructura según la cantidad de aristas presentes y por mejoras en la legibilidad.

Los datos reportados dan fundamento la tabla C.2 en la página 407. Estos valores permitieron establecer la caracterizacion de grafomatas presentada en la sección C.5 en la página 404.

Los grafomatas se listan segun el orden establecido en el menú principal del prototipo PH2.exe.

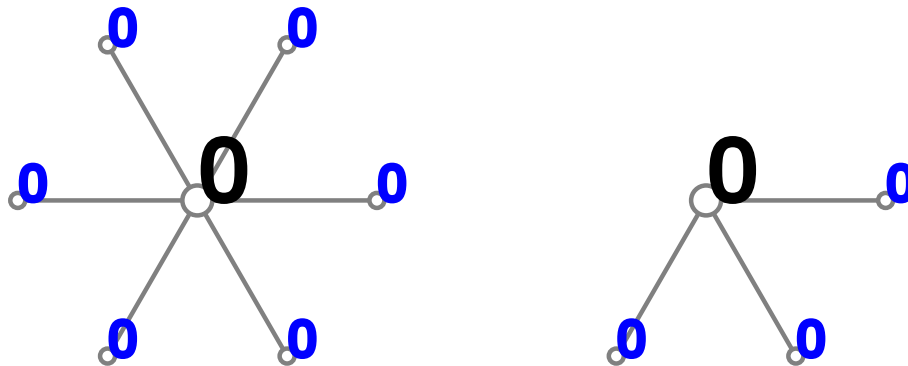
Tabla C.1: Estructura de grafómata y dibúgrafo asociado

Estructura de grafómata				Estructura de dibúgrafo			
Vértice origen V_p	Información relacionada			Vértice origen V_q	Información relacionada		
Poda	Vértice destino	Antipoda	Ubicación respecto al vértice	Poda	Vértice destino	Antipoda	Ubicación respecto al vértice
a0	v0	a0	(0,0)	a1	v0	z1	(x_a, y_a)
a1	va	z1	(x_1, y_1)	a2	va	z2	(x_b, y_b)
a2	vb	z2	(x_2, y_2)	a3	vb	z3	(x_c, y_c)
$\vdots$				$\vdots$			
an	vj	zn	(x_n, y_n)	an	vk	zm	(x_k, y_k)

C.4.1. Grafómata 3.3.3.3.3, malla de triángulos equiláteros

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v0	a4	(120,0)
a1	v0	a4	(120,0)	a2	v0	a5	(60,104)
a2	v0	a5	(60,104)	a3	v0	a6	(-60,104)
a3	v0	a6	(-60,104)				
a4	v0	a1	(-120,0)				
a5	v0	a2	(-60,-104)				
a6	v0	a3	(60,-104)				

Figura C.12: Patrón gráfico del grafómata 3.3.3.3.3



Fuente en Pascal C.8: 3.3.3.3.3

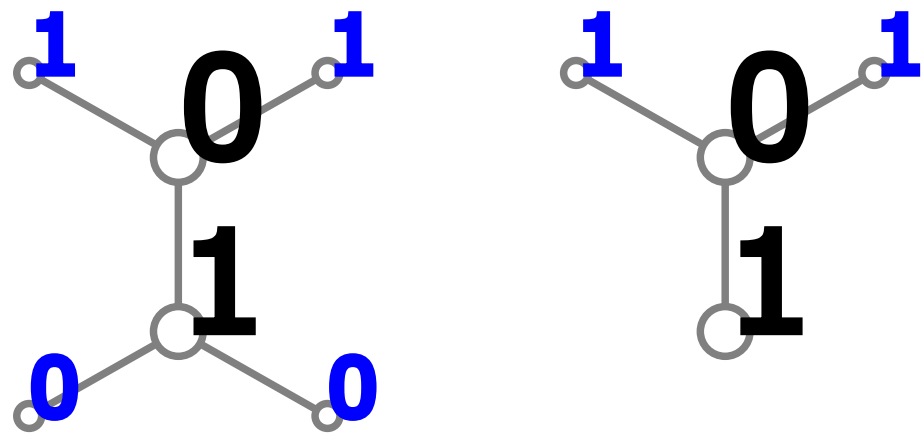
```

1 function Regular1:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   g:=Tgrafomata.create;
8   with g do
9     begin
10      nombre:='333333';
11      agregar_vertice; //v0
12      p:=point(numero_magico,0); agregar_arista(vertice_reciente,p);
13      for i:=1 to 5 do agregar_rotacion_arista(vertice_reciente,i,vertice_reciente,60);
14      construir_grafomata;
15    end;
16    result:=g;
17 end;
```

C.4.2. Grafómata dual 3.3.3.3.3

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a3	(0,70)
a1	v1	a3	(0,70)	a1	v0	a2	(60,34)	a2	v1	a1	(-60,-34)
a2	v1	a1	(-60,-34)	a2	v0	a3	(-60,34)	a3	v1	a2	(60,-34)
a3	v1	a2	(60,-34)	a3	v0	a1	(0,-70)	V1			

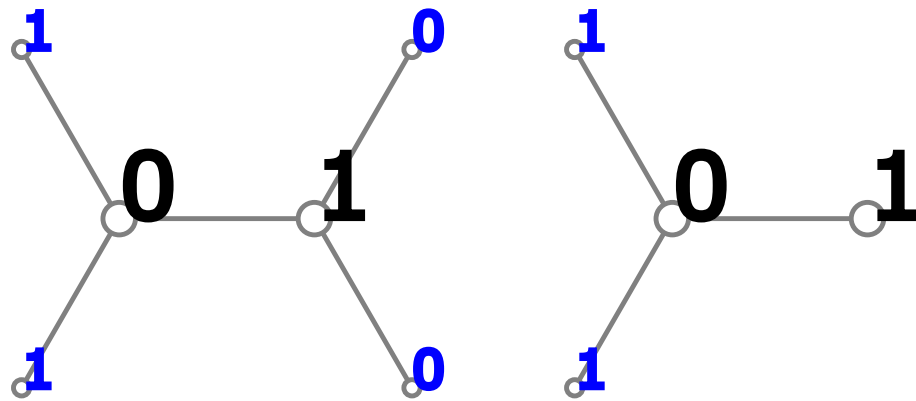
Figura C.13: Patrón gráfico del grafómata dual 3.3.3.3.3



C.4.3. Grafómata 6.6.6, malla de hexágonos

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a2	(120,0)
a1	v1	a2	(120,0)	a1	v0	a3	(60,104)	a2	v1	a3	(-60,104)
a2	v1	a3	(-60,104)	a2	v0	a1	(-120,0)	a3	v1	a1	(-60,-104)
a3	v1	a1	(-60,-104)	a3	v0	a2	(60,-104)	V1			

Figura C.14: Patrón gráfico del grafómata 6.6.6



Fuente en Pascal C.9: 6.6.6

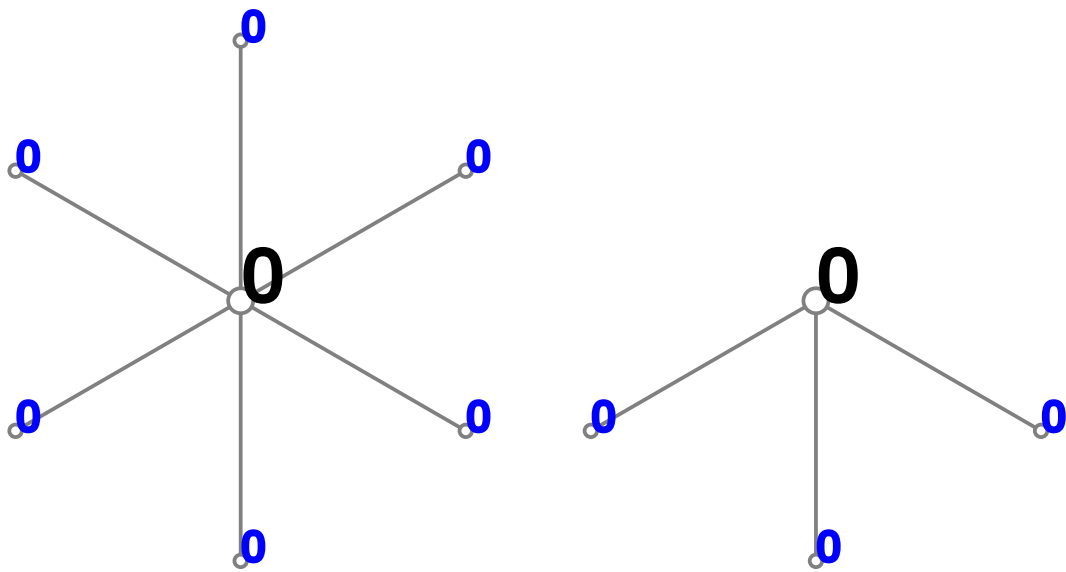
```

1 function Regular3:Tgrafomata;
2 var
3   g:Tgrafomata;
4   p:tpoint;
5 begin
6   g:=Tgrafomata.create;
7   with g do
8     begin
9       nombre:='666';
10      agregar_vertice; //v0
11      p:=point(numero_magico,0);  agregar_arista(1,p);
12      agregar_rotacion_arista(0,arista_reciente,1,120);
13      agregar_rotacion_arista(0,arista_reciente,1,120);
14      agregar_reflexion_vertice(0,0,reflejar_en_xy); //v1, 180° de rotación de las aristas de v0
15      construir_grafomata;
16    end;
17    result:=g;
18 end;
```

C.4.4. Grafómata dual 666

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v0	a4	(180,104)
a1	v0	a4	(180,104)	a2	v0	a5	(0,208)
a2	v0	a5	(0,208)	a3	v0	a6	(-180,104)
a3	v0	a6	(-180,104)				
a4	v0	a1	(-180,-104)				
a5	v0	a2	(0,-208)				
a6	v0	a3	(180,-104)				

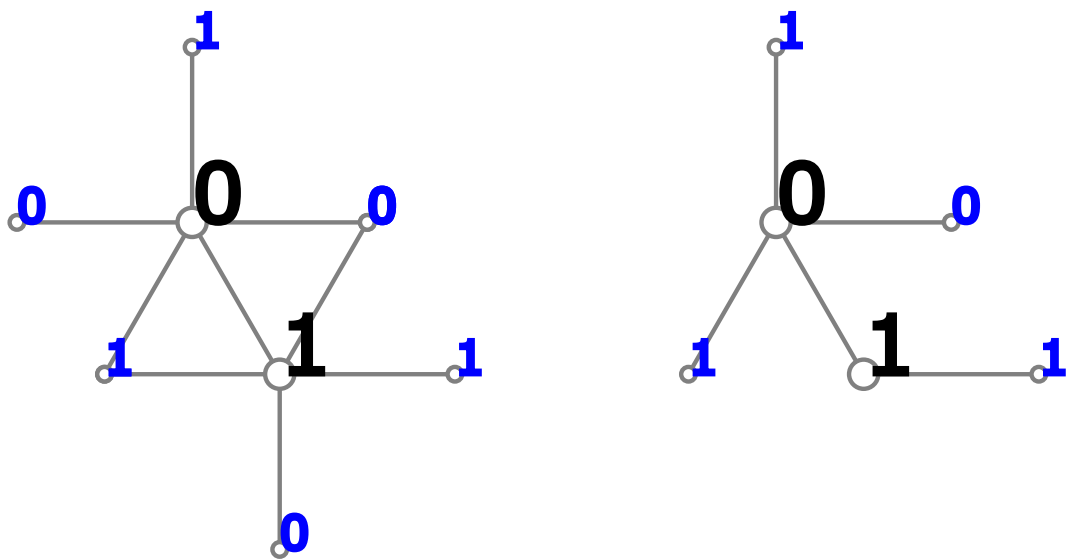
Figura C.15: Patrón gráfico del grafómata dual 6.6.6



C.4.5. Grafómata 3.3.3.4.4

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v0	a4	(120,0)
a1	v0	a4	(120,0)	a1	v1	a3	(120,0)	a2	v1	a4	(60,104)
a2	v1	a4	(60,104)	a2	v0	a5	(0,120)	a3	v1	a5	(-60,104)
a3	v1	a5	(-60,104)	a3	v1	a1	(-120,0)	a5	v1	a2	(0,-120)
a4	v0	a1	(-120,0)	a4	v0	a2	(-60,-104)	V1			
a5	v1	a2	(0,-120)	a5	v0	a3	(60,-104)				

Figura C.16: Patrón gráfico del grafómata



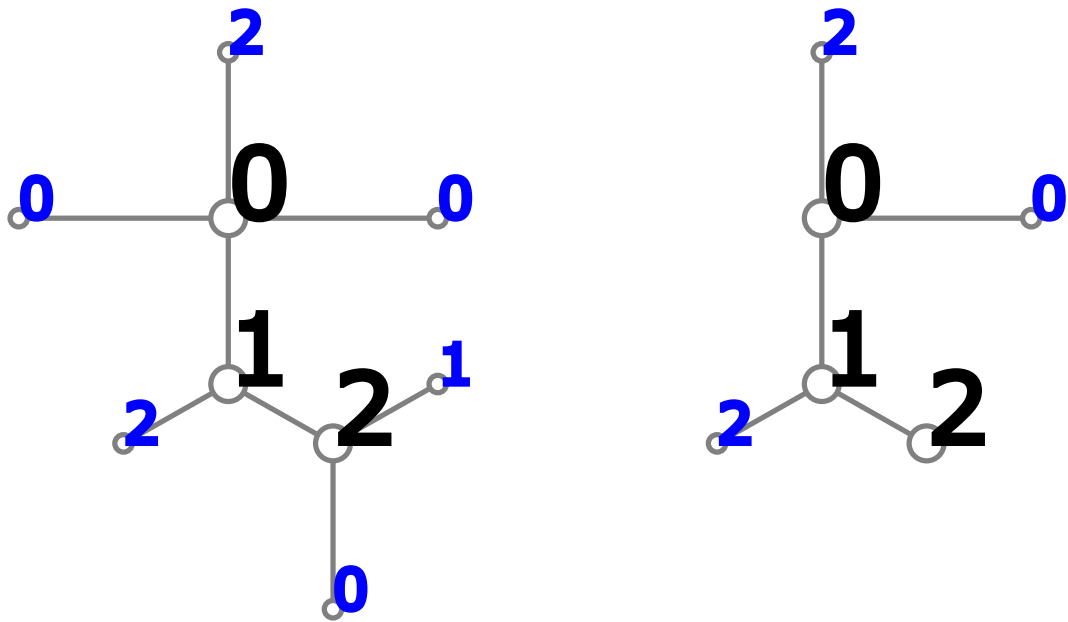
Fuente en Pascal C.10: 3.3.3.4.4

```
1 function Semiregular1:Tgrafomata;
2 var
3   g:Tgrafomata;
4   a,b,c,d,e:tpoint;
5 begin
6   //rotaciones de aristas
7   a:=point(numero_magico,0);
8   b:=rotar(60,a);
9   c:=rotar(60,b);
10  d:=rotar(60,c);
11  e:=rotar(90,d);
12  g:=Tgrafomata.create;
13  with g do
14    begin
15      nombre:='33344';
16      agregar_vertice; //v0
17      agregar_arista(0,a);
18      agregar_arista(1,b);
19      agregar_arista(1,c);
20      agregar_arista(0,d);
21      agregar_arista(1,e);
22      agregar_vertice; //v1
23      agregar_arista(1,reflejar_y(a));
24      agregar_arista(0,reflejar_y(b));
25      agregar_arista(0,reflejar_y(c));
26      agregar_arista(1,reflejar_y(d));
27      agregar_arista(0,reflejar_y(e));
28      construir_grafomata;
29    end;
30    result:=g;
31 end;
```

C.4.6. Grafómata dual 3.3.3.4.4

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v0	a3	(120,0)
a1	v0	a3	(120,0)	a2	v1	a3	(0,95)
a2	v1	a3	(0,95)	a4	v2	a1	(0,-95)
a3	v0	a1	(-120,0)	V1			
a4	v2	a1	(0,-95)	a1	v2	a2	(60,34)
V1				a2	v2	a3	(-60,34)
a0	v1	a0	(0,0)	V2			
a1	v2	a2	(60,34)	a0	v2	a0	(0,0)
a2	v2	a3	(-60,34)	a1	v0	a4	(0,95)
a3	v0	a2	(0,-95)	a2	v1	a1	(-60,-34)
				a3	v1	a2	(60,-34)

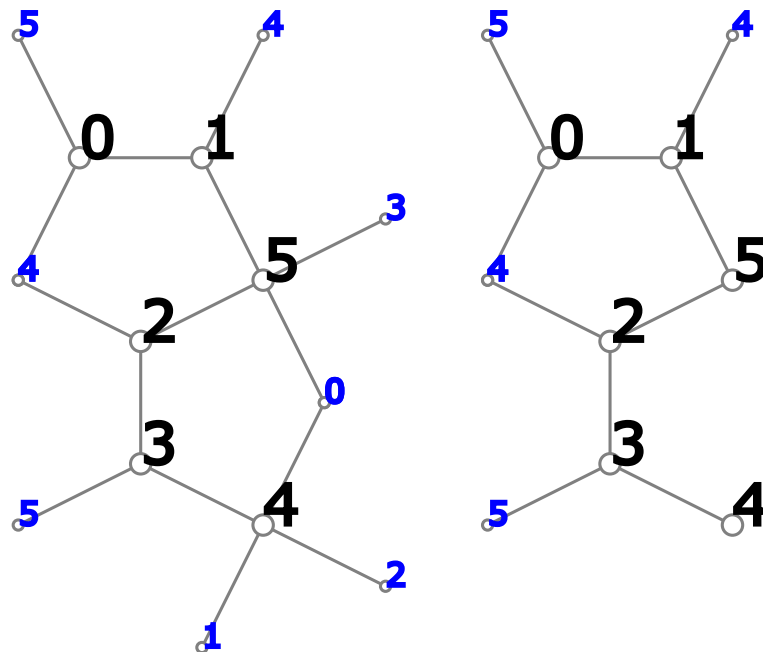
Figura C.17: Patrón gráfico del grafómata dual 3.3.3.4.4



C.4.7. Grafómata el Cairo

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a2	(120,0)
a1	v1	a2	(120,0)	a1	v5	a3	(60,120)	a2	v4	a4	(-60,120)
a2	v4	a4	(-60,120)	a2	v0	a1	(-120,0)	a3	v5	a1	(-60,-120)
a3	v5	a1	(-60,-120)	a3	v4	a2	(60,-120)	V1			
V2				V3				a1	v5	a3	(60,120)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v4	a2	(60,-120)
a1	v3	a3	(0,120)	a1	v4	a3	(120,60)	V2			
a2	v4	a1	(-120,-60)	a2	v5	a4	(-120,60)	a1	v3	a3	(0,120)
a3	v5	a2	(120,-60)	a3	v2	a1	(0,-120)	a2	v4	a1	(-120,-60)
V4				V5				a3	v5	a2	(120,-60)
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	V3			
a1	v2	a2	(120,60)	a1	v0	a3	(60,120)	a1	v4	a3	(120,60)
a2	v1	a3	(-60,120)	a2	v2	a3	(-120,60)	a2	v5	a4	(-120,60)
a3	v3	a1	(-120,-60)	a3	v1	a1	(-60,-120)	V4			
a4	v0	a2	(60,-120)	a4	v3	a2	(120,-60)	V5			

Figura C.18: Patrón gráfico del grafómata el Cairo



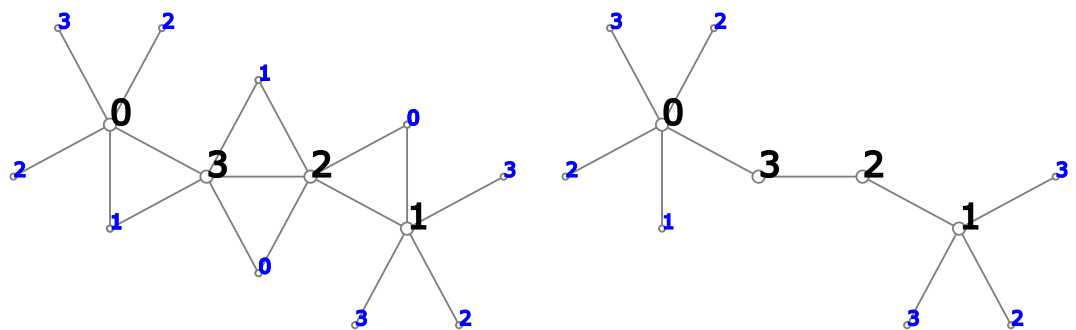
Fuente en Pascal C.11: El Cairo

```
1 function Semiregular2:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   a,b,c,d,e,f,s,t:tpoint;
6 begin
7   //rotaciones de aristas
8   i:=numero_magico;
9   a:=point(i,0);
10  b:=point(-i div 2,-i);
11  c:=reflejar_y(b);
12  d:=reflejar_x(a);
13  e:=reflejar_x(b);
14  f:=reflejar_x(c);
15  g:=Tgrafomata.create;
16  with g do
17    begin
18      nombre:='Cairo';
19      agregar_vertice; //v0
20      agregar_arista(1,a);
21      agregar_arista(5,b);
22      agregar_arista(4,c);
23      agregar_vertice; //v1
24      agregar_arista(0,d);
25      agregar_arista(4,e);
26      agregar_arista(5,f);
27      s:=rotar90d(b);
28      t:=rotar90d(c);
29      agregar_vertice; //v2
30      agregar_arista(3,rotar90d(a));
31      agregar_arista(5,s);
32      agregar_arista(4,t);
33      agregar_vertice; //v3
34      agregar_arista(2,rotar90d(d));
35      agregar_arista(4,rotar90d(e));
36      agregar_arista(5,rotar90d(f));
37      agregar_vertice; //v4
38      agregar_arista(1,c);
39      agregar_arista(3,t);
40      agregar_arista(0,reflejar_xy(c));
41      agregar_arista(2,reflejar_xy(t));
42      agregar_vertice; //v5
43      agregar_arista(0,f);
44      agregar_arista(3,s);
45      agregar_arista(1,reflejar_xy(f));
46      agregar_arista(2,reflejar_xy(s));
47      construir_grafomata;
48    end;
49    result:=g;
50 end;
```

C.4.8. Grafómata dual el cairo, 3.3.4.3.4

Estructura de grafómata				Estructura de dibúgrafo			
V0				V1			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)
a1	v3	a4	(156,84)	a1	v2	a4	(84,156)
a2	v1	a4	(0,168)	a2	v3	a5	(-84,156)
a3	v2	a5	(-156,84)	a3	v2	a1	(-156,-84)
a4	v3	a2	(-84,-156)	a4	v0	a2	(0,-168)
a5	v2	a2	(84,-156)	a5	v3	a3	(156,-84)
V2				V3			
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)
a1	v1	a3	(156,84)	a1	v2	a3	(168,0)
a2	v0	a5	(-84,156)	a2	v0	a4	(84,156)
a3	v3	a1	(-168,0)	a3	v1	a5	(-156,84)
a4	v1	a1	(-84,-156)	a4	v0	a1	(-156,-84)
a5	v0	a3	(156,-84)	a5	v1	a2	(84,-156)
				V0			
				a1	v3	a4	(156,84)
				a2	v1	a4	(0,168)
				a3	v2	a5	(-156,84)
				a4	v3	a2	(-84,-156)
				a5	v2	a2	(84,-156)
				V1			
				a1	v2	a4	(84,156)
				a2	v3	a5	(-84,156)
				a3	v2	a1	(-156,-84)
				a5	v3	a3	(156,-84)
				V2			
				a3	v3	a1	(-168,0)
				V3			

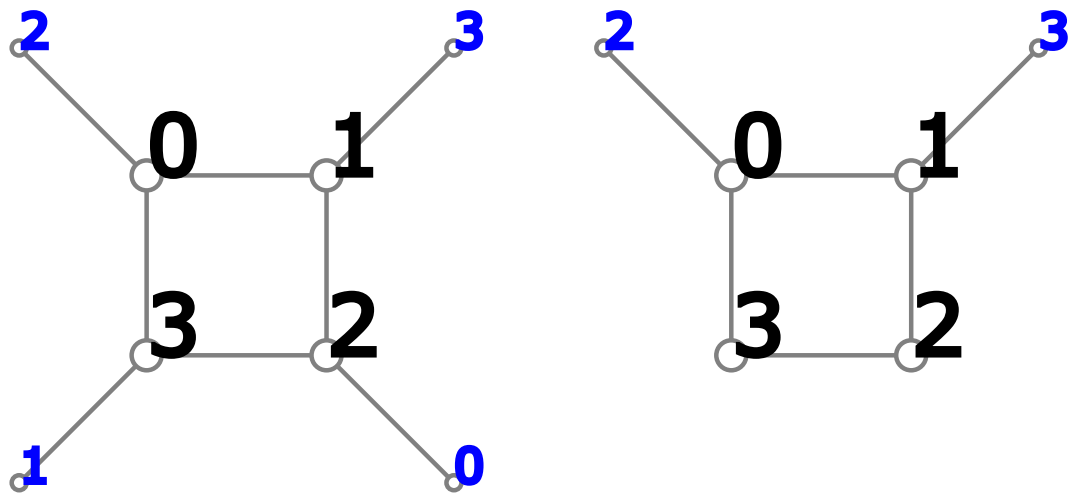
Figura C.19: Patrón gráfico del grafómata dual 3.3.4.3.4



C.4.9. Grafómata 4.8.8

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a2	(120,0)
a1	v1	a2	(120,0)	a1	v2	a3	(0,120)	a2	v3	a3	(0,120)
a2	v3	a3	(0,120)	a2	v0	a1	(-120,0)	a3	v2	a1	(-85,-85)
a3	v2	a1	(-85,-85)	a3	v3	a2	(85,-85)	V1			
V2				V3				a1	v2	a3	(0,120)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v3	a2	(85,-85)
a1	v0	a3	(85,85)	a1	v2	a2	(120,0)	V2			
a2	v3	a1	(-120,0)	a2	v1	a3	(-85,85)	a2	v3	a1	(-120,0)
a3	v1	a1	(0,-120)	a3	v0	a2	(0,-120)	V3			

Figura C.20: Patrón gráfico del grafómata 4.8.8



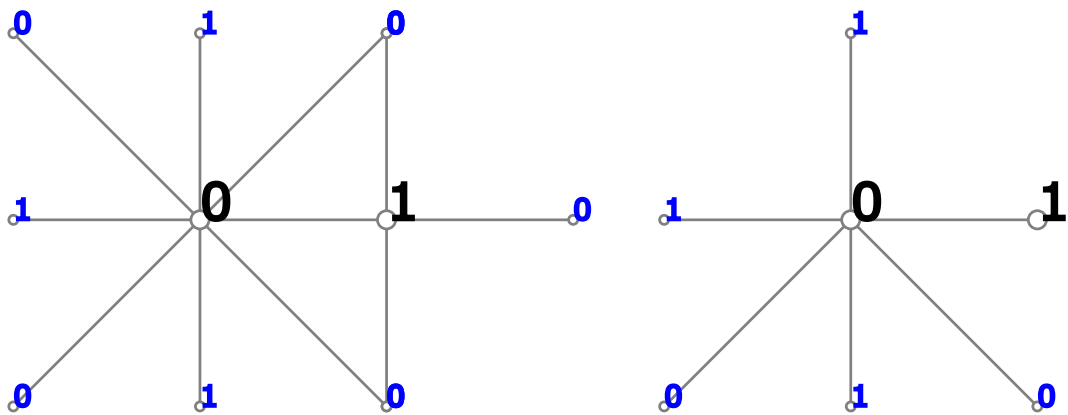
Fuente en Pascal C.12: 4.8.8

```
1 function Semiregular3(angulo:integer):Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   i:=numero_magico;
8   g:=Tgrafomata.create;
9   with g do
10    begin
11     nombre:='488_'+inttostr(angulo);
12     agregar_vertice; //v0
13     p:=point( i, 0); agregar_arista(1,p);
14     p:=point( 0, i); agregar_arista(3,p);
15     p:=rotar(angulo,p); agregar_arista(2,p); //135 forma normal. 45 y 225 = estrella cuatro puntas +
        cuadrados
16     agregar_vertice; //v1
17     agregar_reflexion_arista(0,1,2,rotar_90d);
18     agregar_reflexion_arista(0,2,0,rotar_90d);
19     agregar_reflexion_arista(0,3,3,rotar_90d);
20     agregar_vertice; //v2
21     agregar_reflexion_arista(1,1,3,rotar_90d);
22     agregar_reflexion_arista(1,2,1,rotar_90d);
23     agregar_reflexion_arista(1,3,0,rotar_90d);
24     agregar_vertice; //v3
25     agregar_reflexion_arista(2,1,0,rotar_90d);
26     agregar_reflexion_arista(2,2,2,rotar_90d);
27     agregar_reflexion_arista(2,3,1,rotar_90d);
28     construir_grafomata;
29     end;
30     result:=g;
31 end;
```

C.4.10. Grafómata dual 4.8.8

Estructura de grafómata				Estructura de dibúgrafo			
V0				V1			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)
a1	v1	a3	(205,0)	a1	v0	a5	(205,0)
a2	v0	a6	(205,205)	a2	v0	a7	(0,205)
a3	v1	a4	(0,205)	a3	v0	a1	(-205,0)
a4	v0	a8	(-205,205)	a4	v0	a3	(0,-205)
a5	v1	a1	(-205,0)				
a6	v0	a2	(-205,-205)				
a7	v1	a2	(0,-205)				
a8	v0	a4	(205,-205)				
				V0			
a1	v1	a3	(205,0)	a1	v1	a3	(205,0)
a2	v0	a6	(205,205)	a2	v0	a6	(205,205)
a3	v1	a4	(0,205)	a3	v1	a4	(0,205)
a4	v0	a8	(-205,205)	a4	v0	a8	(-205,205)
a5	v1	a1	(-205,0)	a5	v1	a1	(-205,0)
a7	v1	a2	(0,-205)	a7	v1	a2	(0,-205)
				V1			

Figura C.21: Patrón gráfico del grafómata dual 4.8.8



C.4.11. Grafómata 3.6.3.6

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v2	a3	(120,0)
a1	v2	a3	(120,0)	a1	v0	a4	(60,104)	a2	v1	a3	(60,104)
a2	v1	a3	(60,104)	a2	v2	a4	(-60,104)	a3	v2	a1	(-120,0)
a3	v2	a1	(-120,0)	a3	v0	a2	(-60,-104)	a4	v1	a1	(-60,-104)
a4	v1	a1	(-60,-104)	a4	v2	a2	(60,-104)	V1			
V2											
a0	v2	a0	(0,0)								
a1	v0	a3	(120,0)								
a2	v1	a4	(-60,104)								
a3	v0	a1	(-120,0)								
a4	v1	a2	(60,-104)								
				</							

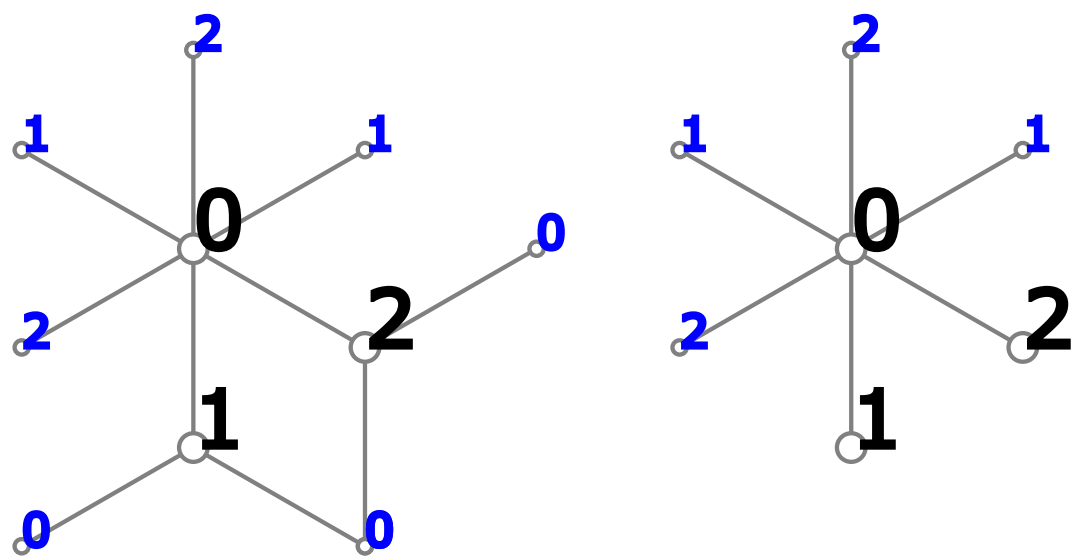
Fuente en Pascal C.13: 3.6.3.6

```
1 function Semiregular4:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   a,b,c,d:tpoint;
6 begin
7   i:=numero_magico;
8   //rotaciones de aristas
9   a:=point(i,0);
10  b:=reflejar_xy(a);
11  c:=rotar(60,a);
12  d:=reflejar_xy(c);
13  g:=Tgrafomata.create;
14  with g do
15    begin
16      nombre:='3636';
17      agregar_vertice; //v0
18      agregar_arista(2,a);
19      agregar_arista(2,b);
20      agregar_arista(1,c);
21      agregar_arista(1,d);
22      agregar_vertice; //v1
23      a:=rotar(60,a); agregar_arista(0,a);
24      b:=rotar(60,b); agregar_arista(0,b);
25      c:=rotar(60,c); agregar_arista(2,c);
26      d:=rotar(60,d); agregar_arista(2,d);
27      agregar_vertice; //v2
28      a:=rotar(60,a); agregar_arista(1,a);
29      b:=rotar(60,b); agregar_arista(1,b);
30      c:=rotar(60,c); agregar_arista(0,c);
31      d:=rotar(60,d); agregar_arista(0,d);
32      construir_grafomata;
33    end;
34    result:=g;
35 end;
```

C.4.12. Grafómata dual 3.6.3.6

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v2	a2	(120,69)
a1	v2	a2	(120,69)	a2	v1	a3	(0,139)
a2	v1	a3	(0,139)	a3	v2	a3	(-120,69)
a3	v2	a3	(-120,69)	a4	v1	a1	(-120,-69)
a4	v1	a1	(-120,-69)	a5	v2	a1	(0,-139)
a5	v2	a1	(0,-139)	a6	v1	a2	(120,-69)
a6	v1	a2	(120,-69)	V1			
V1				V2			
a0	v1	a0	(0,0)	a0	v2	a0	(0,0)
a1	v0	a4	(120,69)	a1	v0	a5	(0,139)
a2	v0	a6	(-120,69)	a2	v0	a1	(-120,-69)
a3	v0	a2	(0,-139)	a3	v0	a3	(120,-69)

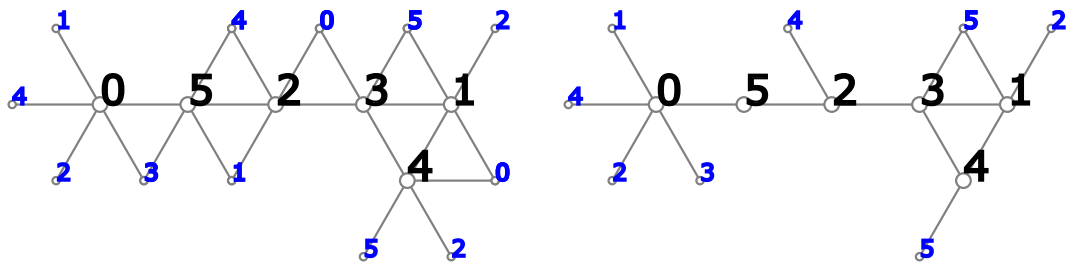
Figura C.23: Patrón gráfico del grafómata dual 3.6.3.6



C.4.13. Grafómata 3.3.3.3.6

Estructura de grafómata								Estructura de dibúgrafo			
V0				V3				V0			
a0	v0	a0	(0,0)	a0	v3	a0	(0,0)	a1	v5	a4	(120,0)
a1	v5	a4	(120,0)	a1	v1	a3	(120,0)	a2	v3	a4	(60,104)
a2	v3	a4	(60,104)	a2	v4	a4	(60,104)	a3	v2	a5	(-60,104)
a3	v2	a5	(-60,104)	a3	v2	a1	(-120,0)	a4	v4	a1	(-120,0)
a4	v4	a1	(-120,0)	a4	v0	a2	(-60,-104)	a5	v1	a1	(-60,-104)
a5	v1	a1	(-60,-104)	a5	v5	a3	(60,-104)	V1			
V1				V4				a2	v4	a5	(-60,104)
a0	v1	a0	(0,0)	a0	v4	a0	(0,0)	a3	v3	a1	(-120,0)
a1	v0	a5	(60,104)	a1	v0	a4	(120,0)	a4	v5	a2	(-60,-104)
a2	v4	a5	(-60,104)	a2	v2	a4	(60,104)	a5	v2	a2	(60,-104)
a3	v3	a1	(-120,0)	a3	v5	a5	(-60,104)	V2			
a4	v5	a2	(-60,-104)	a4	v3	a2	(-60,-104)	a1	v3	a3	(120,0)
a5	v2	a2	(60,-104)	a5	v1	a2	(60,-104)	a3	v5	a1	(-120,0)
V2				V5				a4	v4	a2	(-60,-104)
a0	v2	a0	(0,0)	a0	v5	a0	(0,0)	V3			
a1	v3	a3	(120,0)	a1	v2	a3	(120,0)	a2	v4	a4	(60,104)
a2	v1	a5	(-60,104)	a2	v1	a4	(60,104)	a5	v5	a3	(60,-104)
a3	v5	a1	(-120,0)	a3	v3	a5	(-60,104)	V4			
a4	v4	a2	(-60,-104)	a4	v0	a1	(-120,0)	a3	v5	a5	(-60,104)
a5	v0	a3	(60,-104)	a5	v4	a3	(60,-104)	V5			

Figura C.24: Patrón gráfico del grafómata 3.3.3.3.6



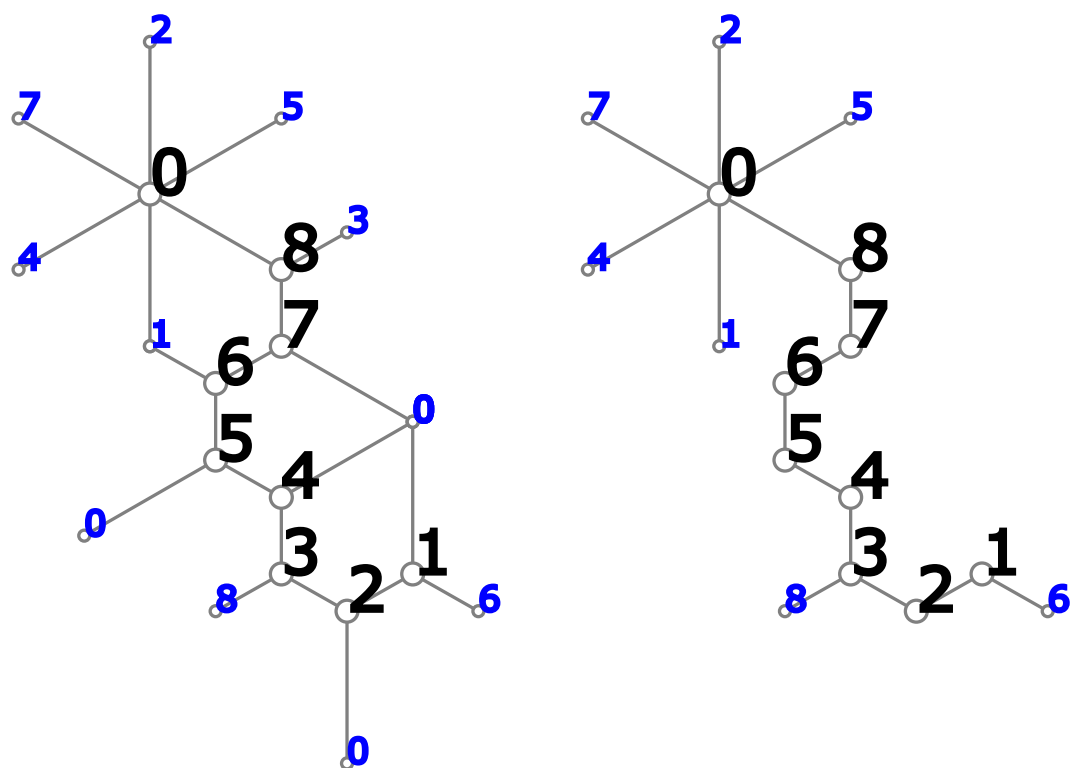
Fuente en Pascal C.14: 3.3.3.3.6

```
1 function Semiregular5:Tgrafomata;
2 var
3   g:Tgrafomata;
4   a,b,c,d,e,f:tpoint;
5 begin
6   //rotaciones de aristas
7   a:=point(numero_magico,0);
8   b:=rotar(60,a);
9   c:=rotar(60,b);
10  d:=rotar(60,c);
11  e:=rotar(60,d);
12  f:=rotar(60,e);
13  g:=Tgrafomata.create;
14  with g do
15    begin
16      nombre:='33336';
17      agregar_vertice; //v0
18      agregar_arista(5,a);
19      agregar_arista(3,b);
20      agregar_arista(2,c);
21      agregar_arista(4,d);
22      agregar_arista(1,e);
23      agregar_vertice; //v1
24      agregar_arista(0,b);
25      agregar_arista(4,c);
26      agregar_arista(3,d);
27      agregar_arista(5,e);
28      agregar_arista(2,f);
29      agregar_vertice; //v2
30      agregar_arista(1,c);
31      agregar_arista(5,d);
32      agregar_arista(4,e);
33      agregar_arista(0,f);
34      agregar_arista(3,a);
35      agregar_vertice; //v3
36      agregar_arista(2,d);
37      agregar_arista(0,e);
38      agregar_arista(5,f);
39      agregar_arista(1,a);
40      agregar_arista(4,b);
41      agregar_vertice; //v4
42      agregar_arista(3,e);
43      agregar_arista(1,f);
44      agregar_arista(0,a);
45      agregar_arista(2,b);
46      agregar_arista(5,c);
47      agregar_vertice; //v5
48      agregar_arista(4,f);
49      agregar_arista(2,a);
50      agregar_arista(1,b);
51      agregar_arista(3,c);
52      agregar_arista(0,d);
53      construir_grafomata;
54    end;
55    result:=g;
56 end;
```

C.4.14. Grafómata dual 3.3.3.3.6

Estructura de grafómata								Estructura de dibúgrafo							
V0								V0							
a0	v0	a0	(0,0)					a1	v8	a2	(120,69)				
a1	v8	a2	(120,69)					a2	v1	a3	(0,139)				
a2	v1	a3	(0,139)					a3	v4	a3	(-120,69)				
a3	v4	a3	(-120,69)					a4	v7	a1	(-120,-69)				
a4	v7	a1	(-120,-69)					a5	v2	a1	(0,-139)				
a5	v2	a1	(0,-139)					a6	v5	a2	(120,-69)				
a6	v5	a2	(120,-69)					V1							
V1								V2							
a0	v1	a0	(0,0)	a0	v2	a0	(0,0)	a1	v6	a2	(60,34)				
a1	v6	a2	(60,34)	a1	v0	a5	(0,139)	a2	v2	a3	(-60,34)				
a2	v2	a3	(-60,34)	a2	v3	a1	(-60,-34)	V2							
a3	v0	a2	(0,-139)	a3	v1	a2	(60,-34)	a2	v3	a1	(-60,-34)				
V3								V3							
a0	v3	a0	(0,0)	a0	v4	a0	(0,0)	a2	v8	a3	(-60,34)				
a1	v2	a2	(60,34)	a1	v3	a3	(0,70)	a3	v4	a1	(0,-70)				
a2	v8	a3	(-60,34)	a2	v5	a1	(-60,-34)	V4							
a3	v4	a1	(0,-70)	a3	v0	a3	(120,-69)	a2	v5	a1	(-60,-34)				
V5								V5							
a0	v5	a0	(0,0)	a0	v6	a0	(0,0)	a3	v6	a1	(0,-70)				
a1	v4	a2	(60,34)	a1	v5	a3	(0,70)	V6							
a2	v0	a6	(-120,69)	a2	v1	a1	(-60,-34)	a3	v7	a2	(60,-34)				
a3	v6	a1	(0,-70)	a3	v7	a2	(60,-34)	V7							
V7								V8							
a0	v7	a0	(0,0)	a0	v8	a0	(0,0)	a3	v8	a1	(0,-70)				
a1	v0	a4	(120,69)	a1	v7	a3	(0,70)	V8							
a2	v6	a3	(-60,34)	a2	v0	a1	(-120,-69)								
a3	v8	a1	(0,-70)	a3	v3	a2	(60,-34)								

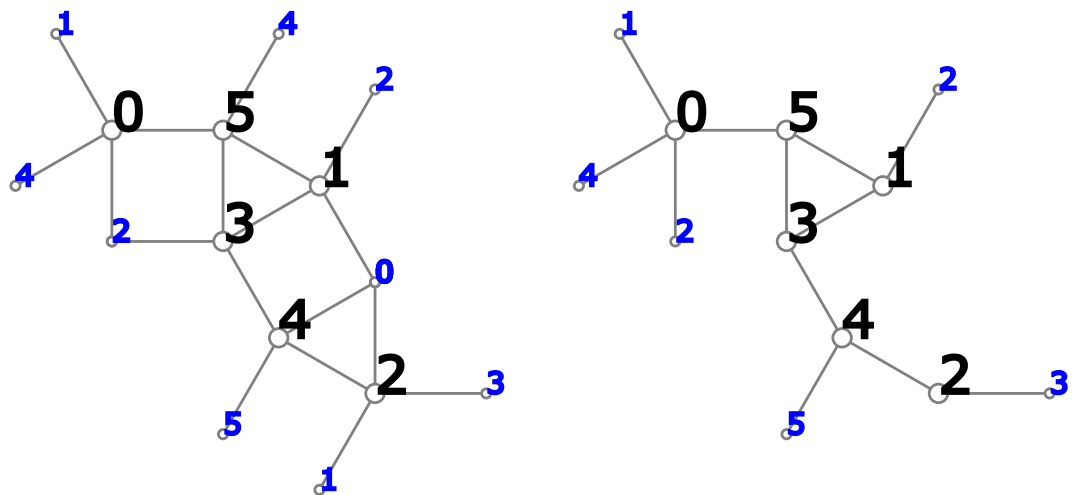
Figura C.25: Patrón gráfico del grafómata dual 3.3.3.3.6



C.4.15. Grafómata 3.4.6.4

Estructura de grafómata								Estructura de dibúgrafo			
V0				V3				V0			
a0	v0	a0	(0,0)	a0	v3	a0	(0,0)	a1	v5	a3	(120,0)
a1	v5	a3	(120,0)	a1	v4	a3	(60,104)	a2	v2	a4	(0,120)
a2	v2	a4	(0,120)	a2	v2	a1	(-120,0)	a3	v4	a4	(-104,60)
a3	v4	a4	(-104,60)	a3	v5	a2	(0,-120)	a4	v1	a1	(-60,-104)
a4	v1	a1	(-60,-104)	a4	v1	a2	(104,-60)	V1			
V1				V4				a2	v3	a4	(-104,60)
a0	v1	a0	(0,0)	a0	v4	a0	(0,0)	a3	v5	a1	(-104,-60)
a1	v0	a4	(60,104)	a1	v2	a3	(104,60)	a4	v2	a2	(60,-104)
a2	v3	a4	(-104,60)	a2	v5	a4	(-60,104)	V2			
a3	v5	a1	(-104,-60)	a3	v3	a1	(-60,-104)	a1	v3	a2	(120,0)
a4	v2	a2	(60,-104)	a4	v0	a3	(104,-60)	a3	v4	a1	(-104,-60)
V2				V5				V3			
a0	v2	a0	(0,0)	a0	v5	a0	(0,0)	a1	v4	a3	(60,104)
a1	v3	a2	(120,0)	a1	v1	a3	(104,60)	a3	v5	a2	(0,-120)
a2	v1	a4	(-60,104)	a2	v3	a3	(0,120)	V4			
a3	v4	a1	(-104,-60)	a3	v0	a1	(-120,0)	a2	v5	a4	(-60,104)
a4	v0	a2	(0,-120)	a4	v4	a2	(60,-104)	V5			

Figura C.26: Patrón gráfico del grafómata 3.4.6.4



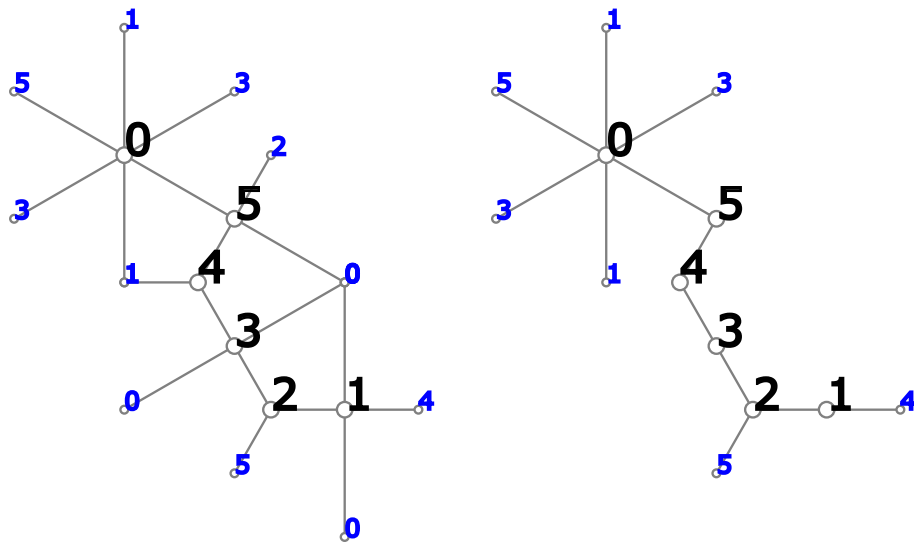
Fuente en Pascal C.15: 3.4.6.4

```
1 function Semiregular6:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   a,b,c,d:tpoint;
6 begin
7   i:=numero_magico;
8   //rotaciones de aristas
9   a:=point(i, 0);
10  b:=point( 0,i);
11  c:=rotar(60,b);
12  d:=rotar(90,c);
13  g:=Tgrafomata.create;
14  with g do
15    begin
16      nombre:='3464';
17      agregar_vertice; //v0
18      agregar_arista(5,a);
19      agregar_arista(2,b);
20      agregar_arista(4,c);
21      agregar_arista(1,d);
22      agregar_vertice; //v1
23      agregar_arista(0,rotar(60,a));
24      agregar_arista(3,rotar(60,b));
25      agregar_arista(5,rotar(60,c));
26      agregar_arista(2,rotar(60,d));
27      agregar_vertice; //v2
28      agregar_arista(1,rotar(120,a));
29      agregar_arista(4,rotar(120,b));
30      agregar_arista(0,rotar(120,c));
31      agregar_arista(3,rotar(120,d));
32      agregar_vertice; //v3
33      agregar_arista(2,rotar(180,a));
34      agregar_arista(5,rotar(180,b));
35      agregar_arista(1,rotar(180,c));
36      agregar_arista(4,rotar(180,d));
37      agregar_vertice; //v4
38      agregar_arista(3,rotar(240,a));
39      agregar_arista(0,rotar(240,b));
40      agregar_arista(2,rotar(240,c));
41      agregar_arista(5,rotar(240,d));
42      agregar_vertice; //v5
43      agregar_arista(4,rotar(300,a));
44      agregar_arista(1,rotar(300,b));
45      agregar_arista(3,rotar(300,c));
46      agregar_arista(0,rotar(300,d));
47      construir_grafomata;
48    end;
49    result:=g;
50 end;
```

C.4.16. Grafómata dual 3.4.6.4

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v5	a3	(142,82)
a1	v5	a3	(142,82)	a1	v4	a2	(95,0)	a2	v1	a4	(0,164)
a2	v1	a4	(0,164)	a2	v0	a5	(0,164)	a3	v3	a4	(-142,82)
a3	v3	a4	(-142,82)	a3	v2	a1	(-95,0)	a4	v5	a1	(-142,-82)
a4	v5	a1	(-142,-82)	a4	v0	a2	(0,-164)	a5	v1	a2	(0,-164)
a5	v1	a2	(0,-164)					a6	v3	a2	(142,-82)
a6	v3	a2	(142,-82)					V1			
V2				V4				a1	v4	a2	(95,0)
a0	v2	a0	(0,0)	a0	v4	a0	(0,0)	a3	v2	a1	(-95,0)
a1	v1	a3	(95,0)	a1	v3	a3	(47,82)	V2			
a2	v5	a4	(-47,82)	a2	v1	a1	(-95,0)	a2	v5	a4	(-47,82)
a3	v3	a1	(-47,-82)	a3	v5	a2	(47,-82)	a3	v3	a1	(-47,-82)
V3				V5				V3			
a0	v3	a0	(0,0)	a0	v5	a0	(0,0)	a3	v4	a1	(-47,-82)
a1	v2	a3	(47,82)	a1	v0	a4	(142,82)	V4			
a2	v0	a6	(-142,82)	a2	v4	a3	(-47,82)	a3	v5	a2	(47,-82)
a3	v4	a1	(-47,-82)	a3	v0	a1	(-142,-82)	V5			
a4	v0	a3	(142,-82)	a4	v2	a2	(47,-82)				

Figura C.27: Patrón gráfico del grafómata dual 3.4.6.4



C.4.17. Grafómata 4.6.12

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v9	a2	(120,0)
a1	v9	a2	(120,0)	a1	v10	a3	(60,104)	a2	v6	a3	(0,120)
a2	v6	a3	(0,120)	a2	v7	a3	(-104,60)	a3	v11	a1	(-60,-104)
a3	v11	a1	(-60,-104)	a3	v6	a2	(60,-104)	V1			
V2				V3				a1	v10	a3	(60,104)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a2	v7	a3	(-104,60)
a1	v7	a2	(120,0)	a1	v8	a3	(60,104)	a3	v6	a2	(60,-104)
a2	v11	a3	(-60,104)	a2	v6	a1	(-120,0)	V2			
a3	v8	a1	(-104,-60)	a3	v9	a1	(0,-120)	a1	v7	a2	(120,0)
V4				V5				a2	v11	a3	(-60,104)
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	a3	v8	a1	(-104,-60)
a1	v9	a3	(-60,104)	a1	v11	a2	(104,60)	V3			
a2	v7	a1	(-60,-104)	a2	v10	a1	(-120,0)	a1	v8	a3	(60,104)
a3	v10	a2	(104,-60)	a3	v8	a2	(60,-104)	a2	v6	a1	(-120,0)
V6				V7				a3	v9	a1	(0,-120)
a0	v6	a0	(0,0)	a0	v7	a0	(0,0)	V4			
a1	v3	a2	(120,0)	a1	v4	a2	(60,104)	a1	v9	a3	(-60,104)
a2	v1	a3	(-60,104)	a2	v2	a1	(-120,0)	a2	v7	a1	(-60,-104)
a3	v0	a2	(0,-120)	a3	v1	a2	(104,-60)	a3	v10	a2	(104,-60)
V8				V9				V5			
a0	v8	a0	(0,0)	a0	v9	a0	(0,0)	a1	v11	a2	(104,60)
a1	v2	a3	(104,60)	a1	v3	a3	(0,120)	a2	v10	a1	(-120,0)
a2	v5	a3	(-60,104)	a2	v0	a1	(-120,0)	a3	v8	a2	(60,-104)
a3	v3	a1	(-60,-104)	a3	v4	a1	(60,-104)	V6			
V10				V11				V7			
a0	v10	a0	(0,0)	a0	v11	a0	(0,0)	V8			
a1	v5	a2	(120,0)	a1	v0	a3	(60,104)	V9			
a2	v4	a3	(-104,60)	a2	v5	a1	(-104,-60)	V10			
a3	v1	a1	(-60,-104)	a3	v2	a2	(60,-104)	V11			

Fuente en Pascal C.16: 4.6.12

```

1 function Semiregular7:Tgrafomata;
2 var
3   g:Tgrafomata;
4   a,b,c,d,e,f:tpoint;
5 begin
6   //rotaciones de aristas
7   a:=point(numero_magico,0);
8   b:=rotar(90,a);
9   c:=rotar(240,a);
10  d:=a;
11  e:=reflejar_y(b);
12  f:=reflejar_y(c);
13  g:=Tgrafomata.create;
14  with g do
15    begin
16      nombre:='4612';
17      agregar_vertice; //v0
18      agregar_arista( 9,a);
19      agregar_arista( 6,b);
20      agregar_arista(11,c);
21      agregar_vertice; //v1
22      agregar_arista(10,rotar(60,a));
23      agregar_arista( 7,rotar(60,b));
24      agregar_arista( 6,rotar(60,c));
25      agregar_vertice; //v2
26      agregar_arista(11,rotar(120,a));
27      agregar_arista( 8,rotar(120,b));
28      agregar_arista( 7,rotar(120,c));
29      agregar_vertice; //v3
30      agregar_arista( 6,rotar(180,a));
31      agregar_arista( 9,rotar(180,b));
32      agregar_arista( 8,rotar(180,c));
33      agregar_vertice; //v4
34      agregar_arista( 7,rotar(240,a));
35      agregar_arista(10,rotar(240,b));
36      agregar_arista( 9,rotar(240,c));
37      agregar_vertice; //v5
38      agregar_arista( 8,rotar(300,a));
39      agregar_arista(11,rotar(300,b));
40      agregar_arista(10,rotar(300,c));
41      a:=d;
42      b:=e;
43      c:=f;
44      agregar_vertice; //v6
45      agregar_arista(3,a);
46      agregar_arista(0,b);
47      agregar_arista(1,c);
48      agregar_vertice; //v7
49      agregar_arista(4,rotar(60,a));
50      agregar_arista(1,rotar(60,b));
51      agregar_arista(2,rotar(60,c));
52      agregar_vertice; //v8
53      agregar_arista(5,rotar(120,a));
54      agregar_arista(2,rotar(120,b));
55      agregar_arista(3,rotar(120,c));
56      agregar_vertice; //v9
57      agregar_arista(0,rotar(180,a));
58      agregar_arista(3,rotar(180,b));
59      agregar_arista(4,rotar(180,c));
60      agregar_vertice; //v10
61      agregar_arista(1,rotar(240,a));
62      agregar_arista(4,rotar(240,b));
63      agregar_arista(5,rotar(240,c));
64      agregar_vertice; //v11
65      agregar_arista(2,rotar(300,a));
66      agregar_arista(5,rotar(300,b));
67      agregar_arista(0,rotar(300,c));
68      construir_grafomata;
69    end;
70    result:=g;
71 end;

```

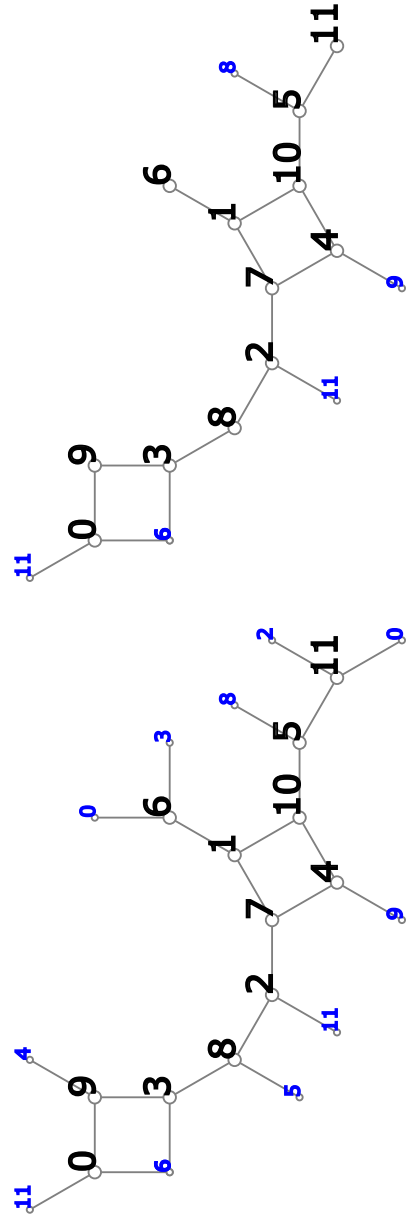


Figura C.28: Patrón gráfico del grafómata 4.6.12

C.4.18. Grafómata dual 4.6.12

Estructura de grafómata								Estructura de dibúgrafo			
V0				V3				V0			
a0	v0	a0	(0,0)	a0	v3	a0	(0,0)	a1	v2	a8	(284,164)
a1	v2	a8	(284,164)	a1	v5	a3	(142,82)	a2	v1	a4	(0,164)
a2	v1	a4	(0,164)	a2	v2	a10	(0,328)	a3	v2	a12	(-284,164)
a3	v2	a12	(-284,164)	a3	v4	a4	(-142,82)	a4	v5	a1	(-142,-82)
a4	v5	a1	(-142,-82)	a4	v2	a2	(-284,-164)	a5	v2	a4	(0,-328)
a5	v2	a4	(0,-328)	a5	v1	a2	(0,-164)	a6	v4	a2	(142,-82)
a6	v4	a2	(142,-82)	a6	v2	a6	(284,-164)	V1			
V1				V2				a1	v2	a7	(284,0)
a0	v1	a0	(0,0)	a0	v2	a0	(0,0)	a2	v3	a5	(0,164)
a1	v2	a7	(284,0)	a1	v1	a3	(284,0)	a3	v2	a1	(-284,0)
a2	v3	a5	(0,164)	a2	v3	a4	(284,164)	V2			
a3	v2	a1	(-284,0)	a3	v4	a3	(142,246)	a2	v3	a4	(284,164)
a4	v0	a2	(0,-164)	a4	v0	a5	(0,328)	a3	v4	a3	(142,246)
V4				a5	v5	a4	(-142,246)	a5	v5	a4	(-142,246)
a0	v4	a0	(0,0)	a6	v3	a6	(-284,164)	a6	v3	a6	(-284,164)
a1	v2	a9	(142,246)	a7	v1	a1	(-284,0)	a9	v4	a1	(-142,-246)
a2	v0	a6	(-142,82)	a8	v0	a1	(-284,-164)	a10	v3	a2	(0,-328)
a3	v2	a3	(-142,-246)	a9	v4	a1	(-142,-246)	a11	v5	a2	(142,-246)
a4	v3	a3	(142,-82)	a10	v3	a2	(0,-328)	V3			
V5				a11	v5	a2	(142,-246)	a1	v5	a3	(142,82)
a0	v5	a0	(0,0)	a12	v0	a3	(284,-164)	a3	v4	a4	(-142,82)
a1	v0	a4	(142,82)					V4			
a2	v2	a11	(-142,246)					V5			
a3	v3	a1	(-142,-82)								
a4	v2	a5	(142,-246)								

C.4.19. Grafómata 3.12.12

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v4	a3	(0,120)
a1	v4	a3	(0,120)	a1	v0	a3	(-60,104)	a2	v3	a3	(-104,60)
a2	v3	a3	(-104,60)	a2	v5	a2	(0,-120)	a3	v1	a1	(60,-104)
a3	v1	a1	(60,-104)	a3	v2	a2	(104,-60)	V1			
V2				V3				a2 v5 a2 (0,-120)			
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v2	a2	(104,-60)
a1	v3	a2	(120,0)	a1	v4	a2	(104,60)	V2			
a2	v1	a3	(-104,60)	a2	v2	a1	(-120,0)	a1	v3	a2	(120,0)
a3	v5	a1	(-104,-60)	a3	v0	a2	(104,-60)	a3	v5	a1	(-104,-60)
V4				V5				V3			
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	a1	v4	a2	(104,60)
a1	v5	a3	(60,104)	a1	v2	a3	(104,60)	V4			
a2	v3	a1	(-104,-60)	a2	v1	a2	(0,120)	a1	v5	a3	(60,104)
a3	v0	a1	(0,-120)	a3	v4	a1	(-60,-104)	V5			

Figura C.30: Patrón gráfico del grafómata 3.12.12

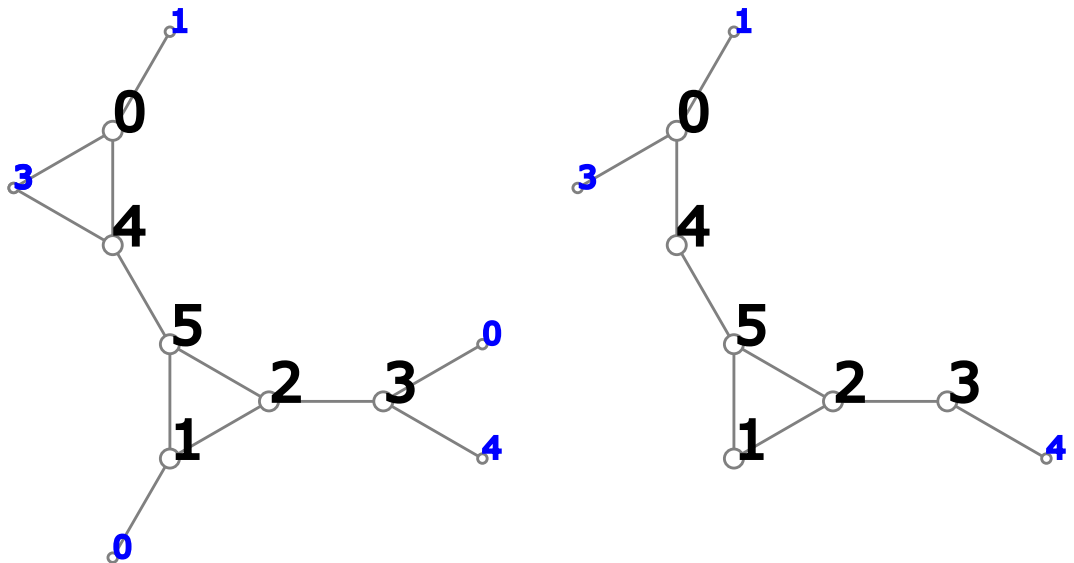
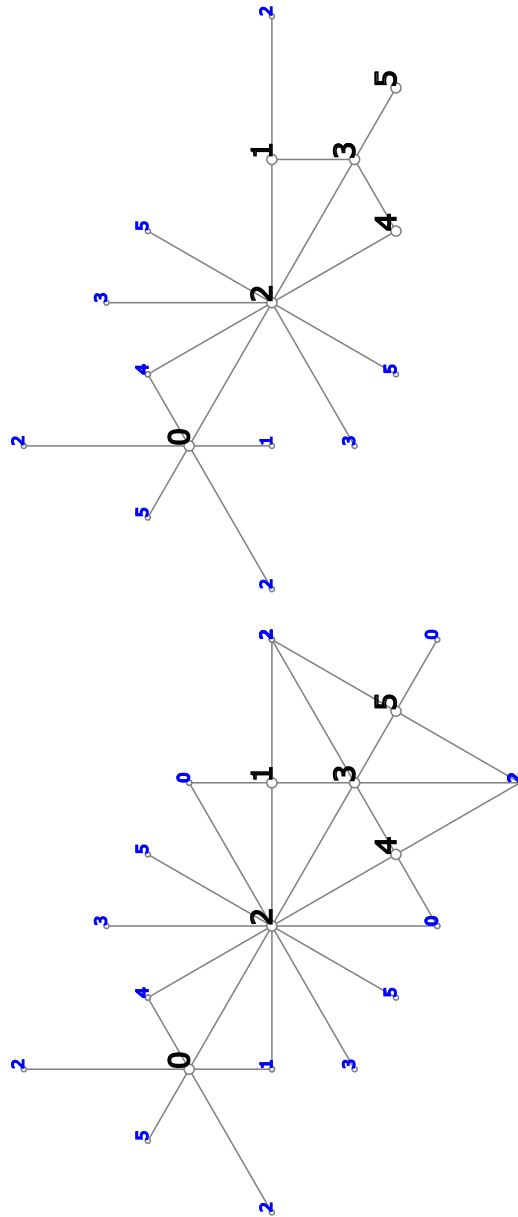


Figura C.29: Patrón gráfico del grafómata dual 4.6.12



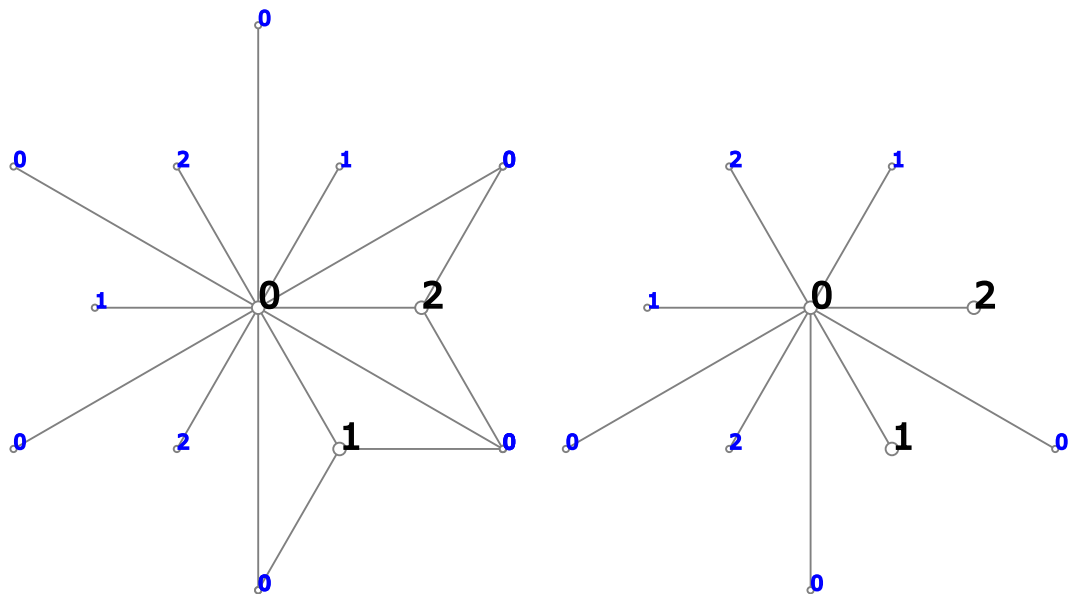
Fuente en Pascal C.17: 3.12.12

```
1 function Semiregular8(angulo:integer):Tgrafomata;
2 var
3   g:Tgrafomata;
4   a,b,c:tpoint;
5 begin
6   g:=Tgrafomata.create;
7   //rotaciones de aristas
8   a:=point(numero_magico,0);
9   b:=rotar(60,a);
10  c:=rotar(angulo,b);// dodecagono y triangulos. angulo 15,30 = estrellitas
11 //rotación extra del conjunto, ajuste del grafomata a los ejes coordinados
12  a:=rotar90d(a);
13  b:=rotar90d(b);
14  c:=rotar90d(c);
15  with g do
16    begin
17      nombre:='31212_'+inttostr(angulo);
18      agregar_vertice; //v0
19      agregar_arista(4,a);
20      agregar_arista(3,b);
21      agregar_arista(1,c);
22      agregar_vertice; //v1
23      agregar_arista(5,rotar(180,a));
24      agregar_arista(2,rotar(180,b));
25      agregar_arista(0,rotar(180,c));
26      agregar_vertice; //v2
27      agregar_arista(1,rotar(60,a));
28      agregar_arista(5,rotar(60,b));
29      agregar_arista(3,rotar(60,c));
30      agregar_vertice; //v3
31      agregar_arista(0,rotar(240,a));
32      agregar_arista(4,rotar(240,b));
33      agregar_arista(2,rotar(240,c));
34      agregar_vertice; //v4
35      agregar_arista(3,rotar(120,a));
36      agregar_arista(0,rotar(120,b));
37      agregar_arista(5,rotar(120,c));
38      agregar_vertice; //v5
39      agregar_arista(2,rotar(300,a));
40      agregar_arista(1,rotar(300,b));
41      agregar_arista(4,rotar(300,c));
42      construir_grafomata;
43    end;
44    result:=g;
45 end;
```

C.4.20. Grafómata dual 3.12.12

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v2	a2	(259,0)
a1	v2	a2	(259,0)	a1	v0	a7	(259,0)	a2	v0	a8	(388,224)
a2	v0	a8	(388,224)	a2	v0	a11	(-129,224)	a3	v1	a3	(129,224)
a3	v1	a3	(129,224)	a3	v0	a3	(-129,-224)	a4	v0	a10	(0,448)
a4	v0	a10	(0,448)	V2				a5	v2	a3	(-129,224)
a5	v2	a3	(-129,224)	a0	v2	a0	(0,0)	a6	v0	a12	(-388,224)
a6	v0	a12	(-388,224)	a1	v0	a9	(129,224)	a7	v1	a1	(-259,0)
a7	v1	a1	(-259,0)	a2	v0	a1	(-259,0)	a9	v2	a1	(-129,-224)
a8	v0	a2	(-388,-224)	a3	v0	a5	(129,-224)	a11	v1	a2	(129,-224)
a9	v2	a1	(-129,-224)					V1			
a10	v0	a4	(0,-448)					V2			
a11	v1	a2	(129,-224)								
a12	v0	a6	(388,-224)								

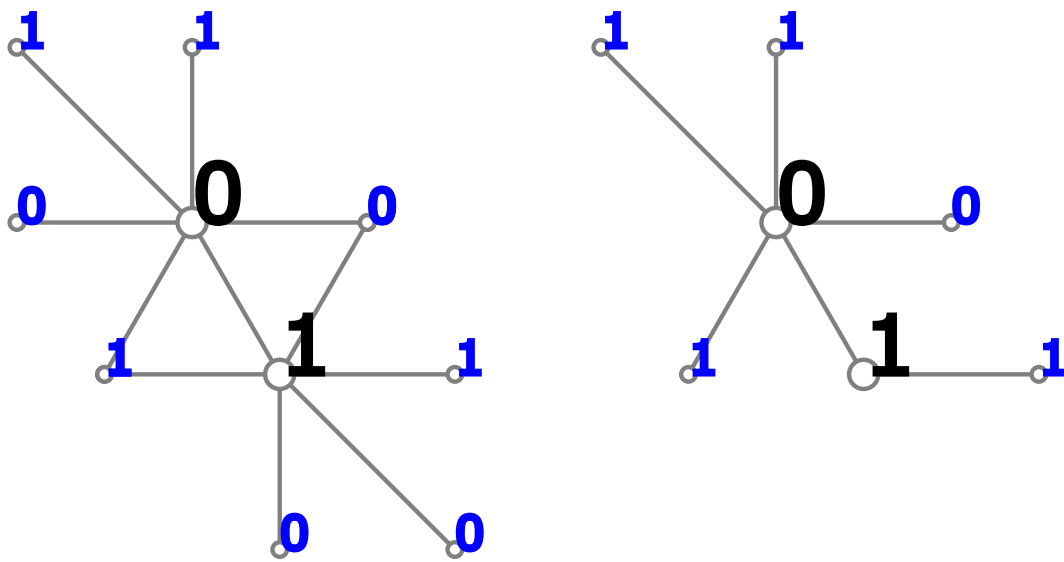
Figura C.31: Patrón gráfico del grafómata dual 3.12.12



C.4.21. Grafómata Variación en triángulos

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v0	a4	(120,0)
a1	v0	a4	(120,0)	a1	v1	a4	(120,0)	a2	v1	a5	(60,104)
a2	v1	a5	(60,104)	a2	v0	a5	(120,120)	a3	v1	a6	(-60,104)
a3	v1	a6	(-60,104)	a3	v0	a6	(0,120)	a5	v1	a2	(-120,-120)
a4	v0	a1	(-120,0)	a4	v1	a1	(-120,0)	a6	v1	a3	(0,-120)
a5	v1	a2	(-120,-120)	a5	v0	a2	(-60,-104)	V1			
a6	v1	a3	(0,-120)	a6	v0	a3	(60,-104)	a1	v1	a4	(120,0)

Figura C.32: Patrón gráfico del grafómata variación en triángulos



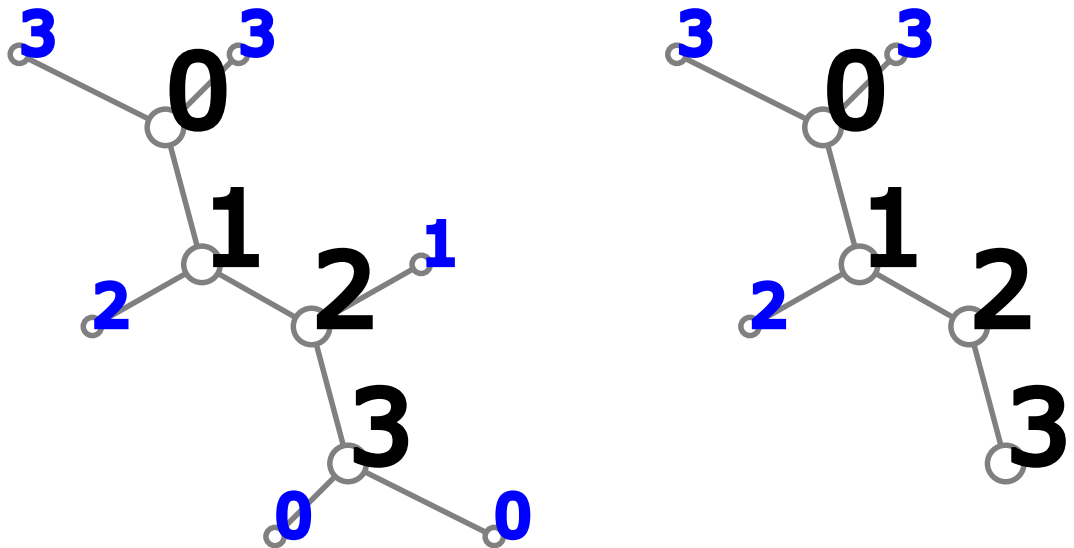
Fuente en Pascal C.18: Variación en triángulos

```
1 function Noregular2:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   a,b,c,d,e,f:tpoint;
6 begin
7   //rotaciones de aristas
8   // i:=10;
9   i:=numero_magico;
10  a:=point(i,0);
11  b:=rotar(60,a);
12  c:=rotar(60,b);
13  d:=rotar(60,c);
14  e:=point(-i,-i);
15  f:=rotar(90,d);
16  g:=Tgrafomata.create;
17  with g do
18    begin
19      nombre:='Var_triangulos';
20      agregar_vertice; //v0
21      agregar_arista(0,a);
22      agregar_arista(1,b);
23      agregar_arista(1,c);
24      agregar_arista(0,d);
25      agregar_arista(1,e);
26      agregar_arista(1,f);
27      agregar_vertice; //v1
28      agregar_arista(1,reflejar_x(a));
29      agregar_arista(0,reflejar_y(b));
30      agregar_arista(0,reflejar_y(c));
31      agregar_arista(1,reflejar_x(d));
32      agregar_arista(0,reflejar_xy(e));
33      agregar_arista(0,reflejar_y(f));
34      construir_grafomata;
35    end;
36    result:=g;
37 end;
```

C.4.22. Grafómata dual variación en triángulos

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a3	(20,75)
a1	v1	a3	(20,75)	a1	v2	a2	(60,34)	a2	v3	a1	(-80,-40)
a2	v3	a1	(-80,-40)	a2	v2	a3	(-60,34)	a3	v3	a2	(40,-40)
a3	v3	a2	(40,-40)	a3	v0	a1	(-20,-75)	V1			
V2				V3				a1	v2	a2	(60,34)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a2	v2	a3	(-60,34)
a1	v3	a3	(20,75)	a1	v0	a2	(80,40)	V2			
a2	v1	a1	(-60,-34)	a2	v0	a3	(-40,40)	a1	v3	a3	(20,75)
a3	v1	a2	(60,-34)	a3	v2	a1	(-20,-75)	V3			

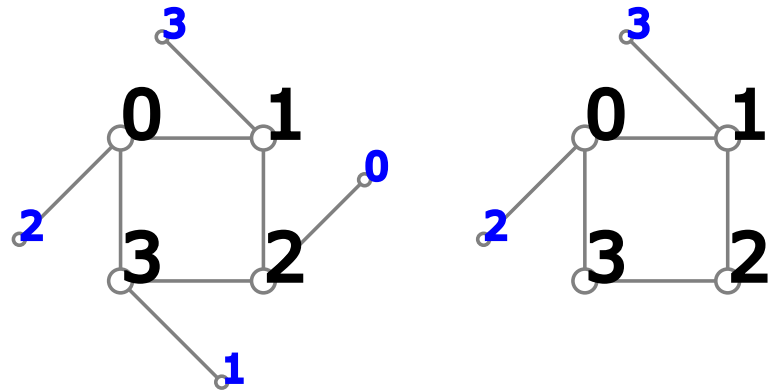
Figura C.33: Patrón gráfico del grafómata dual variación en triángulos



C.4.23. Grafómata Variación en 4.8.8

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a2	(120,0)
a1	v1	a2	(120,0)	a1	v2	a2	(0,120)	a2	v3	a3	(0,120)
a2	v3	a3	(0,120)	a2	v0	a1	(-120,0)	a3	v2	a3	(-85,85)
a3	v2	a3	(-85,85)	a3	v3	a2	(-85,-85)	V1			
V2				V3				a1	v2	a2	(0,120)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v3	a2	(-85,-85)
a1	v3	a1	(-120,0)	a1	v2	a1	(120,0)	V2			
a2	v1	a1	(0,-120)	a2	v1	a3	(85,85)	a1	v3	a1	(-120,0)
a3	v0	a3	(85,-85)	a3	v0	a2	(0,-120)	V3			

Figura C.34: Patrón gráfico del grafómata variación en 4.8.8



Fuente en Pascal C.19: Variación en 4.8.8

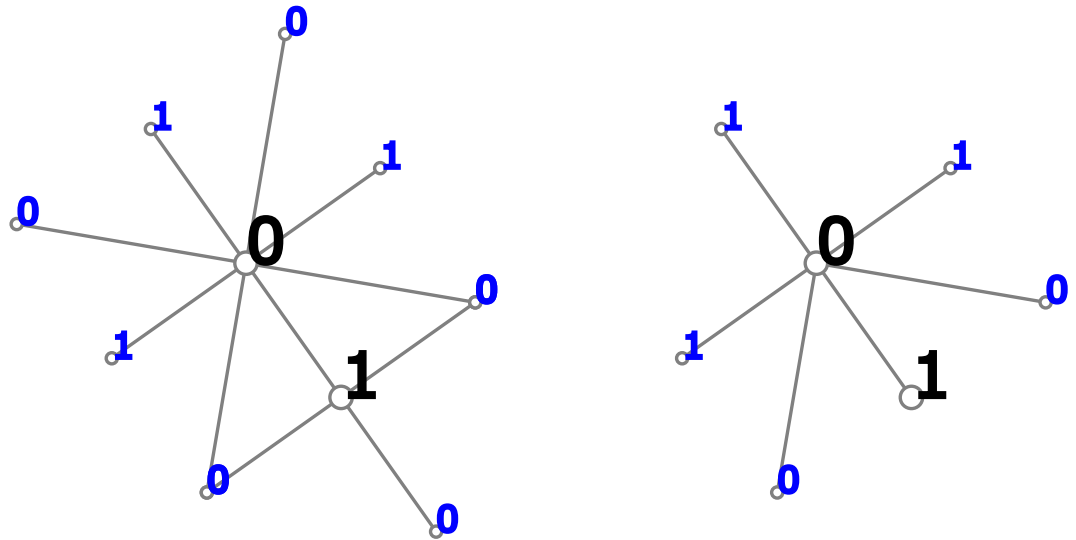
```
1 function Noregular5:Tgrafomata;  
2 begin  
3   result:=semiregular3(45); //estrellas de 8 lados y cuadrados  
4 end;
```

El código fuente de la función semiregular3 se muestra en la página C.12 en la página 350.

C.4.24. Grafómata dual variación en 4.8.8

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v0	a5	(205,35)
a1	v0	a5	(205,35)	a1	v0	a6	(85,120)	a2	v1	a3	(85,120)
a2	v1	a3	(85,120)	a2	v0	a8	(-120,85)	a3	v0	a7	(-35,205)
a3	v0	a7	(-35,205)	a3	v0	a2	(-85,-120)	a4	v1	a4	(-120,85)
a4	v1	a4	(-120,85)	a4	v0	a4	(120,-85)	a6	v1	a1	(-85,-120)
a5	v0	a1	(-205,-35)					a8	v1	a2	(120,-85)
a6	v1	a1	(-85,-120)					V1			
a7	v0	a3	(35,-205)								
a8	v1	a2	(120,-85)								

Figura C.35: Patrón gráfico del grafómata dual variación en 4.8.8



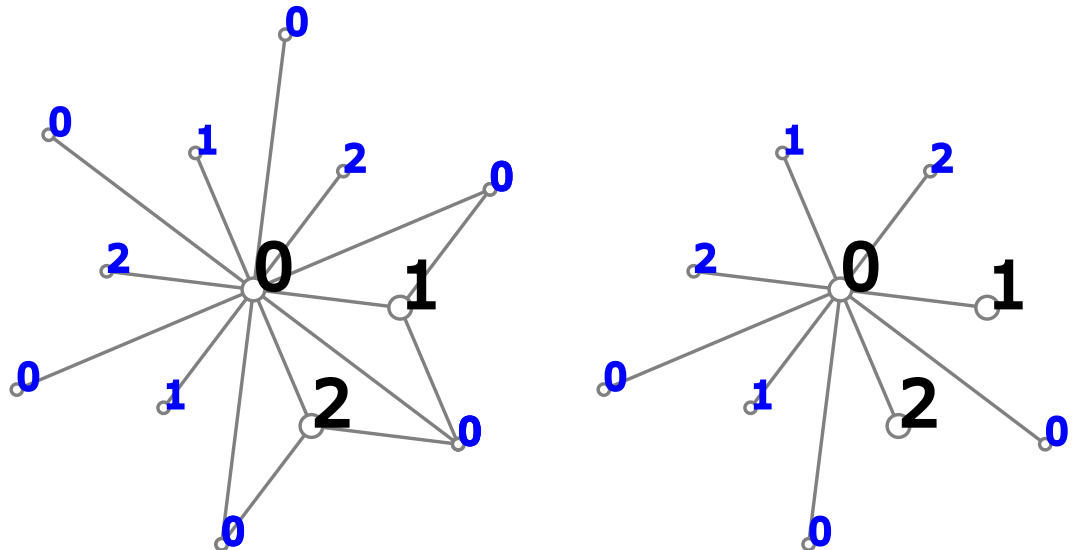
C.4.25. Grafómata variación en 3.12.12

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v4	a2	(0,120)
a1	v4	a2	(0,120)	a1	v0	a3	(120,0)	a2	v3	a3	(-104,60)
a2	v3	a3	(-104,60)	a2	v5	a2	(0,-120)	a3	v1	a1	(-120,0)
a3	v1	a1	(-120,0)	a3	v2	a1	(104,-60)	V1			
V2				V3				a2	v5	a2	(0,-120)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v2	a1	(104,-60)
a1	v1	a3	(-104,60)	a1	v4	a1	(104,60)	V2			
a2	v5	a1	(-104,-60)	a2	v2	a3	(60,104)	a2	v5	a1	(-104,-60)
a3	v3	a2	(-60,-104)	a3	v0	a2	(104,-60)	a3	v3	a2	(-60,-104)
V4				V5				V3			
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	a1	v4	a1	(104,60)
a1	v3	a1	(-104,-60)	a1	v2	a2	(104,60)	V4			
a2	v0	a1	(0,-120)	a2	v1	a2	(0,120)	a3	v5	a3	(60,-104)
a3	v5	a3	(60,-104)	a3	v4	a3	(-60,104)	V5			

C.4.26. Grafómata dual variación 3.12.12

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a2	(129,16)
a1	v1	a2	(129,16)	a1	v0	a9	(51,120)	a2	v0	a8	(180,136)
a2	v0	a8	(180,136)	a2	v0	a1	(-129,-16)	a3	v2	a3	(51,120)
a3	v2	a3	(51,120)	a3	v0	a5	(79,-104)	a4	v0	a10	(-28,224)
a4	v0	a10	(-28,224)	V2				a5	v1	a3	(-79,104)
a5	v1	a3	(-79,104)	a0	v2	a0	(0,0)	a6	v0	a12	(-208,88)
a6	v0	a12	(-208,88)	a1	v0	a7	(129,16)	a7	v2	a1	(-129,-16)
a7	v2	a1	(-129,-16)	a2	v0	a11	(-79,104)	a9	v1	a1	(-51,-120)
a8	v0	a2	(-180,-136)	a3	v0	a3	(-51,-120)	a11	v2	a2	(79,-104)
a9	v1	a1	(-51,-120)					V1			
a10	v0	a4	(28,-224)					V2			
a11	v2	a2	(79,-104)								
a12	v0	a6	(208,-88)								

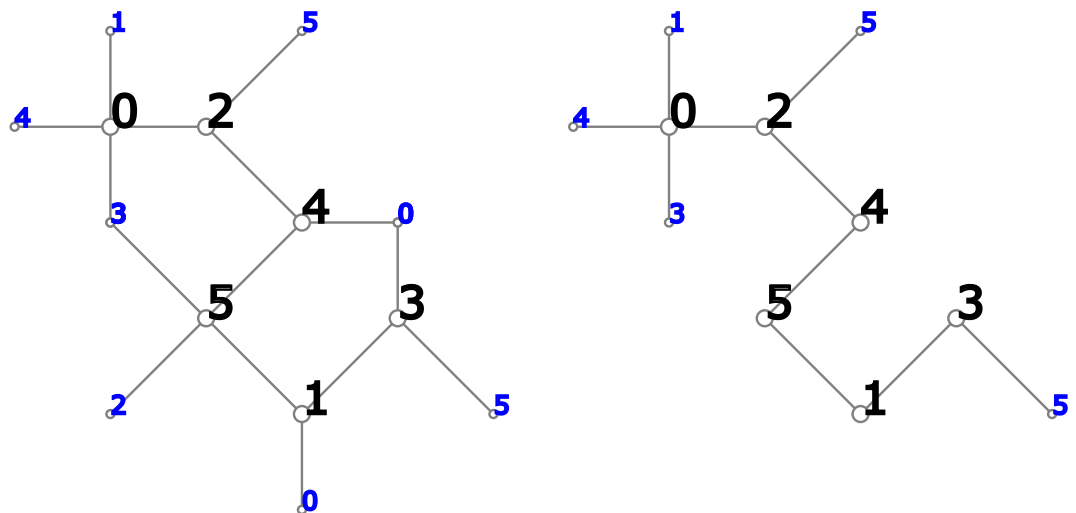
Figura C.37: Patrón gráfico del grafómata dual variación en 3.12.12



C.4.27. Grafómata variación el Cairo

Estructura de grafómata								Estructura de dibúgrafo			
V0				V5				V0			
a0	v0	a0	(0,0)	a0	v5	a0	(0,0)	a1	v2	a2	(120,0)
a1	v2	a2	(120,0)	a1	v1	a2	(120,120)	a2	v3	a3	(0,120)
a2	v3	a3	(0,120)	a2	v2	a3	(-120,120)	a3	v4	a1	(-120,0)
a3	v4	a1	(-120,0)	a3	v3	a1	(-120,-120)	a4	v1	a1	(0,-120)
a4	v1	a1	(0,-120)	a4	v4	a2	(120,-120)	V1			
V1				V2				a2	v5	a1	(-120,-120)
a0	v1	a0	(0,0)	a0	v2	a0	(0,0)	a3	v3	a2	(120,-120)
a1	v0	a4	(0,120)	a1	v4	a3	(120,120)	V2			
a2	v5	a1	(-120,-120)	a2	v0	a1	(-120,0)	a1	v4	a3	(120,120)
a3	v3	a2	(120,-120)	a3	v5	a2	(120,-120)	a3	v5	a2	(120,-120)
V3				V4				V3			
a0	v3	a0	(0,0)	a0	v4	a0	(0,0)	a1	v5	a3	(120,120)
a1	v5	a3	(120,120)	a1	v0	a3	(120,0)	V4			
a2	v1	a3	(-120,120)	a2	v5	a4	(-120,120)	a2	v5	a4	(-120,120)
a3	v0	a2	(0,-120)	a3	v2	a1	(-120,-120)	V5			

Figura C.38: Patrón gráfico del grafómata variación el Cairo



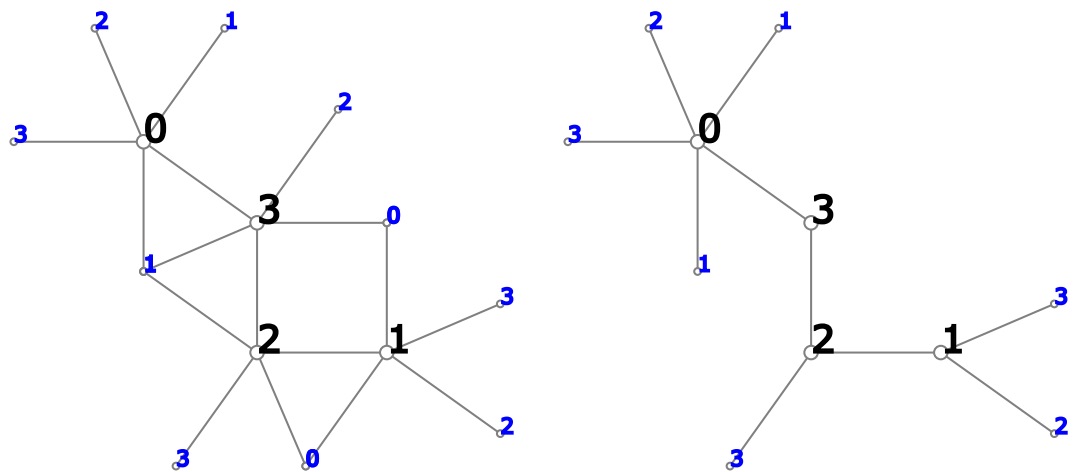
Fuente en Pascal C.21: Variación el cairo

```
1 function Semiregular2:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   a,b,c,d,e,f,s,t:tpoint;
6 begin
7   //rotaciones de aristas
8   i:=numero_magico;
9   a:=point(i,0);
10  b:=point(-i div 2,-i);
11  c:=reflejar_y(b);
12  d:=reflejar_x(a);
13  e:=reflejar_x(b);
14  f:=reflejar_x(c);
15  g:=Tgrafomata.create;
16  with g do
17    begin
18      nombre:='Cairo';
19      agregar_vertice; //v0
20      agregar_arista(1,a);
21      agregar_arista(5,b);
22      agregar_arista(4,c);
23      agregar_vertice; //v1
24      agregar_arista(0,d);
25      agregar_arista(4,e);
26      agregar_arista(5,f);
27      s:=rotar90d(b);
28      t:=rotar90d(c);
29      agregar_vertice; //v2
30      agregar_arista(3,rotar90d(a));
31      agregar_arista(5,s);
32      agregar_arista(4,t);
33      agregar_vertice; //v3
34      agregar_arista(2,rotar90d(d));
35      agregar_arista(4,rotar90d(e));
36      agregar_arista(5,rotar90d(f));
37      agregar_vertice; //v4
38      agregar_arista(1,c);
39      agregar_arista(3,t);
40      agregar_arista(0,reflejar_xy(c));
41      agregar_arista(2,reflejar_xy(t));
42      agregar_vertice; //v5
43      agregar_arista(0,f);
44      agregar_arista(3,s);
45      agregar_arista(1,reflejar_xy(f));
46      agregar_arista(2,reflejar_xy(s));
47      construir_grafomata;
48    end;
49    result:=g;
50 end;
```


C.4.28. Grafómata dual variación el Cairo

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v3	a4	(168,120)
a1	v3	a4	(168,120)	a1	v2	a4	(168,120)	a2	v1	a4	(0,192)
a2	v1	a4	(0,192)	a2	v0	a5	(-120,168)	a3	v3	a1	(-192,0)
a3	v3	a1	(-192,0)	a3	v2	a1	(-192,0)	a4	v2	a2	(-72,-168)
a4	v2	a2	(-72,-168)	a4	v0	a2	(0,-192)	a5	v1	a2	(120,-168)
a5	v1	a2	(120,-168)	a5	v3	a3	(168,-72)	V1			
V2				V3				a1	v2	a4	(168,120)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v2	a1	(-192,0)
a1	v1	a3	(192,0)	a1	v0	a3	(192,0)	a5	v3	a3	(168,-72)
a2	v0	a4	(72,168)	a2	v2	a5	(0,192)	V2			
a3	v3	a5	(-120,168)	a3	v1	a5	(-168,72)	a3	v3	a5	(-120,168)
a4	v1	a1	(-168,-120)	a4	v0	a1	(-168,-120)	a5	v3	a2	(0,-192)
a5	v3	a2	(0,-192)	a5	v2	a3	(120,-168)	V3			

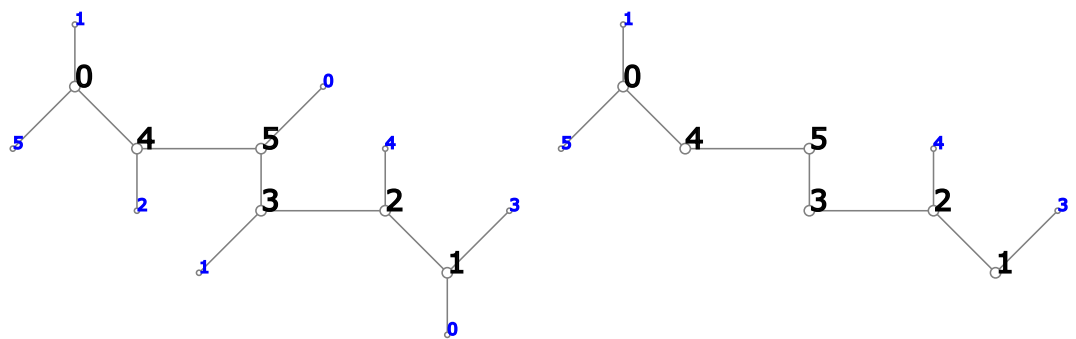
Figura C.39: Patrón gráfico del grafómata dual variación el Cairo



C.4.29. Grafómata pentágonos y octágonos

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v4	a3	(120,120)
a1	v4	a3	(120,120)	a1	v0	a3	(0,120)	a2	v5	a3	(-120,120)
a2	v5	a3	(-120,120)	a2	v2	a1	(-120,-120)	a3	v1	a1	(0,-120)
a3	v1	a1	(0,-120)	a3	v3	a2	(120,-120)	V1			
V2				V3				a2 v2 a1 (-120,-120)			
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v3	a2	(120,-120)
a1	v1	a2	(120,120)	a1	v2	a2	(240,0)	V2			
a2	v3	a1	(-240,0)	a2	v1	a3	(-120,120)	a2	v3	a1	(-240,0)
a3	v4	a2	(0,-120)	a3	v5	a1	(0,-120)	a3	v4	a2	(0,-120)
V4				V5				V3			
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	a3	v5	a1	(0,-120)
a1	v5	a2	(240,0)	a1	v3	a3	(0,120)	V4			
a2	v2	a3	(0,120)	a2	v4	a1	(-240,0)	a1	v5	a2	(240,0)
a3	v0	a1	(-120,-120)	a3	v0	a2	(120,-120)	V5			

Figura C.40: Patrón gráfico del grafómata pentágonos y octágonos



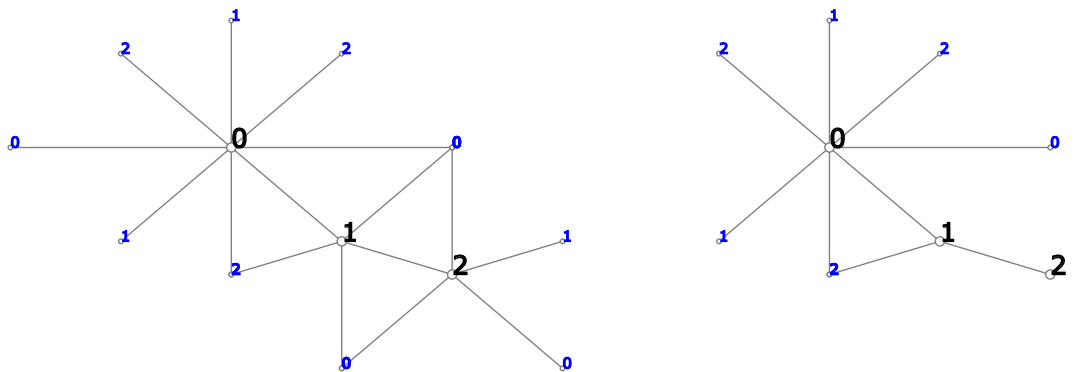
Fuente en Pascal C.22: Pentágonos y octágonos

```
1 function Noregular1:Tgrafomata;
2 var
3   i,j:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   i:=numero_magico;
8   j:=i*2;
9   g:=Tgrafomata.create;
10  with g do
11    begin
12      nombre:='Pentagonos_octagonos';
13      agregar_vertice; //v0
14      p:=point( 0,-i); agregar_arista(1,p);
15      p:=point( i, i); agregar_arista(4,p);
16      p:=point(-i, i); agregar_arista(5,p);
17      agregar_vertice; //v1
18      p:=point( 0, i); agregar_arista(0,p);
19      p:=point(-i,-i); agregar_arista(2,p);
20      p:=point( i,-i); agregar_arista(3,p);
21      agregar_vertice; //v2
22      p:=point(-j, 0); agregar_arista(3,p);
23      p:=point( 0,-i); agregar_arista(4,p);
24      p:=point( i, i); agregar_arista(1,p);
25      agregar_vertice; //v3
26      p:=point( j, 0); agregar_arista(2,p);
27      p:=point(-i, i); agregar_arista(1,p);
28      p:=point( 0,-i); agregar_arista(5,p);
29      agregar_vertice; //v4
30      p:=point(-i,-i); agregar_arista(0,p);
31      p:=point( j, 0); agregar_arista(5,p);
32      p:=point( 0, i); agregar_arista(2,p);
33      agregar_vertice; //v5
34      p:=point( i,-i); agregar_arista(0,p);
35      p:=point( 0, i); agregar_arista(3,p);
36      p:=point(-j, 0); agregar_arista(4,p);
37      construir_grafomata;
38    end;
39    result:=g;
40 end;
```

C.4.30. Grafómata dual pentágonos y octágonos

Estructura de grafómata						Estructura de dibúgrafo			
V0						V0			
a0	v0	a0	(0,0)			a1	v0	a5	(480,0)
a1	v0	a5	(480,0)			a2	v1	a4	(240,204)
a2	v1	a4	(240,204)			a3	v2	a4	(0,276)
a3	v2	a4	(0,276)			a4	v1	a5	(-240,204)
a4	v1	a5	(-240,204)			a6	v2	a1	(-240,-204)
a5	v0	a1	(-480,0)			a7	v1	a2	(0,-276)
a6	v2	a1	(-240,-204)			a8	v2	a2	(240,-204)
a7	v1	a2	(0,-276)			V1			
a8	v2	a2	(240,-204)			a1	v2	a3	(240,72)
V1			V2			a3	v2	a5	(-240,72)
a0	v1	a0	(0,0)			a0	v2	a0	(0,0)
a1	v2	a3	(240,72)			a1	v0	a6	(240,204)
a2	v0	a7	(0,276)			a2	v0	a8	(-240,204)
a3	v2	a5	(-240,72)			a3	v1	a1	(-240,-72)
a4	v0	a2	(-240,-204)			a4	v0	a3	(0,-276)
a5	v0	a4	(240,-204)			a5	v1	a3	(240,-72)

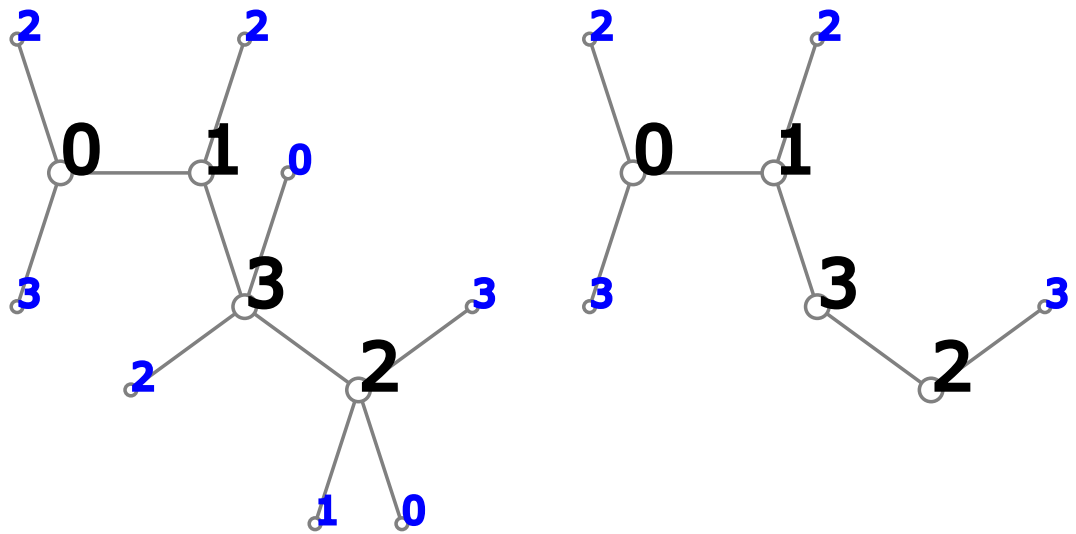
Figura C.41: Patrón gráfico del grafómata dual pentágonos y octágonos



C.4.31. Grafómata pentágonos y rombos

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a2	(120,0)
a1	v1	a2	(120,0)	a1	v3	a3	(37,114)	a2	v3	a4	(-37,114)
a2	v3	a4	(-37,114)	a2	v0	a1	(-120,0)	a3	v2	a1	(-37,-114)
a3	v2	a1	(-37,-114)	a3	v2	a2	(37,-114)	V1			
V2				V3				a1	v3	a3	(37,114)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a3	v2	a2	(37,-114)
a1	v0	a3	(37,114)	a1	v2	a3	(97,71)	V2			
a2	v1	a3	(-37,114)	a2	v2	a4	(-97,71)	a3	v3	a1	(-97,-71)
a3	v3	a1	(-97,-71)	a3	v1	a1	(-37,-114)	a4	v3	a2	(97,-71)
a4	v3	a2	(97,-71)	a4	v0	a2	(37,-114)	V3			

Figura C.42: Patrón gráfico del grafómata pentágonos y rombos



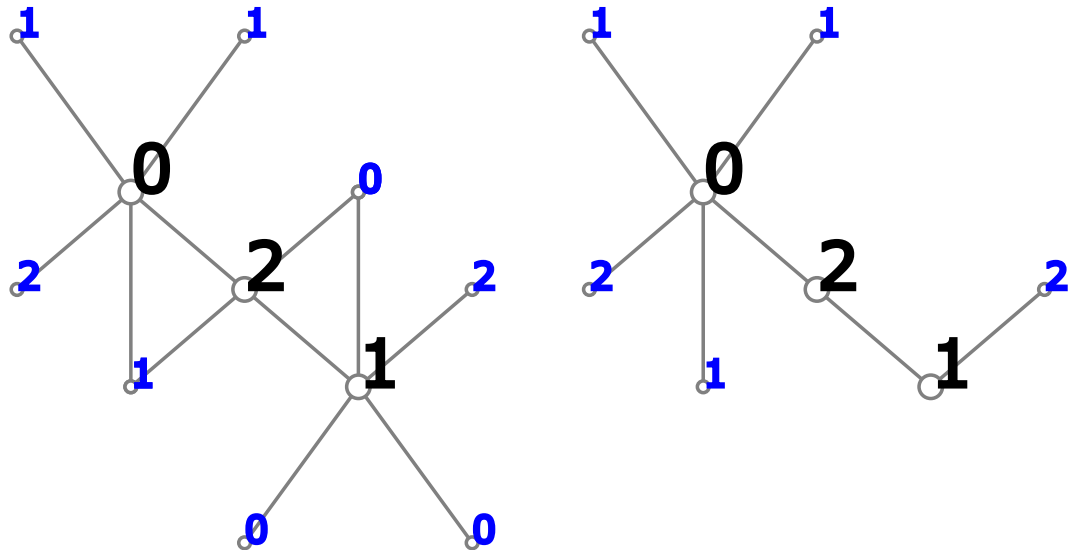
Fuente en Pascal C.23: Pentágonos y rombos

```
1 function Noregular3:Tgrafomata;
2 var
3   g:Tgrafomata;
4   a,b,c,d,e,f:tpoint;
5 begin
6   //rotaciones de aristas
7   a:=point(numero_magico,0);
8   b:=rotar(108,a);
9   c:=reflejar_y(b);
10  g:=Tgrafomata.create;
11  with g do
12    begin
13      nombre:='Pentagonos_rombos';
14      agregar_vertice; //v0
15      agregar_arista(1,a);
16      agregar_arista(3,b);
17      agregar_arista(2,c);
18      agregar_vertice; //v1
19      agregar_arista(0,reflejar_x(a));
20      agregar_arista(3,reflejar_x(b));
21      agregar_arista(2,reflejar_x(c));
22      d:=reflejar_xy(c);
23      e:=b;
24      f:=rotar(216,a);
25      a:=reflejar_x(f);
26      agregar_vertice; //v2
27      agregar_arista(0,d);
28      agregar_arista(1,e);
29      agregar_arista(3,f);
30      agregar_arista(3,a);
31      agregar_vertice; //v3
32      agregar_arista(0,reflejar_y(d));
33      agregar_arista(1,reflejar_y(e));
34      agregar_arista(2,reflejar_y(f));
35      agregar_arista(2,reflejar_y(a));
36      construir_grafomata;
37    end;
38    result:=g;
39 end;
```

C.4.32. Grafómata dual pentágonos y rombos

Estructura de grafómata				Estructura de dibúgrafo			
V0				V1			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)
a1	v2	a3	(97,83)	a1	v0	a4	(97,133)
a2	v1	a4	(0,166)	a2	v0	a5	(-97,133)
a3	v2	a4	(-97,83)	a3	v2	a1	(-97,-83)
a4	v1	a1	(-97,-133)	a4	v0	a2	(0,-166)
a5	v1	a2	(97,-133)	a5	v2	a2	(97,-83)
V2							
a0	v2	a0	(0,0)				
a1	v1	a3	(97,83)				
a2	v1	a5	(-97,83)				
a3	v0	a1	(-97,-83)				
a4	v0	a3	(97,-83)				
				V0			
				a1	v2	a3	(97,83)
				a2	v1	a4	(0,166)
				a3	v2	a4	(-97,83)
				a4	v1	a1	(-97,-133)
				a5	v1	a2	(97,-133)
				V1			
				a3	v2	a1	(-97,-83)
				a5	v2	a2	(97,-83)
				V2			

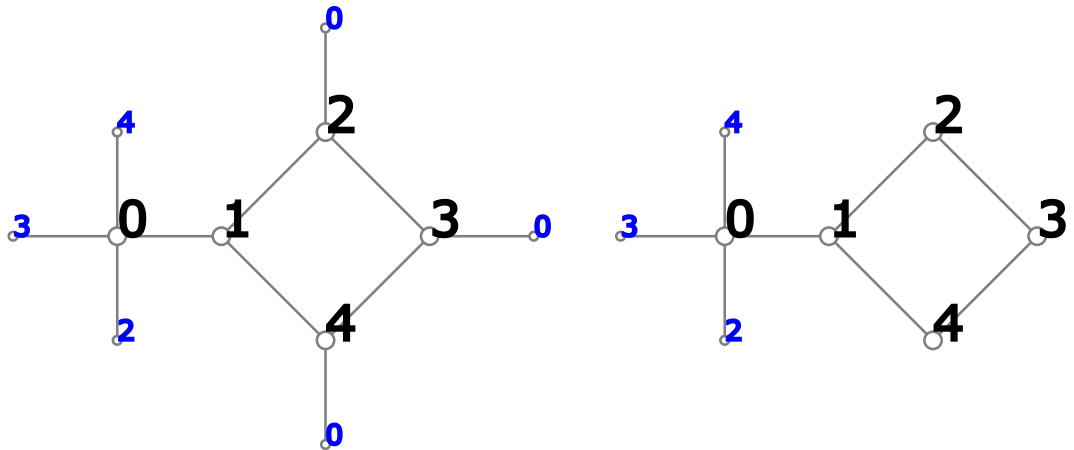
Figura C.43: Patrón gráfico del grafómata dual pentágonos y rombos



C.4.33. Grafómata Cuadrados y hexágonos

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v1	a2	(120,0)
a1	v1	a2	(120,0)	a2	v2	a3	(0,120)
a2	v2	a3	(0,120)	a3	v3	a1	(-120,0)
a3	v3	a1	(-120,0)	a4	v4	a1	(0,-120)
a4	v4	a1	(0,-120)	V1			
V1				a1	v4	a2	(120,120)
a0	v1	a0	(0,0)	a3	v2	a2	(120,-120)
a1	v4	a2	(120,120)	V2			
a2	v0	a1	(-120,0)	a1	v3	a3	(120,120)
a3	v2	a2	(120,-120)	a2	v1	a3	(-120,120)
V3				a3	v0	a2	(0,-120)
a0	v3	a0	(0,0)	V4			
a1	v0	a3	(120,0)	a0	v4	a0	(0,0)
a2	v4	a3	(-120,120)	a1	v0	a4	(0,120)
a3	v2	a1	(-120,-120)	a2	v1	a1	(-120,-120)
				a3	v3	a2	(120,-120)

Figura C.44: Patrón gráfico del grafómata cuadrados y hexágonos



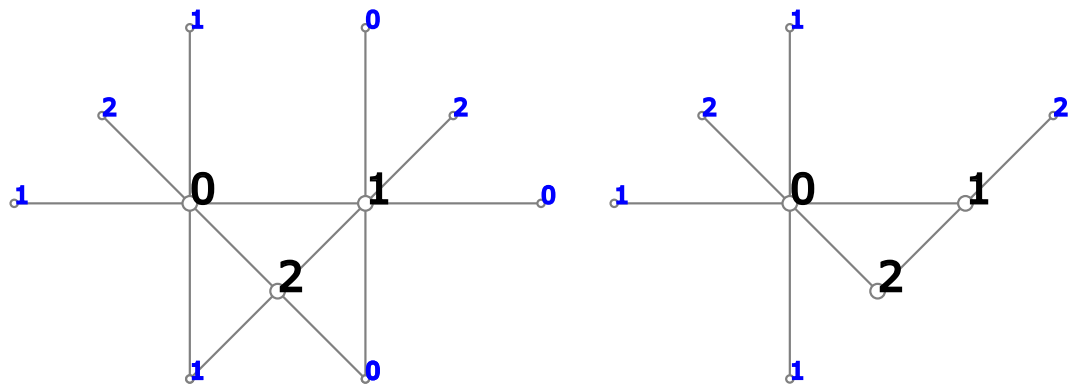
Fuente en Pascal C.24: Cuadrados y hexágonos

```
1 function Noregular4:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   i:=numero_magico;
8   g:=Tgrafomata.create;
9   with g do
10    begin
11     nombre:='cuadrados_hexagonos';
12     agregar_vertice; //v0
13     p:=point(i,0);  agregar_arista(1,p); //bloque central
14     p:=rotar90d(p); agregar_arista(2,p);
15     p:=rotar90d(p); agregar_arista(3,p);
16     p:=rotar90d(p); agregar_arista(4,p);
17     agregar_vertice; //v1 lateral izquierdo
18     p:=point(-i, 0); agregar_arista(0,p);
19     p:=point( i,-i); agregar_arista(2,p);
20     p:=point( i, i); agregar_arista(4,p);
21     agregar_vertice; //v2 lateral superior
22     p:=point( 0,-i); agregar_arista(0,p);
23     p:=point(-i, i); agregar_arista(1,p);
24     p:=point( i, i); agregar_arista(3,p);
25     agregar_vertice; //v3 lateral derecho
26     p:=point( i, 0); agregar_arista(0,p);
27     p:=point(-i,-i); agregar_arista(2,p);
28     p:=point(-i, i); agregar_arista(4,p);
29     agregar_vertice; //v4 lateral inferior
30     p:=point( 0, i); agregar_arista(0,p);
31     p:=point(-i,-i); agregar_arista(1,p);
32     p:=point( i,-i); agregar_arista(3,p);
33     construir_grafomata;
34    end;
35   result:=g;
36 end;
```

C.4.34. Grafómata dual cuadrados y hexágonos

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v1	a4	(240,0)
a1	v1	a4	(240,0)	a1	v0	a4	(240,0)	a2	v2	a3	(120,120)
a2	v2	a3	(120,120)	a2	v0	a6	(0,240)	a3	v1	a5	(0,240)
a3	v1	a5	(0,240)	a3	v2	a4	(-120,120)	a4	v1	a1	(-240,0)
a4	v1	a1	(-240,0)	a4	v0	a1	(-240,0)	a5	v2	a1	(-120,-120)
a5	v2	a1	(-120,-120)	a5	v0	a3	(0,-240)	a6	v1	a2	(0,-240)
a6	v1	a2	(0,-240)	a6	v2	a2	(120,-120)	V1			
V2								a3	v2	a4	(-120,120)
a0	v2	a0	(0,0)					a6	v2	a2	(120,-120)
a1	v0	a5	(120,120)					V2			
a2	v1	a6	(-120,120)								
a3	v0	a2	(-120,-120)								
a4	v1	a3	(120,-120)								

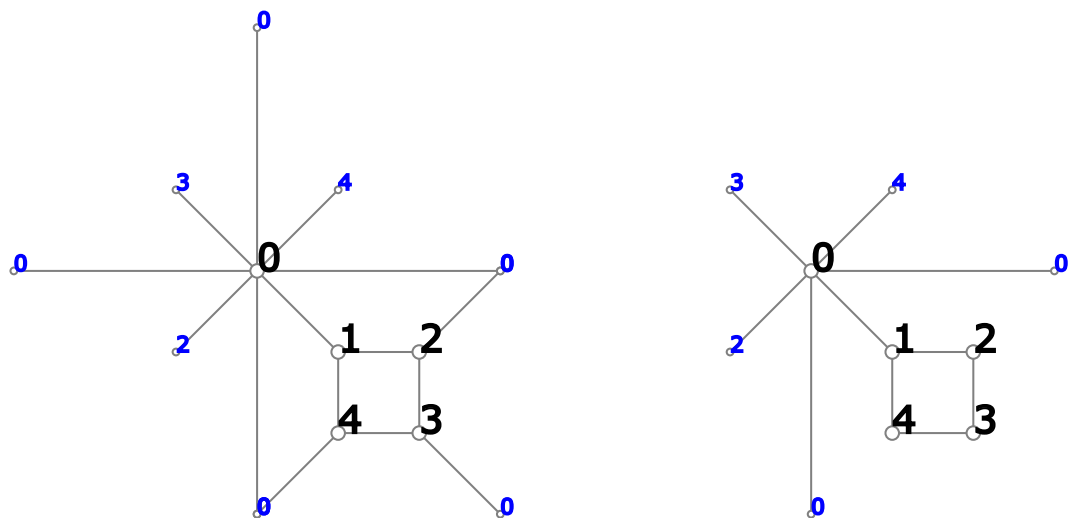
Figura C.45: Patrón gráfico del grafómata dual cuadrados y hexágonos



C.4.35. Grafómata Subcuadros o Waffles

Estructura de grafómata					Estructura de dibúgrafo				
V0					V0				
a0	v0	a0	(0,0)		a1	v0	a5	(360,0)	
a1	v0	a5	(360,0)		a2	v1	a3	(120,120)	
a2	v1	a3	(120,120)		a3	v0	a7	(0,360)	
a3	v0	a7	(0,360)		a4	v2	a3	(-120,120)	
a4	v2	a3	(-120,120)		a6	v3	a1	(-120,-120)	
a5	v0	a1	(-360,0)		a8	v4	a2	(120,-120)	
a6	v3	a1	(-120,-120)		V1				
a7	v0	a3	(0,-360)		a1	v2	a2	(120,0)	
a8	v4	a2	(120,-120)		a2	v4	a3	(0,120)	
V1					V2				
a0	v1	a0	(0,0)		a0	v2	a0	(0,0)	
a1	v2	a2	(120,0)		a1	v3	a3	(0,120)	
a2	v4	a3	(0,120)		a2	v1	a1	(-120,0)	
a3	v0	a2	(-120,-120)		a3	v0	a4	(120,-120)	
V3					V4				
a0	v3	a0	(0,0)		a0	v4	a0	(0,0)	
a1	v0	a6	(120,120)		a1	v3	a2	(120,0)	
a2	v4	a1	(-120,0)		a2	v0	a8	(-120,120)	
a3	v2	a1	(0,-120)		a3	v1	a2	(0,-120)	

Figura C.46: Patrón gráfico del grafómata subcuadros



Fuente en Pascal C.25: Subcuadros

```

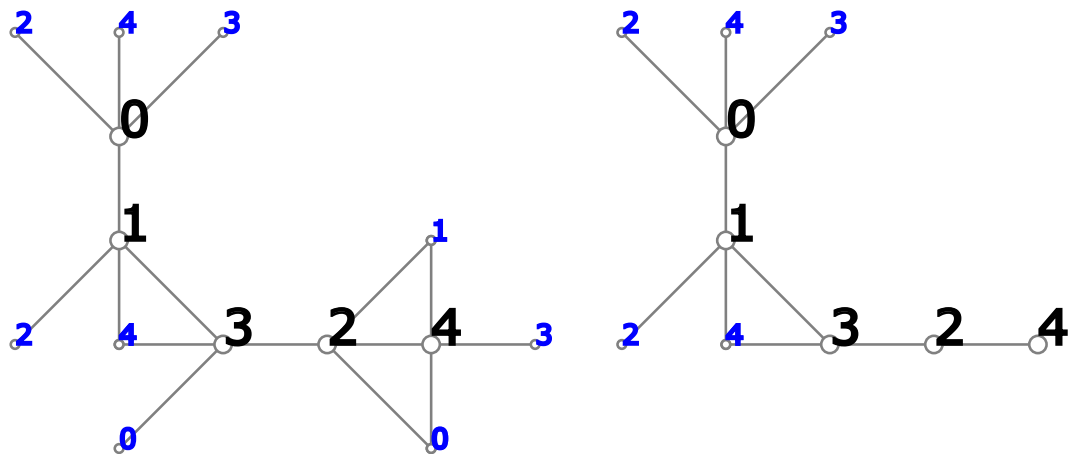
1 function Noregular8:Tgrafomata;
2 var
3   i,j:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   i:=numero_magico;
8   j:=3*i;
9   g:=Tgrafomata.create;
10  with g do
11    begin
12      nombre:='Subcuadro';
13      agregar_vertice; //v0
14      p:=point( 0,j);  agregar_arista(0,p);
15      p:=rotar90d(p);  agregar_arista(0,p);
16      p:=rotar90d(p);  agregar_arista(0,p);
17      p:=rotar90d(p);  agregar_arista(0,p);
18      p:=point(i,i);   agregar_arista(1,p);
19      p:=rotar90d(p);   agregar_arista(2,p);
20      p:=rotar90d(p);   agregar_arista(3,p);
21      p:=rotar90d(p);   agregar_arista(4,p);
22      agregar_vertice; //v1
23      p:=point(-i,-i);  agregar_arista(0,p);
24      p:=point( i, 0);  agregar_arista(2,p);
25      p:=point( 0, i);  agregar_arista(4,p);
26      agregar_vertice; //v2
27      p:=point( i,-i);  agregar_arista(0,p);
28      p:=point(-i, 0);  agregar_arista(1,p);
29      p:=point( 0, i);  agregar_arista(3,p);
30      agregar_vertice; //v3
31      p:=point( i, i);  agregar_arista(0,p);
32      p:=point(-i, 0);  agregar_arista(4,p);
33      p:=point( 0,-i);  agregar_arista(2,p);
34      agregar_vertice; //v4
35      p:=point(-i, i);  agregar_arista(0,p);
36      p:=point( i, 0);  agregar_arista(3,p);
37      p:=point( 0,-i);  agregar_arista(1,p);
38      construir_grafomata;
39    end;
40    result:=g;
41 end;

```

C.4.36. Grafómata dual subcuadros

Estructura de grafómata				Estructura de dibúgrafo			
V0				V1			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)
a1	v1	a4	(0,120)	a1	v3	a4	(120,120)
a2	v2	a2	(-120,-120)	a2	v4	a4	(0,120)
a3	v4	a2	(0,-120)	a3	v2	a4	(-120,120)
a4	v3	a2	(120,-120)	a4	v0	a1	(0,-120)
V2				V3			
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)
a1	v4	a3	(120,0)	a1	v2	a3	(120,0)
a2	v0	a2	(120,120)	a2	v0	a4	(-120,120)
a3	v3	a1	(-120,0)	a3	v4	a1	(-120,0)
a4	v1	a3	(120,-120)	a4	v1	a1	(-120,-120)
V4				V0			
a0	v4	a0	(0,0)	a1	v1	a4	(0,120)
a1	v3	a3	(120,0)	a2	v2	a2	(-120,-120)
a2	v0	a3	(0,120)	a3	v4	a2	(0,-120)
a3	v2	a1	(-120,0)	a4	v3	a2	(120,-120)
a4	v1	a2	(0,-120)	V1			
				a1	v3	a4	(120,120)
				a2	v4	a4	(0,120)
				a3	v2	a4	(-120,120)
				V2			
				a1	v4	a3	(120,0)
				a3	v3	a1	(-120,0)
				V3			
				a3	v4	a1	(-120,0)
				V4			

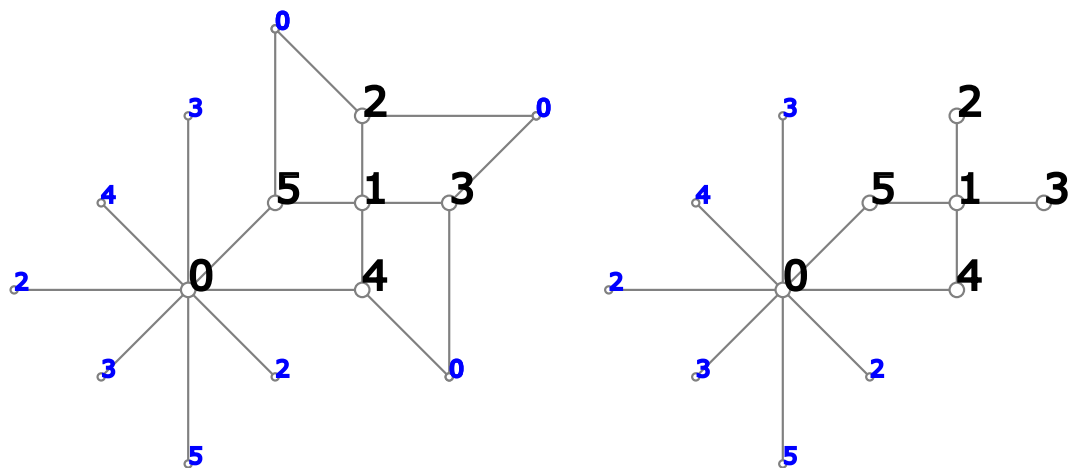
Figura C.47: Patrón gráfico del grafómata dual subcuadros



C.4.37. Grafómata Molinetes

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v4	a2	(240,0)
a1	v4	a2	(240,0)	a1	v3	a2	(120,0)	a2	v2	a3	(120,120)
a2	v2	a3	(120,120)	a2	v4	a3	(0,120)	a3	v5	a3	(0,240)
a3	v5	a3	(0,240)	a3	v5	a1	(-120,0)	a4	v3	a3	(-120,120)
a4	v3	a3	(-120,120)	a4	v2	a2	(0,-120)	a5	v2	a1	(-240,0)
a5	v2	a1	(-240,0)					a6	v4	a1	(-120,-120)
a6	v4	a1	(-120,-120)					a7	v3	a1	(0,-240)
a7	v3	a1	(0,-240)					a8	v5	a2	(120,-120)
a8	v5	a2	(120,-120)					V1			
V2				V3				a1	v3	a2	(120,0)
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a2	v4	a3	(0,120)
a1	v0	a5	(240,0)	a1	v0	a7	(0,240)	a3	v5	a1	(-120,0)
a2	v1	a4	(0,120)	a2	v1	a1	(-120,0)	a4	v2	a2	(0,-120)
a3	v0	a2	(-120,-120)	a3	v0	a4	(120,-120)	V2			
V4				V5				V3			
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	V4			
a1	v0	a6	(120,120)	a1	v1	a3	(120,0)	V5			
a2	v0	a1	(-240,0)	a2	v0	a8	(-120,120)				
a3	v1	a2	(0,-120)	a3	v0	a3	(0,-240)				

Figura C.48: Patrón gráfico del grafómata Molinetes



Fuente en Pascal C.26: Molinetes

```

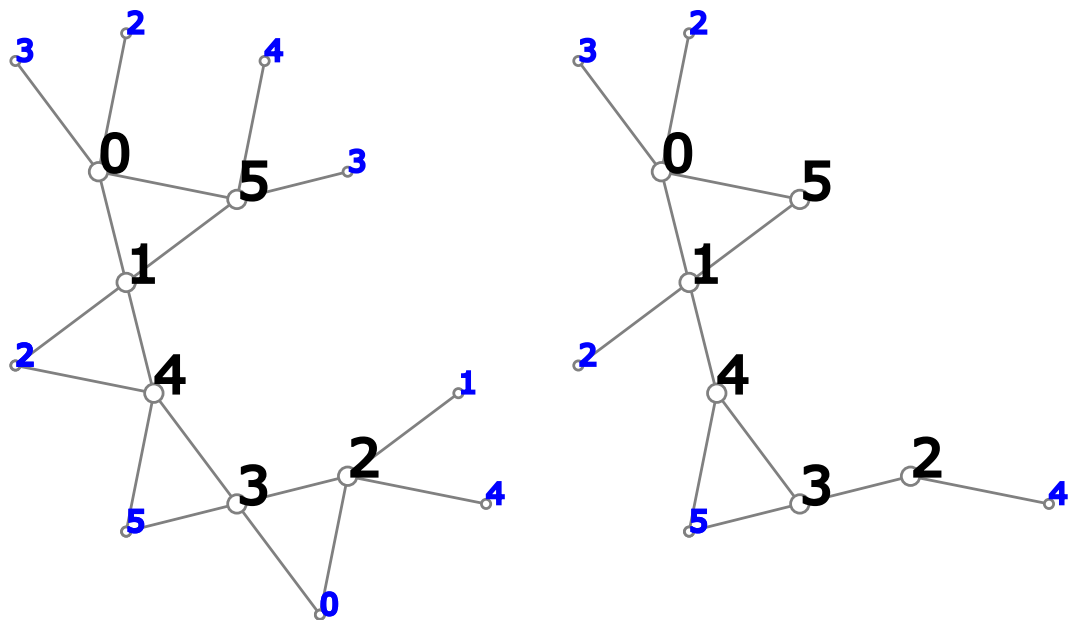
1 function Noregular9:Tgrafomata;
2 var
3   i,j:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   i:=numero_magico;
8   j:=i*2;
9   g:=Tgrafomata.create;
10  with g do
11    begin
12      nombre:='Molinetes';
13      agregar_vertice; //v0
14      p:=point( 0,-j); agregar_arista(3,p);
15      p:=point( i,-i); agregar_arista(5,p);
16      p:=point( j, 0); agregar_arista(4,p);
17      p:=point( i, i); agregar_arista(2,p);
18      p:=point( 0, j); agregar_arista(5,p);
19      p:=point(-i, i); agregar_arista(3,p);
20      p:=point(-j, 0); agregar_arista(2,p);
21      p:=point(-i,-i); agregar_arista(4,p);
22      agregar_vertice; //v1
23      p:=point( 0,-i); agregar_arista(2,p);
24      p:=point( i, 0); agregar_arista(3,p);
25      p:=point( 0, i); agregar_arista(4,p);
26      p:=point(-i, 0); agregar_arista(5,p);
27      agregar_vertice; //v2
28      p:=point( 0, i); agregar_arista(1,p);
29      p:=point( j, 0); agregar_arista(0,p);
30      p:=point(-i,-i); agregar_arista(0,p);
31      agregar_vertice; //v3
32      p:=point(-i, 0); agregar_arista(1,p);
33      p:=point( i,-i); agregar_arista(0,p);
34      p:=point( 0, j); agregar_arista(0,p);
35      agregar_vertice; //v4
36      p:=point( 0,-i); agregar_arista(1,p);
37      p:=point( i, i); agregar_arista(0,p);
38      p:=point(-j, 0); agregar_arista(0,p);
39      agregar_vertice; //v5
40      p:=point( i, 0); agregar_arista(1,p);
41      p:=point(-i, i); agregar_arista(0,p);
42      p:=point( 0,-j); agregar_arista(0,p);
43      construir_grafomata;
44    end;
45    result:=g;
46  end;

```

C.4.38. Grafómata dual molinetes

Estructura de grafómata								Estructura de dibúgrafo			
V0				V1				V0			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)	a1	v5	a2	(150,30)
a1	v5	a2	(150,30)	a1	v4	a4	(30,120)	a2	v1	a3	(30,120)
a2	v1	a3	(30,120)	a2	v2	a4	(-120,90)	a3	v3	a1	(-90,-120)
a3	v3	a1	(-90,-120)	a3	v0	a2	(-30,-120)	a4	v2	a2	(30,-150)
a4	v2	a2	(30,-150)	a4	v5	a1	(120,-90)	V1			
V2				V3				a1			
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)	a1	v4	a4	(30,120)
a1	v4	a3	(150,30)	a1	v0	a3	(90,120)	a2	v2	a4	(-120,90)
a2	v0	a4	(-30,150)	a2	v5	a4	(-120,30)	a4	v5	a1	(120,-90)
a3	v3	a4	(-120,30)	a3	v4	a1	(-90,-120)	V2			
a4	v1	a2	(120,-90)	a4	v2	a3	(120,-30)	a1	v4	a3	(150,30)
V4				V5				a3			
a0	v4	a0	(0,0)	a0	v5	a0	(0,0)	a2	v5	a4	(-120,30)
a1	v3	a3	(90,120)	a1	v1	a4	(-120,90)	a3	v4	a1	(-90,-120)
a2	v5	a3	(-30,150)	a2	v0	a1	(-150,-30)	V4			
a3	v2	a1	(-150,-30)	a3	v4	a2	(30,-150)	a2	v5	a3	(-30,150)
a4	v1	a1	(-30,-120)	a4	v3	a2	(120,-30)	V5			

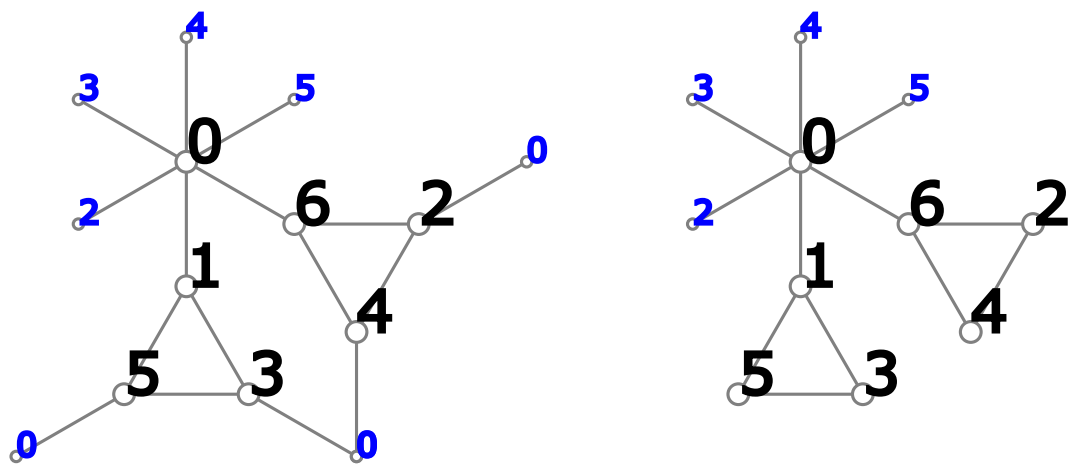
Figura C.49: Patrón gráfico del grafómata dual Molinetes



C.4.39. Grafómata triángulos y hexágonos

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v6	a3	(104,60)
a1	v6	a3	(104,60)	a2	v1	a3	(0,120)
a2	v1	a3	(0,120)	a3	v2	a3	(-104,60)
a3	v2	a3	(-104,60)	a4	v3	a1	(-104,-60)
a4	v3	a1	(-104,-60)	a5	v4	a1	(0,-120)
a5	v4	a1	(0,-120)	a6	v5	a2	(104,-60)
a6	v5	a2	(104,-60)	V1			
V1				a1	v3	a3	(60,104)
a0	v1	a0	(0,0)	a2	v5	a3	(-60,104)
a1	v3	a3	(60,104)	V2			
a2	v5	a3	(-60,104)	a1	v4	a3	(-60,104)
a3	v0	a2	(0,-120)	a2	v6	a1	(-120,0)
V3				a3	v0	a3	(104,-60)
a0	v3	a0	(0,0)	V4			
a1	v0	a4	(104,60)	a0	v4	a0	(0,0)
a2	v5	a1	(-120,0)	a1	v0	a5	(0,120)
a3	v1	a1	(-60,-104)	a2	v6	a2	(-60,-104)
V5				a3	v2	a1	(60,-104)
a0	v5	a0	(0,0)	V6			
a1	v3	a2	(120,0)	a0	v6	a0	(0,0)
a2	v0	a6	(-104,60)	a1	v2	a2	(120,0)
a3	v1	a2	(60,-104)	a2	v4	a2	(60,104)
				a3	v0	a1	(-104,-60)

Figura C.50: Patrón gráfico del grafómata triángulos-hexágonos



Fuente en Pascal C.27: triángulos y hexágonos

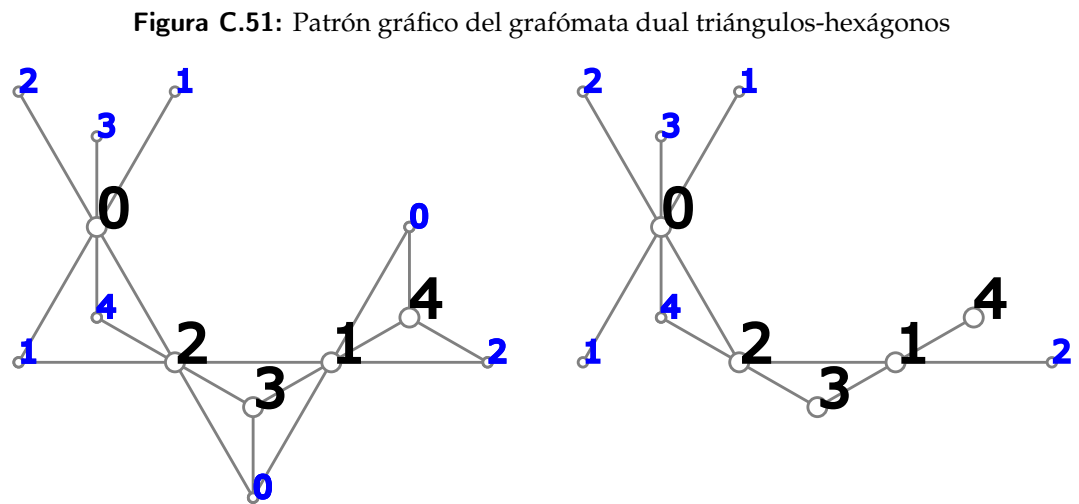
```

1 function Noregular10:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   g:=Tgrafomata.create;
8   with g do
9     begin
10      nombre:='triangulos_hexagonos';
11      p:=point(0,numero_magico); //ok!
12      agregar_vertice; //v0
13      agregar_arista(1,p);
14      for i:=1 to 5 do agregar_rotacion_arista(0,1,i+1,60*i); //rotacion de una unica arista sin
          errores intermedios posibles
15      p:=reflejar_xy(punto_arista(0,1));
16      agregar_vertice; //v1
17      agregar_arista(0,p);
18      agregar_arista(3,rotar(150,p));
19      agregar_arista(5,rotar(210,p));
20      p:=reflejar_xy(punto_arista(0,2));
21      agregar_vertice; //v2
22      agregar_arista(0,p);
23      agregar_arista(4,rotar(150,p));
24      agregar_arista(6,rotar(210,p));
25      p:=reflejar_xy(punto_arista(0,3));
26      agregar_vertice; //v3
27      agregar_arista(0,p);
28      agregar_arista(5,rotar(150,p));
29      agregar_arista(1,rotar(210,p));
30      p:=reflejar_xy(punto_arista(0,4));
31      agregar_vertice; //v4
32      agregar_arista(0,p);
33      agregar_arista(6,rotar(150,p));
34      agregar_arista(2,rotar(210,p));
35      p:=reflejar_xy(punto_arista(0,5));
36      agregar_vertice; //v5
37      agregar_arista(0,p);
38      agregar_arista(1,rotar(150,p));
39      agregar_arista(3,rotar(210,p));
40      p:=reflejar_xy(punto_arista(0,6));
41      agregar_vertice; //v6
42      agregar_arista(0,p);
43      agregar_arista(2,rotar(150,p));
44      agregar_arista(4,rotar(210,p));
45
46      construir_grafomata;
47      end;
48      result:=g;
49 end;

```

C.4.40. Grafómata dual triángulos y hexágonos

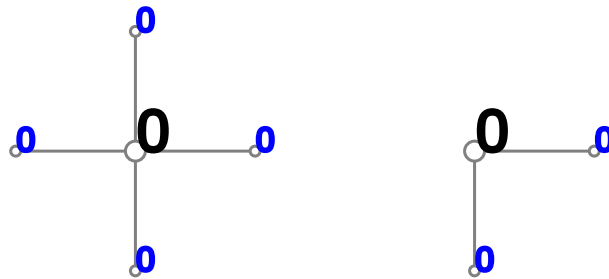
Estructura de grafómata				Estructura de dibúgrafo			
V0				V1			
a0	v0	a0	(0,0)	a0	v1	a0	(0,0)
a1	v2	a6	(82,142)	a1	v2	a4	(164,0)
a2	v4	a3	(0,95)	a2	v0	a6	(-82,142)
a3	v1	a5	(-82,142)	a3	v3	a3	(-82,47)
a4	v2	a3	(-82,-142)	a4	v2	a1	(-164,0)
a5	v3	a1	(0,-95)	a5	v0	a3	(82,-142)
a6	v1	a2	(82,-142)	a6	v4	a2	(82,-47)
V2				V3			
a0	v2	a0	(0,0)	a0	v3	a0	(0,0)
a1	v1	a4	(164,0)	a1	v0	a5	(0,95)
a2	v3	a2	(82,47)	a2	v2	a2	(-82,-47)
a3	v0	a4	(82,142)	a3	v1	a3	(82,-47)
a4	v1	a1	(-164,0)	a4	v4	a6	(-82,142)
a5	v4	a1	(-82,-47)	a5	v3	a3	(-82,142)
a6	v0	a1	(-82,-142)	a6	v2	a6	(-82,-142)
V3				V4			
a0	v3	a0	(0,0)	a0	v4	a0	(0,0)
a1	v0	a5	(0,95)	a1	v2	a5	(82,47)
a2	v2	a2	(-82,-47)	a2	v1	a6	(-82,47)
a3	v1	a3	(82,-47)	a3	v0	a2	(0,-95)



C.4.41. Grafómata 4.4.4.4, malla cuadrada, vecindad de Von Newman

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v0	a3	(120,0)
a1	v0	a3	(120,0)	a2	v0	a4	(0,120)
a2	v0	a4	(0,120)				
a3	v0	a1	(-120,0)				
a4	v0	a2	(0,-120)				

Figura C.52: Patrón gráfico del grafómata vecindario de Von Newman



Fuente en Pascal C.28: 4.4.4.4

```

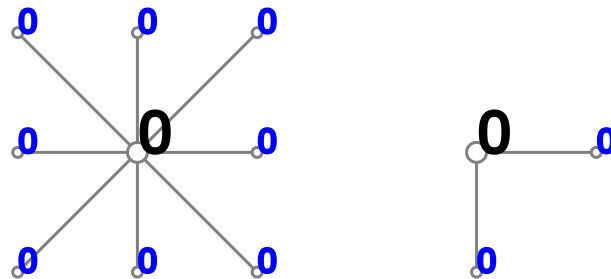
1 function Regular2a:Tgrafomata;
2 var
3   i:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   g:=Tgrafomata.create;
8   with g do
9     begin
10      nombre:='4444';
11      agregar_vertice; //v0
12      p:=point(numero_magico,0); agregar_arista(vertice_reciente,p);
13      for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,i,vertice_reciente,rotar_90d);
14      construir_grafomata;
15    end;
16    result:=g;
17 end;
```

En la malla cuadrada los grafomatas base y dual tienen la misma estructura, difieren en que una está desplazada con respecto a la otra $\frac{|a_k|}{2}$ unidades en cada eje. Este grafomata implementa funcionalmente a la malla de Von Neuman y sirve de base estructural a los grafomatas que implementan los vecindarios de Moore.

C.4.42. Grafómata vecindario de Moore 9 celdas

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v0	a3	(120,0)
a1	v0	a3	(120,0)	a2	v0	a4	(0,120)
a2	v0	a4	(0,120)				
a3	v0	a1	(-120,0)				
a4	v0	a2	(0,-120)				
a5 v0 a0 (120,120)							
a6 v0 a0 (-120,120)							
a7 v0 a0 (-120,-120)							
a8 v0 a0 (120,-120)							

Figura C.53: Patrón gráfico del grafómata vecindario Moore de 9 celdas



Fuente en Pascal C.29: Construcción de vecindario de Moore de 9 celdas

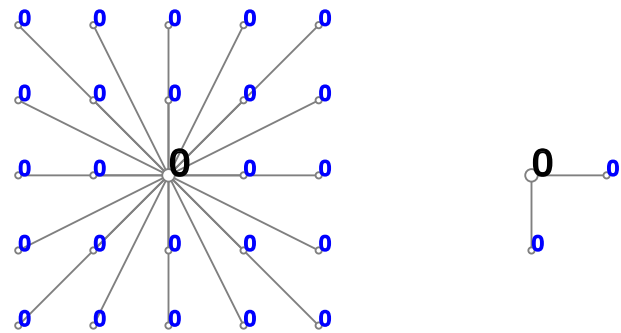
```

1 function Regular2b1:Tgrafomata;
2 var
3   i,j:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   j:=numero_magico;
8   g:=Tgrafomata.create;
9   with g do
10    begin
11     nombre:='4444 Moore 9';
12     //a esta parte del grafómata es posible calcularle su grafómata dual
13     agregar_vertice; //v0
14     p:=point(j,0); agregar_arista(vertice_reciente,p);
15     for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,i,vertice_reciente,rotar_90d);
16     construir_grafomata; //hasta aquí igual que una malla de Von Newman
17 {Esta parte del grafómata no tiene parte en el grafómata dual. Aquí se termina de crear el
18   grafómata, es la capa de aristas que se cruzan}
19     configurar_modo_aristas:=arista_conectiva;
20     p:=point(j,j); agregar_arista(vertice_reciente,p); //línea de conexión con inclinación de 45°
21     for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,arista_reciente,vertice_reciente,
22       rotar_90d);
23   end;
24   result:=g;
25 end;
```

C.4.43. Grafómata vecindario de Moore 25 celdas

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a1	v0	a3	(120,0)
a1	v0	a3	(120,0)	a2	v0	a4	(0,12)
a2	v0	a4	(0,120)				
a3	v0	a1	(-120,0)				
a4	v0	a2	(0,-120)				
a5	v0	a0	(120,120)				
a6	v0	a0	(-120,120)				
a7	v0	a0	(-120,-120)				
a8	v0	a0	(120,-120)				
a9	v0	a0	(240,0)				
a10	v0	a0	(0,240)				
a11	v0	a0	(-240,0)				
a12	v0	a0	(0,-240)				
a13	v0	a0	(240,240)				
a14	v0	a0	(-240,240)				
a15	v0	a0	(-240,-240)				
a16	v0	a0	(240,-240)				
a17	v0	a0	(240,120)				
a18	v0	a0	(-120,240)				
a19	v0	a0	(-240,-120)				
a20	v0	a0	(120,-240)				
a21	v0	a0	(120,240)				
a22	v0	a0	(-240,120)				
a23	v0	a0	(-120,-240)				
a24	v0	a0	(240,-120)				

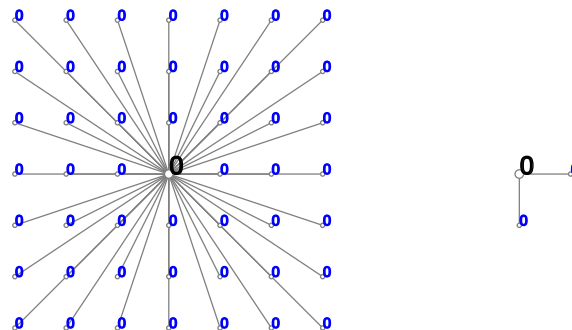
Figura C.54: Patrón gráfico del grafómata vecindario Moore de 25 celdas



C.4.44. Grafómata vecindario de Moore 49 celdas

Estructura de grafómata				Estructura de dibúgrafo			
V0				V0			
a0	v0	a0	(0,0)	a25	v0	a0	(360,0)
a1	v0	a3	(120,0)	a26	v0	a0	(0,360)
a2	v0	a4	(0,120)	a27	v0	a0	(-360,0)
a3	v0	a1	(-120,0)	a28	v0	a0	(0,-360)
a4	v0	a2	(0,-120)	a29	v0	a0	(360,360)
a5	v0	a0	(120,120)	a30	v0	a0	(-360,360)
a6	v0	a0	(-120,120)	a31	v0	a0	(-360,-360)
a7	v0	a0	(-120,-120)	a32	v0	a0	(360,-360)
a8	v0	a0	(120,-120)	a33	v0	a0	(360,120)
a9	v0	a0	(240,0)	a34	v0	a0	(-120,360)
a10	v0	a0	(0,240)	a35	v0	a0	(-360,-120)
a11	v0	a0	(-240,0)	a36	v0	a0	(120,-360)
a12	v0	a0	(0,-240)	a37	v0	a0	(120,360)
a13	v0	a0	(240,240)	a38	v0	a0	(-360,120)
a14	v0	a0	(-240,240)	a39	v0	a0	(-120,-360)
a15	v0	a0	(-240,-240)	a40	v0	a0	(360,-120)
a16	v0	a0	(240,-240)	a41	v0	a0	(360,240)
a17	v0	a0	(240,120)	a42	v0	a0	(-240,360)
a18	v0	a0	(-120,240)	a43	v0	a0	(-360,-240)
a19	v0	a0	(-240,-120)	a44	v0	a0	(240,-360)
a20	v0	a0	(120,-240)	a45	v0	a0	(240,360)
a21	v0	a0	(120,240)	a46	v0	a0	(-360,240)
a22	v0	a0	(-240,120)	a47	v0	a0	(-240,-360)
a23	v0	a0	(-120,-240)	a48	v0	a0	(360,-240)
a24	v0	a0	(240,-120)				

Figura C.55: Patrón gráfico del grafómata vecindario Moore de 49 celdas



C.4.45. Generador de vecindarios de Moore

Se implementó una función en pascal que facilita la creación automática de grafómatas representando las relaciones entre vértices en el espacio cuando se usan vecindarios de Moore. El algoritmo permite crear vecindarios de Moore de N niveles, en el prototipo se hace uso de vecindarios de este tipo desde uno hasta de seis niveles: $(2n + 1)^2 = \{9, 25, 49, 81, 121, 169, \dots\}$. Serie A016754 de OEIS.

En la figura C.56 se muestra el diagrama de flujo de operaciones. El código fuente se muestra en el listado C.30 en la página 403.

Figura C.56: Diagrama de algoritmo de construcción de vecindarios de Moore

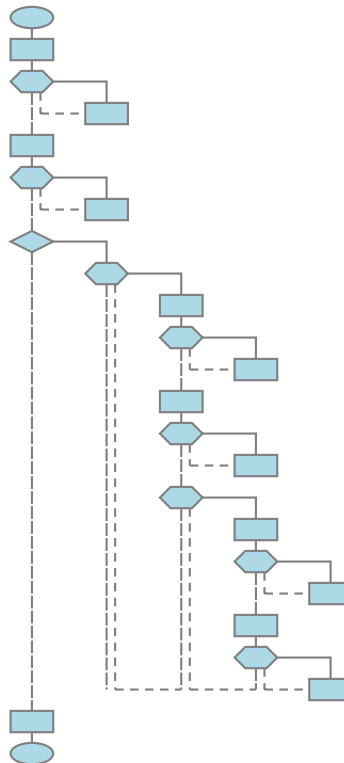


Diagrama de flujo del algoritmo usado para la construcción automática de vecindarios de Moore implementados en PH2.exe. El código fuente asociado se presenta en el listado C.30 en la página 403. El primer ciclo construye las conexiones estructurales, una malla 4.4.4.4. Los restantes ciclos establecen la capa de conectividad según el alcance establecido para el vecindario.

Fuente: Autor.

Fuente en Pascal C.30: Construcción de vecindarios de Moore de N niveles

```

1 function Regular2bn(n:integer):Tgrafomata;
2 var
3   i,j,k,l,m,q:integer;
4   g:Tgrafomata;
5   p:tpoint;
6 begin
7   j:=numero_magico;
8   g:=Tgrafomata.create;
9   with g do
10    begin
11     m:=(2*n+1);
12     m:=m*m;
13     nombre:='4444 Moore '+inttostr(m); //(2*i+1)^2
14     //A esta parte del grafómata es posible calcularle su grafómata dual
15     agregar_vertice; //v0
16     p:=point(j,0);  agregar_arista(vertice_reciente,p);
17     for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,i,vertice_reciente,rotar_90d);
18     construir_grafomata; //hasta aquí igual que una malla de Von Newman
19 {Esta parte del grafómata no tiene parte en el grafómata dual. Aquí se termina de crear el
    grafómata
20 se implementan las aristas que se cruzan
21 primera sección cruzada, malla Moore de 9}
22   configurar_modos_aristas:=arista_conectiva;
23   p:=point(j,j);  agregar_arista(vertice_reciente,p); //línea de conexión con inclinación de 45
    grados
24   for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,arista_reciente,vertice_reciente,
    rotar_90d);
25 {Segunda sección cruzada, ampliación de alcance}
26 //ciclo para agregar conexiones por capas
27   if n>1 then
28     for k:=2 to n do
29       begin
30 //uniones diagonales
31         m:=k*j;
32         p:=point(m,0);  agregar_arista(vertice_reciente,p); //uniones a 90°
33         for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,arista_reciente,vertice_reciente,
    rotar_90d);
34         p:=point(m,m);  agregar_arista(vertice_reciente,p); //uniones a 45°
35         for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,arista_reciente,vertice_reciente,
    rotar_90d);
36 //uniones diagonales intermedias
37         for l:=2 to k do //uniones en ángulos menores que 30°
38           begin
39             q:=j*(l-1);
40             p:=point(m,q);  agregar_arista(vertice_reciente,p);
41             for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,arista_reciente,vertice_reciente,
    rotar_90d);
42             p:=point(m,q);  agregar_arista(vertice_reciente,p);
43             for i:=1 to 3 do agregar_reflexion_arista(vertice_reciente,arista_reciente,vertice_reciente,
    rotar_90d);
44           end;
45         end;
46       end;
47     result:=g;
48 end;

```

C.5. Caracterización de grafómatas

A partir del conjunto implementado de grafómatas se producen las siguientes observaciones:

- Cada grafómata produce un teselado único, si dos grafómatas producen el mismo teselado entonces los grafómatas comparados son estructuralmente iguales⁸.
- Si a partir de un grafómata g se crea otro al cual se le realiza una operación de cambio de orientación de sus aristas sin cambiar su simetría, aunque se produzcan teselados visualmente diferentes los grafómatas comparados tienen la misma estructura subyacente. P.ej: el grafómata 4.8.8 en la sección C.4.1 en la página 339 y la variación de 4.8.8 en la sección C.4.21 en la página 370.
- Si a partir de un grafómata g se crea otro al que se aplican operaciones de traslación, escalado, rotación y demás operaciones tales que alteren la simetría del sistema, aunque los grafómatas comparados produzcan teselados semejantes discrepan en su composición estructural. P.ej: el grafómata 3.3.3.3.3 en la sección C.4.1 en la página 339 y la variación de triángulos en la sección C.4.21 en la página 370.
- Cada grafómata g tiene un dual único $\bar{g}$. En condiciones ideales, el grafómata dual de $\bar{g}$ es el grafómata origen g .
- En los grafómatas siempre hay 3 o mas aristas por cada vertice del sistema.
- La representación grafica de un grafómata depende del orden de enlace de los vertices. Para una estructura con n vertices, se propone que existen (C.1) representaciones diferentes.

$$\prod_{i=1}^v k_i \leq x \leq a_1 \cdot \prod_{i=2}^v (a_i - 1) \mid a_i \geq 3 \text{ y } k_i \leq a_i \quad (\text{C.1})$$

k_i es la cantidad de vertices distintos a los que puede pasar el vertice i y a_i es el total de aristas presentes en el vertice i . El valor exacto es un problema de combinatoria, está en función del total de vertices, la cantidad de aristas y la estructura las conexiones.

- Un grafómata y su dual comparten la misma estructura de malla regular. La diferencia entre mallas se relaciona con la traslación de los vertices.
- Cada grafómata tiene asociado un dibúgrafo. Al igual que con los grafómatas, la selección de aristas en cada vértice permite la existencia de múltiples configuraciones gráficas del dibúgrafo. La malla regular funciona en todas las variantes posibles.

Al analizar la estructura de los grafómatas disponibles, es posible establecer relaciones numéricas que completan la descripción de los elementos componentes de los grafómatas y sus dibúgrafos. Teniendo en cuenta que las aristas son las uniones entre los vértices requeridas por la estructura y

⁸La igualdad se limita a configuración de los vertices y las aristas estructurales.

que los trazos son las líneas necesarias para la construcción grafica de un grafómata, se definen las siguientes elementos:

v_g Total de vértices presentes en el grafómata.

t_g Total de trazos presentes en la estructura, líneas únicas requeridas para el trazado gráfico de la estructura.

a_e Total de aristas que van a vértices fuera de la forma principal del grafómata. Este es un valor par.

a_s Total de aristas que van a vértices dentro de la forma principal del grafomata, estas aristas se solapan con sus antipodas, por eso su cantidad es siempre par. Por definicion en los dibúgrafos no hay aristas que se superpongan.

a_i Total de aristas requeridas para definir una estructura de grafómata o $a_i = a_e + a_s$. Ya que a_s y a_e son pares entonces a_i es también par.

v_d Total de vértices requeridos para la construcción del dibúgrafo. Ya que en el dibugrafo no se hace uso de todos los vértices, $v_d \leq v_g$.

t_d Total de trazos requeridos para la construcción gráfica del dibúgrafo. El total de aristas requeridas por el dibúgrafo es igual a la cantidad de trazos: $a_d = t_d$, el total de trazos de un dibúgrafo es menor que los requeridos para el trazado del grafómata que lo genera: $t_d < t_g$.

$a_i(g)$ Función que determina el total de aristas estructurales de un grafomata g .

$t_d(d)$ Función que determina el total de trazos de un dibúgrafo d .

A partir de estas definiciones se establecen las siguientes relaciones:

- El total de trazos de un grafómata es igual a la suma del total de aristas a vértices externos y la mitad de las aristas entre vértices internos o propios del grafómata.

$$t_g = a_e + \frac{a_s}{2} \quad (C.2)$$

- El total de trazos en un dibúgrafo es igual al promedio de aristas necesarias para construir estructuralmente al grafómata que lo genera.

$$t_d = \frac{(a_e + a_s)}{2} \quad (C.3)$$

- El total de aristas requeridos por un grafómata es igual al doble de trazos requeridos por su dibúgrafo.

$$a_i = 2t_d \quad (C.4)$$

- El total de aristas estructurales de un grafómata es igual al total de aristas estructurales de su grafómata dual

$$a_i(g_b) = a_i(g_d) \quad (C.5)$$

con g_b y g_d como grafómatas base y dual.

- El total de trazos de un dibúgrafo es igual al total de trazos de su dibúgrafo dual.

$$t_d(d_b) = t_d(d_d) \quad (C.6)$$

con d_b y d_d como dibúgrafos base y dual.

- La relación entre el total de aristas estructurales y la cantidad de vértices y trazos presentes en un grafómata es:

$$a_i = v_g + t_g + \{-1, 0, \underline{1}\} \quad (C.7)$$

Esta relación se explica mejor mediante

$$a_i = v_g + t_g + f(v_g, t_g) \quad (C.8)$$

donde

$$f(v_g, t_g) = \begin{cases} 0 & \text{si } \text{par}(v_g) = \text{par}(t_g), \\ \{\underline{1}, -1\} & \text{si } \text{par}(v_g) \neq \text{par}(t_g) \end{cases} \quad (C.9)$$

con

$$\text{par}(x) = \begin{cases} \text{par} & x \in 2n, \\ \text{impar} & x \in 2n + 1 \end{cases} \quad (C.10)$$

La cantidad subrayada corresponde con el valor de ajuste más común, +1. En los grafómatas listados en la tabla C.2, $f(v_g, t_g)$ es negativo en Triángulos y hexágonos, la variación 3.12.12, y el dual de 4.16.12. En este proyecto no se encontró una mejor función de ajuste que explique la diferencia en la adición de términos de la ecuación C.8.

En el anexo E.6.1 en la página 635 y la tabla C.2 en la página siguiente se da soporte a las afirmaciones presentadas mediante la corroboración numérica de las distintas relaciones establecidas. Las relaciones presentadas se predicen en la teoría de grafos, donde se establece que para dos grafos planares mutuamente duales $G = (V, E)$ y $G' = (V', E')$ cuyos conjuntos de regiones son R y R' , se cumple que $|E'| = |E|$, $|V'| = |R|$ y $|R'| = |V|$.

Tabla C.2: Distribución de vértices, trazos y aristas de los gráfomatas y dibúgrafos disponibles en PH2

Id	Gráfomata	Sistema base b						Sistema dual d							
		g_b			d_b			g_d			d_d				
		v_g	t_g	a_i	a_e	a_s	v_d	t_d	v_g	t_g	a_i	a_e	a_s	v_d	t_d
1	3.3.3.3.3.3	1	6	6	6	0	1	3	2	5	6	4	2	1	3
2		4.4.4.4	1	4	4	4	0	1	2	1	4	4	4	0	1
3	6.6.6	2	5	6	4	2	1	3	1	6	6	6	0	1	3
4	3.3.3.4.4	2	9	10	8	2	2	5	3	8	10	6	4	2	5
5	3.3.4.3.4	6	15	20	10	10	4	10	4	17	20	14	6	3	10
6	4.8.8	4	8	12	4	8	3	6	2	11	12	10	2	1	6
7	3.6.3.6	3	9	12	6	6	2	6	3	10	12	8	4	1	6
8	3.3.3.3.6	6	24	30	18	12	5	15	9	22	30	14	16	8	15
9	3.4.6.4	6	18	24	12	12	5	12	6	19	24	14	10	5	12
10	4.6.12	12	24	36	12	24	6	18	6	29	36	22	14	4	18
11	3.12.12	6	12	18	6	12	5	9	3	16	18	14	4	1	9
12	Variación de triángulos	2	11	12	10	2	2	6	4	9	12	6	6	3	6
13		Variación de 4.8.8	4	8	12	4	8	3	6	2	11	12	10	2	1
14	Variación de 3.12.12	6	11	18	4	14	5	9	3	16	18	14	4	1	9
15	Variación el cairo	6	15	20	10	10	5	10	4	17	20	14	6	3	10
16	Pentágonos y octágonos	6	13	18	8	10	5	9	3	16	18	14	4	2	9
17	Pentágonos y rombos	4	11	14	8	6	3	7	3	12	14	10	4	2	7
18	Cuadrados y hexágonos	5	11	16	6	10	4	8	3	13	16	10	6	2	8
19	Subcuadros	5	15	20	10	10	4	10	5	16	20	12	8	4	10
20	Molinetes	6	18	24	12	12	2	12	6	18	24	12	12	5	12
21	Triángulos y hexágonos	7	16	24	8	16	5	12	5	19	24	14	10	3	12
22	Vecindarios de Moore	1	4	4	4	0	1	2	1	4	4	4	0	1	2

Se listan en total 44 gráfomatas, de los cuales 37 son diferentes entre si. La malla cuadrada es su propio dual, la malla triangular y hexagonal son mutuamente duales, las teselaciones regulares y no regulares presentadas tienen duales distintos entre si, las mallas de variaciones muestran alteraciones en la representación gráfica pero repiten las estructuras base y dual de las cuales han sido tomadas. Los vecindarios de Moore tienen el mismo tipo de gráfomata base y dual que la malla cuadrada o 4.4.4.4.

Fuente: Autor.

C.5.1. Conectividad entre vértices de un teselado

La simulación de movimiento de los objetos hace necesario establecer las formas en las cuales es posible pasar de un vertice a otro dentro del espacio de la superficie teselada. Se establecen para los teselados las siguientes configuraciones de conectividad:

- Conectividad base o mínima: Es la que está determinada por las distintas relaciones estructurales entre los vértices del grafómata. Unicamente es posible el paso entre los vertices conectados por las aristas estructurales.
- Conectividad determinada por una función de alcance: Haciendo uso de una función, forma geométrica u otro tipo de relacion de contenencia, se determinan los vértices a los cuales es posible saltar o moverse. P.ej: Un área circular de radio r centrada en el vértice actual, el paso al nivel N de vértices⁹, etc. Esta forma de conectividad es suceptible de ser formalizada mediante la integración en el grafomata de las aristas conectoras correspondientes.
- Conectividad máxima: Esta se da cuando es posible el salto en solo paso de un vértice a cualquier otro dentro del espacio teselado. El paso entre el vértice actual y un vertice destino elegido aleatoriamente dentro de la superficie de teselado imita esta forma de conectividad. Establecer esta capacidad de movimiento solo mediante aristas conectoras es no es práctico.

Las dos primeras formas de conexion presentadas son estáticas, están incorporadas dentro del grafomata. Es posible pensar en funciones que controlen dinamicamente el paso entre vertices en funcion de elementos externos, P.ej: según la cantidad de población presente o la densidad de objetos en un vertice. La conectividad puede modificarse en cada uno de los tipos mostrados impidiendo el paso entre algunos vértices, también es posible inducir movimientos dirigidos haciendo que sea posible el paso entre vertices solo en determinadas direcciones.

Los grafomatas presentados en este proyecto ofrecen enlaces minimos entre vertices y en algunos casos se implementa una capa de aristas de conectividad como en los vecindarios de Moore. Implementar otras formas de conexion distintas a las ofrecidas implica hacer control particular de los componentes del grafomata y del teselado generado. A largo plazo, la conectividad en un teselado depende enteramente de su propósito particular de uso.

⁹Vértices más inmediatos a la vecindad en evaluación, como en los vecindarios de Moore. Ver una progresión de esto desde la forma basica de la malla 4.4.4.4 en la sección C.4.41 en la página 398 hasta un un vecindario de moore de 49 celdas en la sección C.4.44 en la página 401.

C.6. Reconstrucción de un teselado a partir de su grafomata

Reconstruir el teselado al que pertenece un grafomata se resume en tres pasos:

1. Determinar el dibugrafo de la estructura. Ver la sección C.2.2 en la página 317.
2. Determinar la malla regular o de repetición del sistema.
3. Realizar un método o proceso de teselado usando el dibugrafo y la malla regular

Se presenta a continuación el detalle de cada uno de los pasos establecidos para la presentación gráfica del teselado al que pertenece un grafomata dado.

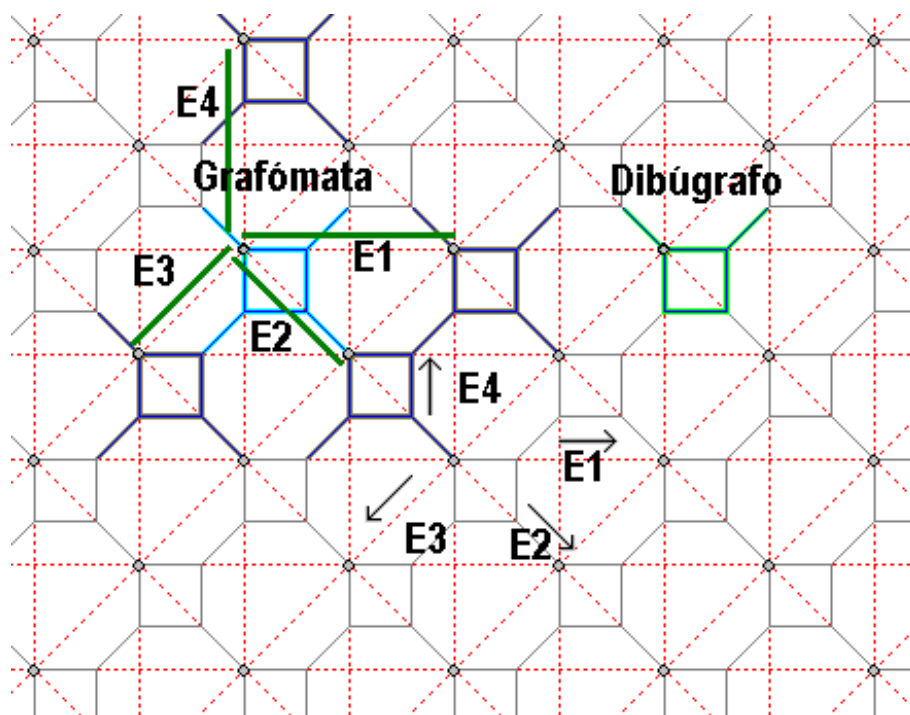
C.6.1. Determinación de la malla regular o de repetición

Para calcular la malla regular asociada a un grafomata se siguen los siguientes pasos:

- Se establece un vértice inicial en el grafomata.
- Desde el primer vértice del grafomata se inicia la búsqueda recursiva de otros vértices del mismo tipo que estén lo más cercanos posible al vértice origen del sistema.
- La línea que une estos vértices establece los puntos sobre los cuales se presenta un eje de repetición.
- Una malla regular puede tener múltiples ejes de movimiento. Se requieren al menos dos ejes para configurar una malla regular. Los ejes a ser usados por el algoritmo de teselado deben ser linealmente independientes y al menos uno de ellos debe ser inmediato, que conecte directamente al vértice inicial con otro vértice del mismo tipo.

La selección de otro vértice cualquiera como punto inicial para la búsqueda produce la misma malla regular, pero centrada en el vértice usado como origen. Una vez se ha establecido un par de ejes que configuren la malla regular, generar el teselado que origina el grafomata es simple. Solo hay que presentar en cada cruce de ejes al dibugrafo correspondiente para reproducir gráficamente la forma del teselado. Ver figuras C.57 y C.58 en la página 411.

Figura C.57: Grafómata, dibúgrafo y malla de repetición

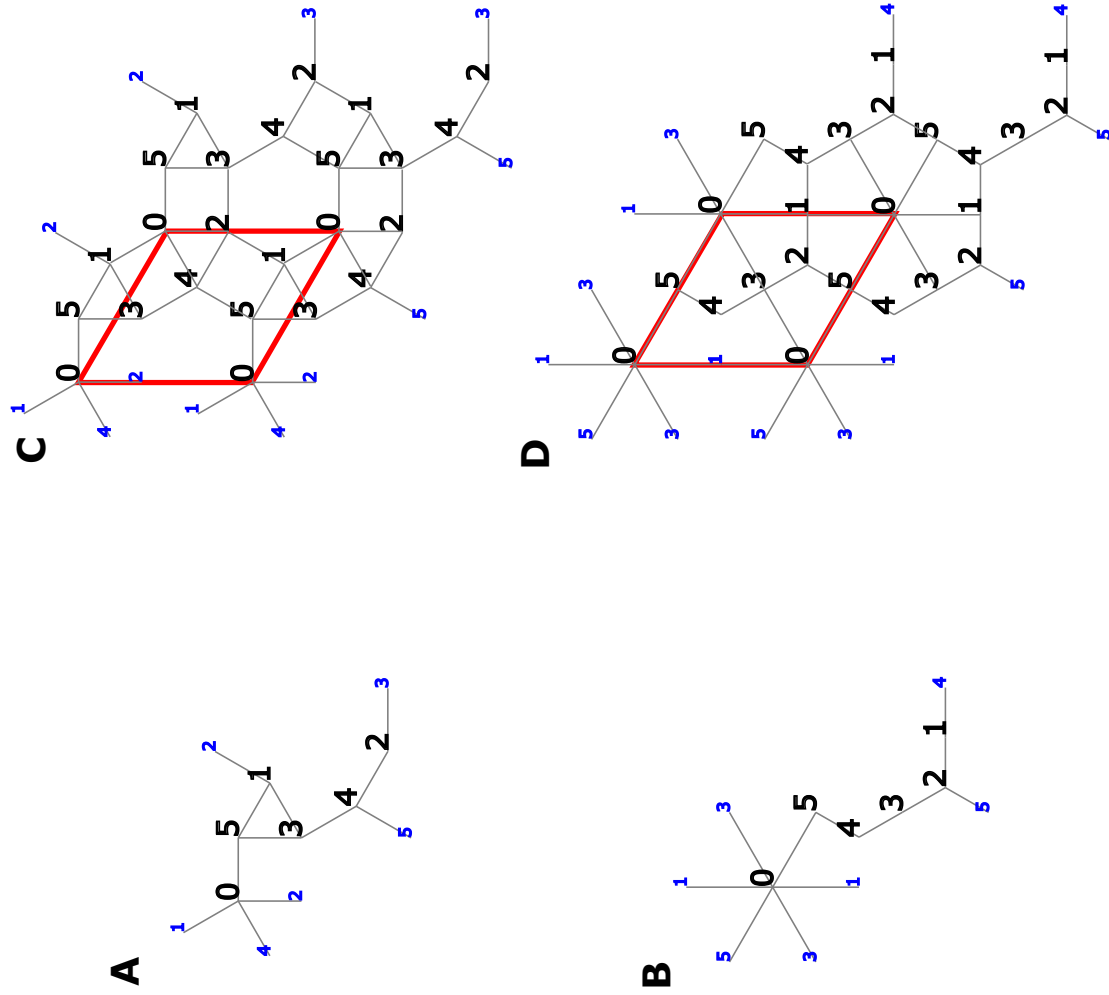


Fuente: Autor.

En la figura C.57, las líneas punteadas en rojo indican los ejes de movimiento de la malla regular o de repetición del grafómata 4.8.8. La estructura de la tesela está en gris, el grafómata está a la izquierda en azul, el dibúgrafo está a la derecha en verde. Los puntos en gris son la intersección entre ejes de movimiento E_1 , E_2 , E_3 y E_4 , los lugares donde se inserta el dibúgrafo.

En la figura C.57 la combinación de ejes E_1 y E_4 no sirve para el propósito de teselado, ya que ambos son no inmediatos y el dibúgrafo con el que se realiza el relleno no es suficiente para conectar en un solo trazo los extremos de los ejes. Los pares formados por la combinación de los demás ejes presentados son viables para ser usados como ejes de movimiento.

Figura C.58: Malla regular en el grafomata base y su dual



A y B son los dibujos, A pertenece al sistema base, C y D son los dibujos en posición según la malla regular subyacente. Nótese la igualdad en la retícula de la malla de repetición.
Fuente: Autor.

C.6.2. Algoritmo de teselado

Con el propósito de presentar gráficamente en el ambiente del prototipo software los distintos grafomatas establecidos en este proyecto, se hizo necesario establecer un algoritmo con el cual realizar dicho proceso. El diagrama de flujo del algoritmo obtenido se muestra en la figura C.59.

El algoritmo de relleno requiere de:

- Las dimensiones de la superficie a rellenar.
- Una ubicación o punto de partida del dentro de la superficie desde el cual realizar las iteraciones de dibujo.
- Los ejes de la malla de repetición propias del grafómata a usar.
- El dibúgrafo o la imagen a repetir en cada punto de dibujo.

El recorrido del algoritmo sobre la superficie de relleno esta controlado por dos ciclos: El ciclo interno controla el proceso de relleno de una mitad de la superficie. Desde el punto origen, el cursor de dibujo avanza una unidad sobre el eje E1 hasta que el siguiente punto de dibujo esté fuera de la superficie de relleno. Cuando eso pasa es momento de moverse una unidad en el eje E2 para volver al eje E1 y moverse en sentido contrario. Este continuo accionar configura un movimiento de relleno en forma de zigzag. El final del proceso de relleno de la porción de la superficie se da cuando ya no es posible hacer volver el cursor al área de relleno.

El ciclo externo consta de dos ciclos, en cada uno hace que el ciclo interno procese una mitad de la superficie a rellenar. El proceso de relleno puede verse en las figuras C.61 en la página 418 y C.60 en la página 416.

El algoritmo prevee varios casos segun la ubicación del punto inicial de relleno:

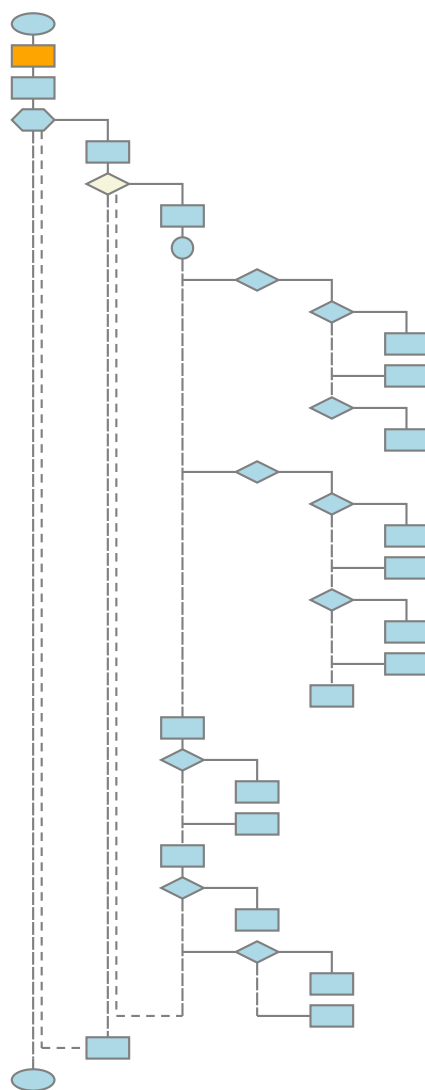
- Fuera de la superficie de relleno: el algoritmo se detiene inmediatamente.
- En un extremo de la superficie de dibujo: Según la orientación de los ejes de movimiento, el algoritmo puede hacer un par de ciclos de prueba hasta determinar la imposibilidad de regreso a la superficie de relleno.

El algoritmo lleva cuenta la cantidad de pasos realizados sobre el eje usado en el ciclo interno: si la condicion de relleno no se cumple, el algoritmo se detiene en un número finito de pasos. El algoritmo obtenido puede rellenar cualquier tipo de superficie rectangular / paralelogramo. Para formas más generales solo hay que inscribir la superficie en un rectangulo que cubra el area deseada. La complejidad del algoritmo es $O(n^2)$.

Durante el proceso de construccion del algoritmo se desarrollo adicionalmente un script grafico que reproduce la forma en la cual se realiza el teselado con el algoritmo. El código fuente de este script se desarrolló en Ejemplos de uso del script se presenta en las figuras C.60 en la página 416 y C.61 en la página 418.

Se lista a continuación el código fuente del algoritmo de teselado. Está implementada en la unidad `Definicion_ambiente.pas`. La función existe en dos versiones, una para el dibujo de los grafómatas (el código mostrado en el listado C.31) y otra para el copiado de regiones de memoria en la pantalla. Esto es así ya que el relleno en el ambiente se hace por etapas. Esto se explica en detalle en la sección C.6.3 en la página 417.

Figura C.59: Algoritmo de relleno de superficies



El diagrama revela el doble ciclo del algoritmo de teselado. Obsérvese la construcción while en el ciclo interno: se desconoce en el cada recorrido hasta donde debe llegar para completar el relleno sobre el eje elegido. En el último bloque del ciclo externo se configura el cambio de sentido de relleno del ciclo interno.

Los detalles del diagrama de flujo se presentan en la sección E.7 en la página 654.

Fuente: Autor.

Fuente en Pascal C.31: algoritmo de relleno de superficies con grafomatas

```

1  //////////////////////////////////////
2  // autómata de relleno de superficie según los ejes de movimiento planteados
3  //////////////////////////////////////
4  procedure dibujar_patron(
5  grf:Tgrafomata;color:Tcolor;estilo:Tpenstyle;ubicaciones_vertice:puntos_xy;bmp:
6  Tbitmap;e1x,e1y,e2x,e2y:integer;gx,gy:integer;area_maxima:Trect;zoom:byte;
7  angulo:precision_angulo
8  );
9  var
10 i,hx,hy,accion,dz,cslim,afuera:integer;
11 en_zona,ox,oy,regresando:boolean;
12 emx,emy,pmx,pmi:integer;
13 begin
14 //el algoritmo requiere que el punto de inicio este dentro del área de relleno
15 assert((gx>area_maxima.left) and (gx<area_maxima.right) and (gy>area_maxima.top) and (gy<
    area_maxima.bottom),'Punto fuera de área de relleno'); //determinar en cada punto si esta
    dentro de cs o no.
16
17 //sentido de avance de relleno
18 ox:=false;
19 oy:=ox;
20
21 emx:=(e1x+e2x)div 2;
22 emy:=(e1y+e2y)div 2;
23
24 // grf.dibujar_dibugrafo(gx,gy,color,estilo,ubicaciones_vertice,bmp,zoom,angulo); //dibujo de
    pieza central
25
26 //gx:=gx+200; gy:=gy+200; //prueba: que pasa si se da un punto fuera de cs??
27
28 for i := 0 to 1 do //realizar relleno de cada seccion [0,1] en el eje de avance
29 begin
30 //ubicación del punto de relleno, el algoritmo asume que el punto esta dentro de cs
31 regresando:=false;
32 cslim:=0; //límites autoestablecidos por el algoritmo
33 afuera:=0; //conteo de pasos fuera de cs
34 accion:=0; //el algoritmo asume que el punto de inicial de dibujo esta dentro de cs, el área de
    relleno
35 dz:=0; //conteo de pasos de relleno sobre el eje actual
36 hx:=gx; //origen de relleno
37 hy:=gy;
38 repeat
39 inc(dz); //contador de longitud maxima
40 case accion of
41 0,1,3: //movimiento en eje de avance principal
42 begin
43 if ox then //movimiento en eje de avance principal
44 begin
45 hx:=hx-e1x; //moverse en eje el negativamente
46 hy:=hy-e1y;
47 accion:=3;
48 end
49 else
50 begin
51 hx:=hx+e1x; //moverse en eje el positivamente
52 hy:=hy+e1y;
53 accion:=1;
54 end;
55 if (dz>cslim) then cslim:=dz; //tope maximo en linea actual
56 end;
57 2://movimiento en eje de avance secundario
58 begin
59 if oy then //movimiento en eje de avance secundario
60 begin
61 hx:=hx+e2x; //moverse en el eje2
62 hy:=hy+e2y;
63 end
64 else
65 begin
66 hx:=hx-e2x; //moverse en el eje2
67 hy:=hy-e2y;
68 end;
69 if ox then dec(accion) else inc(accion); // 3 <- 2 -> 1
70 ox:=not ox; //cambio de sentido de movimiento en eje principal
71 dz:=0; //reinicio del contador de pasos sobre eje1
72 end;

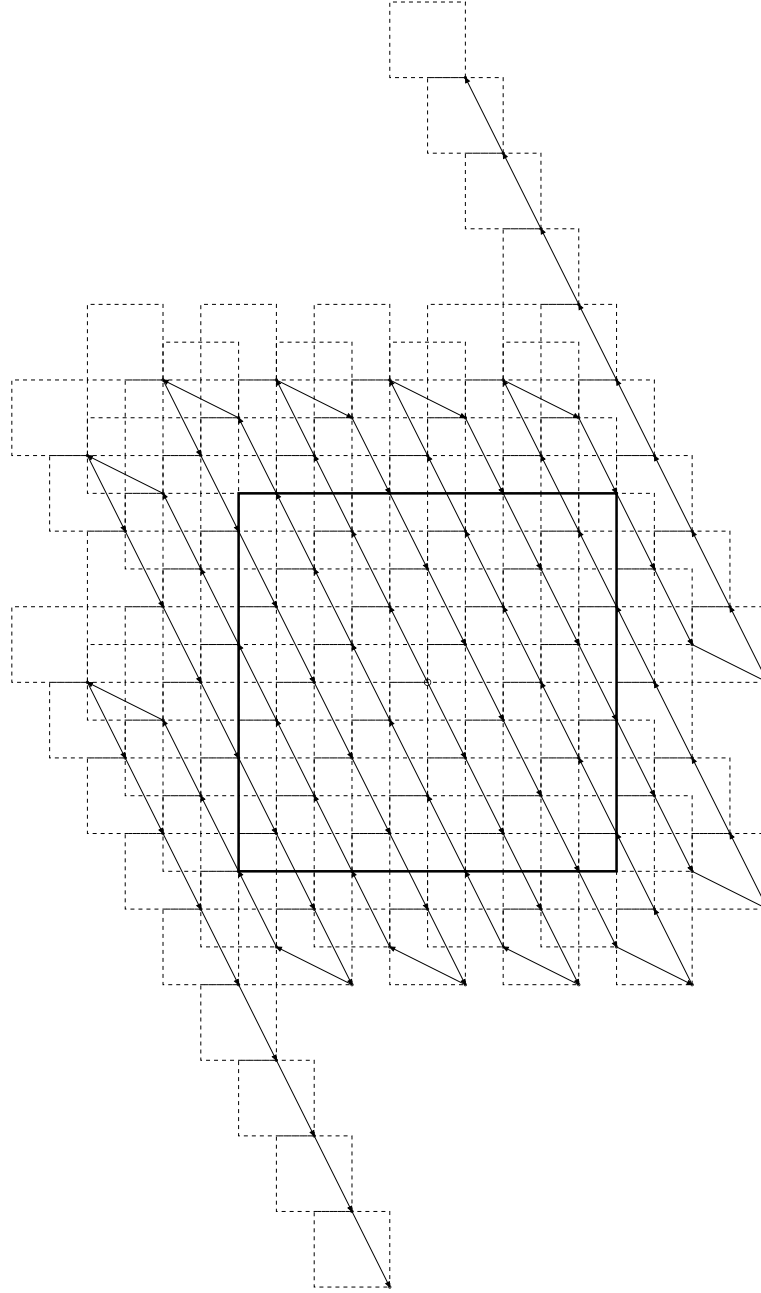
```

```

73     end;
74
75     grf.dibujar_dibugrafo(hx,hy,color,estilo,ubicaciones_vertice,bmp,zoom,angulo); //llamado a la
76         rutina de dibujo del grafómata actual
77
78     //extensión de borde dirigida , en espera de cubrir la mayor cantidad de área gráfica útil
79     posible
80     if ox then
81     begin
82         pmx:=hx+emx;
83         pmy:=hy+emy;
84     end
85     else
86     begin
87         pmx:=hx-emx;
88         pmy:=hy-emy;
89     end;
90
91     en_zona:=(pmx>area_maxima.left) and (pmx<area_maxima.right) and (pmy>area_maxima.top) and (
92         pmy<area_maxima.bottom); //determinar en cada punto si esta dentro de cs o no.
93
94     if en_zona then
95     begin
96         regresando:=false; //operación normal del algoritmo
97         afuera:=0;
98     end
99     else
100     begin
101         if regresando then //el punto actual se detecta como afuera solo cuando se esta regresando
102             inc(afuera)
103         else
104             begin
105                 accion:=2; //solo se hace una vez
106                 regresando:=true; //el algoritmo de recorrido está en espera de regresar a cs
107             end;
108         end;
109     end;
110
111     until (afuera>cslim); //rellenar hasta donde es posible
112
113     //inversión de dirección de avance en ejes , relleno de la siguiente sección
114     ox:=true;
115     oy:=ox;
116 end;
117 end;

```

Figura C.60: Ejemplo de ejecución del algoritmo de relleno de superficie



Vista gráfica del funcionamiento del algoritmo de teselado: El cuadrado de borde sólido es la superficie a rellenar, las flechas indican el sentido de los ejes de avance del algoritmo de relleno indicando su sentido de movimiento. Cada cuadrado punteado representa la imagen gráfica con la cual se llena el espacio. En este ejemplo en particular se nota la superposición de imágenes, ya que los ejes de movimiento usados en el algoritmo producen un relleno no óptimo. Sobre esto ver figura C.62.

Fuente: Autor.

C.6.3. Uso del algoritmo de teselado

Una vez determinado el algoritmo de teselado, se hace necesario establecer las condiciones de uso. En el prototipo software se emplea la tecnica de doble imagen o *double buffer* para el dibujado del teselado en la pantalla: mientras una imagen se muestra en pantalla , la otra es actualizada segun los cambios requeridos. Una vez terminada, la imagen en construcción pasa a ser presentada y la otra se alista para el siguiente ciclo de renovación de imagen.

A continuacion se describe el metodo de relleno del ambiente de simulacion implementado en el prototipo PH2.exe haciendo uso del algoritmo de teselado:

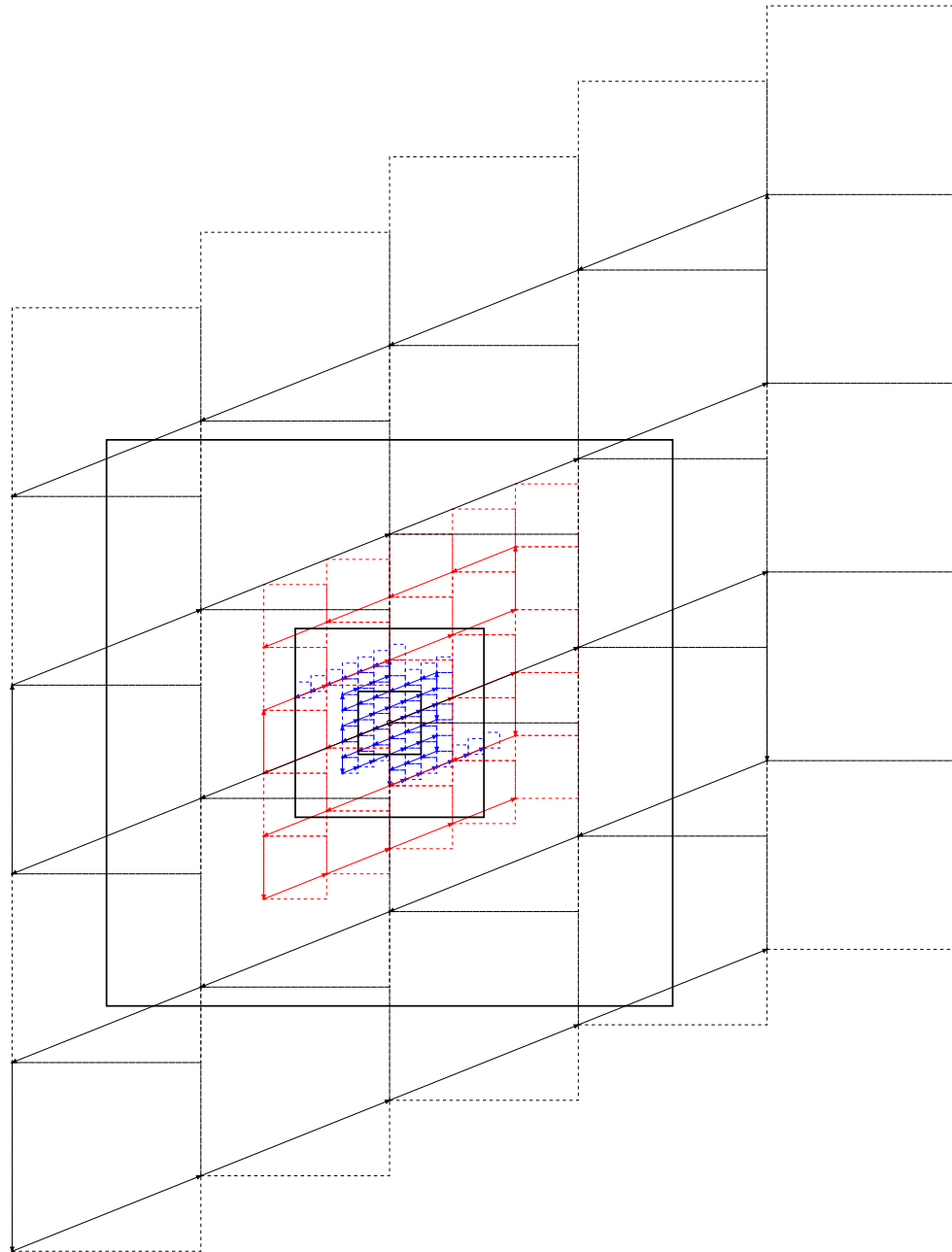
- Etapa inicial: Mientras se presenta la imagen del ciclo anterior en la pantalla principal:
 1. Se inicia la pantalla alterna en blanco.
 2. Se selecciona el grafomata a usar y se calcula la malla regular del grafomata.
 3. Se establece el area minima m requerida por el dibugrafo para la representacion del teselado.
 4. Se traza en m los dibugrafos requeridos para armar una porcion de fondo gráfico haciendo uso del algoritmo de teselado.
- Segunda etapa:
 1. Se copia el area del patron de teselado obtenido.
 2. Se establece el area de la segunda etapa de relleno que se va a rellenar con la copia.
 3. Se traza en ese espacio la copia de la imagen obtenida. De esta forma se expande el area de imagen teselado.

En este punto parte del area a sido cubierta pero aun no completa el espacio total de la imagen. asi que se hace necesario una ultima pasada de algoritmo.
- Tercera etapa:
 1. Se copia el area del patron de teselado extendido obtenido en la segunda etapa.
 2. Se establece el area completa de la imagen a rellenar.
 3. Se traza en ese espacio la copia de la imagen obtenida en la segunda etapa. De esta forma se completa el proceso de teselado de la imagen.

Una vez terminada la imagen, se realizan los distintos cambios relacionados con el funcionamiento del simulador: p.ej: Relocacion de objetos, presentacion de mensajes, etc. Se usa la imagen producida como imagen principal y la imagen actual se recicla, siguiendo el metodo de doble pantalla. El proceso se ilustra en la figura C.61 en la página siguiente.

El teselado se redibuja solo cuando se realizan operaciones de rotacion, escalado o traslacion del ambiente de simulación. El metodo de teselado presentado puede aumentar la cantidad de etapas segun el tamaño del area a rellenar.

Figura C.61: Relleno del espacio mediante uso serial del algoritmo de relleno



Ejemplo gráfico del funcionamiento del algoritmo de relleno usado en el prototipo PH2.

El proceso consta de 3 ejecuciones centradas y en serie del algoritmo de relleno. En azul se muestra la etapa primaria de relleno en la cual mediante el dibúgrafo genera la imagen que sirve de plantilla de relleno. En rojo se muestra la segunda etapa, en la cual se copia en cada punto de dibujo la imagen generada en la etapa azul. En negro se muestra la última etapa, en la cual se copia la imagen generada en la etapa roja. Las líneas en negro muestran el trazado del relleno sobre el superficie global a teselar. La figura presenta un ejemplo de relleno óptimo, no hay solapamientos entre imágenes de una misma etapa de relleno.

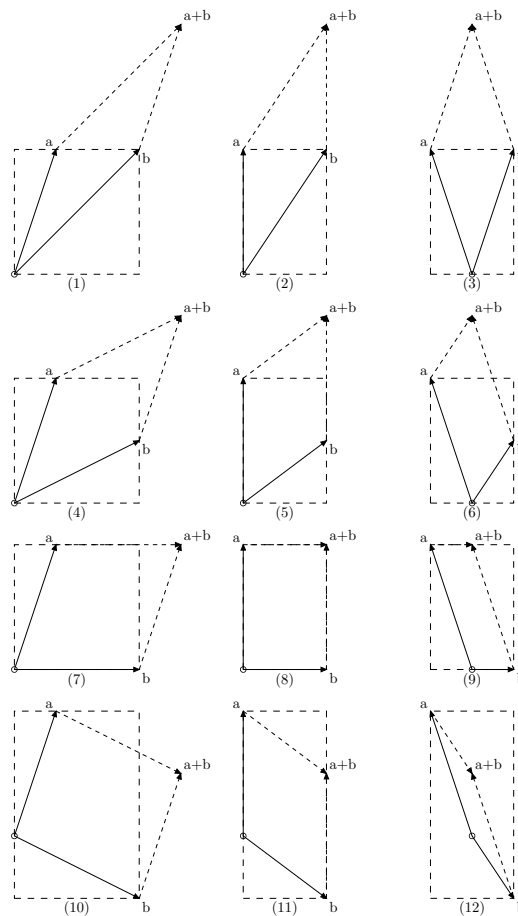
Fuente: Autor.

C.6.4. Teselado óptimo

Las figuras C.6.3 en la página 417 y C.61 en la página anterior muestran un elemento importante para el proceso de teselado: Según la forma del grafomata elegido, este tendrá una configuración de ejes de movimiento tal que permita realizar un relleno óptimo de la superficie a teselar. En un relleno óptimo la region rectangular usada para rellenar el espacio no se superpone a ninguna otra presente en el area de dibujo. Esto garantiza que se realiza la minima cantidad de operaciones de dibujo para rellenar el espacio y que se evitan totalmente los errores causados por el repintado de porciones de imagen.

En proximas iteraciones del prototipo software se esperan mejoras en cuanto al proceso de seleccion de ejes durante la etapa inicial de teselado siguiendo los lineamientos de optimalidad de relleno presentados en la figura C.61 en la página anterior.

Figura C.62: Relacion entre relleno optimo y ejes de movimiento de algoritmo de teselado



En los casos (2), (3), (5), (7) y (8) la configuración de los ejes de movimiento configura un relleno óptimo. El relleno óptimo se produce cuando el ángulo entre los ejes de movimiento es $0^\circ < \alpha \leq 90^\circ$ y al menos uno de los ejes de movimiento está sobre los ejes coordenados X o Y. El caso (3) es particular ya que el ángulo puede estar comprendido entre $0^\circ < \alpha < 180^\circ$, funciona porque el destino de los ejes está a una misma altura, hay igual magnitud de avance en cada uno de los ejes.

Fuente: Autor.

C.6.5. Razones de uso del teselado en 3 etapas en el ambiente

Al observar el metodo de teselado implementado, es natural preguntarse:

- ¿Porqué no se realiza el proceso de teselado en una etapa usando directamente el dibugrafo?
- ¿Porqué se implemento un sistema de 3 etapas en el prototipo?

Para justificar la eleccion realizada, se propone una relacion entre la cantidad de pasadas o etapas de aplicacion de algoritmo de teselado, la cantidad de dibujos realizados en una etapa y el tiempo que tarda cada dibujo en ser realizado. Para k etapas:

$$d_j = \sum_{i=1}^{j-1} n_i^2 + \left(\prod_{i=j}^k n_i \right)^2 \quad (C.11)$$

$$e_j = \sum_{i=1}^{j-1} t_i n_i^2 + t_j \left(\prod_{i=j}^k n_i \right)^2 \quad (C.12)$$

Donde:

k Máximo de etapas del algoritmo requeridas para rellenar el área destino.

j Número de etapas de relleno a implementar de k disponibles.

d_j Total de operaciones de dibujo implementados en la etapa j de k de disponibles.

e_j Total de tiempo requerido por el número de dibujos a realizar en j etapas de k disponibles.

t_i Tiempo requerido para el dibujado de una unidad gráfica en la etapa i .

Ya que el algoritmo cubre generalmente el área de un paralelogramo, geométricamente puede suponerse que la cantidad de dibujos realizados por una etapa del algoritmo será n^2 , donde n_i es el total de puntos de dibujo del lado del área recorrida en la etapa i del algoritmo de teselado.

El total de etapas depende de el area a ser cubierta. Por ejemplo si se usa un mismo tamaño de lado para cada iteracion entonces se sugiere que $k = \frac{\ln(m)}{\ln(n)}$ donde m es el lado del area a teselar en unidades de puntos de dibujo.

En el proceso implementado en el prototipo, la primera etapa consiste en el trazado del dibúgrafo, las demás etapas realizan copia de imágenes. Por ejemplo, para un sistema de $k = 3$ etapas como el implementado en el software con $t_1 = 8$, $t_2 = 1$, $t_3 = 2$ y la misma cantidad de operaciones por lado de area recorrida $n_1 = n_2 = n_3 = 3$, el total de unidades de tiempo necesario para las operaciones de dibujo para los valores dados según (C.11) y (C.12) es:

Tabla C.3: Etapas de teselado, cantidad de operaciones y duracion de proceso

Total de operaciones graficas realizadas. Eq(C.11).	Unidades de tiempo requeridas por las operaciones graficas. Eq(C.12).
$d_1 = \left(\prod_{i=1}^k n_i \right)^2 = 27^2 = 729$	$e_1 = t_1 \cdot \left(\prod_{i=1}^k n_i \right)^2 = 8 \cdot 27^2 = 5832$
$d_2 = n_1^2 + \left(\prod_{i=2}^k n_i \right)^2 = 9 + 9^2 = 90$	$e_2 = t_1 n_1^2 + t_2 \cdot \left(\prod_{i=2}^k n_i \right)^2 = 9 \cdot 8 + 1 \cdot 9^2 = 153$
$d_3 = n_1^2 + n_2^2 + n_3^2 = 9 + 9 + 9 = 27$	$e_3 = t_1 n_1^2 + t_2 n_2^2 + t_3 n_3^2 = 9 \cdot 8 + 9 \cdot 1 + 9 \cdot 2 = 90$

Fuente: Autor.

Los valores usados se justifican de esta forma: representar gráficamente cada dibúgrafo en la primera etapa es costoso¹⁰, aquí se ejemplifica como de 8 unidades de tiempo por dibujo. La copia de imágenes es mucho más rápida; los tiempos presentados se diferencian ya que la última operación de copiado se realiza sobre un área mayor.

Con los datos de ejemplo, el dibujado con 3 etapas resulta ser el que realiza la menor cantidad de operaciones de dibujo, cosa que se traduce en una reducción en el tiempo total de teselado.

Aunque es un ejemplo particular, reproduce una tendencia general: La naturaleza de las funciones hace notar que es mejor realizar sumatorias que productorias, es mas rapido realizar operaciones graficas escaladas que realizar iteraciones con gran numero de operaciones. La experimentación realizada en el prototipo sobre la velocidad de teselado del ambiente confirma los resultados obtenidos.

¹⁰Las operaciones de dibujo realizadas con las funciones gráficas del api de windows son significativamente más lentas que la copia de porciones de memoria gráfica. Seria ideal una forma de dibugrafo lineal ya que haría uso óptimo de la función polypolyline de Delphi.

C.6.6. Teselados y vecindarios producidos por los grafómatas disponibles

Las figuras C.63 en la página siguiente hasta la C.83 en la página 429 ilustran los 21 distintos tipos de vecindarios implementados en el prototipo Homos 2.0¹¹. Cada uno corresponde con las estructuras de grafomatas presentadas en la sección C.4 en la página 337.

Todos los teselados presentados son periodicos. Se incluyen en los ejemplos teselados de poligonos regulares e irregulares y variaciones. Se destaca que la variacion en el vecindario 4.8.8 es un ejemplo de cambio estructural que no altera la simetria presente en el sistema origen, se realizó una rotacion de algunas de las aristas. Cómparese los grafomatas C.4.9 en la página 349 y C.4.23 en la página 373. Los vecindarios comparados tienen la misma configuracion de duales. La variacion en el vecindario de triangulos es un ejemplo de un sistema que exhibe cambios de simetria, el orden alteró mediante traslación de vertices. Comparense los grafomatas C.4.1 en la página 339 y C.4.21 en la página 370. Sus duales estan compuestos de hexágonos, pero se observa una inclinacion en sistema modificado. Si los cambios implementados en un sistema rompen su simetria, entonces se establecen diferencias en el grafomata entre sistemas.

No se presentan en esta lista los vecindarios de Moore ya que tienen como base estructural la malla 4.4.4.4. La diferencia entre estructuras la establecen la capa de aristas conectoras, que son elementos invisibles en el teselado. Los detalles de estas estructuras se presentan desde la sección C.4.42 en la página 399 hasta la seccion C.4.44 en la página 401. La malla 4.4.4.4 es autodual¹², la unica del conjunto en la cual el dual repite la estructura del teselado base. La malla triangular y hexagonal son mutuamente duales. ¿Existe algún otro teselado idempotente / autodual?, ¿Existen otras teselas dualmente cíclicas?, ¿Existen teselas multicíclicas?.

¹¹Ver sección E.8 en la página 657.

¹²Un grafo autodual es aquel que es isomorfo a su dual.

Figura C.63: Vecindario triangular

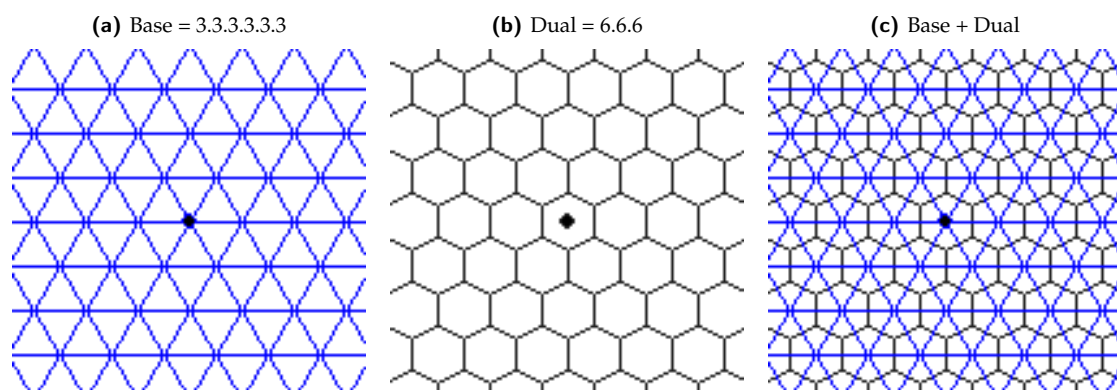


Figura C.64: Vecindario Von Newman

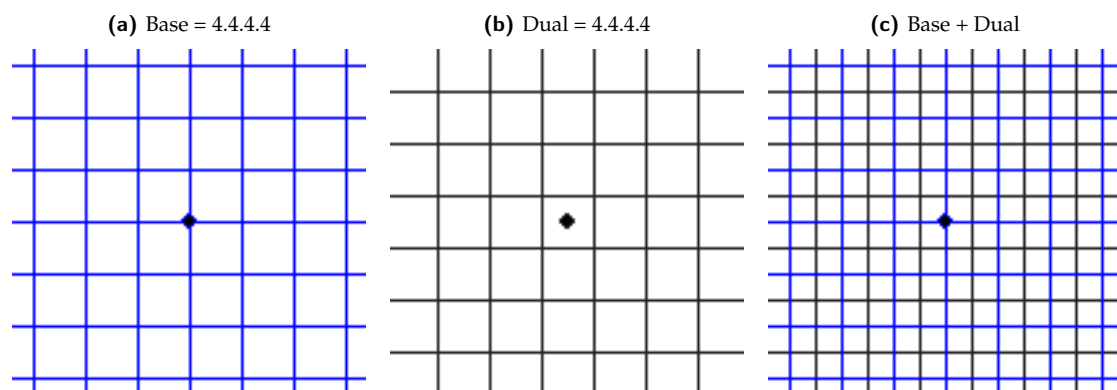


Figura C.65: Vecindario hexagonal

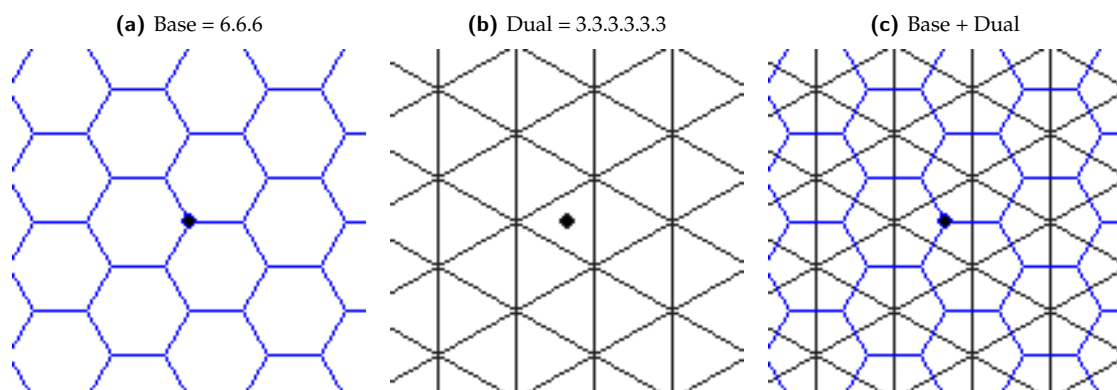


Figura C.66: Vecindario 3.3.3.4.4

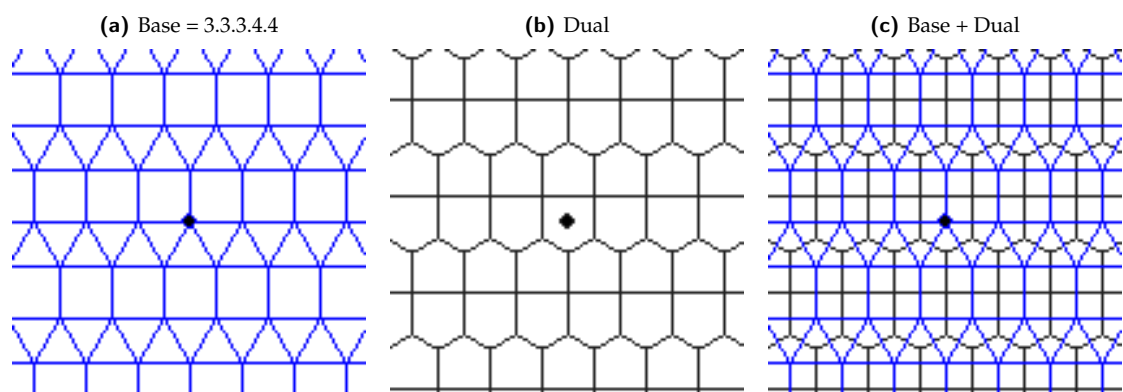


Figura C.67: Vecindario 3.6.3.6

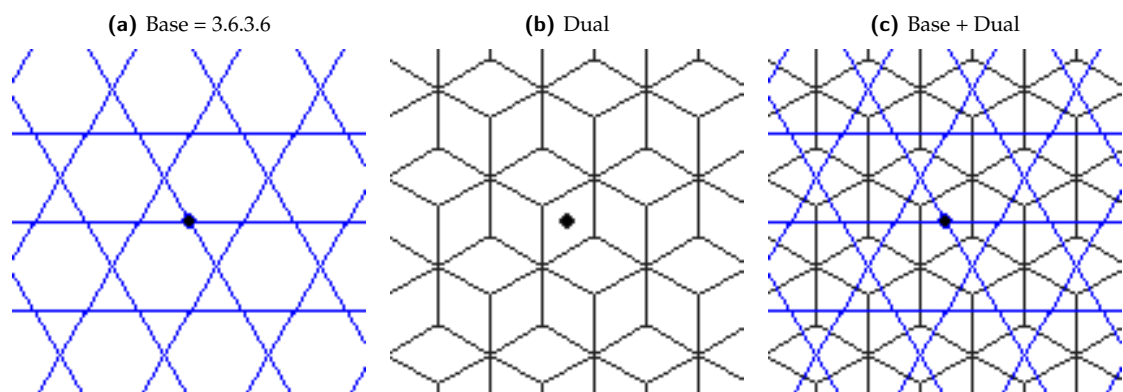


Figura C.68: Vecindario 3.12.12

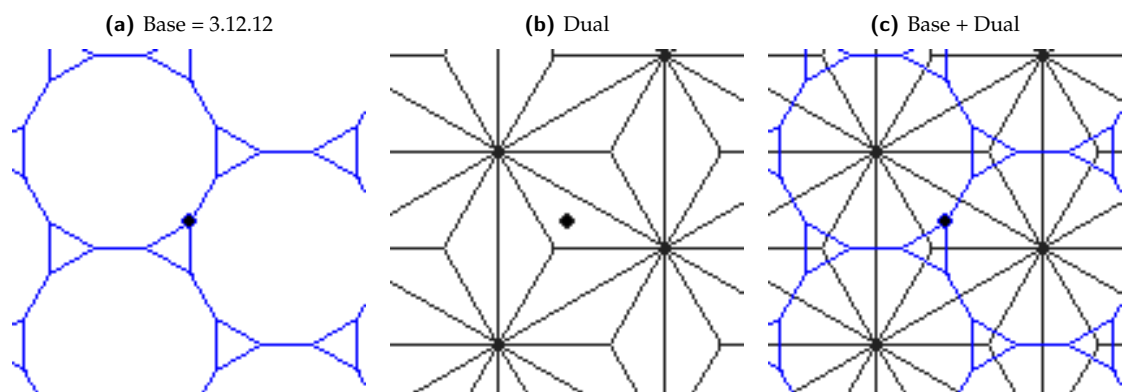


Figura C.69: Estrella de David, Vecindario 3.3.3.3.6

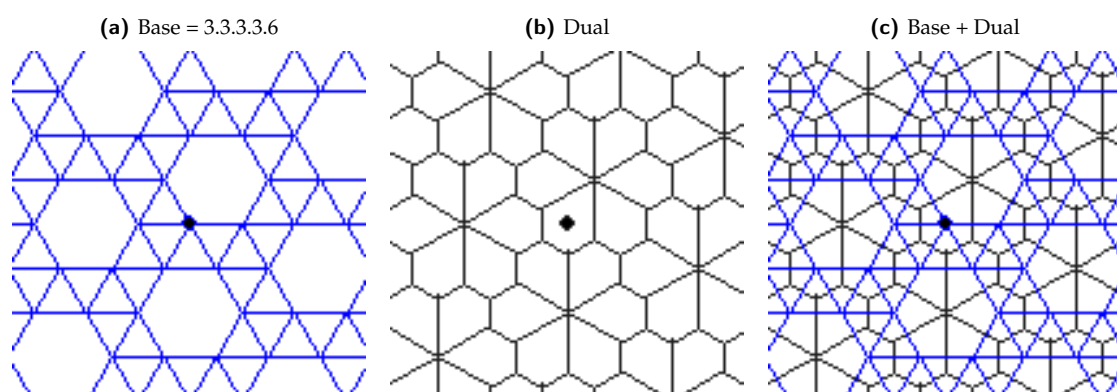


Figura C.70: Vecindario 3.4.6.4

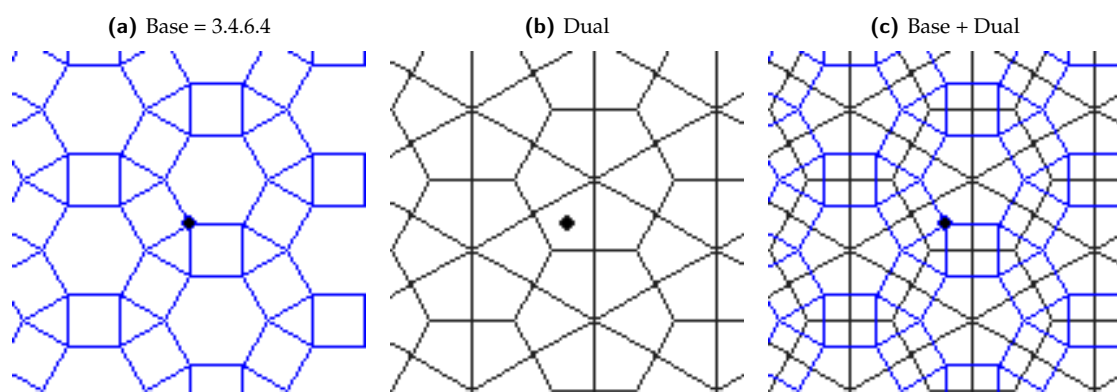


Figura C.71: Vecindario 4.6.12

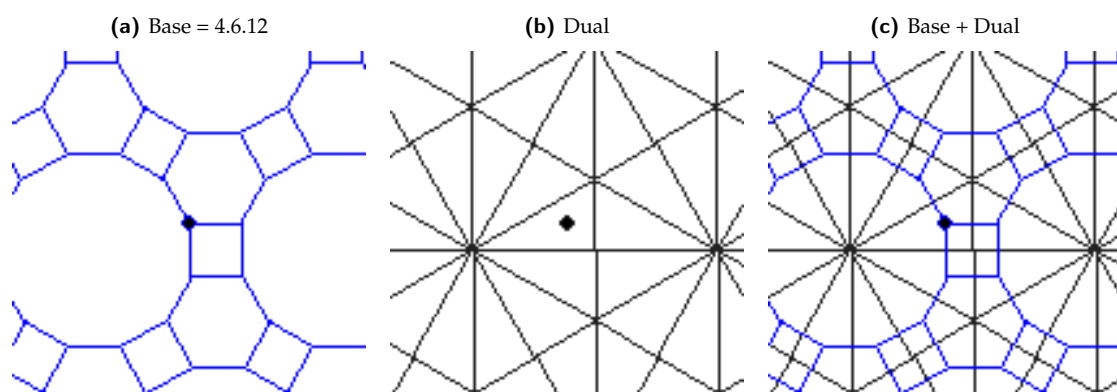


Figura C.72: Cuadrados y octágonos, Vecindario 4.8.8

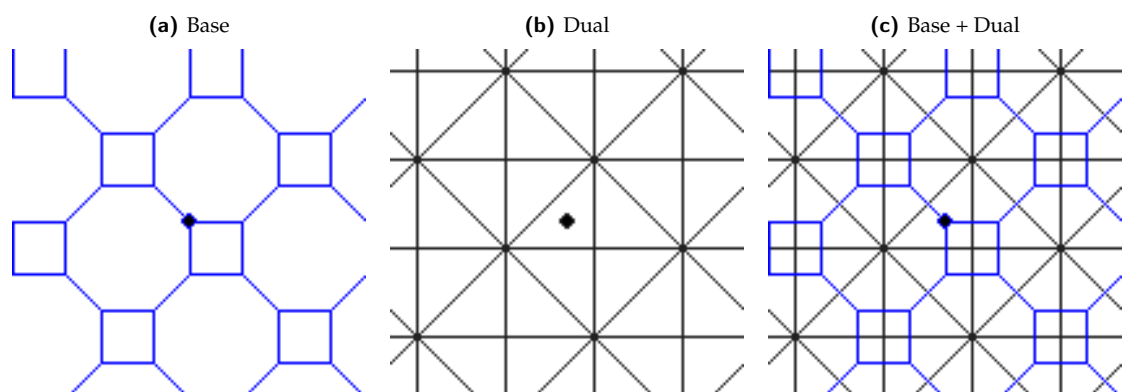


Figura C.73: Cuadrados y hexágonos , Vecindario Ojo de buey

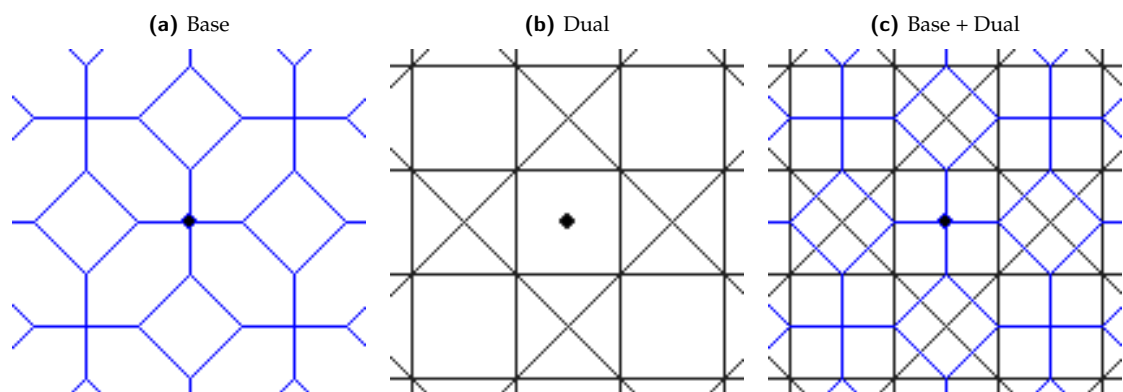


Figura C.74: Cuadrados y trapecios, Vecindario Waffles

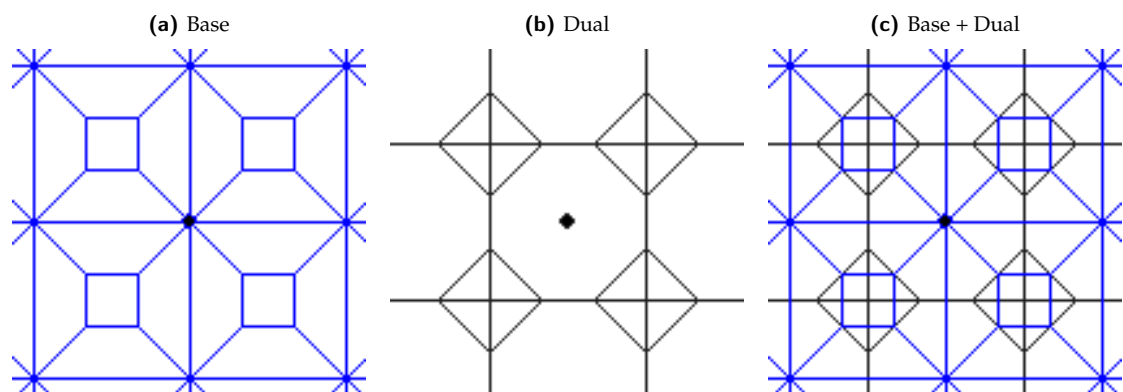


Figura C.75: Vecindario el Cairo

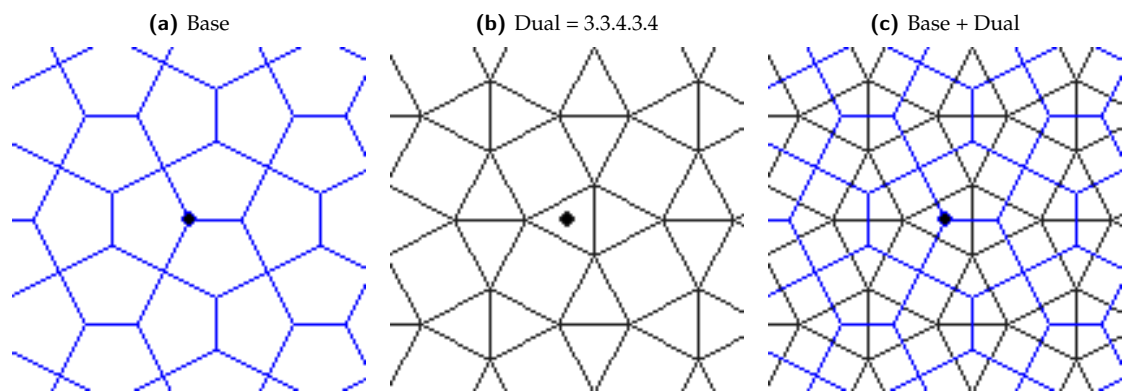


Figura C.76: Variación en Vecindario el Cairo

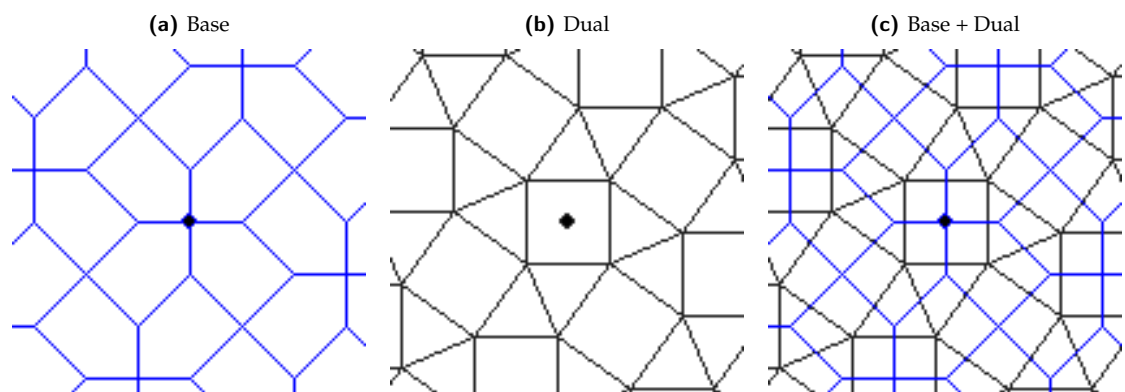


Figura C.77: Vecindario de Molinetes

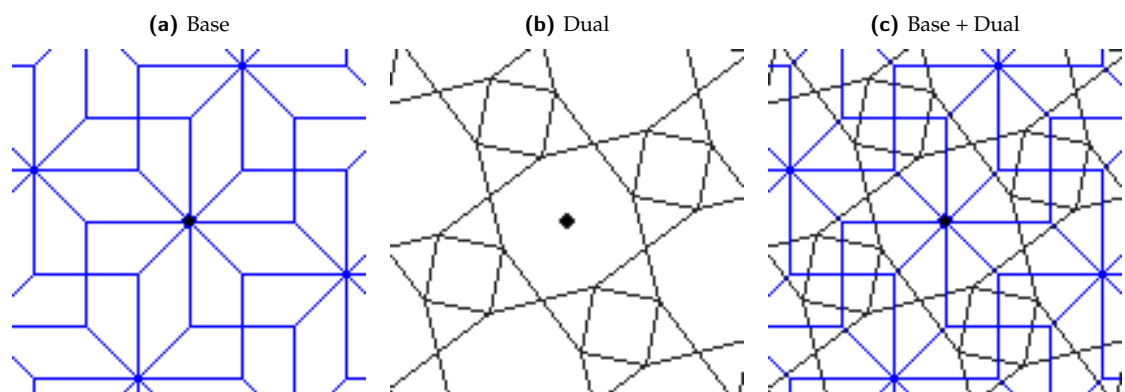


Figura C.78: Variación en Vecindario de triángulos

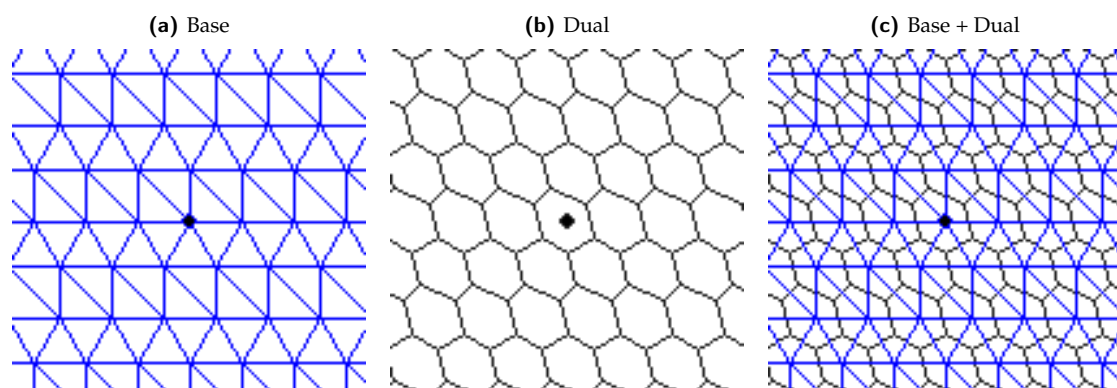


Figura C.79: Estrellas de 4 puntas, variación en Vecindario de 4.8.8

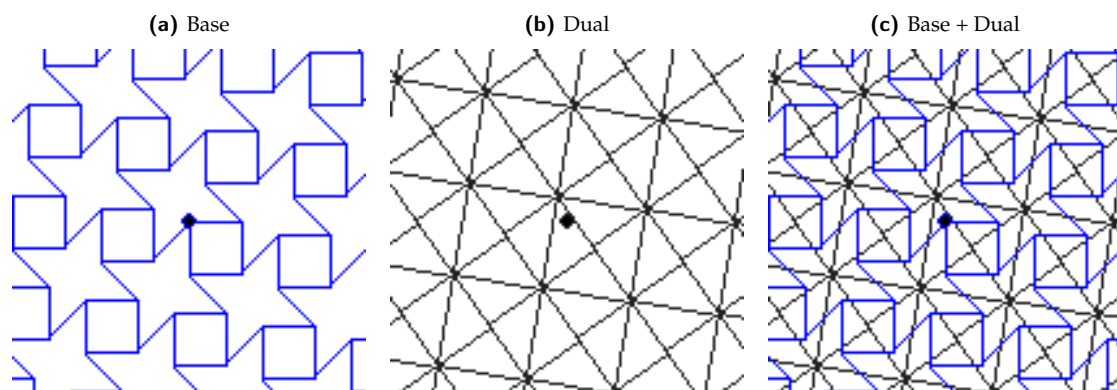


Figura C.80: Estrellas de 6 puntas, Variación en Vecindario 3.12.12

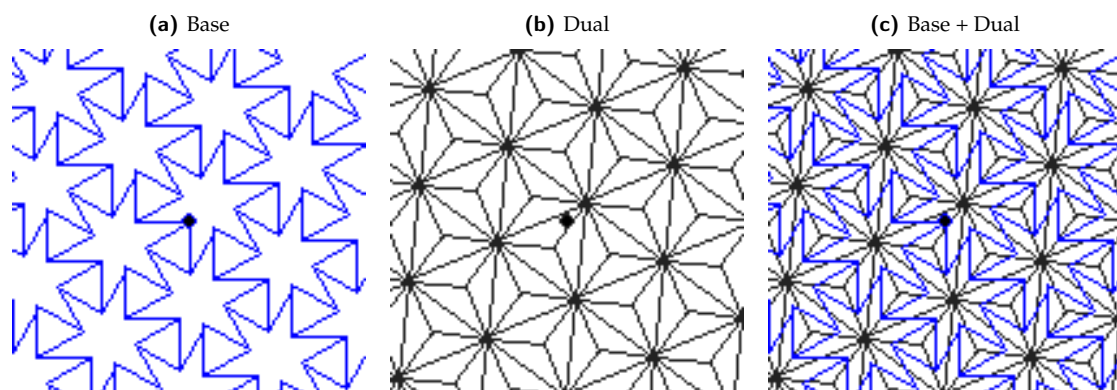


Figura C.81: Vecindario de triángulos y hexágonos

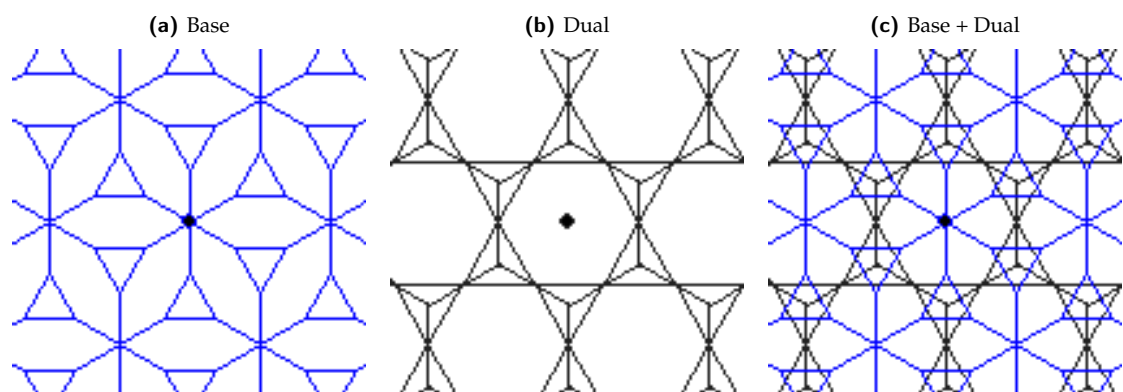


Figura C.82: Vecindario de pentágonos y rombos

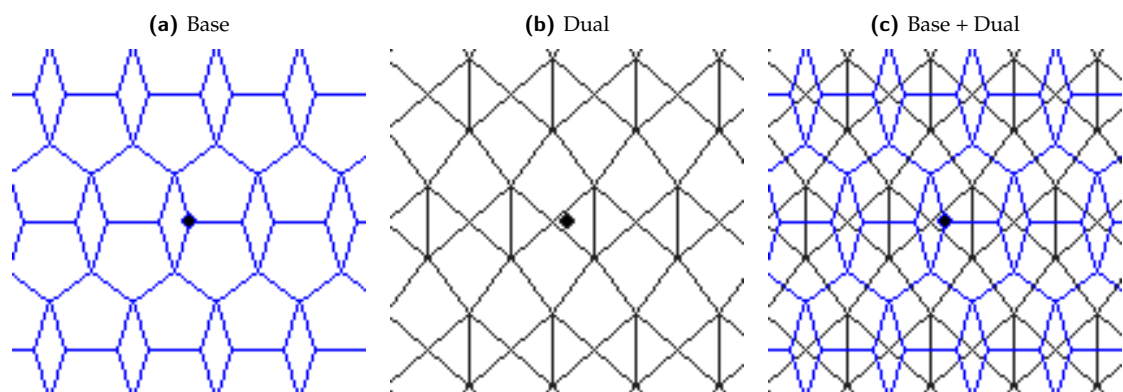
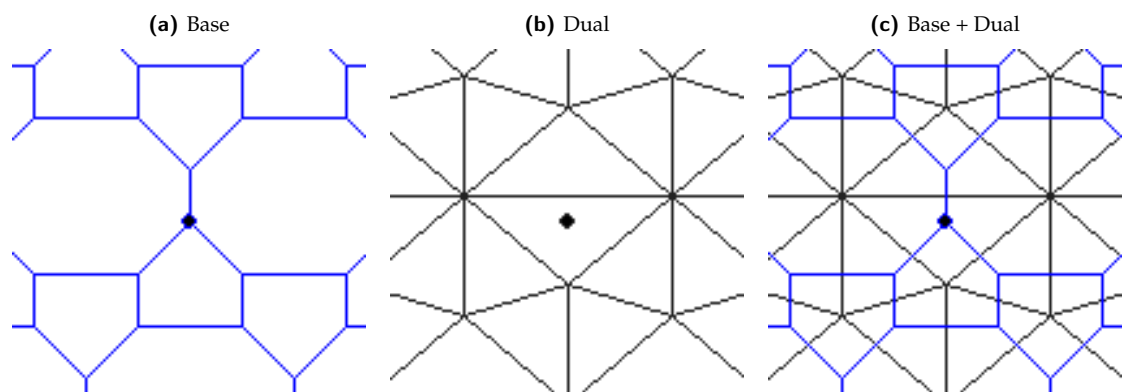


Figura C.83: Vecindario de octágonos y pentágonos



C.7. Extensión y expansión de uso del concepto de grafomata

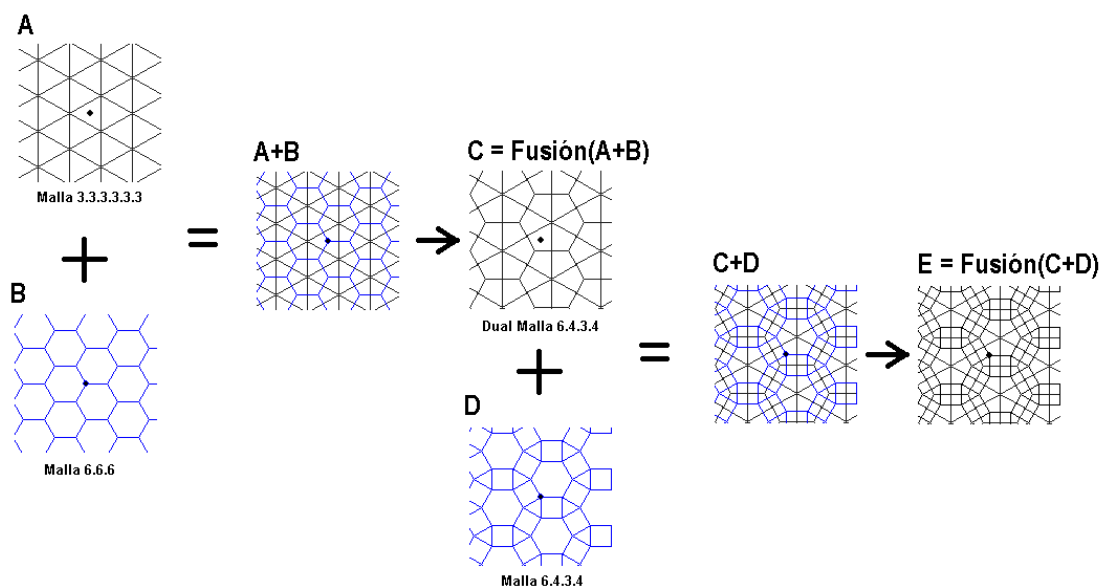
C.7.1. Relaciones entre teselados

Obsérvese el resultado de la superposición los grafomatas base y dual presentado en la subfigura c de cada teselado presentado en la sección C.6.6 en la página 422. Cada una puede usarse como fuente para la estructura de un nuevo grafomata, el cual estaría caracterizado por las propiedades particulares de las estructuras participantes y los vertices y aristas generados por sus intersecciones. La integración entre distintas estructuras de grafomatas definen un grafomata de *fusión*.

Una fusión puede interpretarse tambien como la aplicacion de una forma de subdivision de la tesela base o dual que la origina. Un ejemplo de fusión de teselados se presenta en la figura C.84.

A partir de la figura C.84 puede suponerse la existencia de teselas primas, aquellas que no se derivan de otros teselados, ni son productos de la fusion u otro tipo de operaciones entre teselas. De la figura se observa que A, B y D son teselas primas. Ya que estas teselas no se generan por formas previas, es de suponer que todas las teselaciones aperiódicas pertenecen a esta categoría. ¿Como se determina si una tesela es prima?, ¿Que otras relaciones existen entre teselados?.

Figura C.84: Ejemplo de relaciones entre de teselados implementados



Nótese las relaciones entre las distintas teselas o mallas. La malla C ha sido escalada para hacer posible la fusión en la forma deseada con la malla 6.4.3.4.

Fuente: Autor.

C.7.2. Teselas y costuras

Cuando en un espacio dado se hace uso de dos o mas grafomatas en una misma superficie, surge la pregunta: ¿Qué es lo que acontece en las zonas de contacto entre los grafómatas? Se da un fenómeno de paso entre teselados. Debido al aspecto irregular de las uniones y al parecido de la superficie con una colcha de retazos, los puntos donde se encuentran los limites los teselados se los denomina bordes de costura o simplemente costuras. La figura C.85 en la página siguiente muestra varios ejemplos de objetos cercanos a costuras.

La realización efectiva de la continuidad de movimiento de los objetos en un ambiente con varios sistemas de teselado integrados se da mediante los siguientes pasos:

- A partir de la ubicacion en el teselado actual se determina con el grafomata asociado el lugar en el espacio del siguiente nodo o vertice.
- Se determina si la ubicación o nodo elegido está dentro del espacio que delimita el teselado actual. Si está fuera de área delimitada entonces se debe determinar el sistema de teselado al cual se hace paso.
- En el teselado destino se determina el vértice más cercano al punto deseado. Una vez encontrado y si se cumplen las distintas condiciones de paso entonces se permite el movimiento del objeto entre teselados.

El objeto Tgrafomata implementado en este proyecto incluye varias funciones a ese proposito¹³, ofrece la busqueda recursiva del punto destino haciendo uso de distancias metropolitana y euclidiana a partir del punto origen del teselado y la busqueda mediante exploracion recursiva de los vertices del teselado desde una ubicacion de vertice conocida.

El proceso de acercamiento a un punto dado es como sigue: Mediante la malla regular del teselado destino es posible acercarse hasta el vertice inicial del grafomata mas cercano al punto de llegada con la menor cantidad de calculos posibles. Una vez en esta ubicación, se realiza la busqueda recursiva en la estructura destino hasta alcanzar el vertice mas cercano al punto deseado.

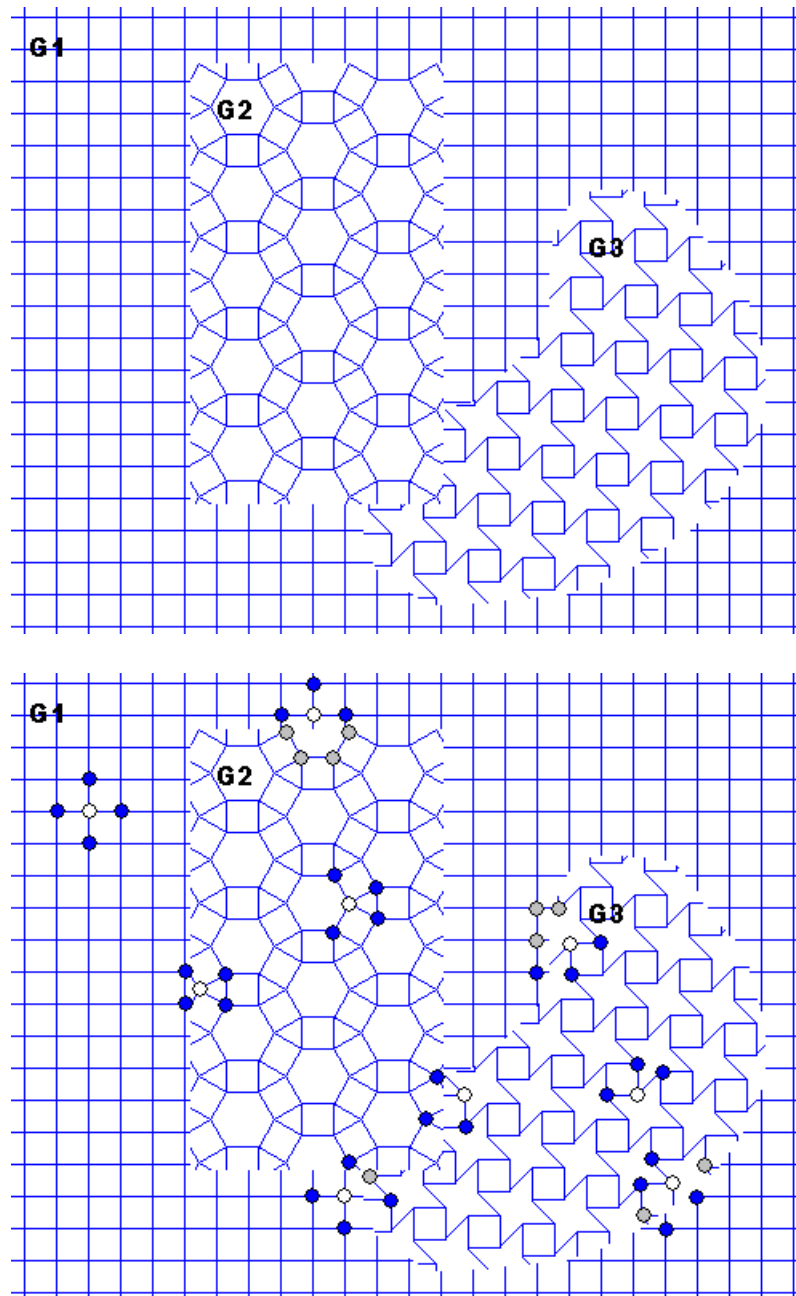
Como puede verse en la figura C.85 en la página siguiente, cada teselado tendrá configuraciones distintas¹⁴. El uso practico de varios teselados simultaneos sobre una superficie corresponde con la implementacion en el ambiente de cambios en la estructura de material o del medio de movimiento en el que estan inmersos los objetos. Otros usos son posibles.

Se espera implementar en las siguientes iteraciones del prototipo el uso de mas de una tesela sobre una superficie y mejoras en los distintos algoritmos de ubicacion de vertices en los teselados.

¹³Ver secciones C.3.4 en la página 333 y C.3.5 en la página 333

¹⁴Sistema coordenado y origen de coordenadas, rotacion, dimensiones, ... etc.

Figura C.85: Ejemplo de bordes de costura, paso entre áreas de grafómatas



G1, G2 y G3 son porciones de superficie cada una con un grafómata distinto. Observense los límites entre geometrías, un objeto en el borde de cualquiera de las áreas debe determinar el salto o paso al siguiente sistema de movimiento. Los puntos blancos en la imagen son objetos, los puntos azules son sus pasos inmediatos, los puntos grises son ubicaciones indirectamente alcanzables por el objeto.

Fuente: Autor.

Conversión idempotente: $g_{\text{base}} = g_{\text{dual}}$

Grafómata base $\longleftrightarrow$ Grafómata base

Conversión ideal: $g_{\text{base}} \neq g_{\text{dual}}$

Grafómata base $\longleftrightarrow$ Grafómata dual

Conversión real: $g_{\text{base}} \neq g_{\text{dual}}$

Grafómata base $\longrightarrow$ Grafómata dual

Grafómata base₁ $\longrightarrow$ Grafómata dual₁

Grafómata base₂ $\longrightarrow$ Grafómata dual₂

...

...

Cambio, oscilación, deformación progresiva

Fuente: Autor.

Al calcular en forma iterada el dual de un grafomata, se obtienen varios fenómenos: la conversión idempotente, la conversión ideal o perfecta y la imperfecta o variante.

La otra forma es la imperfecta. Al pretender obtener la base a partir de un dual existente, se obtiene una estructura similar a la original pero alterada, debido a problemas de redondeo, truncamiento y a la forma misma de la estructura usada como base para generar el dual. Cualquier grafomata puede presentar este comportamiento, mediante el ajuste de su estructura por ejemplo usando magnitudes irracionales en las aristas o usando poligonos no regulares. Entre los cambios producidos por las alteraciones se observan oscilaciones, degradaciones y hasta cambio estructural mediante eliminación de vertices por reduccion de la distancia entre vertices o por coincidencia locacional.

433

C.7.4. Mas allá de Tgrafomata

Con las facilidades previstas en el objeto Tgrafomata presentado en este proyecto es posible construir manualmente las estructuras de grafómatas. Hacer esto directamente sobre código fuente es una actividad lenta y engorrosa, propensa a errores y en general poco práctica.

Para facilitar la creación de grafómatas hace falta un editor especializado en dicha labor¹⁵. Con tal programa se simplifican y automatizan algunas de las tareas propias de la construcción de los grafómatas p.ej: la generación de la estructura del grafómata dual. Esta herramienta permitiría además el análisis automático de las estructuras producidas p.ej: la determinación de dibúgrafos asociados al grafómata, visualizar la forma de malla de repetición, etc. También permitiría la construcción de estructuras de vecindarios arbitrarias, específicas o de propósito particular. Un ejemplo de tales estructuras es la malla vial de una ciudad. Otras operaciones posibles serían la fusión y la subdivisión de estructuras de grafomatas.

El desarrollo del editor de grafómatas es una necesidad y una mejora al proceso manual de creación de grafómatas. Pensando a largo plazo, se fomentaría la masificación del uso de dichas estructuras si existiera alguna forma de creación automática de grafómatas. Esta actividad permitiría aumentar la cantidad y variedad de vecindarios disponibles para una posterior aplicación práctica.

Los requisitos primarios en el proceso de construcción automatizada de grafómatas son los siguientes:

- Determinar el espacio geométrico sobre el cual se realizará la búsqueda, por ejemplo en las teselaciones regulares, las teselaciones aperiódicas¹⁶, el espacio hiperbólico¹⁷, superficies esféricas, etc. Inicialmente se esperan resultados en el espacio bidimensional, pero pueden extenderse a espacios en dimensiones superiores.
- Establecer un metodo simple de construcción de grafómatas. Una forma general para determinar el grafomata a partir de una teselado es:
 1. Establecer una nube de puntos / vertices suficientemente significativa como para que contenga en su interior el patron regular que define la estructura
 2. Determinar y clasificar los distintos vertices existentes y sus conexiones
 3. Filtrar los vertices obtenidos manteniendo su conectividad hasta encontrar la forma fundamental o grafomata que define el teselado.
- Crear una herramienta especializada dedicada a la búsqueda de las estructuras. El proceso de búsqueda de grafómatas podría ser tanto en profundidad como en amplitud dentro del espacio geométrico de interés. En semejanza con los procesos de construcción realizados con los SPRT¹⁸ presentados en este proyecto, determinar la estructura de un grafómata complejo puede requerir de un importante aporte de poder computacional para su realización práctica.

¹⁵Un ejemplo operaciones sobre teselados se muestra en *Tiling Transformer*, <http://www.angelfire.com/mn3/anisohedral/tile/Tile.htm>
Una compilacion de software de edicion de teselados en *Computer Software for Tiling*:
<http://www.geom.uiuc.edu/software/tilings/TilingSoftware.html>

¹⁶Teselaciones de Penrose y de Pinwheel.

¹⁷En estos espacios se permiten uniones de poligonos regulares imposibles en el espacio euclidiano.

¹⁸Que está compuesto principalmente por elementos de construcción de combinaciones y estructuras de grafos.

Para facilitar el proceso de realizacion del editor, se proponen además:

- Determinar una forma de nomenclatura que permita dar nombre y establecer las propiedades particulares de cada grafómata encontrado ademas de un medio de representación digital de las estruturas encontradas.
- Organizar los resultados obtenidos del proceso mediante catálogos u otras estructuras de registro taxonómico.

En conclusión, una herramienta software que soporte plenamente los grafómatas debería permitir:

- La edición manual de grafómatas.
- La construcción automática de grafómatas en un espacio de búsqueda determinado, p.ej: teselaciones regulares.
- La consulta de catálogos de estructuras previamente existentes.

Con los catálogos y el trabajo que se acumule sobre los grafómatas será posible la selección automatizada del vecindario más conveniente para la representación del espacio en el desarrollo de modelos BOR o de otro tipo.

Al final, se espera que los grafómatas trasciendan su uso como estructuras de representación de espacio y puedan ser usados en otros contextos, p.ej: como funciones índice de estructuras de almacenamiento de datos, como sistemas de referencia de coordenadas no cartesianas, como mallas de discretización, *MESH* en análisis numérico, etc.

C.7.5. Beneficios de los grafomatas

Se resumen a continuacion las ventajas y desventajas encontradas durante el proceso de implementacion y uso de grafomatas:

Ventajas:

- Relaciona teselados y grafos. El grafomata es una estructura finita de vertices y aristas que abstrae un patron que se repite sobre una superficie infinita.
- Permite representar multiples tipos de vecindarios y sus duales permiten establecer los espacios de influencia de cada vertice presente en el sistema base.
- Las coordenadas en el espacio son el enlace con las ubicaciones en la nube de puntos del teselado.
- La malla regular y la estructura del grafomata permiten la ubicacion sobre cualquiera de los vertices presentes en el espacio de teselado.
- No hace uso de matrices u otros sistemas de almacenamiento para la representacion de ubicaciones, solo se requiere la estructura de datos que define al grafomata y su dual.
- Un mismo algoritmo de construccion de dibugrafos, estructuras duales y reconstruccion del teselado es aplicable a cualquier estructura de grafomata válida.
- Permite la incorporacion rapida de nuevas teselaciones, si ya existe una implementacion funcional del sistema que soporta el uso de grafomatas.
- Por ser una estructura vectorial puede aplicarse sobre ella trasformaciones como rotaciones, escalados, traslaciones, etc.

Desventajas:

- No hay un mecanismo automatico de generacion de grafomatas a partir de la nube de puntos de un teselado. El proceso hasta ahora se realiza manualmente.
- Se requiere una estructura de vertices particular para cada tipo de teselado.
- Se requieren estructuras de datos y algoritmos especializados para su implementacion. En esta versión se implemento un objeto y distintas rutinas en Dephi.
- El sistema puede no ser inmediatamente compatible con aplicaciones que realizen calculos de ubicaciones en el espacio mediante otros metodos.
- Se requieren mejoras en su formalismo teorico.

El ejercicio de la aplicacion de las ideas presentadas en este documento hará validas o no las distintas consideraciones realizadas sobre la practicidad de los grafomatas.

C.8. Conclusiones obtenidas

Luego de la presentacion realizada sobre los aspectos concernientes a los grafomatas tratados en este documento, se afirma que cualquier tipo de teselado construido mediante una estructura invariante que se repita en forma regular puede ser implementada mediante un grafómata. En este anexo se reunen a partir del concepto de grafomata:

- Una estructura simple que puede generalizarse para representar formas regulares de division del espacio y un metodo para su uso.
- Una caracterizacion de las relaciones mas basicas que describen la estructura.
- Un objeto y funciones que permiten su realizacion práctica y uso.
- Los algoritmos necesarios para la generacion de las construcciones asociadas y el dual.
- Un algoritmo para presentar graficamente el teselado asociado con la estructura.

Durante el desarrollo del concepto se produjeron algunas preguntas que sirven al propósito de dar continuidad al estudio:

- Dada una estructura de grafo, ¿como se determina que es un grafomata? ¿como se prueba que tesela el plano?
- ¿Como se construye automaticamente un grafomata a partir de un teselado?
- ¿Cuántas representaciones gráficas distintas tiene el grafomata de un teselado? y ¿Cuántas representaciones gráficas distintas tiene un dibugrafo?
- ¿Es posible generar un dibúgrafo tal que la conexión de sus aristas sea como una línea quebrada pero continua? ¿que condiciones debe cumplir el grafómata para que esto sea posible?
- ¿Es posible construir teselaciones aperiódicas mediante un metodo similar al de los grafómatas? ¿Es posible aplicar el metodo a espacios hiperbólicos u otro tipo de espacios?

Apéndice D

SPRT: Sistema de producción de reglas de transición

Introducción

Este anexo contiene una propuesta de representación teórica para la expresión de transiciones según los lineamientos declarados e inferibles presentes en la metodología de modelamiento basada en objetos y reglas, MBOR. El propósito de esta propuesta es llenar el vacío teórico existente en la metodología al establecer mediante un formalismo escrito, la representación, construcción y estudio de sistemas de transición.

Inicialmente se establece un punto de partida o génesis del sistema, se identifican los distintos conceptos y supuestos sobre los cuales descansa toda la construcción teórica: símbolo, acción de cambio y dimensión de evaluación. Con estos elementos se establece un orden y una forma de representación simples, se implementan los conceptos de primitiva y regla de transición, el cuerpo de los SPRT. En razón a la naturaleza combinatorial de la formación de las reglas, se caracterizan sus regularidades mediante patrones de transición.

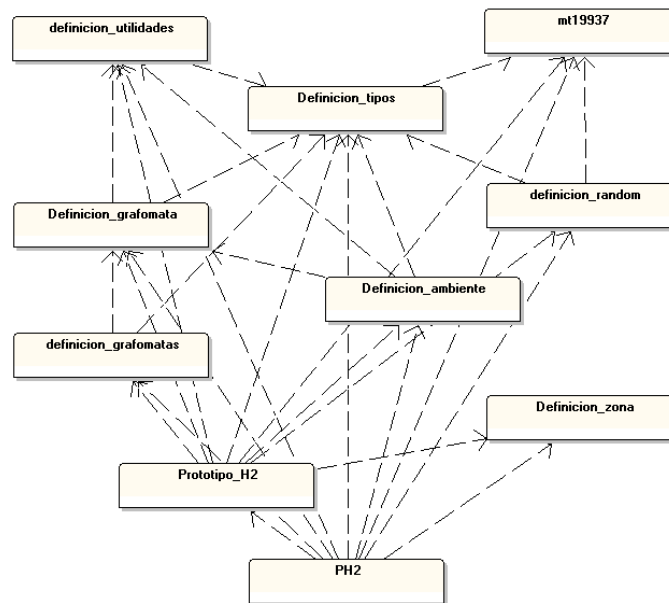
Se declara una nomenclatura particular y distintiva para la notación los SPRT presentados en este documento según la cantidad de símbolos, primitivas y dimensiones de evaluación requeridos para su construcción. Esto permite la organización de los sistemas mediante un arreglo, al que se denominó Matriz SPRT. Se analiza y caracteriza matemáticamente esta matriz, se reconocen en ella conjuntos de sistemas con propiedades particulares, como son los sistemas de referencias y los sistemas de límite de patrones, así como se presenta una extensión general de la matriz SPRT relacionada con el conjunto de primitivas a usar en cada SPRT.

Una vez conocidos los principales elementos teóricos, se da paso a la descripción formal del juego de reglas de transición existentes en la MBOR en términos de los conceptos componentes de los SPRT. La construcción teórica realizada y sus alcances junto a la formalización de las reglas de

transicion presentes en la mbor constituyen el aporte principal de esta tesis al tema de investigacion, comprenden la accion de innovacion mas importante realizada sobre el tema hasta la fecha. El diseño general para una nueva version de Homos planteado en esta tesis se basa integramente en los elementos descritos.

En la seccion E.2 en la página 517 se realiza un ejercicio de aplicación práctica de los conceptos desarrollados mediante la implementación de un generador de sistemas SPRT, el cual da soporte concreto a los elementos teóricos planteados en el presente anexo.

Figura D.1: Desarrollo del concepto de SPRT



Las flechas indican las dependencias inmediatas entre módulos.

Fuente: Autor.

D.1. Génesis de un sistema de representación de transiciones

entidades a interactuar, relacion entre entidades y orden de relación

el significado de los simbolos usados es arbitrario, constituyen el alfabeto del sistema de transiciones

los simbolos usados en en sistema son terminales. cada uno representa una entidad conocida, identificable

caracterizacion

se desarrolla una notacion para la representacion de transiciones, explicando los supuestos por los cuales se establecen los distintos elementos notacionales.

a usar por el sistema de representacion de transiciones

La incorporación de un símbolo se denotará mediante su presentación entre corchetes [*Símbolo*] seguido de su significado.

Para la creación del sistema de representación planteado en este documento, se siguió el siguiente método de construcción:

1. Establecer los elementos básicos del sistema: establecer su alfabeto.
2. Establecer relaciones entre elementos y realizar construcciones con los elementos dados
3. Determinar propiedades básicas y extensiones
4. Agregar extensiones para cubrir elementos faltantes, según estos sean requeridos.

: complejidad debido a representatividad?

símbolos y significados básicos

se describen transiciones entre símbolos

Todos los símbolos son terminales, representan una entidad o estado conocido específico.

paso entre conocido y desconocido

paso entre cosas conocidas,

determinación del sistema a usar

D.1.1. Cantidad de conocimiento en Naturaleza de las transiciones

estado de conocimiento de la expresión de transición

en adelante se notarán las expresiones de transición como primitivas, ya que nos interesa conocer la descripción de la transición, la causa y el efecto relacionados

los símbolos a igual que las expresiones de transición pueden ser conocidos o desconocidos, pasar de una causa desconocida a una conocida y así en cuatro variantes

?:?

?:C

C:?

C:C

estado de conocimiento sobre elementos en transición	Descripción según conocimiento sobre los componentes de transición		Uso en simulación
? : ?	Causa y efecto desconocidos	estados desconocidos	Sin uso práctico
? : C	De causa desconocida a efecto conocido	Descripción de estado inicial	Configuración
C : ?	De causa conocida a efecto desconocido	Descripción de estado final	Fin de simulación
C : C	Causa y efecto conocidos	Descripción de estados intermedios	Paso entre transiciones

siendo las tres últimas formas las que se usarán última

D.1.2. Representación simbólica

Uso de símbolos como representaciones

el primer símbolo a usar es la representación general de lo desconocido o [?].

Se hace uso de letras mayúsculas y minúsculas, palabras y otras denominaciones alfabéticas como representaciones de símbolos arbitrarios, cuyo significado o valor es establecido por el modelador en cada caso.

separación natural entre lo que es y no es

identificable, perceptible o no

limitaciones

representación de transiciones de cosas conocidas o al menos distinguibles. lo que no puede ser representado simbólicamente está excluido del sistema de transiciones

identificable y desconocido

elementos usables como símbolos para transiciones

símbolos a usar en el sistema de transiciones

Naturaleza de los símbolos	identificable	no identificable
conocido	C	?
desconocido	?	excluidas ya que no son representables

conocido identificable: identificación de una cosa, sometido al grado de conocimiento que se tenga sobre ella.

Tabla D.1: representación

Representación abstracta	Significado	Representación particular	Significado
$\bigcirc, \triangle, \square, \dots$	Cosas discernibles	$?$	Cosa desconocida, abstracción general de lo desconocido, expresión por defecto
$\triangle_k$	Cosa discernible k	$?_k$	Cosa desconocida k
$\triangle_1, \triangle_2, \triangle_3, \dots, \triangle_n$	Conjunto de cosas discernibles diferentes de tipo $\triangle$	$?_1, ?_2, ?_3, \dots, ?_n$	Conjunto de cosas desconocidas distintas y numeradas
		C	Cosa conocida
		C_k	Cosa conocida k
		$C_1, C_2, C_3, \dots, C_n$	Conjunto de cosas conocidas distintas y numeradas de tipo C
		$C_1 = a, C_2 = b, \dots, C_n =$ $a, b, c, \dots, n$	Metaforización - igualdad simbólica Conjunto de cosas conocidas distintas

Fuente: Autor.

desconocido identificable: cosa que aun no ha sido racionalizada pero que es perceptible e individualizable, tratable como una cosa.

conocido no identificable: corresponde con la descripcion de intangibles y colectivos

desconocido no identificable: cosa no representable simbolicamente, autoexcluida del sistema de transiciones debido a su irrepresentabilidad

el estatus conocido/desconocido implica la existencia de las cosas referenciadas. la identificabilidad esta relacionada con la percepcion de las cosas por nuestros sentidos e intelecto.

soporte de transiciones

El paso de simbolos en el sistema de transiciones se entiende inicialmente como solo entre simbolos, descripciones. la explicacion del paso de elementos representados metaforicamente por los simbolos se explica fenomenologicamente, donde cada transicion hace explicito que es lo que relaciona, la entrada y salida de una funcion de caja negra (la transicion), obviando los detalles y particularidades que explican como es que se realiza la transicion.

transiciones como cajas negras, solo se ve lo que entra y sale de cada caja, no se explican los detalles de las transiciones

en igualdad de condici

$Promedio (?_1, ?_2, ?_2, \dots, ?_n) = ?$

$C_1, C_2, C_3, \dots, C_n = a, b, c, \dots, n$

$? = Promedio \left(?_1, ?_2, ?_3, \dots, ?_n \right) C = Promedio \left(c_1, c_2, c_3, \dots, c_n \right)$

$a, b, c, \dots$ Una vez que se determina la entidad, se la nombra, de da la simbolización, seleccion de lo desconocido. caracterizacion descripcion identificacion particularización y generalidad

en general para este documento se usaran letras mayusculas y minusculas como representaciones de simbolos. se hace asi para simplificar el identificador de cada simbolo, por generalidad y conveniencia.

el triangulo simboliza a una cosa. se usa esta figura para representar una cosa debido a su universalidad

C cosa conocida

? cosa desconocida

cantidad de conocimiento sobre expresiones

construccion de transiciones a opartur de conocimiento sobre transiciones

mediante nS se identifica la cantidad de simbolos distintos usados

D.1.3. Acción de cambio o transición

estado1:estado2

Tabla D.2: Metáforas de transición

	→	
Identificador precursor	Sentido de avance de transición	Identificador consecuente
Entrada	Función de Caja negra	Salida
Parámetro	operador de cambio	Resultado
Inicial	:	Final
Actual	:	Siguiente
Presente	:	Futuro
Fuente / Origen	:	Destino
...		

Fuente: Autor.

momento1:momento2

[:] simbolo de transicion, separador de la accion de cambio entre estados

por convencion se establece que el elemento a la izquierda del simbolo de paso es lo que se confirma inicialmente para dar paso a lo siguiente o final

descripcion de estados relacionados

con respecto al tiempo, las transiciones tienen dos componentes principales

momento de transición: instante en el cual se realiza la transición, con respecto a un marco temporal previamente establecido.

duración de transición: cantidad de tiempo requerido para ejecución completa de la transición a partir del momento en que esta se inicia.

una vez realizada la transición esta permanece hasta la siguiente oportunidad de cambio, según lo establecido en el sistema de transiciones.

las transiciones en los sprt pueden ser temporales, aplicables en secuencia dentro de un sistema referencial de tiempo o atemporales, con propósitos descriptivos.

es una relación 1 a 1 entre símbolos enlazados por la causalidad de su relación

Transición descripción de un evento, en el cual se cambia de un estado a otro, de una descripción a otra, de una cosa a otra. Una transición puede tener o no una asociación con el tiempo.

Transiciones atemporales Explicitan una relación de cambio, pero no tiene en cuenta el paso del tiempo. Hace manifiesta una separación de estados pero no dice nada acerca de la duración del evento.

Transiciones temporales Representa el paso de un momento a otro en el tiempo, es una transición atemporal a la cual se le da un marco temporal. La duración de la transición se mide en unidades de tiempo previamente establecidas, estas pueden ser arbitrarias o específicas.

D.1.3.1. Relación causa y efecto

expresión de transición

conocida

desconocida

simple o compleja = simple, solo una transición compleja varias transiciones simultáneas

A partir de la descripción de cambio o la transición, se establece lo que es inicial y final al proceso de cambio.

para que sea posible el cambio descrito, se requiere que lo inicial esté presente para que pueda darse el paso a lo final descrito en la transición

identifiquemos una entidad conocida C

[causa]:[efecto]

construcción de complejidad de transición mediante adición de dimensiones de evaluación

las dimensiones de evaluación se dan simultáneamente toda la expresión se evalúa al unísono

[?:?]

????C

???C?

??C?? independencia entre expresiones de transición

?C???

C????

cada dimensión de evaluación agrega independencia al sistema?

estado de conocimiento sobre la transición

combinaciones de transiciones

en la siguiente tabla se establecen

Tabla D.4: Expresiones de transición y estado de conocimiento de la dimensión de evaluación

Estado de conocimiento de las dimensiones de evaluación que componen una expresión de transición		Ordenamiento de dimensiones de evaluación										Expresiones de transición según permutaciones en el orden de las dimensiones de evaluación que las componen	
	?	1	1	1	1	2	1	1	3	3	1	?	
0D	?											?	
1D	?			C								?	C
2D	??		??	CC								??	??C CC
3D	???		???	??C	CCC							???	??C ??C C?
4D	????		????	??CC	??CC	CCCC		1	4	6	4	???	???C ??C? ?C?? C???
5D	?????		?????	???CC	??CCC	??CCCC	CCCCC	1	5	10	10	5	1
...													
%	100%?	$n\%? + (n - 100)\% C$										100%?	$n\%? + (n - 100)\% C$

Los símbolos α y γ representan el estado de conocimiento asociado a las dimensiones de evaluación que componen cada expresión de transición. Solo las expresiones compuestas por transiciones conocidas son útiles al propósito de descripción fenomenológico de los SPRT. P.ej: CCC es una expresión de transición completamente conocida de tres dimensiones de evaluación. Las expresiones con elementos desconocidos pueden usarse como puente entre expresiones con diferente cantidad de dimensiones de evaluación.

Fuente: Autor.

cada ? y C describe una dimension de evaluacion.

se hace notar que en la practica, los sistemas de transiciones usan solo las combinaciones completamente conocidas

pensamiento combinatorio en descripcion de transiciones primera

[?:C]

[C:?]

[C:C]

expresion de transicion

D.1.4. Dimensiones de evaluación

cada relacion causa efecto es una dimension de evaluacion

para la representacion de entidades, se comprende que coexisten simultaneamente multiples

que estas entidades se interrelacionan y que pueden describirse sus interacciones de forma descriptiva causa y efecto

cada relacion de causa y efecto es una dimension de evaluacion.

la agrupacion relaciones simultaneas de cambio

orden y evaluacion

mediante la particula nD donde n indican la cantidad de evaluaciones simultaneas

la cantidad de dimensiones de evaluacion se representa con nD donde

dimension de evaluacion

relacion uno a uno entre causa efecto

la dimension cero es la no evaluacion

la dimension 1 es la autoevaluacion

en general a partir de

para n dimensiones de evaluacion simultaneas, habra una dimension de autoevaluacion y las demas estan relacionadas con los vecinos

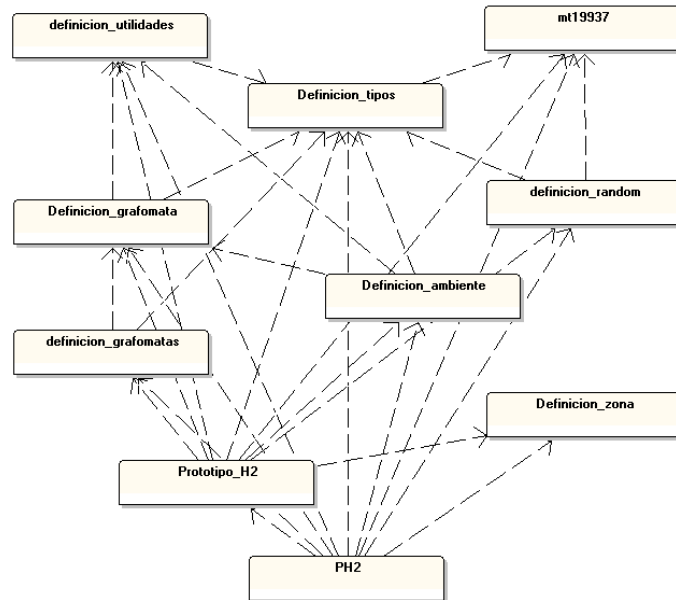
a partir de dos o mas dimensiones se tienen en cuenta a los vecinos, autoevaluacion y un vecino

? y C

porque se llaman dimensiones de evaluacion?

D.2. Fundamentos de un SPRT

Figura D.2: diseño de SPRT



Las flechas indican las dependencias inmediatas entre módulos.

Fuente: Autor.

D.2.1. Protonotación

D.2.2. Componentes

simbolos, dimensiones de evaluacion, simultaneidad

D.2.3. Primitivas de transición

Es la descripción mínima de una transición entre símbolos, implementa la idea de transición expuesta en [tal]

para escribir una primitiva se deben determinar los símbolos causa y efecto, separándolos mediante el operador de cambio [:] dos puntos

cumple la función gramatical de separar dos preposiciones independientes

Tabla D.5: Patrones de transición fundamentales

Notación	Significado	Cambio poblacional	Cambio espacial
A:A	Idempotencia de A	Sin cambios	Sin cambios
A:B	Remplazo de A por B	-A +B	Sin cambios

Tabla D.6: Primitivas de transición fundamentales

Transiciones fundamentales

Notación	Significado	Cambio poblacional	Cambio espacial
A:A	Idempotencia de A	Sin cambios	Sin cambios
A:B	Remplazo de A por B	-A +B	Sin cambios

Primitivas de transición fundamentales

Notación	Significado	Cambio poblacional	Cambio espacial
_:A	Instanciación de A	+1A	-1E
A:A	Permanencia de A	Sin cambios	Sin cambios
A:B	Cambio de A por B	-1A , +1B	Sin cambios
A:_	Terminación de A	-1A	+1E

Fuente: Autor.

cada primitiva describe el aporte de cambio que

D.2.4. Reglas de transición

notacion de adición o combinacion de primitivas

notacion de orden o permutación de primitivas

En la notacion de orden se usan los corchetes para delimitar la expresion de transición.

aporte de las reglas de transicion

Las reglas de transicion de los sistemas con una dimension son notacionalmente iguales a las primitivas de transición. a partir de la segunda dimension se hace notar la simultaneidad de la notacion mediante la ordenacion escrita de las primitivas dentro de la expresion de transicion

la expresion de transicion mas simple es la regla de transicion unidimensional, que es equivalente a una primitiva de transicion.

Tabla D.8: Patrones de transición en primitivas de transición

Notación	Significado	Cambio poblacional	Cambio espacial
A:A	Idempotencia de A	Sin cambios	Sin cambios
A:B	Remplazo de A por B	-A +B	Sin cambios

Estos son los patrones de transicion presentes en los sistemas SPRT nS1D

Tabla D.7: Formas de notación de reglas de transición

Notación de Adición / Combinación de primitivas		Notación de Orden / Permutación de primitivas
Por presencia de términos	Por orden de términos	
a:b	a:b	[a:b]
a:c + a:d	a:c + a:d a:d + a:c	[a,a:c,d] [a,a:d,c]
a:c + b:d	a:c + b:d b:d + a:c	[a,b:c,d] [b,a:d,c]
a:a + a:a + a:a	a:a + a:a + a:a	[a,a,a:a,a,a]
a:a + a:a + a:c	a:a + a:a + a:c a:a + a:c + a:a a:c + a:a + a:a	[a,a,a:a,a,c] [a,a,a:a,c,a] [a,a,a:c,a,a]
a:a + a:a + b:c	a:a + a:a + b:c a:a + b:c + a:a b:c + a:a + a:a	[a,a,b:a,a,c] [a,b,a:a,c,a] [b,a,a:c,a,a]
a:d + b:e + c:f	a:d + b:e + c:f a:d + c:f + b:e b:e + a:d + c:f b:e + c:f + a:d c:f + a:d + b:e c:f + b:e + a:d	[a,b,c:d,e,f] [a,c,b:d,f,e] [b,a,c:e,d,f] [b,c,a:e,f,d] [c,a,b:f,d,e] [c,b,a:f,e,d]

Fuente: Autor.

D.2.5. Patrones de transición

D.2.5.1. Determinación de patrones a partir de reglas de transición

Es posible determinar el patrón de transicion al que pertenece una regla de transición mediante el siguiente procedimiento:

Tabla D.9: Notación de patrones de transición

Ejemplo	Notación de Adición / Combinación de primitivas	Notación de Orden / Permutación de primitivas	Reglas formadas con los símbolos d,e,f,g,h,i siguiendo la estructura de los patrones de transición según cada forma de notación	
e_1	A:B	A:B	d:e g:i h:f	d:e g:i h:f
e_2	A:C + A:D	A,A:C,D A,A:D,C	f:e + f:i h:d + h:g	f,f:e,i h,h:d,g
e_3	A:C + B:D	A,B:C,D B,A:D,C	d:f + e:g	d,e:f,g e,d:g,f
e_4	A:A + A:A + A:A	A,A,A:A,A,A	h:h + h:h + h:h	h,h,h:h,h,h
e_5	A:A + A:A + A:C	A,A,A:A,A,C A,A,A:A,C,A A,A,A:C,A,A	g:g + g:g + g:h	g,g,g:g,g,h g,g,g:g,h,g g,g,g:h,g,g
e_6	A:A + A:A + B:C	A,A,B:A,A,C A,B,A:A,C,A B,A,A:C,A,A	e:e + e:e + h:i	e,e,h:e,e,i e,h,e:e,i,e h,e,e:i,e,e
e_7	A:D + B:E + C:F	A,B,C:D,E,F A,C,B:D,F,E B,A,C:E,D,F B,C,A:E,F,D C,A,B:F,D,E C,B,A:F,E,D	d:g + e:h + f:i	d,e,f:g,h,i d,f,e:g,i,h e,d,f:h,g,i e,f,d:h,i,g f,d,e:i,g,h f,e,d:i,h,g

Fuente: Autor.

Para una regla de transición dada:

1. Indizar identificadores: Se establece un índice único para cada símbolo presente en la regla.
2. Permutar índices: Se genera una tabla que contiene la representación de la regla en términos de la permutación de los índices asignados. El total de permutaciones a realizar es igual al factorial de la cantidad de símbolos distintos presentes en la regla.
3. Ordenar primitivas en regla: Se ordenan las filas de la tabla anterior según el valor de los índices de las primitivas que componen a la regla de transición. Esta ordenación determina si se busca una representación máxima o mínima de índices del patrón de transición.
4. Determinar expresión máxima o mínima del patrón: Se ordenan las expresiones obtenidas, se determina la mayor o menor expresión de acuerdo con el orden de primitivas elegido en el paso anterior.
5. Una vez elegida la representación, esta se almacena para realizar con ella futuras comparaciones.

El procedimiento descrito se aplica a todas las reglas a evaluar. En la tabla D.10 se muestra un ejercicio de aplicación de búsqueda de patrones de transición. Para detectar los patrones solo se requiere la

transformación uniforme de las reglas de transición a una de las representaciones, la máxima o la mínima. Se muestran ambas formas con fines didácticos.

Las reglas $R_1 \dots R_5$ usan la notación de combinación. En el ejemplo, solo R_3 y R_4 pertenecen al mismo tipo de patrón de transición. Se señala con [M] y [m] la combinación de permutación de índices según un orden establecido de primitivas con la cual se determina una secuencia que normaliza la identificación de patrones de transición.

En esta sección los patrones de transición obtenidos se representan mediante índices numéricos. Para propósitos de notación, se hace uso de las letras mayúsculas para identificar expresiones de patrones de transición. Las reglas de transición se notarán en minúsculas. Esta convención se observará con frecuencia más adelante en este documento.

D.3. Sistemas de SPRT

nSkPmD

SPRT							
	Incompletos	$k < n^2$	Hay seleccion de primitivas, se omiten primitivas			nSmD	
nSkPmD	Completos	$k = n^2$	Se hace uso de todas las primitivas disponibles			nRD	
	Redundantes	$k \geq n^2$	Hace uso de todas las primitivas y se repiten algunas			2mSmD	
		$k \neq n^2$					

según el uso que se haga de las primitivas de transición, los SPRT pueden ser completos, incompletos o redundantes.

un SPRT completo es aquel que hace uso de todas las primitivas de transición que es posible configurar con el conjunto de símbolos

de transición incompleta o incompletos,

se hace selección de primitivas, esto es, se usan solo algunas primitivas de transición de todo el repertorio que es posible generar con la tabla de símbolos disponible

los modelos, por la selección que hacen de reglas de transición, pueden verse como expresiones de sistemas incompletos, ya que hacen uso de un conjunto limitado de primitivas de transición.

de transición completa

Tabla D.10: Determinación de patrones a partir de reglas de transición

$R_1 = a:a + a:a$					$Patrón_m$					$Patrón_M$				
Indíces		Permutaciones			$p_i < p_{i+1}$					$p_i > p_{i+1}$				
a		a	a	a	a					s_1	s_2	s_3	s_4	R
1		1	1	1	1	1	1	1	1	1	1	1	1	M

$R_2 = a:a + b:a$					$Patrón_m$					$Patrón_M$				
Indíces		Permutaciones			$p_i < p_{i+1}$					$p_i > p_{i+1}$				
a	b	a	a	b	a					s_1	s_2	s_3	s_4	R
1	2	1	1	2	1	1	2	1	m	2	1	1	1	M
2	1	2	2	1	2	1	2	2		2	2	1	2	

$R_3 = a:a + b:c$					$Patrón_m$					$Patrón_M$						
Indíces		Permutaciones			$p_i < p_{i+1}$					$p_i > p_{i+1}$						
a	b	c	a	a	b	c	s_1	s_2	s_3	s_4	R	s_1	s_2	s_3	s_4	R
1	2	3	1	1	2	3	1	1	2	3	m	2	3	1	1	M
1	3	2	1	1	3	2	1	1	3	2		3	2	1	1	
2	1	3	2	2	1	3	1	3	2	2		2	2	1	3	
2	3	1	2	2	3	1	2	2	3	1		3	1	2	2	
3	1	2	3	3	1	2	1	2	3	3		3	3	1	2	
3	2	1	3	3	2	1	2	1	3	3		3	3	2	1	

$R_4 = c:a + b:b$					$Patrón_m$					$Patrón_M$						
Indíces		Permutaciones			$p_i < p_{i+1}$					$p_i > p_{i+1}$						
a	b	c	c	a	b	b	s_1	s_2	s_3	s_4	R	s_1	s_2	s_3	s_4	R
1	2	3	3	1	2	2	2	2	3	1	m	3	1	2	2	M
1	3	2	2	1	3	3	2	1	3	3		3	3	2	1	
2	1	3	3	2	1	1	1	1	3	2		3	2	1	1	
2	3	1	1	2	3	3	1	2	3	3		3	3	1	2	
3	1	2	2	3	1	1	1	1	2	3		2	3	1	1	
3	2	1	1	3	2	2	1	3	2	2		2	2	1	3	

$R_5 = c:a + b:a$					$Patrón_m$					$Patrón_M$						
Indíces		Permutaciones			$p_i < p_{i+1}$					$p_i > p_{i+1}$						
a	b	c	c	a	b	a	s_1	s_2	s_3	s_4	R	s_1	s_2	s_3	s_4	R
1	2	3	3	1	2	1	2	1	3	1	m	3	1	2	1	M
1	3	2	2	1	3	1	2	1	3	1		3	1	2	1	
2	1	3	3	2	1	2	1	2	3	2		3	2	1	2	
2	3	1	1	2	3	2	1	2	3	2		3	2	1	2	
3	1	2	2	3	1	3	1	3	2	3		2	3	1	3	
3	2	1	1	3	2	3	1	3	2	3		2	3	1	3	

Fuente: Autor.

Tabla D.11: Determinación de patrones en reglas de transición: Diferencias de representación y relaciones entre símbolos

Representación de un patrón según símbolos disponibles										Prueba de generalidad													
Regla		[a, b:a, c]			[b, c:b, a]			[b, a:c, b]			[b, a:c, a]			S2			[f, d:f, b]						
S1		E1			E2			E3			E4			S2			E5						
a	b	c	a	b	a	c	b	a	c	b	b	a	c	a	b	d	f	d	f	b	R		
1	2	3	1	2	1	3	m	2	3	2	1	2	1	3	1	1	2	3	2	3	1	M	
1	3	2	1	3	1	2		3	2	3	1	3	1	2	1	1	3	2	2	3	2	1	
2	1	3	2	1	2	3		1	3	1	2	1	2	3	2	2	1	3	3	1	3	2	
2	3	1	2	2	2	1		2	1	2	2	2	2	1	2	2	3	1	1	3	1	2	
3	1	2	3	1	3	2		1	2	1	3	1	3	2	3	3	1	2	2	1	2	3	
3	2	1	3	2	3	1	M	2	1	2	3	2	3	1	3	3	2	1	1	2	1	3	m

Se señala la representación R mayor [M] y menor [m] del patrón de transición en términos de los índices de permutación asignados a cada símbolo presente en la regla de transición en análisis. Se requiere que cada símbolo se asocie a un índice único, no importa el orden en que se realicen las asignaciones de los índices. El patrón al que pertenece la regla de transición aparece identificado de forma única. Patrón(E1) = Patrón(E2) = Patrón(E4) ≠ Patrón(E3)

Tabla D.12: Determinación de patrones en reglas de transición: Índices e Invariabilidad de representación

Representación de un patrón según símbolos disponibles															
Regla S1			[a,b:a,c] E1					S1'			[a,b:a,c] E1				
a	b	c	a	b	a	c	R	b	a	c	a	b	a	c	R
1	2	3	1	2	1	3	m	1	2	3	2	1	2	3	
1	3	2	1	3	1	2		1	3	2	3	1	3	2	
2	1	3	2	1	2	3		2	1	3	1	2	1	3	m
2	3	1	2	2	2	1		2	3	1	3	2	3	1	M
3	1	2	3	1	3	2		3	1	2	1	3	1	2	
3	2	1	3	2	3	1	M	3	2	1	2	3	2	1	

Se señala la representación R mayor [M] y menor [m] del patrón de transición en términos de los índices de permutación asignados a cada símbolo presente en la regla de transición en análisis. Se requiere que cada símbolo se asocie a un índice único, no importa el orden en que se realicen las asignaciones de los índices. El patrón al que pertenece la regla de transición aparece identificado de forma única. Patrón(E1) = Patrón(E2) = Patrón(E4) ≠ Patrón(E3)

el sistema hace uso de todas y cada una de las primitivas de transición disponibles, es posible hacer transición de un símbolo a otro cualquiera en un solo paso.

de transición redundante

por redundancia de símbolos

razones de existencia de sistemas redundantes

un SPRT es redundante cuando en él existen repeticiones de una o más primitivas de transición. esto puede producirse por la existencia de identificadores repetidos dentro del conjunto de símbolos disponibles en el SPRT, ya que cada primitiva es una función uno a uno de paso entre los símbolos disponibles, provocando la repetición de primitivas.

por redundancia explícita de primitivas

se introducen en la configuración del sistema repeticiones de una o más primitivas ya existentes, con el propósito de aumentar el aporte de la primitiva repetida en la estructura de reglas de transición del SPRT.

las diferencias mostradas entre sistemas se hacen con fines expositivos.

En general y a causa de

debido a la selección de reglas que se realiza durante el para la construcción de modelos, estos pueden interpretarse como versiones reducidas de SPRT incompletos o redundantes.

D.3.1. Características generales

Tabla D.13: Clasificación de SPRT según primitivas de transición

Caracterización de un SPRT					
Nombre del Sistema	Nulo, sin transiciones 0S0D 0SnD nS0D	nSkPmD			
		Incompleto		Completo	
Primitivas en uso	$P = 0$	$1 \leq P < S^2$		$P = S^2$	
Total de conjuntos	1	p^2	Dependiente de forma	1	Dependiente de forma
Sistemas según redundancia / alienación		Sin redundancia	Con redundancia	Sin redundancia	Con redundancia
Tendencia del sistema		Desbalanceado debido a la presencia o ausencia de aportes	Desbalance por los aportes de repeticiones y la ausencia o presencia de primitivas	En equilibrio, hay igualdad de aportes de todos los símbolos	Desbalance causado por los aportes de las repeticiones

Fuente: Autor.

D.3.1.1. La Matriz SPRT

Figura D.3: Definición completa de la matriz SPRT $nSmD$

$$M_{nSmD} = \begin{pmatrix} 0S0D & 0S1D & 0S2D & 0S3D & \cdots & 0SmD \\ 1S0D & 1S1D & 1S2D & 1S3D & \cdots & 1SmD \\ 2S0D & 2S1D & 2S2D & 2S3D & \cdots & 2SmD \\ 3S0D & 3S1D & 3S2D & 3S3D & \cdots & 3SmD \\ 4S0D & 4S1D & 4S2D & 4S3D & \cdots & 4SmD \\ \cdots & \cdots & \cdots & \cdots & \ddots & \cdots \\ nS0D & nS1D & nS2D & nS3D & \cdots & nSmD \end{pmatrix}$$

Fuente: Autor.

matriz sprt completa?

cantidad de reglas de transicion

cantidad de reglas independientes?

cantidad de patrones de transicion?

conjuntos particulares de sprt contenidos en la matriz sprt

de referencias

de limite de patrones

que pasa con los sistemas con sleccion de primitivas? con mas o menos primitivas que $n^{2??}$

D.3.1.2. Total de reglas independientes en sistemas SPRT

D.4. SPRT de símbolos - nSmD

Figura D.4: Sistemas de Símbolos: Matriz de SPRT nSmD

$$M_{nSmD} = \begin{pmatrix} 1S1D & 1S2D & 1S3D & \dots & 1SmD \\ 2S1D & 2S2D & 2S3D & \dots & 2SmD \\ 3S1D & 3S2D & 3S3D & \dots & 3SmD \\ 4S1D & 4S2D & 4S3D & \dots & 4SmD \\ \dots & \dots & \dots & \ddots & \dots \\ nS1D & nS2D & nS3D & \dots & nSmD \end{pmatrix}$$

Fuente: Autor.

descripcion de la matriz SPRT

la primera fila contiene los sistemas permanentes

la primera columna describe todos los sistemas

la diagonal principal de la matriz describe los sistemas de referencias

los elementos referenciados en la matriz con 2mSmM esto es

para los sistemas en los cuales la cantidad de simbolos es igual al doble de las dimensiones del sistema describe el sistema limite de representacion de patrones en cada columna de la matriz SPRT. en estos sistemas se genera la mayor cantidad posible de patrones de transicion que es posible representar con el sistema de transiciones. los sistemas mayores a los 2nSnD

En general las primitivas describen relaciones de cambio.

la permanencia es en si misma una operacion de cambio en la cual no se realiza cambio alguno. En la operacion de cambio se realiza el remplazo de un simbolo por otro

Tabla D.14: Distribución de reglas de transición independientes en sistemas de SPRT nSmD con selección de primitivas

S	$P \leq \xi^2$	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	1	2	3	4	5	6	7	8	9	10	11
	3	1	3	6	10	15	21	28	36	45	55	66
	4	1	4	10	20	35	56	84	120	165	220	286
	5	1	5	15	35	70	126	210	330	495	715	1.001
3	6	1	6	21	56	126	252	462	792	1.287	2.002	3.003
	7	1	7	28	84	210	462	924	1.716	3.003	5.005	8.008
	8	1	8	36	120	330	792	1.716	3.432	6.435	11.440	19.448
	9	1	9	45	165	495	1.287	3.003	6.435	12.870	24.310	43.758
4	10	1	10	55	220	715	2.002	5.005	11.440	24.310	48.620	92.378
	11	1	11	66	286	1.001	3.003	8.008	19.448	43.758	92.378	184.756
	12	1	12	78	364	1.365	4.368	12.376	31.824	75.582	167.960	352.716
	13	1	13	91	455	1.820	6.188	18.564	50.388	125.970	293.930	646.646
5	14	1	14	105	560	2.380	8.568	27.132	77.520	203.490	497.420	1.144.066
	15	1	15	120	680	3.060	11.628	38.760	116.280	319.770	817.190	1.961.256
	16	1	16	136	816	3.876	15.504	54.264	170.544	490.314	1.307.504	3.268.760
	17	1	17	153	969	4.845	20.349	74.613	245.157	735.471	2.042.975	5.311.735
6	18	1	18	171	1.140	5.985	26.334	100.947	346.104	1.081.575	3.124.550	8.436.285
	19	1	19	190	1.330	7.315	33.649	134.596	480.700	1.562.275	4.686.825	13.123.110
	20	1	20	210	1.540	8.855	42.504	177.100	657.800	2.220.075	6.906.900	20.030.010
	21	1	21	231	1.771	10.626	53.130	230.230	888.030	3.108.105	10.015.005	30.045.015
7	22	1	22	253	2.024	12.650	65.780	296.010	1.184.040	4.292.145	14.307.150	44.352.165
	23	1	23	276	2.300	14.950	80.730	376.740	1.560.780	5.852.925	20.160.075	64.512.240
	24	1	24	300	2.600	17.550	98.280	475.020	2.035.800	7.888.725	28.048.800	92.561.040
	25	1	25	325	2.925	20.475	118.755	593.775	2.629.575	10.518.300	38.567.100	131.128.140
8	26	1	26	351	3.276	23.751	142.506	736.281	3.365.856	13.884.156	52.461.256	183.579.396
	27	1	27	378	3.654	27.405	169.911	906.192	4.272.048	18.156.204	70.607.460	254.186.856
	28	1	28	406	4.060	31.465	201.376	1.107.568	5.379.616	23.535.820	94.143.280	348.330.136
	29	1	29	435	4.495	35.960	237.336	1.344.904	6.724.520	30.260.340	124.403.620	472.733.756
9	30	1	30	465	4.960	40.920	278.256	1.623.160	8.347.680	38.608.020	163.011.640	635.745.396
	31	1	31	496	5.456	46.376	324.632	1.947.792	10.295.472	48.903.492	211.915.132	847.660.528
	32	1	32	528	5.984	52.360	376.992	2.324.784	12.620.256	61.523.748	273.438.880	1.121.099.408
	33	1	33	561	6.545	58.905	435.897	2.760.681	15.380.937	76.904.685	350.343.565	1.471.442.973

Tabla D.15: Distribución de reglas de transición independientes en sistemas de SPRT nSmD con selección de primitivas

s	$p \leq s^2$	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
6	34	1	34	595	7.140	66.045	501.942	3.262.623	18.643.560	95.548.245	445.891.810	1.917.334.783
	35	1	35	630	7.770	73.815	575.757	3.838.380	22.481.940	118.030.185	563.921.995	2.481.256.778
	36	1	36	666	8.436	82.251	658.008	4.496.388	26.978.328	145.008.513	708.930.508	3.190.187.286
	37	1	37	703	9.139	91.390	749.398	5.245.786	32.224.114	177.232.627	886.163.135	4.076.350.421
7	38	1	38	741	9.880	101.270	850.668	6.096.454	38.320.568	215.553.195	1.101.716.330	5.178.066.751
	39	1	39	780	10.660	111.930	962.598	7.059.052	45.379.620	260.932.815	1.362.649.145	6.540.715.896
	40	1	40	820	11.480	123.410	1.086.008	8.145.060	53.524.680	314.457.495	1.677.106.640	8.217.822.536
	41	1	41	861	12.341	135.751	1.221.759	9.366.819	62.891.499	377.348.994	2.054.455.634	10.272.278.170
	42	1	42	903	13.244	148.995	1.370.754	10.737.573	73.629.072	450.978.066	2.505.433.700	12.777.711.870
	43	1	43	946	14.190	163.185	1.533.939	12.271.512	85.900.584	536.878.650	3.042.312.350	15.820.024.220
	44	1	44	990	15.180	178.365	1.712.304	13.983.816	99.884.400	636.763.050	3.679.075.400	19.499.099.620
	45	1	45	1.035	16.215	194.580	1.906.884	15.890.700	115.775.100	752.538.150	4.431.613.550	23.930.713.170
	46	1	46	1.081	17.296	211.876	2.118.760	18.009.460	133.784.560	886.322.710	5.317.936.260	29.248.649.430
	47	1	47	1.128	18.424	230.300	2.349.060	20.338.520	154.143.080	1.040.465.790	6.358.402.050	35.607.051.480
	48	1	48	1.176	19.600	249.900	2.598.960	22.957.480	177.100.560	1.217.566.350	7.575.968.400	43.183.019.880
	49	1	49	1.225	20.825	270.725	2.869.685	25.827.165	202.927.725	1.420.494.075	8.996.462.475	52.179.482.355
	50	1	50	1.275	22.100	292.825	3.162.510	28.989.675	231.917.400	1.652.411.475	10.648.873.950	62.828.356.305
	51	1	51	1.326	23.426	316.251	3.478.761	32.468.436	264.385.836	1.916.797.311	12.565.671.261	75.394.027.566
	52	1	52	1.378	24.804	341.055	3.819.816	36.288.252	300.674.088	2.217.471.399	14.783.142.660	90.177.170.226
	53	1	53	1.431	26.235	367.290	4.187.106	40.475.358	341.149.446	2.558.620.845	17.341.763.505	107.518.933.731
8	54	1	54	1.485	27.720	395.010	4.582.116	45.057.474	386.206.920	2.944.827.765	20.286.591.270	127.805.525.001
	55	1	55	1.540	29.260	424.270	5.006.386	50.063.860	436.270.780	3.381.098.545	23.667.689.815	151.473.214.816
	56	1	56	1.596	30.856	455.126	5.461.512	55.252.372	491.796.152	3.872.894.697	27.540.584.512	179.013.799.328
	57	1	57	1.653	32.509	487.635	5.949.147	61.474.519	553.270.671	4.426.165.568	31.966.749.880	210.980.549.208
	58	1	58	1.711	34.220	521.855	6.471.002	67.945.521	621.216.192	5.047.381.560	37.014.131.440	247.994.680.648
	59	1	59	1.770	35.990	557.845	7.028.847	74.974.368	696.190.560	5.743.572.120	42.757.703.560	290.752.384.208
	60	1	60	1.830	37.820	595.665	7.624.512	82.598.880	778.789.440	6.522.361.560	49.280.065.120	340.032.449.328
	61	1	61	1.891	39.711	635.376	8.239.888	90.858.768	869.648.208	7.392.009.768	56.672.074.888	396.704.524.216
	62	1	62	1.953	41.664	677.040	8.936.928	99.795.696	969.443.904	8.361.453.672	65.033.528.560	461.738.052.776
	63	1	63	2.016	43.680	720.720	9.657.648	109.453.344	1.078.897.248	9.440.350.920	74.473.879.480	536.211.032.256
	64	1	64	2.080	45.760	766.480	10.424.128	119.877.472	1.198.774.720	10.639.125.640	85.113.005.120	621.324.937.376
	65	1	65	2.145	47.905	814.385	11.238.513	131.115.985	1.329.890.705	11.969.016.345	97.082.021.465	718.406.958.841
	66	1	66	2.211	50.116	864.501	12.103.014	143.218.999	1.473.109.704	13.442.126.049	110.524.147.514	828.931.106.355
	67	1	67	2.278	52.394	916.895	13.019.909	156.238.908	1.629.348.612	15.071.474.661	125.595.622.175	954.526.728.530

Tabla D.16: Distribución de reglas de transición independientes en sistemas de SPRT nSmD con selección de primitivas

s	$P \leq s^2$	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
68	1	68	2.346	54.740	971.635	13.991.544	170.230.452	1.799.579.064	16.871.053.725	142.466.675.900	1.096.993.404.430
69	1	69	2.415	57.155	1.028.790	15.020.334	185.250.786	1.984.829.850	18.855.883.575	161.322.559.475	1.258.315.963.905
70	1	70	2.485	59.640	1.088.430	16.108.764	201.359.550	2.186.189.400	21.042.072.975	182.364.632.450	1.440.680.596.355
71	1	71	2.556	62.196	1.150.626	17.259.390	218.618.940	2.404.808.340	23.446.881.315	205.811.513.765	1.646.492.110.120
72	1	72	2.628	64.824	1.215.450	18.474.840	237.093.780	2.641.902.120	26.088.783.435	231.900.297.200	1.878.392.407.320
73	1	73	2.701	67.525	1.282.975	19.757.815	256.851.595	2.898.753.715	28.987.537.150	260.887.834.350	2.139.280.241.670
74	1	74	2.775	70.300	1.353.275	21.111.090	277.962.685	3.176.716.400	32.164.253.550	293.052.087.900	2.432.332.329.570
75	1	75	2.850	73.150	1.426.425	22.537.515	300.500.200	3.477.216.600	35.641.470.150	328.693.558.050	2.761.025.887.620
76	1	76	2.926	76.076	1.502.501	24.040.016	324.540.216	3.801.756.816	39.443.226.966	368.136.785.016	3.129.162.672.656
77	1	77	3.003	79.079	1.581.580	25.621.596	350.161.812	4.151.918.628	43.595.145.594	411.731.930.610	3.540.894.603.246
78	1	78	3.081	82.160	1.663.740	27.285.336	377.447.148	4.529.365.776	48.124.511.370	459.856.441.980	4.000.751.045.226
79	1	79	3.160	85.320	1.749.060	29.034.396	406.481.544	4.935.847.320	53.060.358.690	512.916.800.670	4.513.667.845.896
80	1	80	3.240	88.560	1.837.620	30.872.016	437.353.560	5.373.200.880	58.433.559.570	571.350.360.240	5.085.018.206.156
81	1	81	3.321	91.881	1.929.301	32.801.517	470.155.077	5.843.355.957	64.276.915.527	635.627.275.767	5.720.445.481.903
82	1	82	3.403	95.284	2.024.785	34.826.302	504.981.379	6.348.337.336	70.625.252.863	706.252.528.630	6.426.898.010.533
83	1	83	3.486	98.770	2.123.555	36.949.857	541.931.236	6.890.268.572	77.515.521.435	783.768.050.065	7.210.666.060.598
84	1	84	3.570	102.340	2.225.895	39.175.752	581.106.988	7.471.375.560	84.986.896.995	868.754.947.060	8.079.421.007.658
85	1	85	3.655	105.995	2.331.890	41.507.642	622.614.630	8.093.990.190	93.080.887.185	961.835.834.245	9.041.256.841.903
86	1	86	3.741	109.736	2.441.626	43.949.268	666.563.898	8.760.554.088	101.841.441.273	1.063.677.275.518	10.104.934.117.421
87	1	87	3.828	113.564	2.555.190	46.504.458	713.068.356	9.473.622.444	111.315.063.717	1.174.992.339.235	11.279.926.456.656
88	1	88	3.916	117.480	2.672.670	49.177.128	762.245.484	10.235.867.928	121.550.931.645	1.296.543.270.880	12.576.469.727.536
89	1	89	4.005	121.485	2.794.155	51.971.283	814.216.767	11.050.084.695	132.601.016.340	1.429.144.287.220	14.005.614.014.756
90	1	90	4.095	125.580	2.919.735	54.891.018	869.107.785	11.919.192.480	144.520.208.820	1.573.664.496.040	15.579.278.510.796
91	1	91	4.186	129.766	3.049.301	57.940.519	927.048.304	12.846.240.784	157.366.449.604	1.731.030.945.644	17.310.309.456.440
92	1	92	4.278	134.044	3.183.545	61.124.064	988.172.368	13.854.413.152	171.200.862.756	1.902.231.808.400	19.212.541.264.840
93	1	93	4.371	138.415	3.321.960	64.446.024	1.052.618.392	14.887.031.544	186.087.894.300	2.088.319.702.700	21.300.860.967.540
94	1	94	4.465	142.880	3.464.840	67.910.864	1.120.529.256	16.007.560.800	202.095.455.100	2.290.415.157.800	23.591.276.125.340
95	1	95	4.560	147.440	3.612.280	71.523.144	1.192.052.400	17.199.613.200	219.295.068.300	2.509.710.226.100	26.100.986.351.440
96	1	96	4.656	152.096	3.764.376	75.287.520	1.267.339.920	18.466.953.120	237.762.021.420	2.747.472.247.520	28.848.458.598.960
97	1	97	4.753	156.849	3.921.225	79.208.745	1.346.548.665	19.813.301.785	257.575.523.205	3.005.047.770.725	31.853.506.369.685
98	1	98	4.851	161.700	4.082.925	83.291.670	1.429.840.335	21.243.342.120	278.818.865.325	3.283.866.636.050	35.137.373.005.735
99	1	99	4.950	166.650	4.249.575	87.541.245	1.517.381.580	22.760.725.700	301.579.589.025	3.585.446.225.075	38.722.819.230.810
100	1	100	5.050	171.700	4.421.275	91.962.520	1.609.344.100	24.370.067.800	325.949.656.825	3.911.395.881.900	42.634.215.112.710
101	1	101	5.151	176.851	4.598.126	96.560.646	1.716.904.746	26.075.972.546	352.025.629.371	4.263.421.511.271	46.897.636.623.981

D.4.1. Significado operativo de las primitivas de transición de un SPRT

La relacion general causa-efecto entre identificadores permite construir una matriz de primitivas de transicion entre simbolos, como se ilustra en la tabla D.17.

Tabla D.17: Matriz general de primitivas de transición

→	S_1	S_2	S_3	S_4	...	S_i
S_1	$P(S_1)$	$C(S_1, S_2)$	$C(S_1, S_3)$	$C(S_1, S_4)$		$P(S_i)$
S_2	$C(S_2, S_1)$	$P(S_2)$	$C(S_2, S_3)$	$C(S_2, S_4)$		
S_3	$C(S_3, S_1)$	$C(S_3, S_2)$	$P(S_3)$	$C(S_3, S_4)$		$C(S_i, S_j)$
S_4	$C(S_4, S_1)$	$C(S_4, S_2)$	$C(S_4, S_3)$	$P(S_4)$		
$\vdots$					$\ddots$	
S_j	$C(S_j, S_1)$		$C(S_i, S_j)$			$P(S_i)$

La columna e_1 hace uso de todas las primitivas disponibles, el sistema 2S2D base tiene transición completa entre simbolos. Las demas columnas $e_2 \dots e_4$ hacen uso de la selección de primitivas, los sistemas producidos no tienen transicion completa entre simbolos. Al final de la tabla se sumaliza la cantidad de reglas dependientes e independientes en cada sistema.

Fuente: Autor.

Representacion matricial del paso entre simbolos de izquierda a derecha. La ausencia de uno o más elementos de la matriz indica transicion incompleta entre simbolos.

Permanencia, $P(S_k) = C(S_k, S_k)$. No realiza cambios sobre la población de S_k .

Cambio, $C(S_i, S_j)$. Altera simultáneamente las poblaciones, reduciendo S_i e incrementado S_j en una unidad de cada tipo.

La diagonal principal de la matriz hace referencia las primitivas de permanencia, el simbolo relacionado en $P(S_k)$ se mantiene fijo o idempotente entre transiciones. Las demás relaciones presentes en la matriz son operaciones de cambio, explicitan el paso de un simbolo a otro.

En un SPRT completo, no existen espacios libres en la matriz de primitivas de transicion. cualquier ausencia denota manipulacion / supresion de primitivas y por ende un sistema de transiciones incompleto, ya que no es posible el cambio en un solo paso de todos los simbolos disponibles en el sistema.

Sin la creacion de simbolos especiales, las operaciones de permanencia y cambio mantienen los significados ya establecidos para todas las posibles combinaciones.

Una matriz de primitivas de transición completa relaciona todos los simbolos entre si, es una matriz cuadrada. por esta razon se suprime el termino de cantidad de primitivas del nombre de un sistema sprt

en un sistema de transiciones completo, $nSkPmD$ con $k = n^2k$ nSn^2PmD ya que es la cantidad de primitivas es una caracteristica basica de completitud, se la omite en la nomenclatura del sistema

es posible pensar en sistemas de n simbolos con mas de n^2 primitivas, pero esto significa repeticion o redundancia de primitivas, algunas de ellas se repiten en el sistema.

la aparicion del termino kP en el nombre implica necsariamente la seleccion de primitivas en el sistema. esto se verá con mas detalle en tal

primitivaUna matriz incompleta relaciona algunos de los simbolos, presenta vacios o no transiciones en su estructura.

los SPRT pueden ser redundantes o no redundantes

Las transiciones entre simbolos sin significado especial estan

Significado de primitivas de transición de un SPRT de símbolos e identificador de vacío

Tabla D.18: Matriz de primitivas de transición e identificador de vacío

→	$\_$	S_1	S_2	S_3	...	S_i		
$\_$	$P(\_)$	$I(S_1)$	$I(S_2)$	$I(S_3)$		$I(S_i)$	$\_ : \_$	$P(\_)$
S_1	$T(S_1)$	$P(S_1)$	$C(S_1, S_2)$	$C(S_1, S_3)$			$S_1 : \_$	$T(S_1)$
S_2	$T(S_2)$	$C(S_2, S_1)$	$P(S_2)$	$C(S_2, S_3)$		$C(S_i, S_j)$	$\_ : S_3$	$I(S_3)$
S_3	$T(S_3)$	$C(S_3, S_1)$	$C(S_3, S_2)$	$P(S_3)$			$S_3 : S_2$	$C(S_3, S_2)$
$\vdots$					$\ddots$			
S_j	$T(S_j)$		$C(S_i, S_j)$			$P(S_i)$		

La columna e_1 hace uso de todas las primitivas disponibles, el sistema 2S2D base tiene transición completa entre simbolos. Las demas columnas $e_2 \dots e_4$ hacen uso de la selección de primitivas, los sistemas producidos no tienen transicion completa entre simbolos. Al final de la tabla se sumaria la cantidad de reglas dependientes e independientes en cada sistema.

Fuente: Autor.

Representacion matricial del paso entre simbolos de izquierda a derecha

Instanciación, $I(S_k) = C(\_, S_k)$. Adiciona un elemento a la poblacion de S_i , reduce la cantidad de espacios vacios en una unidad.

Permanencia, $P(S_k) = C(S_k, S_k)$. No realiza cambios sobre la poblacion de S_i

Terminación, $T(S_k) = C(S_k, \_)$. Elimina un elemento a la poblacion de S_i , aumenta la cantidad de espacios vacios en una unidad.

Cambio, $C(S_i, S_j)$. Altera simultáneamente las poblaciones, reduciendo S_i e incrementado S_j en una unidad de cada tipo.

al adicionar un simbolo especial para hay una operacion de permanencia, las demas son metáforas de cambio.

Cuando en el conjunto de identificadores de la matriz de primitivas de transicion se hace uso de un simbolo para representar el vacío o $\_$ aparecen nuevos significados operativos de primitivas de transición.

Tabla D.19: Selección de primitivas y subconjuntos SRT de un 2S2D

		Primitivas			
		$p_1 = a:a$	$p_2 = a:b$	$p_3 = b:a$	$p_4 = b:b$
		$I(a)$	$R(a,b)$	$R(b,a)$	$I(b)$
		e_1	e_2	e_3	e_4
Regla	Protonotación	$s_1 = p_1, p_2, p_3, p_4$	$s_2 = p_1, p_2, p_3$	$s_3 = p_1, p_3$	$s_4 = p_1$
r_1	$a:a + a:a$ [a,a:a,a]	$2*I(a)$	$I(a)$	$I(a)$	$I(a)$
r_2	$a:a + a:b$ [a,a:a,b]	$I(a)+R(a,b)$	$I(a)+R(a,b)$	-	
r_3	$a:a + b:a$ [a,b:a,a]	$I(a)+R(b,a)$	$I(a)+R(b,a)$	$I(a)+R(b,a)$	
r_4	$a:a + b:b$ [a,b:a,b]	$I(a)+I(b)$			
r_5	$a:b + a:a$ [a,a:b,a]	Igual que r_2	Igual Que r_2		
r_6	$a:b + a:b$ [a,a:b,b]	$2*R(a,b)$	$2*R(a,b)$		
r_7	$a:b + b:a$ [a,b:b,a]	$R(a,b) + R(b,a)$	$R(a,b) + R(b,a)$		
r_8	$a:b + b:b$ [a,b:b,b]	$R(a,b)+I(b)$			
r_9	$b:a + a:a$ [b,a:a,a]	Igual que r_3	Igual Que r_3	Igual que r_3	
r_{10}	$b:a + a:b$ [b,a:a,b]	Igual que r_7	Igual Que r_7	-	
r_{11}	$b:a + b:a$ [b,b:a,a]	$2*R(b,a)$	$2*R(b,a)$	$2*R(b,a)$	
r_{12}	$b:a + b:b$ [b,b:a,b]	$R(b,a)+I(b)$			
r_{13}	$b:b + a:a$ [b,a:b,a]	Igual que r_4			
r_{14}	$b:b + a:b$ [b,a:b,b]	Igual que r_8			
r_{15}	$b:b + b:a$ [b,b:b,a]	Igual que r_{12}			
r_{16}	$b:b + b:b$ [b,b:b,b]	$2*I(b)$			
$\Sigma dependientes$		6	3	1	0
$\Sigma independientes$		10	6	3	1
$\Sigma = P^2$		16	9	4	1

La columna e_1 hace uso de todas las primitivas disponibles, el sistema 2S2D base tiene transición completa entre símbolos. Las demás columnas $e_2 \dots e_4$ hacen uso de la selección de primitivas, los sistemas producidos no tienen transición completa entre símbolos. Al final de la tabla se resume la cantidad de reglas dependientes e independientes en cada sistema.

Fuente: Autor.

El paso de un símbolo a vacío es una operación de cambio en la que el símbolo deja de existir para la simulación, siendo reemplazado por un vacío en el espacio. Esto se denomina Terminación o $T(S_k)$. La aparición de un símbolo a partir de un espacio vacío se denomina Instanciación o $I(S_k)$.

Por medio de las primitivas instanciación, cambio, permanencia y terminación se describen todas s

Para este trabajo el manejo de símbolos generales y la adición de un símbolo de vacío en el espacio constituyen los elementos necesarios mínimos para la representación de transiciones. Al adicionar otro tipo de símbolos especiales es posible establecer nuevos tipos de primitivas de transición.

Tabla D.20: Construcción de SRTs según selección de subconjuntos de primitivas de transición en un 2S2D

Total de conjuntos	Primitivas relacionadas	Subconjunto de primitivas	Descripción fundamental del sistema generado
1	4	$\{p_1, p_2, p_3, p_4\}$	Uso de todas las primitivas disponibles, es posible pasar de un símbolo a otro en un paso
4	3	$\{p_1, p_2, p_3\}$ $\{p_1, p_2, p_4\}$ $\{p_1, p_3, p_4\}$ $\{p_2, p_3, p_4\}$	Las relaciones entre cambio e idempotencia descritas en las primitivas en uso establecen la tendencia general del sistema
6	2	$\{p_1, p_2\}\{p_1, p_3\}$ $\{p_1, p_4\}\{p_2, p_3\}$ $\{p_2, p_4\}\{p_3, p_4\}$	Sistemas de transición binarios
4	1	$\{p_1\}\{p_2\}$ $\{p_3\}\{p_4\}$	Solo hay una transición disponible en cada caso, sistemas estacionarios
1	0	$\{\}$	No se hace uso de las primitivas disponibles
$\Sigma = 2^P = 2^4 = 16$			

Fuente: Autor.

D.4.1.1. Total de reglas independientes en sistemas SPRT

se muestra el total de reglas independentes de sistemas sprt completos de 17 simbolos y hasta 9 dimensiones, como las previstas en el vecindario de homs 1.0

D.4.1.2. Distribución de patrones de transición en SPRT nSmD completos

La tabla D.22 presenta cantidad de patrones de transicion presente en cada SPRT nSmd completo.

La tabla D.23 presenta la relación entre las series A136564 y A144047. La serie A144047 que contiene los valores máximos de la serie A136564. Se muestra la construccion progresiva de la serie A144047 en el interior de la tabla, mediante estos datos se determinan los números encerrados en la tabla D.22.

Características de la distribucion de patrones

calcular directamente la cantidad de patrones de transicion presentes en nSmD es un proceso lento, sobre todo para sistemas con grandes valores de S y D. Conocer mas terminos de la serie A136564 y de la serie A144047 permitiria determinar rapidamente estos valores.

Observe que la distribucion de patrones tiene un limite, que depende de la cantidad de simbolos distintos que pueden ser incluidos simultaneamente en una expresion de transicion. una vez que se

Tabla D.21: Distribución de reglas de transición independiente en SPRT nSmD completos

S	P = S ²	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D
0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1
2	4	1	4	10	20	35	56	84	120	165	220
3	9	1	9	45	165	495	1.287	3.003	6.435	12.870	24.310
4	16	1	16	136	816	3.876	15.504	54.264	170.544	490.314	1.307.504
5	25	1	25	325	2.925	20.475	118.755	593.775	2.629.575	10.518.300	38.567.100
6	36	1	36	666	8.436	82.251	658.008	4.496.388	26.978.328	145.008.513	708.930.508
7	49	1	49	1.225	20.825	270.725	2.869.685	25.827.165	202.927.725	1.420.494.075	8.996.462.475
8	64	1	64	2.080	45.760	766.480	10.424.128	119.877.472	1.198.774.720	10.639.125.640	85.113.005.120
9	81	1	81	3.321	91.881	1.929.501	32.801.517	470.155.077	5.843.355.957	64.276.915.527	635.627.275.767
10	100	1	100	5.050	171.700	4.421.275	91.962.520	1.609.344.100	24.370.067.800	325.949.656.825	3.911.395.881.900
11	121	1	121	7.381	302.621	9.381.251	234.531.275	4.925.156.775	89.356.415.775	1.429.702.652.400	20.492.404.684.400
12	144	1	144	10.440	508.080	18.671.940	552.689.424	13.725.120.696	294.109.729.200	5.551.321.138.650	93.755.645.897.200
13	169	1	169	14.365	818.805	35.208.615	1.218.218.079	35.328.324.291	883.208.107.275	19.430.578.360.050	382.134.707.747.650
14	196	1	196	19.306	1.274.196	63.391.251	2.535.650.040	84.944.276.340	2.451.249.117.240	62.200.446.349.965	1.409.876.783.932.540
15	225	1	225	25.425	1.923.825	109.658.025	5.022.337.545	192.522.939.225	6.353.256.994.425	184.244.452.838.325	4.769.884.167.925.520
16	256	1	256	32.896	2.829.056	183.181.376	9.525.431.552	414.356.272.512	15.508.763.342.592	509.850.594.887.712	14.955.617.450.039.500
17	289	1	289	41.905	4.064.785	296.729.305	17.388.337.273	852.028.526.377	35.906.916.468.745	1.328.555.909.343.560	43.842.345.008.337.600

Fuente: Autor.

alcanza el maximo de simbolos distintos en la expresion, ya no es posible generar nuevos tipos de patrones, su numero se hace constante, sin importar si hay mas simbolos disponibles que combinar.

Al observar el final de cada columna, se nota que hay

una cantidad limite de patrones de transicion.

La cantidad límite de patrones por dimension de evaluación se expresa en la serie A052171

Se desconoce la distribucion de reglas independientes y patrones de transicion para los SPRT nSkPmD incompletos. Esta labor se deja a trabajos futuros.

D.4.1.3. Límites de representación de SPRT: Sistemas 2mSmD

patrones de transición A050927

La tabla puede leerse como la distribucion del numero de patrones de transicion aportados segun el total de simbolos usados en una cantidad dada de dimensiones de evaluacion disponibles.

D.5. SPRT de referencias de símbolos - nRD

Figura D.5: Sistemas de Referencias: Vector de SPRT nRD

$$M_{nRD} = \left(\begin{array}{cccc} 1RD & 2RD & 3RD & \cdots & nRD \end{array} \right)$$

Representación de primitivas de transición de referencias como funciones de Permanencia y Dominación

Fuente: Autor.

D.5.1. Primitivas de transición en sistemas de referencias

permanencia y dominación

Tabla D.22: Distribución de patrones de transición en SPRT nSmD completos

S	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	10D	11D	12D	13D	OEIS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A000004
1	<u>1</u>	1	1	1	1	1	1	1	1	1	1	1	1	1	A000012
2	1	<u>2</u>	6	10	19	28	44	60	85	110	146	182	231	280	A005993
3	1	2	10	31	90	222	520	1090	2180	4090	7356	12660	21105	34020	A050927
4	1	2	<u>11</u>	47	198	713	2423	7388	21003	55433	137944	324659	729022	1567139	A050929
5	1	2	11	51	269	1270	5776	24032	93067						
6	1	2	11	<u>52</u>	291	1596	8838	46384							
7	1	2	11	52	295	1697	10425	63419							
8	1	2	11	52	<u>296</u>	1719	10922	71401							
9	1	2	11	52	296	1723	11033	73834							
10	1	2	11	52	296	<u>1724</u>	11055	74389							
11	1	2	11	52	296	1724	11059	74500							
12	1	2	11	52	296	1724	<u>11060</u>	74522							
13	1	2	11	52	296	1724	11060	74526							
14	1	2	11	52	296	1724	11060	74527							
15	1	2	11	52	296	1724	11060	74527							
16	1	2	11	52	296	1724	11060	74527							
17	1	2	11	52	296	1724	11060	74527							
18	1	2	11	52	296	1724	11060	74527							
19	1	2	11	52	296	1724	11060	74527							
20	1	2	11	52	296	1724	11060	74527							
21	1	2	11	52	296	1724	11060	74527							
22	1	2	11	52	296	1724	11060	74527							
23	1	2	11	52	296	1724	11060	74527							
24	1	2	11	52	296	1724	11060	74527							
25	1	2	11	52	296	1724	11060	74527							
26	1	2	11	52	296	1724	11060	74527							

¿?

Fuente: Autor.

Tabla D.23: Distribución de diferencias entre filas de la tabla

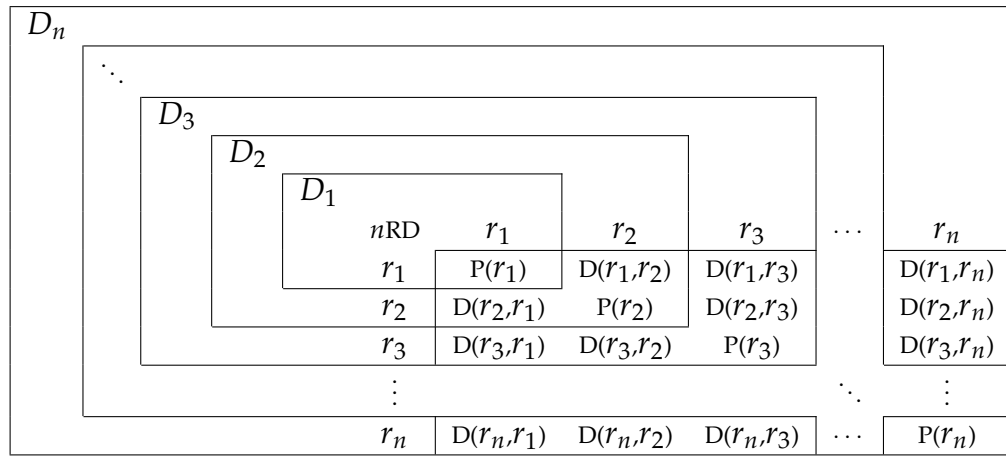
S	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	10D	11D	12D	13D	OEIS
0															
1	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	1	1	1	1	1	
2	0	<u>1</u>	<u>5</u>	<u>9</u>	<u>18</u>	<u>27</u>	<u>43</u>	<u>59</u>	<u>84</u>	109	145	181	230	279	
3	0	0	<u>4</u>	<u>21</u>	<u>71</u>	<u>194</u>	<u>476</u>	<u>1030</u>	<u>2095</u>	3980	7210	12478	20874	33740	
4	0	0	<u>1</u>	<u>16</u>	<u>108</u>	<u>491</u>	<u>1903</u>	<u>6298</u>	<u>18823</u>	51343	130588	311999	707917	1533119	
5	0	0	0	<u>4</u>	<u>71</u>	<u>557</u>	<u>3353</u>	<u>16644</u>	<u>72064</u>						
6	0	0	0	<u>1</u>	<u>22</u>	<u>326</u>	<u>3062</u>	<u>22352</u>							
7	0	0	0	0	<u>4</u>	<u>101</u>	<u>1587</u>	<u>17035</u>							
8	0	0	0	0	<u>1</u>	<u>22</u>	<u>497</u>	<u>7982</u>							
9	0	0	0	0	0	<u>4</u>	<u>111</u>	<u>2433</u>							
10	0	0	0	0	0	<u>1</u>	<u>22</u>	<u>555</u>							
11	0	0	0	0	0	0	<u>4</u>	<u>111</u>				¿?			
12	0	0	0	0	0	0	<u>1</u>	<u>22</u>							
13	0	0	0	0	0	0	0	<u>4</u>							
14	0	0	0	0	0	0	0	<u>1</u>							
15	0	0	0	0	0	0	0	0							
16	0	0	0	0	0	0	0	0							
17	0	0	0	0	0	0	0	0							
18	0	0	0	0	0	0	0	0							
19	0	0	0	0	0	0	0	0							
20	0	0	0	0	0	0	0	0							
21	0	0	0	0	0	0	0	0							
22	0	0	0	0	0	0	0	0							
23	0	0	0	0	0	0	0	0							
24	0	0	0	0	0	0	0	0							
25	0	0	0	0	0	0	0	0							
26	0	0	0	0	0	0	0	0							

A136564

A144047

Fuente: Autor.

Figura D.6: Cobertura de dimensiones de evaluación y primitivas de transición en un SPRT de referencias



Representación de primitivas de transición de referencias como funciones de Permanencia y Dominación

Fuente: Autor.

matriz de primitivas de transición en sistemas de referencias

permanencia de referencia y dominación de referencias

Tabla D.24: Interpretación matricial de primitivas de transición del SPRT 1RD y 2RD

1RD	2RD			
P1	P1	P2	P3	NO
P_1	$P_i + D_{ij}$	I_{ij}	$P_i + P_j$	Válidos
1 r_1	1 $r_1 \ r_2$	2 $r_1 \ r_2$	3 $r_1 \ r_2$	$\times$ $r_1 \ r_2$
r_1 1	r_1 1	r_1 1	r_1 1	r_1 1 1
	r_2 1	r_2 1	r_2 1	r_2
	4 $r_1 \ r_2$			$\times$ $r_1 \ r_2$
	r_1 1			r_1
	r_2 1			r_2 1 1

P_i : Permanencia de referencia i.

D_{ij} : Dominación de referencia i sobre j.

I_{ij} : Intercambio entre referencias i y j, $I_{ij} = I_{ji}$. Notación alternativa de $D_{ij} + D_{ji}$.

R_{ijk} : Rotación, Intercambio entre referencias i, j y k. Notación alternativa de $D_{ij} + D_{jk} + D_{ki}$.

id : Identificador de la regla de transición representada en cada la matriz, según los SPRT 1RD y 2RD reportados en en anexo E.2.5.1 en la página 557 y E.2.5.2 en la página 558.

Fuente: Autor.

Tabla D.25: Significado de Primitivas de transición de un SPRT de referencias

	R_1	R_2	R_3	R_4	...	R_i
R_1	$P(R_1)$	$D(R_1, R_2)$	$D(R_1, R_3)$	$D(R_1, R_4)$		
R_2	$D(R_2, R_1)$	$P(R_2)$	$D(R_2, R_3)$	$D(R_2, R_4)$		
R_3	$D(R_3, R_1)$	$D(R_3, R_2)$	$P(R_3)$	$D(R_3, R_4)$		$D(R_i, R_j)$
R_4	$D(R_4, R_1)$	$D(R_4, R_2)$	$D(R_4, R_3)$	$P(R_4)$		
$\vdots$					$\ddots$	
R_j			$D(R_i, R_j)$			$P(R_i)$

Permanencia= $D(R_i, R_i)$ se mantiene la referencia R_i durante la transición.

Dominación = $D(R_i, R_j)$ cambio de referencia por imposición o dominación, permanece R_j .

Intercambio = $D(R_i, R_j) + D(R_j, R_i)$, cambio simultáneo, bidimensional con conservación del total de referencias.

Rotación = $D(R_i, R_j) + D(R_j, R_k) + D(R_k, R_i)$, cambio simultáneo, multidimensional con conservación del total de referencias.

Fuente: Autor.

D.5.2. Sistemas de notación

D.5.2.1. Notación escrita de representación de reglas de transición

notación de primitivas de transición

notación de reglas de transición

notación de patrones de transición

D.5.2.2. Notación gráfica, Diagrama BOR

D.5.3. Formas de representación de modelos

D.5.3.1. Notación H: Estructura intermedia entre las formas escrita y gráfica de representación de transiciones

Notación H de representación escrita de reglas de transición

Esta notación se presenta como una alternativa a la representación estrictamente gráfica. Sirve de etapa intermedia entre la notación escrita de transiciones a la representación de interacciones de transición en el Diagrama BOR. La notación H facilita el paso entre la forma escrita y gráfica de representación de transiciones.

El nombre de notación H deriva de la forma más común que adopta la representación de una regla de transición, que es cuando se compone de primitivas de permanencia y cambio.

D.6. Sistemas SPRT 2mSmD - Estructuras presentes en reglas de transición de sistemas de referencias

permanencias

intercambios

rotaciones

rotaciones inconclusas, filas o series

dominaciones

creacion y destrucción de referencias

combinación de operaciones en una misma regla de transición

Tabla D.27: Ejemplos de metáforas de transición en términos de reglas de transiciones

SPRT	Metáfora	Patrón y Variantes		Condiciones	E	P	C	A	B
2S2D	Movimiento en MBOR	A: _ + _ : A	[A, _ : _, A]	A, _			2		
3S2D	Reemplazo en movimiento	A: _ + _ : B	[A, _ : _, B]	A, B, _			2	-1	+1
2S2D		a: _ + _ : a	[a, _ : _, a]	a=b			2		
3S2D		a: _ + _ : b	[a, _ : _, b]	a≠b			2	-1	+1
2S2D	Movimiento mediante reubicación específica	1: _ + _ : 1	[1, _ : _, 1]				0		
		a: _ + _ : 1	[a, _ : _, 1]	&1=a			0		
		a: 2 + _ : 1	[a, _ : 2, 1]	&1=a y &2=_			0		
2S2D	Idempotencia de ubicación	1: 1 + 2: 2	[1, 2: 1, 2]				0		
	Intercambio de 1 y 2	1: 2 + 2: 1	[1, 2: 2, 1]				0		
[1..3]S3D	Idempotencia de ubicación	1: 1 + 2: 2 + 3: 3	[1, 2, 3: 1, 2, 3]				0		
	Intercambio de 2 y 3 ante permanencia de 1	1: 1 + 2: 3 + 3: 2	[1, 2, 3: 1, 3, 2]				0		
	Intercambio de 1 y 2 ante permanencia de 3	1: 2 + 2: 1 + 3: 3	[1, 2, 3: 2, 1, 3]				0		
	Rotación izquierda de 3 símbolos	1: 2 + 2: 3 + 3: 1	[1, 2, 3: 2, 3, 1]				0		
	Rotación derecha de 3 símbolos	1: 3 + 2: 1 + 3: 2	[1, 2, 3: 3, 1, 2]				0		
	Intercambio de 1 y 3 ante permanencia de 2	1: 3 + 2: 2 + 3: 1	[1, 2, 3: 3, 2, 1]				0		
[1..2]S2D	Dominación de 1 sobre 2 ante permanencia de 1	1: 1 + 2: 1	[1, 2: 1, 1]				1		
[1..3]S3D	Dominación de 1 sobre 2 y 3	1: 1 + 2: 1 + 3: 1	[1, 2, 3: 1, 1, 1]				2		
	Dominación de 1 sobre 2 ante permanencia de 1 y 3	1: 1 + 2: 1 + 3: 3	[1, 2, 3: 1, 1, 3]				1		
[1..4]S4D	Dominación de 1 sobre 2 y 3 a 4 ante permanencia de 1 y 3	1: 1 + 2: 1 + 3: 3 + 4: 3	[1, 2, 3, 4: 1, 1, 3, 3]				2		
1S1D	Finalización de referencia	1: 0	[1: 0]		+1	-1	1	-&1	
1S1D	Ejemplo de metacambio	1: {a, b, c...n}	[1: {a, b, c...n}]				1	-&1 + ?	

Fuente: Autor.

D.6.1. Patrones de transición de sistemas nSmD completos

la presentación de los patrones de transición de los sistemas nSmD se realiza mediante
se listan a continuación los patrones de transición presentes en los sistemas
se listan estos patrones por ser

2mSmD

2S1D,4S2D,6S3D,8S4D

En las tablas D.28, D.29, D.30, D.31, D.32 y D.33, #S es la cantidad de símbolos presente en el patrón y
F es la frecuencia de patrón en el SPRT al que pertenece. Para cada patrón se muestra en notación de
adición de primitivas y con su representación transparente asociada.

Tabla D.28: Patrones, metáforas y transparencias de 2S1D

Id	Patrón de transición	F	#S	Metáfora	Transparencia
1	A:A	2	1	Idempotencia de símbolo	A
2	A:B	2	2	Reemplazo de símbolo	$[A \rightarrow B]$

Tabla D.29: Patrones, metáforas y transparencias de 4S2D

Id	Patrón de transición	F	#S	Metáfora	Transparencia
1	A:A + A:A	4	1	Doble permanencia uniforme	2A
2	A:A + B:A	24	2	Alteración positiva	$\left[B \xrightarrow{A} A \right]$
3	A:B + A:B	12	2	Doble cambio uniforme	$2[A \rightarrow B]$
4	A:B + C:B	24	3	Unificación par	$\left[\begin{array}{c} A \\ C \end{array} \rightarrow \begin{array}{c} B \\ B \end{array} \right]$
5	A:A + A:B	24	2	Alteración negativa	$\left[A \xrightarrow{A} B \right]$
6	A:A + B:B	12	2	Doble permanencia de símbolos distintos	A, B
7	A:A + B:C	48	3	Alteración indiferente	$\left[B \xrightarrow{A} C \right]$
8	A:B + B:A	12	2	Intercambio	$\left[\begin{array}{c} A \\ B \end{array} \rightarrow \begin{array}{c} B \\ A \end{array} \right]$
9	A:B + B:C	48	3	Fila	$\left[\begin{array}{c} A \\ B \end{array} \rightarrow \begin{array}{c} B \\ C \end{array} \right]$
10	A:B + A:C	24	3	Disociación par	$\left[\begin{array}{c} A \\ A \end{array} \rightarrow \begin{array}{c} B \\ C \end{array} \right]$
11	A:B + C:D	24	4	Doble cambio de símbolos distintos entre si	$\left[\begin{array}{c} A \\ C \end{array} \rightarrow \begin{array}{c} B \\ D \end{array} \right]$

Tabla D.30: Patrones, metáforas y transparencias de 6S3D - Patrones 1..16

Id	Patrón de transición	F	#S	Metáfora	Transparencia
1	A:A + A:A + A:A	6	1	Triple permanencia uniforme	$3A$
2	A:A + A:A + B:A	90	2	Alteración positiva: Doble permanencia y cambio	$\begin{bmatrix} B & \xrightarrow{2A} & A \end{bmatrix}$
3	A:A + B:A + B:A	90	2	Alteración positiva: Permanencia y doble cambio uniforme	$\begin{bmatrix} 2B & \xrightarrow{A} & 2A \end{bmatrix}$
4	A:A + B:A + C:A	360	3	Alteración positiva: Permanencia y unificación	$\begin{bmatrix} B & \xrightarrow{A} & A \\ C & & A \end{bmatrix}$
5	A:B + A:B + A:B	30	2	Triple cambio uniforme	$3[A \rightarrow B]$
6	A:B + A:B + C:B	360	3	Unificación triple con repetición de cambio	$\begin{bmatrix} 2A & \rightarrow & 2B \\ C & & B \end{bmatrix}$
7	A:B + C:B + D:B	360	4	Unificación triple	$\begin{bmatrix} A & & B \\ C & \rightarrow & B \\ D & & B \end{bmatrix}$
8	A:A + A:A + A:B	90	2	Alteración negativa: Doble permanencia y cambio	$\begin{bmatrix} A & \xrightarrow{2A} & B \end{bmatrix}$
9	A:A + A:A + B:B	90	2	Triple permanencia no uniforme	$2A, B$
10	A:A + A:A + B:C	360	3	Alteración indiferente: Doble permanencia y cambio independiente	$\begin{bmatrix} B & \xrightarrow{2A} & C \end{bmatrix}$
11	A:A + A:B + B:A	180	2	Intercambio y permanencia	$\begin{bmatrix} A & \xrightarrow{A} & B \\ B & & A \end{bmatrix}$
12	A:A + A:B + B:B	180	2	Alteración negativa: Doble permanencia no uniforme y cambio	$\begin{bmatrix} A & \xrightarrow{A, B} & B \end{bmatrix}$
13	A:A + B:A + C:B	720	3	Alteración positiva: Permanencia y fila positiva	$\begin{bmatrix} B & \xrightarrow{A} & A \\ C & & B \end{bmatrix}$
14	A:A + A:B + C:A	720	3	Alteración negativa: Permanencia y fila indiferente	$\begin{bmatrix} A & \xrightarrow{A} & B \\ C & & A \end{bmatrix}$
15	A:A + B:A + C:C	720	3	Alteración positiva: Doble permanencia no uniforme y cambio	$\begin{bmatrix} B & \xrightarrow{A, C} & A \end{bmatrix}$
16	A:A + B:A + B:C	720	3	Alteración positiva: permanencia y disociación	$\begin{bmatrix} B & \xrightarrow{A} & A \\ B & & C \end{bmatrix}$

Tabla D.31: Patrones, metáforas y transparencias de 6S3D - Patrones 17..29

Id	Patrón de transición	F	#S	Metáfora	Transparencia
17	A:A + B:A + C:D	2160	4	Alteración positiva: permanencia y doble cambio independiente	$\begin{bmatrix} B & \xrightarrow{A} & A \\ C & & D \end{bmatrix}$
18	A:B + A:B + B:A	90	2	Intercambio y cambio	$\begin{bmatrix} 2A & \rightarrow & 2B \\ B & & A \end{bmatrix}$
19	A:A + A:B + A:B	90	2	Alteración negativa: permanencia y doble cambio uniforme	$[2A \xrightarrow{A} 2B]$
20	A:B + A:B + C:A	360	3	Triple cambio: Doble cambio uniforme y cambio	$\begin{bmatrix} 2A & \rightarrow & 2B \\ C & & A \end{bmatrix}$
21	A:B + B:A + C:A	720	3	Intercambio y cambio	$\begin{bmatrix} A & & B \\ B & \rightarrow & A \\ C & & A \end{bmatrix}$
22	A:A + A:B + C:B	720	3	Alteración negativa: permanencia y unificación	$\begin{bmatrix} A & \xrightarrow{A} & B \\ C & & B \end{bmatrix}$
23	A:B + A:C + B:C	720	3	Fila y cambio	$\begin{bmatrix} A & & B \\ A & \rightarrow & C \\ B & & C \end{bmatrix}$
24	A:B + B:C + D:C	2160	4	Fila y cambio independiente, Unificación y cambio	$\begin{bmatrix} A & & B \\ B & \rightarrow & C \\ D & & C \end{bmatrix}$
25	A:B + A:B + B:C	360	3	Fila y cambio, Doble cambio uniforme y cambio	$\begin{bmatrix} 2A & \rightarrow & 2B \\ B & & C \end{bmatrix}$
26	A:A + B:C + B:C	360	3	Alteración indiferente: permanencia y doble cambio uniforme independiente	$[2B \xrightarrow{A} 2C]$
27	A:B + A:B + A:C	360	3	Disociación con repetición de primitiva	$\begin{bmatrix} 2A & \rightarrow & 2B \\ A & & C \end{bmatrix}$
28	A:B + A:B + C:D	1080	4	Triple cambio: Doble cambio uniforme con cambio independiente	$\begin{bmatrix} 2A & \rightarrow & 2B \\ C & & D \end{bmatrix}$
29	A:B + B:C + D:B	1080	4	Fila con cambio, Unificación con cambio	$\begin{bmatrix} A & & B \\ B & \rightarrow & C \\ D & & B \end{bmatrix}$

Tabla D.32: Patrones, metáforas y transparencias de 6S3D - Patrones 30..42

Id	Patrón de transición	F	#S	Metáfora	Transparencia
30	A:A + B:C + D:C	1080	4	Alteración indiferente: Permanencia y unificación independiente	$\begin{bmatrix} B & \xrightarrow{A} & C \\ D & & C \end{bmatrix}$
31	A:B + A:C + D:B	2160	4	Disociación y cambio independiente, Unificación y cambio	$\begin{bmatrix} A & & B \\ A & \rightarrow & C \\ D & & B \end{bmatrix}$
32	A:B + C:B + D:E	2160	5	Unificación y cambio independiente	$\begin{bmatrix} A & & B \\ C & \rightarrow & B \\ D & & E \end{bmatrix}$
33	A:A + A:B + A:C	360	3	Alteración negativa: permanencia y cambio, Disociación y permanencia	$\begin{bmatrix} A & \xrightarrow{A} & B \\ A & & C \end{bmatrix}$
34	A:A + A:B + B:C	720	3	Alteración negativa: permanencia y cambio, Fila y permanencia	$\begin{bmatrix} A & \xrightarrow{A} & B \\ B & & C \end{bmatrix}$
35	A:A + A:B + C:C	720	3	Alteración negativa: doble permanencia no uniforme y cambio	$\left[A \xrightarrow{A,C} B \right]$
36	A:A + A:B + C:D	2160	4	Alteración negativa: permanencia y doble cambio no uniforme, Alteración negativa con cambio independiente	$\begin{bmatrix} A & \xrightarrow{A} & B \\ C & & D \end{bmatrix}$
37	A:A + B:B + C:C	120	3	Triple permanencia no uniforme	A, B, C
38	A:A + B:B + C:D	1080	4	Alteración indiferente: doble permanencia no uniforme y cambio independiente	$\left[C \xrightarrow{A,B} D \right]$
39	A:A + B:C + C:B	360	3	Alteración indiferente: Permanencia e intercambio independiente	$\begin{bmatrix} B & \xrightarrow{A} & C \\ C & & B \end{bmatrix}$
40	A:A + B:C + C:D	2160	4	Alteración indiferente: Permanencia y fila independiente	$\begin{bmatrix} B & \xrightarrow{A} & C \\ C & & D \end{bmatrix}$
41	A:A + B:C + B:D	1080	4	Alteración indiferente: Permanencia y disociación independiente	$\begin{bmatrix} B & \xrightarrow{A} & C \\ B & & D \end{bmatrix}$
42	A:A + B:C + D:E	2160	5	Alteración indiferente: Permanencia y doble cambio independiente	$\begin{bmatrix} B & \xrightarrow{A} & C \\ D & & E \end{bmatrix}$

Tabla D.33: Patrones, metáforas y transparencias de 6S3D - Patrones 43..52

Id	Patrón de transición	F	#S	Metáfora	Transparencia
43	$A:B + A:C + B:A$	720	3	Disociación y cambio, Intercambio y cambio	$\begin{bmatrix} A & B \\ A \rightarrow C \\ B & A \end{bmatrix}$
44	$A:B + B:A + C:D$	1080	4	Intercambio con cambio independiente	$\begin{bmatrix} A & B \\ B \rightarrow A \\ C & D \end{bmatrix}$
45	$A:B + B:C + C:A$	240	3	Rotación de tres símbolos	$\begin{bmatrix} A & B \\ B \rightarrow C \\ C & A \end{bmatrix}$
46	$A:B + B:C + C:D$	2160	4	Fila de tres puestos	$\begin{bmatrix} A & B \\ B \rightarrow C \\ C & D \end{bmatrix}$
47	$A:B + A:C + D:A$	1080	4	Disociación y cambio, Fila y cambio	$\begin{bmatrix} A & B \\ A \rightarrow C \\ D & A \end{bmatrix}$
48	$A:B + A:C + B:D$	2160	4	Fila y cambio, Disociación y cambio	$\begin{bmatrix} A & B \\ A \rightarrow C \\ B & D \end{bmatrix}$
49	$A:B + B:C + D:E$	4320	5	Fila y cambio independiente	$\begin{bmatrix} A & B \\ B \rightarrow C \\ D & E \end{bmatrix}$
50	$A:B + A:C + A:D$	360	4	Disociación triple	$\begin{bmatrix} A & B \\ A \rightarrow C \\ A & D \end{bmatrix}$
51	$A:B + A:C + D:E$	2160	5	Disociación con cambio independiente	$\begin{bmatrix} A & B \\ A \rightarrow C \\ D & E \end{bmatrix}$
52	$A:B + C:D + E:F$	720	6	Triple cambio no uniforme	$\begin{bmatrix} A & B \\ C \rightarrow D \\ E & F \end{bmatrix}$

Tabla de patrones de transición de SPRT(6S,3D) , Existen en promedio 3.9906121399177 símbolos distintos en cada regla del SPRT

para este ejemplo no se ofrece ejemplo de cada metáfora

determinación de aportes brutos y netos a partir de notación transparente

El símbolo a la izquierda, s_i , s_{c_h} disminuye su población m_h unidades en cada ejecución de la regla de transición

El símbolo a la derecha s_j , s_{e_h} incrementa su población m_h unidades en cada ejecución de la regla de transición

Los términos $s_h, s_{p_1} \dots s_{p_q}$ no presentan cambio poblacional, son permanencias.

En un paso de simulación, cada población relacionada por las primitivas que componen la regla de transición R_k gana o pierde $n_k \cdot m$ unidades.

$$n_k \left[-s_i \xrightarrow{s_h} +s_j \right]$$

$$n_k \left[-m \cdot s_i \xrightarrow{s_h} +m \cdot s_j \right]$$

Conformación de regla de transición en términos de símbolos componentes de primitivas y su caracterización en diferencias:

$$n_k \left[\begin{array}{ccc} -m_1 \cdot s_{c_1} & & +m_1 \cdot s_{e_1} \\ -m_2 \cdot s_{c_2} & s_{p_1} \dots s_{p_q} \xrightarrow{\quad} & +m_1 \cdot s_{e_2} \\ \vdots & & \vdots \\ -m_h \cdot s_{c_h} & & +m_h \cdot s_{e_h} \end{array} \right]$$

población condicionante = permanencia de símbolo

alteración = influencia de uno sobre otro = permanencia + cambio

intercambio = doble cambio simultáneo entre causas y efectos

repetición de operaciones

Establecer metáforas para los patrones de transición es difícil, puesto que la metáfora apunta a una descripción global d

según se observa en la descripción de los distintos patrones de transición encontrados, es posible una deducción automática de la metáfora de cambio representada por cada regla de transición

la cantidad límite de patrones de transición en los SPRT completos es 1,2,11,52 esta serie aparece en OEIS identificada

esta relacionada con

como A050927

con cada uno de las transparencias presentadas se construyen las representaciones transparentes

D.6.2. notación transparente

$$R_k = \begin{cases} \left[\begin{array}{cc} e_1 & s_1 \\ e_2 & s_2 \\ \vdots & \vdots \\ e_n & s_n \end{array} \right] & \text{si } e_i = s_i \\ \left[\begin{array}{cc} e_1 & s_1 \\ e_2 & \xrightarrow{Perm(R_k)} s_2 \\ \vdots & \vdots \\ e_n & s_n \end{array} \right] = \left[\begin{array}{cc} e_1 & s_1 \\ e_2 & \xrightarrow{[e_a, e_b, \dots, e_m]} s_2 \\ \vdots & \vdots \\ e_n & s_n \end{array} \right] & \text{si } e_i \neq s_i \\ \left[\begin{array}{cc} e_1 & s_1 \\ e_2 & s_2 \\ e_3 & \rightarrow s_3 \\ \vdots & \vdots \\ e_n & s_n \end{array} \right] & \text{si } e_i \neq s_i \end{cases}$$

D.6.3. Representación transparente de un modelo BOR

La representación transparente es una forma de notación de reglas de transición cuyo propósito es permitir la interpretación directa e inmediata de las reglas de transición de un modelo BOR. Lo de transparente alude a que en el formato implementado para la notación no se oculta la descripción de la regla de transición, se la muestra tal y como es, es plenamente visible, clara, sin ofuscaciones, se ve inmediatamente, es *transparente* la estructura de la transición descrita por la regla.

Para representar un modelo BOR de forma transparente, hay que conocer explícitamente las reglas de transición del modelo.

Para obtener la representación transparente del modelo solo hay que determinar cual es el patrón al que pertenece cada regla existente y actualizar los valores de las clases - símbolos existentes particulares de cada expresión de transición.

La representación transparente de cada regla aparece cuando se determina cual es el patrón de transición al que pertenece la regla. Se usa la transparencia asociada a cada patrón para construir la notación y se actualizandola según los valores de las clases - símbolos existentes particulares de la expresión de transición. este proyecto propone una lista de transparencias asociadas a los patrones existentes en los sistemas 2S1D, 4S2D y 6D3D. Estas se muestran en las tablas .

Este es un ejemplo de la representación transparente de las reglas de transición de un modelo BOR:

$$\begin{aligned}
Modelo &= \sum n_i R_i \\
&= n_1 R_1 + n_2 R_2 + n_3 R_3 + \dots + n_i R_i \\
&= n_1 \left[\begin{array}{cc} A & B \\ C & \rightarrow D \\ E & F \end{array} \right] + \dots \\
&\quad + n_2 \left[\begin{array}{ccc} B & \xrightarrow{A} & A \\ C & & A \end{array} \right] + \dots \\
&\quad + n_3 \left[B \xrightarrow{2A} A \right] + \dots \\
&\quad + n_4 3A + \dots + n_i 3F
\end{aligned} \tag{D.1}$$

El modelo representado en la ecuación (D.1) supone que contiene reglas que pertenecen como máximo a un SPRT 6S3D. Cuando se construyen este tipo de representaciones, se recomienda la agrupación de términos según su naturaleza de transición. La ecuación (D.1) es un ejemplo conceptual del uso de la notación con agrupación de términos ordenados de mayor a menor complejidad.

Un modelo BOR es la suma de todas las reglas de transición que lo componen.

Se nota claramente por comprensión en la primera línea de la ecuación (D.1).

la segunda línea hace lo mismo por extensión: cada regla de transición tiene un número asociado particular, un coeficiente de ejecución o en un término propuesto por este proyecto, un coeficiente de actividad. Cada regla se realizará n veces durante un paso de simulación dado.

La tercera igualdad ya expresa cada regla de transición en forma transparente. reemplaza cada R por su respectiva expresión transparente.

D.6.4. resumen de estructuras sprt encontradas

nSmD

nRD = nSnD

2mSmD

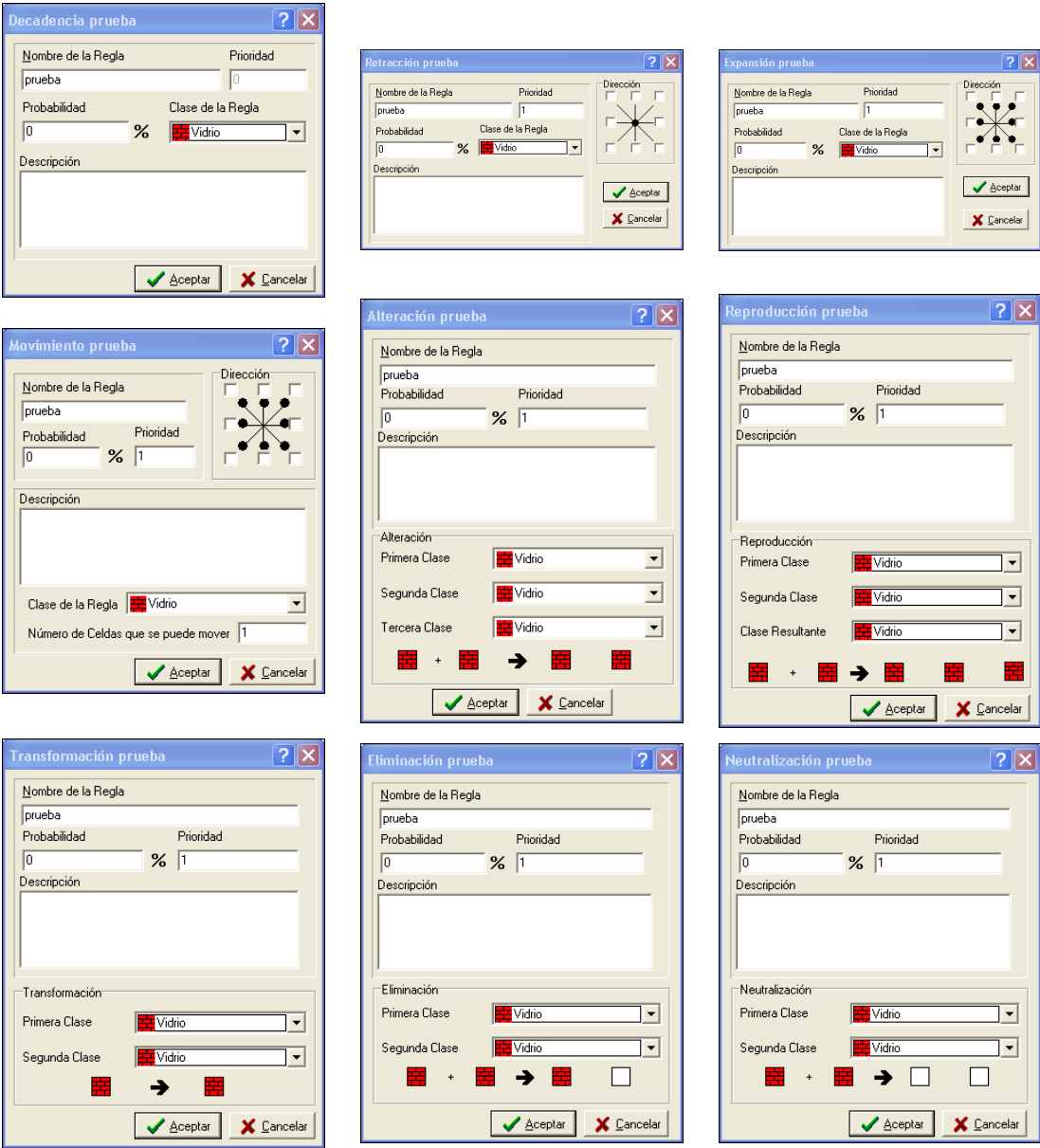
0S0D, 0SnD y nS0D

SPRT completo	nSmD	nSnD = nRd	2mSmD	kPmD		
Reglas de transicion						
reglas de transicion independientes						
patrones de transicion						

D.7. El SPRT de la MBOR y Homos 1.0

descripcion teorica de las reglas de transición de la MBOR

Figura D.7: Reglas de transición e interfaz gráfica de Homos 1.0



En orden de izquierda a derecha y hacia abajo: Cuadros de diálogo de las reglas Decadencia, Retracción, Expansión, Movimiento, Alteración, Reproducción, Transformación, Eliminación y Neutralización.
A partir de estas estructuras se determinaron las distintas variantes de cada tipo de regla de transición existentes en Homos 1.0.

Fuente: Autor.

Tabla D.34: Representación básica de reglas de transición de la MBOR en formalismo de transición de SPRT

SPRT	Nombre de Regla	Notación de	
		Combinación	Permutación
1S1D	<i>Idempotencia</i>	a : a	[a : a]
2S1D	Instanciación	_ : a	[_ : a]
	Decadencia	a : _	[a : _]
	Transformación	a : b	[a : b]
2S2D	Retracción	a : a + a : _	[a , a : a , _]
	Expansión	a : a + _ : a	[a , _ : a , a]

Tabla D.35: Conjunto completo de reglas independientes y patrones de transición presentes en Homos 1.0

Id	Reglas de transición independientes	Patrón base	#S	Id	Patrón de transición	Uso	#S
1	_:_	1	1	1	A:A	2	1
2	_:a	2	2	2	A:B	3	2
3	a:_	2	2	3	A:_ + _:A	1	2
4	a:a	1	1	4	A:_ + A:_	1	2
5	a:b	2	2	5	A:_ + B:_	1	3
6	a:_ + _:a	3	2	6	A:A + A:A	1	1
7	a:a + _:a	8	2	7	A:A + A:B	2	2
8	a:_ + a:_	4	2	8	A:A + B:A	2	2
9	a:_ + b:_	5	3	9	A:A + B:B	1	2
10	a:a + a:_	7	2	10	A:A + B:C	2	3
11	a:a + b:_	10	3	11	A:A + A:A + _:A	1	2
12	a:a + a:a	6	1	12	A:A + A:A + _:B	1	3
13	a:a + a:b	7	2	13	A:A + A:B + _:A	1	3
14	a:a + b:a	8	2	14	A:A + A:B + _:B	1	3
15	a:a + b:b	9	2	15	A:A + A:B + _:C	1	4
16	a:a + b:c	10	3				
17	a:a + a:a + _:a	11	2				
18	a:a + a:a + _:b	12	3				
19	a:a + b:b + _:a	13	3				
20	a:a + b:b + _:b	14	3				
21	a:a + b:b + _:c	15	4				

$$\sum Uso(P_i) = 21$$

$$f(P_i) = Uso(P_i) \cdot \#S(P_i)$$

$$\frac{\sum f(P_i)}{\#R} = \frac{47}{21} \approx 2.238$$

$$\frac{\sum \#S(R_i)}{\#R} = \frac{47}{21} \approx 2.238$$

Los patrones y las reglas de transición posibles en el sistema se deducen directamente de las opciones implementadas en la interfaz grafica de Homos 1.0. Ver la figura D.7 en la página 489.

Fuente: Autor.

Tabla D.36: Primitivas de transición presentes en MBOR y Homos 1.0

Id	Primitiva	Descripción	#S
1	_:_	Idempotencia de vacío	1
2	a:a	Idempotencia de a	1
3	b:b	Idempotencia de b	1
4	_:a	Instanciación de a	2
5	_:b	Instanciación de b	2
6	_:c	Instanciación de c	2
7	a:b	Reemplazo de a por b	2
8	b:a	Reemplazo de b por a	2
9	b:c	Reemplazo de b por c	2
10	a:_	Terminación de a	2
11	b:_	Terminación de b	2
$\frac{\sum \#S(p_i)}{\#p} = \frac{19}{11} \approx 1.7272...$			

La primitiva _:_ es implícita en el sistema, está implementada en el algoritmo de simulación. Las demás son explícitas: pueden deducirse directamente de la estructura de los distintas ventanas de configuración de las reglas de transición presentes en la interfaz de la aplicación. El sistema maneja 4 símbolos distintos = $4^2 = 16$ primitivas posibles, solo usa 11 para describir sus transiciones. $p < s^2$, El SPRT de Homos 1.0 es incompleto y no redundante.

Fuente: Autor.

Tabla D.37: Patrones y variantes de transición presentes en la MBOR de Homos 1.0

SPRT	Metáfora en MBOR	Patrones y variantes de uso Por combinación	Por permutación	Condiciones	Aporte diferencial					
					E	P	C	A	B	C
1S1D	Idempotencia	A:A	[A:A]	A			0			
2S1D	Decendencia	A:_	[A:_]	A, _	+1	-1	1	-1	-1	+1
2S1D	Instanciación	_:A	[_:A]	A, _	-1	+1	1	+1	-1	-1
2S1D	Transformación	A:B	[A:B]	A,B			1	-1	-1	+1
1S1D		a:a	[a:a]	a=b			0			
2S1D		a:b	[a:b]	a≠b			1	-1	-1	+1
2S2D	Expansión	A:A + _:A	[A, _:A,A]	A, _			1			
2S2D	Retracción	A:A + A:_	[A,A:A,_]	A, _			1			
3S2D	Neutralización	A:_ + B:_	[A,B:_,_]	A,B, _	+2	-2	2	-1	-1	+2
2S2D		a:_ + a:_	[a,a:_,_]	a=b	+2	-2	2	-2	-2	+2
3S2D		a:_ + b:_	[a,b:_,_]	a≠b	+2	-2	2	-1	-1	+2
3S2D	Eliminación	A:A + B:_	[A,B:A,_]	A,B, _	+1	-1	1		-1	+1
2S2D		a:a + a:_	[a,a:a,_]	a=b	+1	-1	1	-1	-1	+1
3S2D		a:a + b:_	[a,b:a,_]	a≠b	+1	-1	1	-1	-1	+1
3S2D	Alteración	A:A + B:C	[A,B:A,C]	A,B,C			1	-1	-1	+1
1S2D		a:a + a:a	[a,a:a,a]	a=b=c			0			
2S2D		a:a + a:c	[a,a:a,c]	a=b≠c			1	-1		+1
2S2D		a:a + b:a	[a,b:a,a]	a=c≠b			1	+1	-1	
2S2D		a:a + b:b	[a,b:a,b]	a≠c=b			0			
3S2D		a:a + b:c	[a,b:a,c]	a≠b≠c			1	-1	-1	+-
4S3D	Reproducción	A:A + B:B + _:C	[A,B, _:A,B,C]	A,B,C, _	-1	+1	1		+1	-1
2S3D		a:a + a:a + _:a	[a,a, _:a,a,a]	a=b=c	-1	+1	1	+1		-1
3S3D		a:a + a:a + _:c	[a,a, _:a,a,c]	a=b≠c	-1	+1	1		+1	-1
3S3D		a:a + b:b + _:a	[a,b, _:a,b,a]	a=c≠b	-1	+1	1	+1		-1
3S3D		a:a + b:b + _:b	[a,b, _:a,b,b]	a≠c=b	-1	+1	1		+1	-1
4S3D		a:a + b:b + _:c	[a,b, _:a,b,c]	a≠b≠c	-1	+1	1		+1	-1

Fuente: Autor.

En las dos últimas entradas de la tabla (D.38) se propone el uso del símbolo Ampersand “&” para diferenciar notacionalmente a las referencias de los símbolos y las cantidades numéricas. También se plantea el uso del cero como símbolo para la expresión de terminación de referencia, un análogo funcional al uso del símbolo vacío “_” en los nSmD.

En el ejemplo de regla de metacambio, el símbolo “?” hace alusión al incremento positivo en el resultado indeterminado obtenido de selección del símbolo con el cual remplazar la referencia 1.

Se lista en la tabla D.39 en la página 498 las series OEIS relacionadas con los SPRT en este documento. Estos datos aparecen consignados en el archivo contenido\anexos\scripts\Excel\sprt.xls.

Tabla D.38: Representación de movimiento y ejemplo de acciones mediante referencias

SPRT	Metáfora	Patrón y Variantes	Condiciones	E	P	C	A	B
2S2D	Movimiento en MBOR	A: _ + _ : A	[A, _ : _ , A]	A, _		2		
3S2D	Reemplazo en movimiento	A: _ + _ : B	[A, _ : _ , B]	A, B, _		2	-1	+1
2S2D		a: _ + _ : a	[a, _ : _ , a]	a=b		2		
3S2D		a: _ + _ : b	[a, _ : _ , b]	a≠b		2	-1	+1
2S2D	Movimiento mediante reubicación específica	1: _ + _ : 1	[1, _ : _ , 1]			0		
		a: _ + _ : 1	[a, _ : _ , 1]	&1=a		0		
		a: 2 + _ : 1	[a, _ : 2, 1]	&1=a y &2=_		0		
2S2D	Idempotencia de ubicación	1: 1 + 2: 2	[1, 2: 1, 2]			0		
	Intercambio de 1 y 2	1: 2 + 2: 1	[1, 2: 2, 1]			0		
[1..3]S3D	Idempotencia de ubicación	1: 1 + 2: 2 + 3: 3	[1, 2, 3: 1, 2, 3]			0		
	Intercambio de 2 y 3 ante permanencia de 1	1: 1 + 2: 3 + 3: 2	[1, 2, 3: 1, 3, 2]			0		
	Intercambio de 1 y 2 ante permanencia de 3	1: 2 + 2: 1 + 3: 3	[1, 2, 3: 2, 1, 3]			0		
	Rotación izquierda de 3 símbolos	1: 2 + 2: 3 + 3: 1	[1, 2, 3: 2, 3, 1]			0		
	Rotación derecha de 3 símbolos	1: 3 + 2: 1 + 3: 2	[1, 2, 3: 3, 1, 2]			0		
	Intercambio de 1 y 3 ante permanencia de 2	1: 3 + 2: 2 + 3: 1	[1, 2, 3: 3, 2, 1]			0		
[1..2]S2D	Dominación de 1 sobre 2 ante permanencia de 1	1: 1 + 2: 1	[1, 2: 1, 1]			1		
[1..3]S3D	Dominación de 1 sobre 2 y 3	1: 1 + 2: 1 + 3: 1	[1, 2, 3: 1, 1, 1]			2		
	Dominación de 1 sobre 2 ante permanencia de 1 y 3	1: 1 + 2: 1 + 3: 3	[1, 2, 3: 1, 1, 3]			1		
[1..4]S4D	Dominación de 1 sobre 2 y 3 a 4 ante permanencia de 1 y 3	1: 1 + 2: 1 + 3: 3 + 4: 3	[1, 2, 3, 4: 1, 1, 3, 3]			2		
1S1D	Finalización de referencia	1: 0	[1: 0]	+1	-1	1	-&1	
1S1D	Ejemplo de metacambio	1: {a, b, c...n}	[1: {a, b, c...n}]			1	-&1 +?	

Fuente: Autor.

D.8. extensiones de protonotacion

caracteristicas sugeridas del lenguaje

Tabla D.39: Secuencias OEIS y SPRT

Serie	Descripción	Contexto SPRT
A007318	Triángulo de pascal organizado por filas.	Cantidad de reglas de transición independientes presentes en sistemas SPRT nSkPmD en general.
A000004	Secuencia de ceros	Cantidad de patrones de transición en sistemas SPRT nSmD completos
A000012	Secuencia de unos	
A005993	Número de multígrafos dirigidos con ciclos de 2 nodos con n arcos.	
A050927	Número de multígrafos dirigidos con ciclos de 3 nodos con n arcos.	
A050929	Número de multígrafos dirigidos con ciclos de 4 nodos con n arcos.	
...	Se reconoce inmediatamente que se relacionan las series de multígrafos dirigidos con n arcos de ciclos de 5 nodos o más. Estas series adicionales no aparecen listadas / identificadas en OEIS	
A061711	$k!k^k$ =Cantidad de ciclos requeridos para la construcción de sistemas nRD	
A052171	Número de multígrafos dirigidos con ciclos de n arcos y k vértices.	Cantidad de patrones de transición en sistemas SPRT 2mSmD completos
A136564	Número de multígrafos dirigidos con ciclos de n arcos, k vértices y ningún vértice de grado 0.	Diferencias entre filas en patrones de transición de sistemas SPRT nSmD completos
A144047	Valores límite de la serie A136564.	
A008794	Cuadrados repetidos	Diferencias entre columnas de patrones de transición de sistemas SPRT nSmD completos

Fuente: Autor.

Apéndice E

Útiles de investigación

Con el propósito de probar algunas ideas surgidas durante el desarrollo de la investigación y la necesidad de corroborar el funcionamiento de algunas de las facilidades presentes en el software Homos 1.0, se planteó la necesidad de encontrar una herramienta o un lenguaje de programación con el cual fuese posible:

- Construir programas o scripts implementados como archivos de texto con la mayor flexibilidad, portabilidad e independencia de plataforma disponible.
- Contar con características de programación tales como arrays asociativos, excelente manejo de cadenas, flexibilidad en la conversión entre tipos de datos.
- Facilidades para la presentación de resultados como páginas html/texto.
- Amplio acceso a documentación y soporte técnico.
- Acceso simplificado a la capa OLE/COM/COM+.
- La velocidad de ejecución no es una restricción importante, puesto que inicialmente se desea realizar prototipado, prueba de conceptos y reportes.

De todas las opciones de lenguajes disponibles¹, se eligió finalmente a PHP². Aunque depurar código PHP sin editores o herramientas especializadas no sea algo trivial, cumplió con los requisitos establecidos y facilitó muchas de las labores de construcción de código requeridas por este proyecto.

Todo el software producido en este proyecto de investigación se ejecuta sobre Windows XP 32 bits o superior.

¹Delphi, Visual basic, C, Java, Perl, Lua, etc ...

²<http://www.php.net>

En este anexo se describen en orden de integración al documento:

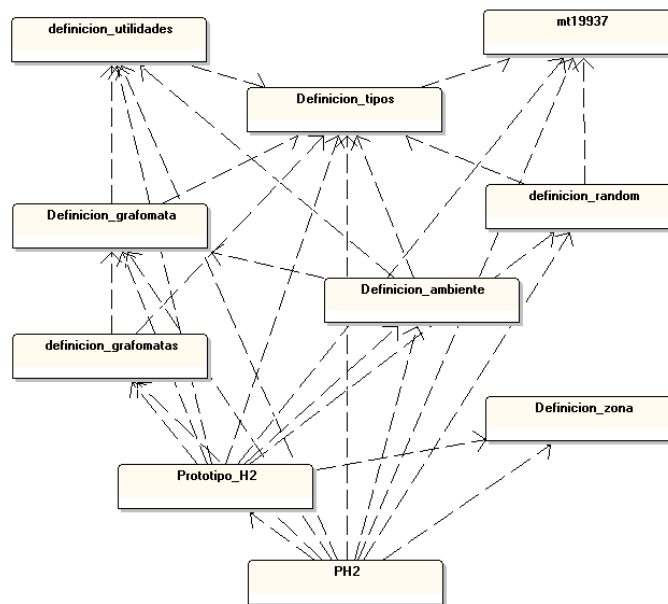
- AegesPHP: Un script para la edición y ejecución de código PHP.
- GSPRT: Un script para la construcción reglas de transición mediante SPRT³.
- Cinco modelos-script de ejemplo de uso de la interfaz OLE/COM⁴ de Homos 1.0.

Para cada tipo de script PHP presentado se indica su propósito, forma de uso, resultados obtenidos y el código fuente.

Adicionalmente se incluyen:

- Un script desarrollado en METAPOST que visualiza el funcionamiento interno del algoritmo de teselado de superficies usando grafómatas presentado en el anexo C.6.2 en la página 412.
- Dos hojas de cálculo: `sprt.xls` y `mbor.xls`, en las cuales se muestran distintas distribuciones numéricas asociadas con los SPRT y una prueba conceptual de la caracterización en diferencias de los modelos BOR propuesta en este proyecto.

Figura E.1: Utiles de investigación



Las flechas indican las dependencias inmediatas entre módulos.

Fuente: Autor.

³Sistemas de Producción de Reglas de Transición, Anexo D en la página 438.

⁴Como una composición de resultados entre el análisis de la interfaz OLE y la teoría de los SPRT.

Recursos requeridos para la ejecución de los scripts de investigación

Los elementos software requeridos para la ejecución exitosa de los scripts en PHP incluidos en este documento son:

- Una instalación del motor de lenguaje PHP. En el desarrollo de este proyecto se usó el servidor web liviano Uniform Server⁵, del cual se probaron versiones desde la 3.5 hasta la 5.6. Esta última edición implementa Apache/2.2.15 (Win32) + mod_ssl/2.2.15 + OpenSSL/0.9.8l + PHP/5.3.2 DAV/2.
- Para los scripts de prueba de funcionamiento de la interfaz OLE se requiere la instalación de la versión completa de Homos 1.0: Ejecutable principal con interfaz OLE implementada y activada en el registro de Windows. Los detalles de instalación se encuentran en la sección A.1.4.2 en la página 101.
- Un browser para visualizar los resultados. Se probaron Internet explorer 6 y 7, Google Chrome, Firefox 3.X y Opera desde la versión 7 en adelante. Opera fué el browser principal de prueba y desarrollo.
- Un editor de texto para la realización de cambios sobre el código. Se usó principalmente el EditPad lite 6.6.4⁶ y el CodePad 4.1⁷.
- Los scripts asumen que la instalación de Homos 1.0 y sus recursos se encuentran en la carpeta: C:\Program Files\UIS\Homos\Recursos\

En cada programa-script están ausentes los delimitadores de inicio y fin de script PHP `<? ... ?>`, estos no se requieren ya que son manejados internamente por el script AegesPHP.

Para la ejecución de los scripts en METAPOST se requiere:

- Un editor de texto para creación del script.
- La instalación de una distribución $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ ⁸ para Microsoft Windows XP que incluya los fuentes y binarios requeridos para usar METAPOST. Para ese propósito se usó a MiK $\text{T}_{\text{E}}\text{X}$ 2.8⁹, software que da soporte a L γ X 1.6.10¹⁰, un procesador de documentos de código abierto con el cual se construyó este documento de proyecto.
- Un archivo *.bat que facilite la compilación de los scripts en METAPOST, con el siguiente contenido¹¹:

⁵<http://www.uniformserver.com>

⁶<http://www.editpadlite.com>

⁷<http://shicola.wz.cz/codepad/download.php>

⁸Este software existe para otras plataformas.

⁹<http://www.miktex.org/>

¹⁰<http://www.lyx.org/>

¹¹Tip for push and pop cd dir. <http://blogs.msdn.com/b/oldnewthing/archive/2005/01/28/362565.aspx>

Archivo auxiliar do.Bat

```
@echo OFF
set OLDDIR=%CD%
cd "C:\MiKTeX_2.8\miktex\bin"
mpost "%1" -output-directory="%OLDDIR%" --halt-on-error
chdir /d "%OLDDIR%" &rem restore current directory
```

El archivo *.bat tiene como parámetro %1 el nombre del script METAPOST a procesar por los binarios presentes en la ubicación de la instalación de MiKTeX dada.

Para visualizar los archivos de *.xls se requiere de una instalación de Office 97 o superior. Los archivos *.xlsx requieren Office 2007 o superior.

El código fuente y los resultados obtenidos de la ejecución de los scripts reportados en este documento se encuentran en la carpeta contenido\anexos\scripts\

E.1. AegesPHP: Abrir, Editar, Guardar y Ejecutar Scripts PHP

El script AegesPHP se creó como un útil para el prototipado rápido de pequeñas aplicaciones, las requeridas para la exploración de ideas y la afirmación de ciertos conceptos encontrados durante el desarrollo del proyecto. El nombre de la aplicación es una sigla que corresponde con la acción de sus funciones principales. Los scripts *.php se editan en el AegesPHP porque reúne el proceso de editar, ejecutar y obtener resultados texto del script todo en un solo sitio.

E.1.1. Descripción de la interfaz

El diseño de la interfaz es minimalista: sin gráficos, solo texto y elementos de formulario. Se presenta en la figura E.3 en la página 509. El script hace uso práctico de las siguientes facilidades existentes en el browser anfitrión:

- Las operaciones hacer y deshacer, salvado de texto, copiar, pegar, modificar, eliminar y demás opciones de edición disponibles en el browser.
- Uso de los botones de historial, avance y retroceso de página presentes en el browser como una forma de revisión de código / ejecuciones anteriores del sub-script en edición.

Los botones / controles implementados en la interfaz son los siguientes:

- Selección de documento en carpeta de trabajo. En Opera este aparece como [Elegir...].
- Ejecución del script, [Ejecutar]. Este botón se repite en la interfaz, la idea es facilitar el acceso global a esta función.
- El botón [Salvar] realiza la operación de almacenamiento. Se ofrecen tres opciones básicas en la lista de selección de función [Salvar documento..]:
 - Acumular la versión, crear un nuevo documento al salvar. Esta es la opción por defecto.
 - Acumular una versión del documento al ejecutar el sub-script.
 - Sobre escribir el documento inicial.
- Los botones de ubicación en el documento permiten ir:
 - Al cuadro de edición de texto [Editar script].
 - Al inicio [Resultado[|<-]] o al final [Resultado[|->]] de los resultados producidos por la ejecución del sub-script.
- El botón [Nuevo] produce una nueva instancia de la interfaz AegesPHP.
- Los botones para dar [Reset] y [Borrar] el contenido del cuadro de texto.
- El botón [cerrar[X]] termina la instancia actual de la interfaz.

- Los botones de opciones de visualización de código fuente se encuentran en la parte inferior media del cuadro de texto, permiten visualizar el contenido del sub-script en una ventana separada en tres formas:
 - [Ver Script]: Visualiza el código del script como aparece en el editor.
 - [Script SC]: Presenta el sub-script Sin Comentarios.
 - [Script 123]: Permite ver el sub-script con sus líneas numeradas, es especialmente útil cuando se depura código.

Cuando se ejecuta el sub-script se adiciona al bloque de visualización el botón [Ver resultado], esta función presenta en una ventana el contenido de los resultados producidos.

- El botón [Abrir] permite cargar los documentos relacionados en la lista de documentos recientes... Esta opción hace uso de los documentos creados mediante la acumulación de versiones.

La organización funcional de los botones en la interfaz es:

- En la parte superior del cuadro de edición de texto se encuentra:
 - A la izquierda: las opciones de carga, ejecución, almacenamiento e instanciación de nuevo editor del script.
 - A la derecha: las opciones de reinicio, borrado y cierre del script.
- En la parte inferior del cuadro de edición de texto se presentan:
 - A la izquierda: Opciones de ubicación en la interfaz, ir al editor, ir al inicio o fin de los resultados, ejecutar el script.
 - Al medio: Aparecen las opciones de visualización de código del sub-script.
 - A la derecha: la opción de recuperación de versiones previamente guardadas del sub-script en edición. Al realizar una ejecución del script, aparece en la parte baja del espacio una copia de los botones de opción de ubicación en la interfaz.
- En la parte inferior de la interfaz, el script informa el tiempo de ejecución en segundos del sub-script en edición y el del propio script AegesPHP.

Por defecto todos los botones tienen el color estándar proporcionado por el browser, se encontró conveniente resaltar los botones [Ejecutar] en gris, el botón [Salvar] en verde oliva y el botón [Cerrar] en rojo. La elección de colores busca realzar la importancia particular de esas funciones en la interfaz.

E.1.2. Contenido por defecto del cuadro de texto del editor

Por defecto el cuadro de texto del editor inicia con un sub-script configurado con una cabecera automática:

Cabecera automática de scripts en AegesPHP

```
1 //descripción del script
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = "c:\_phplets"; //directorio por defecto
4 $phplet_slf_ = "Aeges_2011_02_10-1297351668.php"; //nombre archivo
5 //$phplet_lib_ = ""; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 [... aqui va el script ...]
```

En la cabecera se establecen:

`set_time_limit(0)`: Configura el tiempo máximo de ejecución del script. Inicialmente se establece que no hay tal límite de tiempo.

`$phplet_dir_`: Determina la ubicación por defecto del directorio de trabajo.

`$phplet_slf_`: Establece el nombre del sub-script. El formato del nombre automático es:

Aeges_ + fecha en formato (año_mes_día) + "_" + tiempo en segundos + .php.

`$phplet_lib_`: Determina la ubicación de las librerías requeridas por el script, esta opción aparece desactivada por defecto. Los archivos de librerías relacionados se cargan siempre al inicio de la ejecución del sub-script.

`$phplet_rows_`: Fija la cantidad de filas a visualizar en el cuadro de edición de texto. Por defecto se establece de 30 líneas de alto.

La información automática proporcionada es fácilmente ajustable a los propósitos deseados del sub-script.

E.1.3. Notas de implementación

- En algunos browsers aparecerá un campo texto seguido de un botón etiquetado como [Seleccionar] | [Seleccionar archivo] | [Examinar...] desde el cual se presentará un cuadro de diálogo abrir. El documento seleccionado es cargado en el campo de edición de texto.
- El editor contiene elementos mínimos de estilo CSS, para evitar problemas de codificación entre browsers. Un ejemplo notable de esto es la identificación del color gris con etiquetas gray | grey | #808080. El código de color RGB evade el problema.
- Se implementaron *tooltips* o pequeños mensajes que aclaran la función de un elemento señalado con el mouse dentro de la interfaz del AegesPHP, pero algunos browsers pueden omitir la presentación de esa información.

- Por defecto se ofrece una interfaz de alto contraste: fondo negro y letras blancas, se implementó una versión blanca o amistosa con la impresora que se activa adicionando `?bn` al final del texto usado para la carga del script `AegesPHP` en la barra de direcciones del browser.
- El directorio por defecto se establece internamente en el archivo `aegesphp.php` en la variable `$dir_phplets`. Durante el desarrollo del proyecto los scripts se almacenaron en `c:\_phplets`.
- En las vistas de código del sub-script, el script `AegesPHP` hace uso del *tag* `<Xmp>`: este hace que el browser presente una secuencia literal de caracteres en un tipo letra de ancho fijo, ideal para presentar el texto tal y como está escrito. Aunque este identificador ha sido marcado como obsoleto^{12 13}, resulta ser la opción más conveniente en cuanto a visualización de código fuente.
- Las opciones de almacenamiento del sub-script configuran un CVS¹⁴ mínimo. Si el documento no existe se crea uno con el nombre establecido por `$phplet_slf_`. El editor `AegesPHP` no hace evaluación de si el documento a salvar existe previamente o no en el directorio de almacenamiento. La operación de acumular una versión del código consiste en adicionar al final del nombre del sub-script un símbolo `#` seguido de un número consecutivo que informa del número versión inmediata, salvando el archivo con el nuevo nombre. Los nombres de estas versiones son los que aparecen en la lista de documentos recientes, el registro se lleva en un archivo identificado como `@nombre_script.txt` que se almacena en el directorio raíz donde se encuentra el `aegesphp.php`. El editor no ofrece ninguna opción de adicional de administración de estos archivos, esta operación debe hacerse manualmente.
- Las vistas de código fuente están vigentes solo mientras la ventana que reporta los datos esta activa o tiene el enfoque, al perderlo se cierran automáticamente. Esto es así para evitar el apilamiento de ventanas en el browser, también porque luego de satisfacer la razón por la cual se visualiza el código, los cambios serán inmediatos. Así que en general no se requiere que la vista de código fuente exista por largos periodos de tiempo. El tiempo de duración antes del cierre de las vistas de código es en milisegundos y se establece en `$popup_time` en `aegesphp.php`
- Al ejecutar un script y en caso de que exista uno o más errores dentro del código, PHP detiene la ejecución y presenta los mensajes de error correspondientes. La corrección consiste en la edición de la instrucción o sentencia PHP en el script que causa los mensajes de error. Este un proceso dinámico altamente *adictivo*, es la razón de la existencia de las distintas opciones de vistas de código incluidas en el editor.
- En la figura E.2 en la página 508 se hace un resumen del proceso de instalación y uso del editor `AegesPHP`.

¹²Ver <http://www.w3.org/TR/REC-html32#xmp>

¹³Ver http://www.w3.org/MarkUp/html-spec/html-spec_5.html#SEC5.5.2.1

¹⁴Concurrent Versions System, ver <http://www.cvshome.org/> y <http://www.nongnu.org/cvs/>

E.1.4. Flujo de trabajo con el editor AegesPHP

Los pasos de instalación del script se muestran en la figura E.2 en la página siguiente. El método de trabajo con el editor AegesPHP es el siguiente:

1. Etapa de instanciación: Carga del script en el browser, inyección del sub-script en el editor (cargado o pegado directamente), configuración de los valores iniciales.
2. Etapa de edición: Modificación del sub-script contenido en el cuadro texto y uso de los distintos botones de control. Las funciones principales son:
 - La ejecución del código: Inyectar el código del sub-script en AegesPHP, realizar procesos estipulados en el sub-script, inyectar en el cuerpo html de AegesPHP el texto producido como resultados de la ejecución del sub-script.
 - Borrar todo el texto en edición, Instanciar un nuevo editor y presentar en una nueva ventana el código en edición: sin cambios, sin comentarios o con sus líneas numeradas.
3. Terminación de ejecución al cerrar la instancia del editor.

Se deduce del flujo de trabajo que la fase de edición predomina sobre las otras, la operación interna más común será la edición de código fuente y la ejecución del sub-script.

E.1.5. Sobre el desarrollo del editor

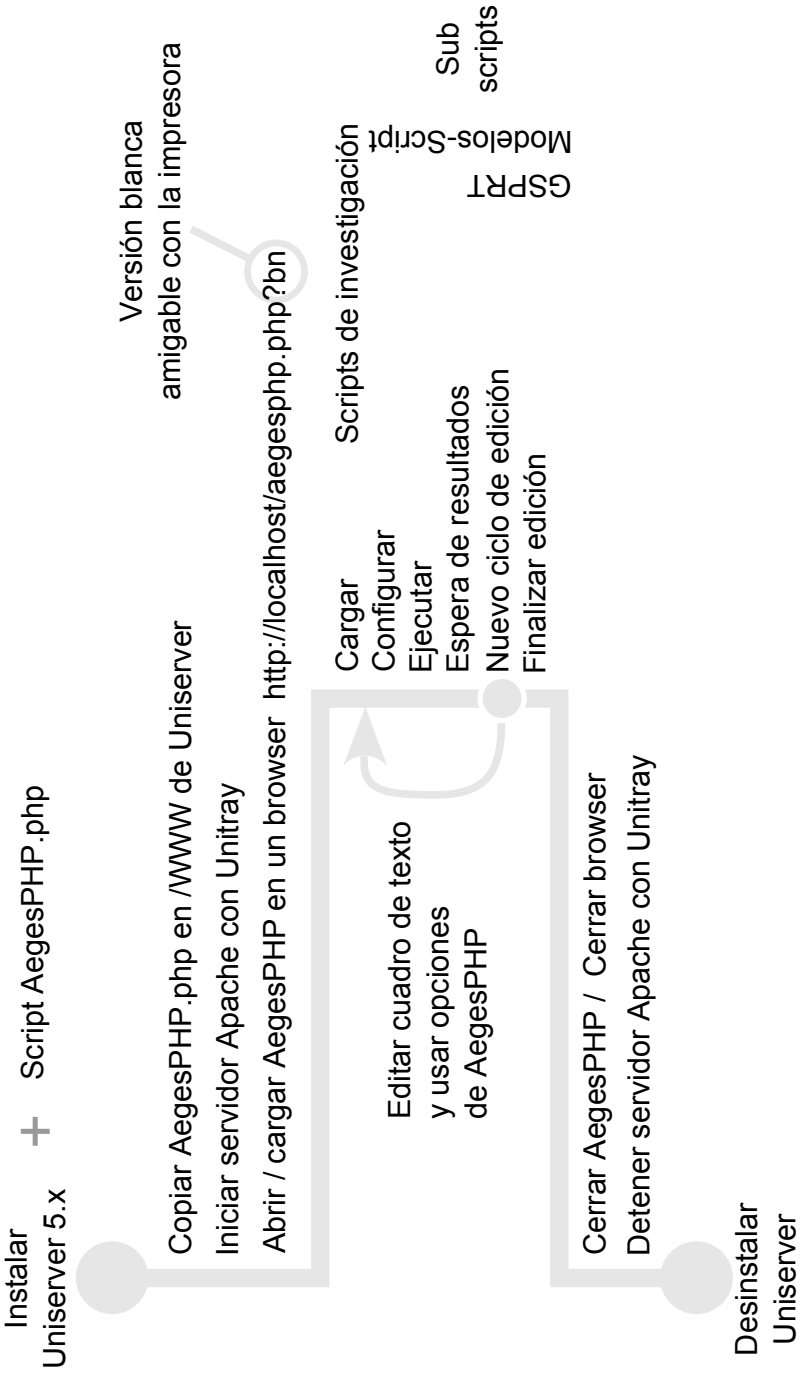
Inicialmente se buscó la forma de ejecutar código PHP directamente desde el cuadro de texto de un formulario en un browser. El script AegesPHP inserta dentro de sí el código del sub-script en edición mediante la función `eval` de PHP. Esta función es la que hace posible la ejecución, ya que procesa una cadena de texto como código PHP. Una vez alcanzada la funcionalidad básica, se propició la creación de los demás elementos de la interfaz, que en su orden fueron:

- Se establecieron los botones de ejecución de código, las opciones de localización rápida dentro del editor y la integración de los resultados obtenidos en la misma ventana de trabajo.
- Se añadieron las opciones de visualización de código y la creación de una nueva instancia del editor.
- Se implementaron los botones de borrado y cerrado del script, seguidos de los botones de apertura de archivo y la realización práctica de las distintas opciones de almacenamiento.

A pesar de los distintos refinamientos realizados, el script AegesPHP aún se encuentra en etapas iniciales de desarrollo. Nuevas opciones y mejoras a las funciones implementadas son posibles. En esta versión del script se usó código PHP + JavaScript + Html + CSS.

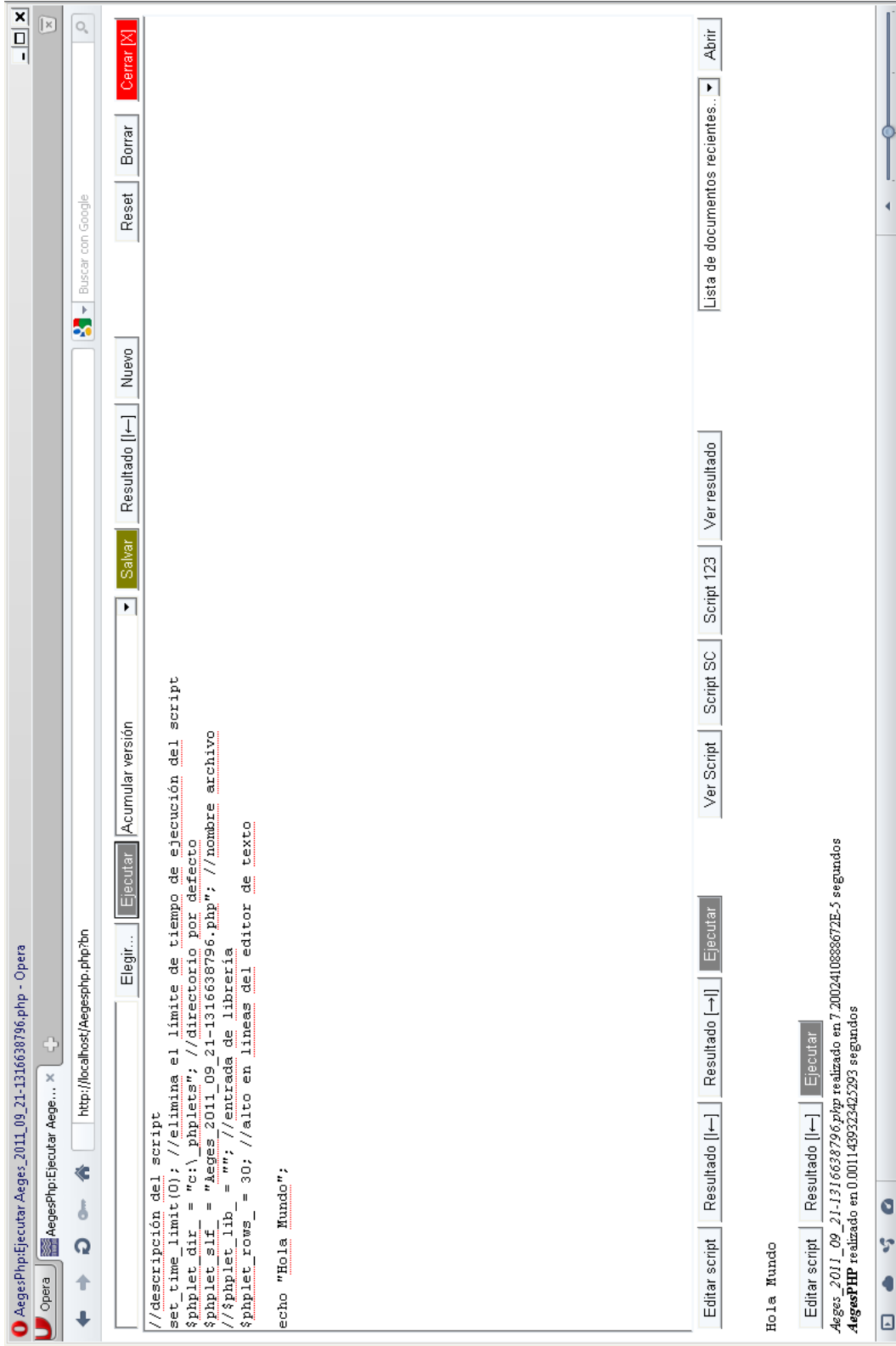
Figura E.2: Guía de instalación y uso de AegesPHP

Útil de desarrollo Script AegesPHP



AegesPHP es un editor de scripts *.PHP de propósito general.
Fuente: Autor.

Figura E.3: Interfaz gráfica del script aegeshp.php



Versión blanca de la interfaz gráfica del script Aegeshp.php. Este formato se activa al agregar `?bn` en el llamado del script. `http://localhost/Aegeshp.php?bn`
 El color normal de esta interfaz es fondo negro, letras blancas. Se muestra la versión blanca ya que es amistosa con la impresora.
 Fuente: Autor.

Script E.10: Editor AegesPHP

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
3 <!-- AegesPHP por Angel Maria Valdes Peña Email:anmaval@hotmail.com anmaval@gmail.com -->
4 <?php
5 $ta_ = microtime(true); //tiempo de proceso del editor AegesPHP
6 ob_start();
7 //*****
8 //configuracion de variables internas del editor
9 $aeges_name="<strong><em>Aeges</em>PHP</strong>";
10 $dir_phplets = "c:\_phplets"; //editar el directorio por defecto aqui
11
12 $phplet_popup_time = 300; //tiempo de permanencia en milisegundos
13 $phplet_file = $_SERVER["SCRIPT_NAME"];
14 $phplet_break_name = Explode('/', $phplet_file);
15 $phplet_self_name = ucfirst($phplet_break_name[count($phplet_break_name) - 1]);
16
17 $ini_rows_ = 30;
18 $autonombre = "Aeges_".date('Y_m_d')."_.time().".php"; //nombre automatico de documento
19
20 $phplet_ini_ = //cabecera descriptiva de cada phplet, presentada solo en documentos nuevos
21 " //descripción del script\n".
22 "set_time_limit(0); //elimina el límite de tiempo de ejecución del script\n".
23 "\$phplet_dir_ = \"\$dir_phplets\"; //directorío por defecto\n".
24 "\$phplet_slf_ = \"\$autonombre\"; //nombre archivo\n".
25 "\$phplet_lib_ = \"\"; //entrada de librería\n".
26 "\$phplet_rows_ = \$ini_rows_; //alto en lineas del editor de texto\n";
27
28 //*****
29 //lectura de variables del formulario editor
30
31 if(isset($_POST['archivo_editar'])) $ae_ = $_POST['archivo_editar']; else $ae_ = '';
32 if(isset($_POST['archivo_actual'])) $aa_ = $_POST['archivo_actual']; else $aa_ = '';
33
34 //centraliza la recepcion de ordenes del formulario
35 if(isset($_POST['boton'])) $accion = $_POST['boton']; else $accion = '';
36 if(isset($_POST['extra'])) $extra = $_POST['extra']; else $extra = '';
37
38 //uso de entrada de comando alternativo, para permitir el uso del boton ejecutar fuera de
39 //formulario
40 if ($extra<>'') $accion = $extra;
41
42 if(isset($_POST['salvar'])) $salvar_modos=$_POST['salvar']; else $salvar_modos = '';
43 if(isset($_POST['cvs'])) $cvs = $_POST['cvs']; else $cvs = '';
44
45 //preprocesamiento necesario para captura de direccion del script
46 if(isset($_POST['aeges_script'])) $txt_ = $_POST['aeges_script']; else $txt_ = '';
47
48 if ($ae_==""){ //si no se ha asignado ningun documento para abrir
49     if($cvs!=""){ //hay un cvs por abrir?
50         $ae_=$cvs; //reseñar cvs
51     }
52 }
53
54 //operaciones antes de presentar el documento
55 if($ae_!="") { //si no hay un nombre de archivo entonces no se hace nada
56     if($ae_!=$aa_){ //solo se abre un archivo si el nombre elegido es diferente al abierto
57         actualmente
58         $aa_ = $ae_;
59         if($cvs!=""){
60             $filename = $ae_; //esta es la carpeta de trabajo por defecto, direccion
61             //completa si existe cvs
62         } else {
63             $break_ae = Explode('\\', $ae_);
64             $ae_ = $break_ae[count($break_ae) - 1]; //resuelve un problema de
65             //compatibilidad con el internet explorer, que envia la localizacion
66             //completa de un documento
67             $filename = "$dir_phplets/$ae_"; //esta es la carpeta de trabajo por
68             //defecto
69         }
70         $fd = fopen($filename, "r") or die("Fallo al abrir la libreria $filename<br />\n");
71         $txt_ = fread($fd, filesize ($filename)) or die("Fallo al leer la libreria
72             $filename<br />\n");
73         fclose($fd) or die("Fallo al cerrar la libreria $filename<br />\n");
74         $accion="Abrir";
75     } else $ae_ = ""; //ignora la re-apertura del archivo

```

```

69 }
70
71 //conservar el codigo mientras la accion no sea un documento nuevo, abrir un documento existente o
  borrar el documento actual
72 if(isset($_POST['aeges_script']) && $ae_==" && $accion!='Borrar') { $txt_ = $txt_; }
73
74 switch($accion){
75 case 'Borrar': //borrado = crear codigo para un nuevo script
76 case '': $txt_=$phplet_ini_; break; //no hay accion, se asume un nuevo script
77 case 'Ejecutar': //inclusion manual de las librerias especificadas en el programa antes de la
  evaluacion del codigo unicamente al ejecutar el phplet
78     preg_match_all("(?!\\|\\|)\\s*\\[\\$\\]phplet_lib_\\s*=\\s*(\\|'\\|")(\\.*)\\|1;/isU",$txt_,$phplet_tz_)
  ;
79     while (list($id,$phplib_) = each ($phplet_tz_[2])) {
80         $fnx=$phplib_;
81         $fdx = fopen($fnx, "r") or die("Fallo al abrir $fnx<br />\n");
82         $txt_libphp_ = fread($fdx, filesize ($fnx)) or die("Fallo al leer $fnx<br />\n");
83         fclose($fdx) or die("Fallo al cerrar $fnx<br />\n");
84         eval($txt_libphp_);
85     };
86     break;
87 };
88
89 //lee el numero de lineas de la ventana de edicion
90 preg_match("(?!\\|\\|)\\s*\\[\\$\\]phplet_rows_\\s*=\\s*(\\d+)\\s*/;isU",$txt_,$phplet_rows_);
91 if(!isset($phplet_rows_[1])) $phplet_rows_[1]=$ini_rows_; //si no se ha definido se establece el
  valor por defecto
92
93 //lectura del nombre del phplet
94 preg_match("(?!\\|\\|)\\s*\\[\\$\\]phplet_slf_\\s*=\\s*(\\|'\\|")((\\.|\\|)\\.\\.\\.?)\\|1;/isU",$txt_,
  $phplet_aa_);
95 if(!isset($phplet_aa_[2])){
96     echo "No se ha especificado un nombre para el phplet<br />\n hacerlo con \\$phplet_slf_ = \"
  nombrearchivo.php\";<br />";
97     exit;
98 }
99
100 //lectura del historial particular de cada documento
101 $hist_file="@.$phplet_aa_[3].".txt"; //nombre de archivo de historial cvs
102 $hist_doc[0]=array();
103 $sel_hist="";
104
105 if(file_exists($hist_file)) {
106     $fd_ = @fopen($hist_file, "r") or die("Fallo al abrir el documento $hist_file<br />\n");
107     $hist_txt_ = @fread($fd_,filesize($hist_file)) or die("Fallo al leer el documento
  $hist_file<br />\n");
108     fclose($fd_) or die("Fallo al cerrar el documento $hist_file<br />\n");
109     preg_match_all('/.*\\S/', $hist_txt_,$hist_doc); //traslacion de lineas de texto a elementos
  de un vector
110
111     while (list ($id_h, $el) = each ($hist_doc[0])) { $sel_hist.="\\t\\t\\t<option>$el</option>\\
  n"; } //leer cada linea del historial
112 }
113
114 //mensajes de interfaz aeges
115 $tooltip['ejecutar'] = 'title="Ejecutar el script actual"';
116 $tooltip['resultados'] = 'title="Ir a seccion de resultados"';
117 $tooltip['ver'] = 'title="Presentar el script actual"';
118 $tooltip['salvar'] = 'title="Salvar el script segun las condiciones establecidas"';
119 $tooltip['nuevo'] = 'title="Crear un nuevo documento en ventana adicional"';
120 $tooltip['reset'] = 'title="Volver el formulario al estado inicial"';
121 $tooltip['borrar'] = 'title="Borra el script actual"';
122 $tooltip['cvs'] = 'title="Abrir documento seleccionado"';
123 $tooltip['cerrar'] = 'title="Cerrar AegesPHP"';
124 $tooltip['editor'] = 'title="Volver al editor AegesPHP"';
125 $tooltip['ver_script'] = 'title="Presentar el código del Script"';
126 $tooltip['script_sc'] = 'title="Presentar el script sin comentarios"';
127 $tooltip['script_123'] = 'title="Presentar el script con numeracion de lineas"';
128 $tooltip['ver_resultados'] = 'title="Presentar los resultados de ejecución"';
129 $tooltip['fin_resultados'] = 'title="Ir al final de los resultados"';
130
131 ?><head>
132 <title>AegesPhp:<?php echo "$accion ". $phplet_aa_[2]; ?></title>
133 <style TYPE="text/css">
134 <!--
135 /* definiciones para cuerpo principal del documento */
136 @media print {

```

```

137         body          { background-color:white;          color:black;      font-size:12pt; }
138         textarea       { background-color:white;          color:black;      font-size:12pt; width:100%;
139         #contentarea    { width:100%; }
140     }
141     @media screen {
142     <?php if(isset($_GET['bn'])) { //selector que permite presentar la interfaz en un modo amigable con
143         la impresora ?>
144         body          { background-color:white;          color:black; }
145         textarea       { background-color:white;          color:black;      width:100%; }
146     <?php } else {
147     ?>         body          { background-color:black;      color:white; }
148         textarea       { background-color:black;          color:white;      width:100%; }
149     <?php } ?>
150     /* clases para los botones */
151     .cerrar           { background-color:red;              color:white; }
152     .ejecutar          { background-color:#808080;          color:white; } /* es gris.. gray o grey =
153     nombre de color no estandar entre browsers*/
154     .guardar           { background-color:olive;           color:white; }
155 }
156 -->
157 </style>
158 </head>
159 <script type="text/javascript">
160 //caracterisiticas de impresion
161 var impresion_css = "<style type='text/css'>
162 /* definiciones para cuerpo principal del documento */
163 @media print {
164     body          { background-color:white;          color:black;      font-size:12pt; }
165     #contentarea    { width:100%; }
166 }
167 @media screen {
168     body          { background-color:black;          color:white; }
169 }</style>";
170
171 function nuevo_aeges(){
172     var uniqueId = 'NW_' + new Date().getTime(); //+ '_' + new Date().getMilliseconds();
173     newwin=window.open('<?php echo $phplet_self_name; ?>',uniqueId,'toolbar,location,
174     directories,status,menubar,scrollbars,resizable');
175     newwin.focus();
176 }
177
178 function ver_txt(titulo,este_txt,tag){
179     //     regExp=/\n/gi;
180     //     if(tag=='xmp') {crlf='\n';} else {crlf='<br />';}
181     //     este_txt = este_txt.replace(regExp,crlf);
182     //     var uniqueId = 'NW_' + new Date().getTime();
183     //     newwin=window.open('',uniqueId,'toolbar,location,directories,status,menubar,scrollbars,
184     //     resizable');
185     //     newwin.document.open("text/html","replace");
186     //     newwin.document.writeln('<html>\n<head>\n<title>'+titulo+' <?php echo $phplet_aa_[2]; ?></
187     //     title>\n'+impresion_css+'\n</head>');
188     //     if(tag!='') este_txt="<"+tag+">\n"+este_txt+"\n</"+tag+">"; //aplicacion de algun tag de
189     //     presentacion
190     //     newwin.document.writeln('<body onBlur="setTimeout(\''window.close()'\',<?php echo
191     //     $phplet_popup_time; ?>)">\n'+este_txt+'\n</body>\n</html>');
192     //     newwin.document.close();
193     //     newwin.focus();
194 }
195
196 function num_txt(titulo,este_txt,tag){
197     //     regExp=/\n/gi;
198     //     if(tag=='xmp') {crlf='\n';} else {crlf='<br />';}
199     //     este_txt = este_txt.replace(regExp,crlf);
200     //     linea_txt = este_txt.split('\n');
201     //     sz=linea_txt.length;
202     //     sz_num = sz + ' ';
203     //     este_txt='';
204     //     for(ln=0;ln<sz;ln++){
205     //         num = ln+1+' '; //conversion de numero a cadena
206     //         este_txt = este_txt + Array(sz_num.length + 1 - num.length).join(' ')+ num +
207     //         linea_txt[ln] + crlf;
208     //     }
209     //     var uniqueId = 'NW_' + new Date().getTime();
210     //     newwin=window.open('',uniqueId,'toolbar,location,directories,status,menubar,scrollbars,
211     //     resizable');
212     //     newwin.document.open("text/html","replace");

```



```

258         <input type="button" name="close" value="Cerrar [X]" class="cerrar" onclick="window
                .close();" <?php echo $tooltip['cerrar']; ?> />
259     </td>
260 </tr>
261 </table>
262 <textarea name="aeges_script" rows="<?php echo $phplet_rows_[1]; ?>"><?php echo $txt_; ?></
    textarea>
263 <table id="aeges_resultado" width="100%">
264 <tr>
265 <td>
266         <a href="#aeges_editor"><input type="button" name="boton" value="Editar script" <?
                php echo $tooltip['editor']; ?> /></a>
267         <a href="#aeges_resultado"><input type="button" name="boton" value="Resultado
                [&#124;&larr;]" <?php echo $tooltip['resultados']; ?> /></a>
268         <a href="#aeges_resultado_final"><input type="button" name="boton" value="Resultado
                [&rarr;&#124;]" <?php echo $tooltip['fin_resultados']; ?> /></a>
269         <input type="submit" name="boton" value="Ejecutar" class="ejecutar" <?php echo
                $tooltip['ejecutar']; ?> />
270     </td>
271 <td>
272         <input type="button" id="ver_script" value="Ver Script" <?php echo $tooltip['
                ver_script']; ?> onclick="ver_txt ('Script',document.aeges_editor.
                aeges_script.value,'xmp');" />
273         <input type="button" id="clear_script" value="Script SC" <?php echo $tooltip['
                script_sc']; ?> onclick="clear_txt('Script sin comentarios',document.
                aeges_editor.aeges_script.value,'xmp');" />
274         <input type="button" id="numerar_script" value="Script 123" <?php echo $tooltip['
                script_123']; ?> onclick="num_txt('Numeracion de lineas en Script',document.
                aeges_editor.aeges_script.value,'xmp');" />
275         <input type="button" id="ver_resultado" value="Ver resultado" <?php echo $tooltip['
                ver_resultados']; ?> onclick="ver_txt('Resultados',document.getElementById('
                output').innerHTML,'pre');" />
276     </td>
277 <td align="right">
278         <select name="cvs">
279             <option value="">Lista de documentos recientes..</option>
280             <?php echo "\n".$sel_hist;?>
281         </select>
282         <input type="button" name="boton" value="Abrir" onclick="if(document.aeges_editor.
                cvs.selectedindex!=='') { document.aeges_editor.submit() }" <?php echo $tooltip
                ['cvs']; ?> />
283     </td>
284 </tr>
285 </table>
286 </form>
287 <?php
288 //operaciones posteriores a la carga del formulario relacionadas con el contenido del documento
289 //simplisimo CVS, busca en el historial la ultima version numerada del documento
290 function CVS($hist,$doc){
291     reset($hist);//reiniciar el puntero interno del array
292     $maxid=0; //contador de versiones
293     while (list ($id, $el) = each ($hist)) {
294         if(preg_match("/\b".$doc."\b(?:\#(\d+))?\b/i",$el,$resp)) //capturar el id del
                registro
295             if (isset($resp[1])) if($resp[1]>$maxid) $maxid=$resp[1]; //mayor id
                obtenido
296     }
297     $maxid++; //incrementar al siguiente id
298 return $maxid;
299 };
300
301 if(($accion=="Ejecutar")and($salvar_modo==2)) $accion="Salvar_ejecutar"; //salvar el phplet y
    ejecutarlo
302 switch($accion){
303     case 'Abrir' :
304         if($ae_=="") echo "No se ha elegido/seleccionado un archivo<br />\n";
305         break;
306     case 'Salvar_ejecutar':
307     case 'Salvar':
308         if(isset($phplet_aa_[2])) {
309             $phplet_t0_="";
310             preg_match("/((?<!\|\/)\s*[\$]phplet_dir\s*=\s*([\'\"])(.*)\\1\s*;\/isU",
                $txt_, $phplet_t0_); //leer el directorio actual
311             switch($salvar_modo){ //
312                 case 1: //salvado con versiones acumulativas, establecer el nombre
313                 case 2: if(count($hist_doc[0])>0){ //si no existen registros,
                        guardar el documento con su primer nombre original

```



```

314                                     $phplet_aa_[2]=$phplet_aa_[3]."#.CVS($hist_doc[0],
315                                     $phplet_aa_[3]).".$phplet_aa_[4];
316                                     }
317                                     break;
318                                     case 3: //salvar sobre el documento original
319                                     }
320                                     //nombre del documento a actualizar/crear determinado segun $salvar_modos
321                                     $self_=$phplet_aa_[2];
322                                     $phplet_document="$phplet_t0_[2]\\$phplet_aa_[2]";
323                                     $phplet_lf_document=$phplet_document."\n";
324
325                                     //creacion del archivo especificado
326                                     $file_=@fopen($phplet_t0_[2]."/".$phplet_aa_[2],"w") or die("Fallo al abrir
327                                     /crear $self_");
328                                     fwrite($file_,$txt_) or die("Fallo al escribir en $self_");
329                                     fclose($file_) or die("Fallo al guardar $self_");
330                                     echo "<small><em>$self_</em> ha sido guardado con exito</small><br />";
331
332                                     //guardado del nombre de documento en el historial, solo almacenar los
333                                     nombres distintos entre si
334                                     if (!in_array($phplet_document,$hist_doc[0])){
335                                         $fd = @fopen($hist_file, "a") or die("Fallo al abrir el documento
336                                         $hist_file<br />\n");
337                                         fwrite($fd,$phplet_lf_document) or die("Fallo al actualizar el
338                                         documento $hist_file<br />\n");
339                                         fclose($fd) or die("Fallo al cerrar el documento $hist_file<br />\n
340                                         ");
341                                     }
342                                     } else echo "No se ha establecido un nombre de archivo. Definalo mediante \
343                                     $phplet_slf_ = \"nombrearchivo.php\";<br />\nPara establecer el directorio de
344                                     trabajo usar \$phplet_dir_ = \"Directorio_Arch\";<br />\n";
345                                     if($accion=="Salvar") break; //interrumpe la ejecucion si solo es salvar
346
347 case 'Ejecutar':
348 //esta seccion solo aparece cuando se ejecuta el codigo php del textarea
349 ?>
350 <!-- seccion de resultados -->
351 <pre id="output">
352 <?php
353     $ta = microtime(true);
354     ob_start();
355     eval($txt_); //aqui se ejecuta el codigo php en edicion
356     ob_end_flush();
357     $tb = microtime(true);
358     $tx = $tb - $ta;
359 ?>
360 </pre>
361
362 <table id="aeges_resultado_final">
363 <tr>
364 <td>
365
366     <a href="#aeges_editor"><input type="button" name="boton" value="Editar script" <?
367     php echo $tooltip['editor']; ?>></a>
368     <a href="#aeges_resultado"><input type="button" name="boton" value="Resultado
369     [&#124;&larr;]" <?php echo $tooltip['resultados']; ?>></a>
370     <input type="button" name="run" value="Ejecutar" class="ejecutar" onclick="document
371     .aeges_editor.extra.value='Ejecutar'; document.aeges_editor.submit();" <?php
372     echo $tooltip['ejecutar']; ?> />
373
374 </td>
375 </tr>
376 </table>
377
378 <?php
379 break;
380 };
381 ob_end_flush();
382 ob_end_clean();
383 $tb_ = microtime(true);
384 if (isset($tx)) $tx_ = $tb_ - $ta_ - $tx; else $tx_ = $tb_ - $ta_;
385 if ($ae_=="") $aegesphtml = $phplet_aa_[2]; else $aegesphtml = $ae_; //reporte de ejecucion
386 ?>
387 <small>
388 <?php echo "<t<em>$aegesphtml</em>";
389 if (isset($tx)) echo " realizado en $tx segundos";
390 echo "<br />\n<t$aeges_name realizado en $tx_ segundos\n"; ?>
391 </small>
392 <script type="text/javascript">
393 if(document.getElementById('output')==null) { document.getElementById('ver_resultado').style.
394     visibility = 'hidden';} else { document.getElementById('ver_resultado').style.visibility = '
395     visible'}

```

```
377 document.aeges_editor.aeges_script.focus(); //enfoue predeterminado en el boton ejecutar
378 </script>
379 </body>
380 </html>
```

E.2. GSPRT: Generador de Sistemas Producción de Reglas de Transición

Este script recrea en forma práctica la teoría de construcción de reglas de transición propuesta en el anexo D en la página 438. Construye y describe un SPRT a partir de:

- La estructura de relaciones entre símbolos del sistema: Estas se establecen mediante la definición de una tabla de combinaciones $\$z$, con ella se determina la cantidad S e identidad $s_1, s_2, \dots$ de cada uno de los símbolos a usar en el sistema.
- La dimensión máxima D , cantidad de dimensiones de evaluación, evaluaciones simultáneas o de vecindad a usar en la definición de cada regla del sistema de transiciones.

Las relaciones entre símbolos determinan la estructura de las primitivas de transición, ellas establecen el número y la configuración de la estructura de las primitivas que van a combinarse para armar las reglas de transición del sistema.

El propósito principal del script es generar SPRT completos, aquellos en los cuales la cantidad de primitivas es máxima e igual a $p = S^2$. La igualdad representa la capacidad contenida en las primitivas de realizar transiciones de un símbolo a cualquier otro del sistema sin restricciones. Igualmente puede generar otro tipo de sistemas, SPRT según su completitud (completos $p = S^2$ | incompletos $p < S^2$) y por su redundancia (con o sin repetición de primitivas en el sistema, Pej: $p > S^2$). Ejemplos de estas otras configuraciones posibles se muestran en la sección E.2.4 en la página 522.

La descripción de las primitivas se realiza por comprensión y se deduce automáticamente a partir de las relaciones entre los símbolos indicados en $\$z$. Para configurar el GSPRT se modifica según sea necesario el siguiente segmento de código, presente al inicio del script:

Script E.11: Configuración de GSPRT: SPRT completos

```
1 ...
2 //en este ejemplo se configura un SPRT 4S3D completo
3 //Definición de Tabla de símbolos
4 $o = array('_', 'A', 'B', 'C'); //tres símbolos distintos entre si y vacío
5
6 /* Configuración del vecindario / dimensiones / simultaneidad por comprensión
7 estructura de $z
8 c1,c2..cn = causas y e1,e2..en = efectos => c1:e1 + c2:e2 + ... + cn:en
9 z = c1,c2,..,cn,e1,e2,..,en = siempre en pares, en notación de permutaciones/simultaneidad/orden
10 */
11 $vecindad = 3; //dimensión máxima del SPRT
12 for($i=0;$i<$vecindad;$i++) {
13     $z[]=$o;$z[]=$o;
14 } //creación de la tabla de combinaciones $z, parejas en vecindad
15 ...
```

E.2.1. Reporte de resultados

Una vez configurado, la ejecución el script reporta las primitivas de transición, el conjunto de reglas de transición que es posible crear con las primitivas disponibles y el conjunto de patrones de transición contenido en el sistema. El detalle de los resultados obtenidos es el siguiente:

- Encabezado del reporte:
 - El nombre del sistema de producción de reglas de transición determinado por los símbolos y dimensión indicados.
 - La tabla de símbolos, la dimensión, la cantidad de combinaciones de símbolos a generar y la cantidad de reglas independientes. Si el SPRT es incompleto, el script indicará que existe selección de primitivas.
 - La cantidad de primitivas y la lista de primitivas de transición presentes en el SPRT.
 - La estructura general de construcción de reglas de transición mediante la representación por comprensión de las primitivas de transición del sistema usando para ello la configuración de símbolos en la vecindad o dimensiones disponibles.
- La tabla de reglas independientes del sistema: contiene la cantidad y configuración de cada una de las reglas de transición independientes presentes en el sistema, descritas mediante las siguientes columnas:
 - Un identificador numérico para la regla de transición.
 - El identificador del patrón de transición que genera la regla de transición.
 - Descripción de la regla de transición en notación de combinación/adición y de permutación/simultaneidad/orden de primitivas de transición.
P.ej: $A:B + A:A$ y $[A, A:B, A]$.
 - Un identificador de acción de primitivas de transición.
 - Un listado de acción de primitivas en función de los símbolos presentes en la regla.
 - El valor neto de aportes espacial y poblacional de la regla de transición. $+/-nE$ y $+/-nP$.
 - Cantidad de cambio realizado por la regla de transición. $+/-nC$.
 - El valor neto de los aportes poblacionales de los símbolos relacionados en la regla de transición. $+/-nA$, $+/-nB$, ... etc y espacio vacío = $+/-n_$.
 - Una descripción texto de la cualidad principal de la regla de transición.
P.ej: *Triple Instanciación*.
- La tabla de reglas repetidas del sistema:
 - Una descripción de las reglas de transición en notación de combinación y permutación de primitivas de transición.
 - El identificador de la regla de transición a la que iguala semánticamente. R[id] P.ej: R3.

■ Estadísticas del SPRT:

- Balance global de símbolos: presenta el aporte poblacional neto de cada uno de los símbolos del SPRT obtenido de la sumatoria de aportes poblacionales de todas las reglas disponibles en el sistema. Los valores presentados son enteros con signo.
P.ej: $+/-nA$, $+/-nB$, ...etc.
- Balance global de Espacio, Población, Cambio¹⁵ e Inercia¹⁶: presenta el aporte poblacional neto del espacio, población, acción de remplazo y de idempotencia de símbolos del SPRT obtenido mediante la sumatoria de aportes poblacionales de todas las reglas disponibles en el sistema. Los valores de espacio y población son enteros con signo, el remplazo y la idempotencia son valores reales y van desde cero hasta el número de dimensiones disponibles en el SPRT.
- Balance global de acción de las primitivas de transición Permanencia, Instanciación, Cambio y Finalización: presenta el porcentaje de acción de las primitivas indicadas mediante la sumatoria de aportes poblacionales de todas las reglas disponibles en el sistema. Estos son valores reales y están comprendidos entre cero y el número de dimensiones usadas en el SPRT.

■ Una tabla de aplicación de patrones de transición:

- Un identificador numérico para el patrón de transición.
- Descripción del patrón de transición en notación de combinación y permutación de primitivas de transición.
- Un identificador de acción de primitivas de transición.
- Un listado de acción de primitivas en función de los símbolos presentes en la regla.
- Una descripción texto de la cualidad principal de la regla de transición.
P.ej: *Permanencia + Doble Finalización*.

■ La cantidad e identidad de los patrones de transición presentes en el SPRT:

- Un identificador numérico para el patrón de transición.
- Una descripción del patrón en notación de combinación de primitivas de transición. Se usan letras mayúsculas como identificadores, se busca resaltar las diferencias estructurales.
- Un conteo de uso del patrón dentro del cuerpo de reglas de transición del SPRT.
- La cantidad de símbolos distintos usados en el patrón.
- Presentación del aporte neto poblacional del patrón de transición: P.ej: $+/-nA$, $+/-nB$, ...etc. Se deja un espacio en blanco cuando el patrón no tiene influencia poblacional, p.ej: Una regla de referencias como movimiento o intercambio. estos valores fundamentan la caracterización en diferencias del modelo BOR. Ver la sección E.6.3 en la página 638.

■ El promedio de símbolos distintos usados en la construcción de las reglas del sistema.

¹⁵Como operación de remplazo de símbolos.

¹⁶Como operación de idempotencia de símbolos.

E.2.2. Resultados de uso del script GSPRT y documentación asociada

Se incluyen los siguientes resultados obtenidos del uso del script GSPRT:

- Generación de los SPRT nSmD 1S1D, 2S1D, 2S2D, 3S2D, 3S3D, 4S2D y 4S3D.
- Generación de un ejemplo de SPRT incompleto para un sistema 4S2D.
- Generación de los SPRT de referencias 1RD, 2RD, 3RD, 4RD, 5RD y 6RD. Los tres últimos sistemas nRD no se incorporan al documento de proyecto.
- Generación de los SPRT 2mSmD 6S3D y 8S4D, presentando solo la tabla de patrones de transición presentes en los sistemas indicados.
- Al final de la sección se presenta el código fuente del script GSPRT.php.

El cuerpo de los distintos sistemas reportados y el código fuente del script GSPRT se incluyen en contenido\anexos\scripts\SPRT. En los resultados reportados no se muestran las reglas repetidas del sistema de transiciones.

E.2.3. Matriz SPRT de los sistemas nSmD, 2mSmD y nRD reportados

Esta es la ubicación lógica de los sistemas SPRT generados en el desarrollo de este proyecto.

$$M_{nSmD} = \begin{pmatrix} 1S1D & \dots & & & \\ 2S1D & 2S2D & \dots & \dots & \\ \vdots & 3S2D & 3S3D & \dots & \\ & 4S2D & 4S3D & \dots & \\ & \vdots & \vdots & & \\ & & 6S3D & \dots & \\ & & \vdots & \vdots & \\ & & & 8S4D & \end{pmatrix} \quad | \quad 2mSmD \in nSmD$$

$$M_{nRD} = \begin{pmatrix} 1RD & 2RD & 3RD & 4RD & 5RD & 6RD \end{pmatrix}$$

Los sistemas 1SnD cuentan con solo una regla de transición, la permanencia pura. El 1S1D describe funcionalmente estos SPRT.

Las reglas de transición de Homos 1.0 están comprendidas dentro de los SPRT 1S1D, 2S1D, 2S2D, 3S2D, 3S3D y 4S3D. Esta es la razón de su inclusión de estos SPRT en este documento.

Referencialmente Homos 1.0 llega hasta solo hasta el 2RD, esto es, solo se requieren hasta dos dimensiones de evaluación para representar las reglas de movimiento (de cambios de referencia) presentes en el sistema de transiciones. Los demás sistemas incluidos en este apartado se realizaron con fines de exploración y de prueba de funcionamiento de algoritmo implementado en GSPRT.php.

Los SPRT bajo el sistema 2mSmD en cada dimensión de evaluación presentan un número constante de patrones de transición. El 6S3D contiene el conjunto completo de patrones que es posible construir con 3 dimensiones de evaluación, Homos 1.0 hace uso de esa misma cantidad de dimensiones para la representación de sus reglas de transición. El sistema 8S4D se incluye como ejemplo del poder proyectivo de los SPRT. Muestra los límites de representación existentes del siguiente nivel de complejidad transicional, al adicionar una nueva dimensión de evaluación a la estructura de las reglas de transición.

E.2.4. SPRT nSmD: Sistemas de transición entre símbolos

Son el objetivo de construcción principal del script GSPRT. El siguiente listado ilustra formas de configuración de GSPRT para la implementación de sistemas nSmD, en concordancia con las definiciones dadas en la tabla D.13 en la página 457:

Script E.12: Ejemplos de configuración de GSPRT para un SPRT de referencias

```
1 ...
2 //////////////////////////////////////
3 // Representación de SPRT nSmD y configuración de tablas de símbolos //
4 //////////////////////////////////////
5
6 //a0) Un SPRT nSmD nulo no tiene símbolos o no tiene dimensiones o ambos. No se implementan en
   GSPRT
7
8 //a1) Una definición manual de Tabla de símbolos para un nSmD completo
9 //definición de un 3S3D completo
10 $o = array('A','B','C');
11 //construcción efectiva del sistema
12 $z=array($o,$o,$o,$o,$o,$o,$o); //paso completo entre símbolos
13 $vecindad=count($z)/2;
14
15 //a2) Una definición manual de Tabla de símbolos para un nSmD incompleto
16 //definición de un 3S3D incompleto
17 $o = array('A','B','C');
18 $p = array('A','B');
19 $z=array($o,$o,$o,$p,$p,$p); //no hay paso a la referencia 'C'
20 $vecindad=count($z)/2;
21
22 //a3) Una definición manual de Tabla de símbolos para un nSmD completo redundante / alienado
23 //definición de un 3S3D con alienación de causa del símbolo 'B'
24 $o = array('A','B','B','B','C'); // 'B' es 3/5 mas probable que 'A' y 'C'
25 $p = array('A','B','C');
26 $z=array($o,$o,$o,$p,$p,$p); //paso completo entre símbolos
27 $vecindad=count($z)/2;
28
29 //a4) Una definición manual de Tabla de símbolos para un nSmD incompleto redundante / alienado
30 //definición de un 3S3D con alienación de efecto del símbolo 'A'
31 $o = array('A','B','C');
32 $p = array('A','A','A','B'); // 'A' es 1/2 mas probable que 'A' y 'C'
33 $z=array($o,$o,$o,$p,$p,$p); //no hay paso a la referencia 'C'
34 $vecindad=count($z)/2;
35
36 //a5) Definición de un 9S3D incompleto con redundancias en causas y efectos de transición
37 //prueba de versatilidad de configuración de algoritmo en GSPRT
38 $o1 = array('A','B','C','C','C'); //redundancia de causa 'C'
39 $o2 = array('D','E','F');
40 $o3 = array('G','H','I');
41 $p1 = array('D','E','F');
42 $p2 = array('D','E','F','G','H','I');
43 $p3 = array('A','B','B','B','C','D','E','F','G','H','I'); //redundancia de efecto 'B'
44 $z=array($o1,$o2,$o3,$p1,$p2,$p3);
45 $vecindad=count($z)/2;
46
47 //a6) Un SPRT 4S5D completo con símbolo de vacío = '_'
48 $o = array('_', 'A','B','C');
49 //construcción efectiva del sistema
50 $z=array($o,$o,$o,$o,$o,$o,$o,$o,$o,$o,$o,$o,$o,$o,$o); //paso completo entre símbolos
51 $vecindad=count($z)/2;
52 ...
```


E.2.4.1. SPRT 1S1D

Al establecer solo un símbolo en el sistema, es posible una forma de transición: la permanencia pura, dimensional de ese símbolo entre pasos de transición. En el reporte E.1 se usó el símbolo de vacío, representa un sistema en el cual no hay perturbaciones, solo hay espacio disponible y formas de vacío en igualdad de condiciones, el estado natural de un ambiente sin objetos.

Para 1SnD, el manejo de un símbolo en el sistema garantiza la permanencia de tipo ante cualquier número de dimensiones y pasos de simulación. Es claro que solo existe una forma de patrón de transición en este sistema, que puede interpretarse como permanencia o idempotencia multidimensional pura.

Reporte E.1: SPRT 1S1D de vacío

```
SPRT(1S,1D)
Tabla de símbolos = {_} = 1 símbolos
Dimensiones, simultaneidad, vecindario máximo = 1 = Se combinarán 2 vectores
elementos en cada vector [ 1 1 ] = 1 combinaciones
Existen 1 reglas independientes

Se usan 1 primitivas de transición
-:-

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario  Dimensión  Entrada/Causa  ->  Salida/Efecto  Notación
Principal             Celda[0]      {_}          ->    {_}        {-:-}

1 1 _:- [-:-] P P(_)          Permanencia
Se realizaron 1 combinaciones, existen 1 reglas independientes = permutaciones de primitivas
diferentes

Existen 0 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
  0_

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
  OE      OP      0.00C = 0.00% Cambio      1.00I = 100.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
  1.00P = 100.00% 0.00I = 0.00%      0.00C = 0.00%      0.00F = 0.00%

Ejemplos de aplicación: Existen 1 patrones de transición
1 _:- [-:-] P P(_)      Permanencia

Tabla de patrones de transición de SPRT(1S,1D)
1 A:A 1 1
Existen en promedio 1 símbolos distintos en cada regla del SPRT
```

E.2.4.2. SPRT 2S1D

Dependiendo de la naturaleza de los símbolos a usar en las combinaciones se pueden establecer las siguientes configuraciones del sistema: para dos símbolos iguales, dos símbolos distintos y un símbolo de clase e identificador de vacío¹⁷, se presenta un ejemplo de cada caso.

- Dos símbolos iguales: Configura un sistema permanente, iguala las construcciones de 1S1D.
- Dos símbolos distintos: Configura un sistema en el cual únicamente existen cambios y permanencias de símbolos. Nótese que en la tabla de reglas independientes reportada no aparecen los indicadores de cambio poblacional ni espacial.

Reporte E.2: SPRT 2S1D de símbolos distintos

```
SPRT(2S,1D)
Tabla de símbolos = {a,b} = 2 símbolos
Dimensiones, simultaneidad, vecindario máximo = 1 = Se combinarán 2 vectores
elementos en cada vector [ 2 2 ] = 4 combinaciones
Existen 4 reglas independientes

Se usan 4 primitivas de transición
a:a      a:b      b:a      b:b

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario  Dimensión  Entrada/Causa  ->  Salida/Efecto  Notación
Principal            Celda[0]      {a,b}         ->   {a,b}       {a,b:a,b}

1 1 a:a [a:a] P P(a)      Permanencia
2 2 b:a [b:a] C C(b,a) 1C 1a-1b Cambio
3 2 a:b [a:b] C C(a,b) 1C -1a 1b Cambio
4 1 b:b [b:b] P P(b)      Permanencia
Se realizaron 4 combinaciones, existen 4 reglas independientes = permutaciones de primitivas
diferentes

Existen 0 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
  0a  0b

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
  0E   0P   0.50C = 50.00% Cambio   0.50I = 50.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
  0.50P = 50.00%   0.00I = 0.00%   0.50C = 50.00%   0.00F = 0.00%

Ejemplos de aplicación: Existen 2 patrones de transición
1 a:a [a:a] P P(a)      Permanencia
2 b:a [b:a] C C(b,a) Cambio

Tabla de patrones de transición de SPRT(2S,1D)
1 A:A 2 1
2 A:B 2 2 -1A +1B
Existen en promedio 1.5 símbolos distintos en cada regla del SPRT
```

- Un símbolo de clase y vacío: Representa una variación del sistema anterior causada por el significado operativo de uno de los símbolos, al usar el identificador de vacío como símbolo alternativo o de contraste. El sistema se caracteriza mediante permanencia, instanciación y finalización de símbolos. Al igual que en el caso anterior, solo contiene dos patrones de transición, permanencia y cambio puros.

¹⁷Otros casos son posibles si se definen nuevos tipos de símbolos.

Reporte E.3: SPRT 2S1D de un símbolo y vacío

```

SPRT(2S,1D)
Tabla de símbolos = {_,a} = 2 símbolos
Dimensiones, simultaneidad, vecindario máximo = 1 = Se combinarán 2 vectores
elementos en cada vector [ 2 2 ] = 4 combinaciones
Existen 4 reglas independientes

Se usan 4 primitivas de transición
_:_      _:a      a:_      a:a

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario   Dimensión   Entrada/Causa   ->   Salida/Efecto   Notación
Principal              Celda[0]      {_,a}          ->   {_,a}          {_,a:_:,a}

1 1 _:_ [_:_] P P(_)      Permanencia
2 2 a:_ [a:_] F F(a)      1E-1P 1C 1_-1a Finalización
3 2 _:a [_:a] I I(a)      -1E 1P 1C -1_ 1a Instanciación
4 1 a:a [a:a] P P(a)      Permanencia
Se realizaron 4 combinaciones, existen 4 reglas independientes = permutaciones de primitivas
diferentes

Existen 0 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
  0_      0a

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
  OE      OP      0.50C = 50.00% Cambio      0.50I = 50.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
  0.50P = 50.00% 0.25I = 25.00% 0.00C = 0.00% 0.25F = 25.00%

Ejemplos de aplicación: Existen 2 patrones de transición
1 _:_ [_:_] P P(_)      Permanencia
2 a:_ [a:_] F F(a)      Finalización

Tabla de patrones de transición de SPRT(2S,1D)
1 A:A 2 1
2 A:B 2 2 -1A +1B
Existen en promedio 1.5 símbolos distintos en cada regla del SPRT

```

E.2.4.3. SPRT 2S2D

Dependiendo de la naturaleza del tipo de símbolo a usar en las combinaciones se pueden establecer las siguientes configuraciones del sistema: para dos símbolos iguales, dos símbolos distintos y un símbolo de clase e identificador de vacío¹⁸, se presenta un ejemplo de los dos últimos casos. A partir de la bidimensionalidad del sistema, la simultaneidad hace realidad las operaciones interdimensionales y el duplicado de tipos de transiciones, aunque mantiene la tendencia de comportamiento de presente en 2S1D.

La cantidad de símbolos dada no es suficiente para la generación y coexistencia de todos los tipos de transición posibles.

- Dos símbolos iguales: Configura un sistema permanente, iguala las construcciones de 1S2D.
- Dos símbolos distintos: Configura un sistema en el cual existen cambios, permanencias y duplicaciones por simultaneidad de estas operaciones. En este caso la tabla de reglas independientes no reporta los indicadores de cambio poblacional ni espacial.

Reporte E.4: SPRT 2S2D de símbolos distintos

```
SPRT (2S,2D)
Tabla de símbolos = {a,b} = 2 símbolos
Dimensiones, simultaneidad, vecindario máximo = 2 = Se combinarán 4 vectores
elementos en cada vector [ 2 2 2 2 ] = 16 combinaciones
Existen 10 reglas independientes

Se usan 4 primitivas de transición
a:a      a:b      b:a      b:b

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario  Dimensión  Entrada/Causa  ->  Salida/Efecto  Notación
Principal             Celda[0]       {a,b}         ->  {a,b}         {a,b:a,b}
Vecina 1              Celda[1]       {a,b}         ->  {a,b}         {a,b:a,b}

1 1 a:a + a:a [a,a:a,a] PP P(a) P(a) Doble Permanencia
2 2 b:a + a:a [b,a:a,a] CP C(b,a) P(a) 1C 1a-1b Permanencia + Cambio
3 3 b:a + b:a [b,b:a,a] CC C(b,a) C(b,a) 2C 2a-2b Doble Cambio
4 4 a:b + a:a [a,a:b,a] CP C(a,b) P(a) 1C -1a 1b Permanencia + Cambio
5 5 b:b + a:a [b,a:b,a] PP P(b) P(a) Doble Permanencia
6 6 a:b + b:a [a,b:b,a] CC C(a,b) C(b,a) 2C Doble Cambio
7 4 b:b + b:a [b,b:b,a] PC P(b) C(b,a) 1C 1a-1b Permanencia + Cambio
8 3 a:b + a:b [a,a:b,b] CC C(a,b) C(a,b) 2C -2a 2b Doble Cambio
9 2 b:b + a:b [b,a:b,b] PC P(b) C(a,b) 1C -1a 1b Permanencia + Cambio
10 1 b:b + b:b [b,b:b,b] PP P(b) P(b) Doble Permanencia

Se realizaron 16 combinaciones, existen 10 reglas independientes = permutaciones de primitivas
diferentes

Existen 6 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
0a 0b

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
0E 0P 1.00C = 50.00% Cambio 1.00I = 50.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
1.00P = 50.00% 0.00I = 0.00% 1.00C = 50.00% 0.00F = 0.00%

Ejemplos de aplicación: Existen 6 patrones de transición
1 a:a + a:a [a,a:a,a] PP P(a) P(a) Doble Permanencia
2 b:a + a:a [b,a:a,a] CP C(b,a) P(a) Permanencia + Cambio
3 b:a + b:a [b,b:a,a] CC C(b,a) C(b,a) Doble Cambio
```

¹⁸Otros casos son posibles si se definen nuevos tipos de símbolos.

```

4 a:b + a:a [a,a:b,a] CP C(a,b) P(a) Permanencia + Cambio
5 b:b + a:a [b,a:b,a] PP P(b) P(a) Doble Permanencia
6 a:b + b:a [a,b:b,a] CC C(a,b) C(b,a) Doble Cambio

```

Tabla de patrones de transición de SPRT(2S,2D)

```

1 A:A + A:A 2 1
2 A:A + B:A 4 2 +1A -1B
3 A:B + A:B 2 2 -2A +2B
4 A:A + A:B 4 2 -1A +1B
5 A:A + B:B 2 2
6 A:B + B:A 2 2

```

Existen en promedio 1.875 símbolos distintos en cada regla del SPRT

- Un símbolo de clase y vacío: El sistema se caracteriza mediante la combinación y duplicación de las transiciones de permanencia, instanciación y finalización de símbolos.
Nótese que comparado con el sistema del listado E.4, los patrones de transición son iguales y las reglas de transición difieren por el significado operativo de los símbolos empleados para su construcción.

Reporte E.5: SPRT 2S2D de símbolo y vacío

SPRT(2S,2D)

Tabla de símbolos = {_,a} = 2 símbolos

Dimensiones, simultaneidad, vecindario máximo = 2 = Se combinarán 4 vectores
elementos en cada vector [2 2 2 2] = 16 combinaciones

Existen 10 reglas independientes

Se usan 4 primitivas de transición

_: _ _: a a: _ a: a

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa	->	Salida/Efecto	Notación
Principal	Celda[0]	{_,a}	->	{_,a}	{_,a:_,a}
Vecina 1	Celda[1]	{_,a}	->	{_,a}	{_,a:_,a}

```

1 1 _:_ + _:_ [_:_:_] PP P(_) P(_) Doble Permanencia
2 2 a:_ + _:_ [a,_:_] FP F(a) P(_) 1E-1P 1C 1_-1a Permanencia + Finalización
3 3 a:_ + a:_ [a,a:_] FF F(a) F(a) 2E-2P 2C 2_-2a Doble Finalización
4 4 _:a + _:_ [_:_:a_] IP I(a) P(_) -1E 1P 1C -1_ 1a Permanencia + Instanciación
5 5 a:a + _:_ [a,_:a_] PP P(a) P(_) Doble Permanencia
6 6 _:a + a:_ [_:a:a_] IF I(a) F(a) 2C Instanciación + Finalización
7 4 a:a + a:_ [a,a:a_] PF P(a) F(a) 1E-1P 1C 1_-1a Permanencia + Finalización
8 3 _:a + _:a [_:_:a,a] II I(a) I(a) -2E 2P 2C -2_ 2a Doble Instanciación
9 2 a:a + _:a [a,_:a,a] PI P(a) I(a) -1E 1P 1C -1_ 1a Permanencia + Instanciación
10 1 a:a + a:a [a,a:a,a] PP P(a) P(a) Doble Permanencia

```

Se realizaron 16 combinaciones, existen 10 reglas independientes = permutaciones de primitivas diferentes

Existen 6 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos
0_ 0a

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
0E 0P 1.00C = 50.00% Cambio 1.00I = 50.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
1.00P = 50.00% 0.50I = 25.00% 0.00C = 0.00% 0.50F = 25.00%

Ejemplos de aplicación: Existen 6 patrones de transición

```

1 _:_ + _:_ [_:_:_] PP P(_) P(_) Doble Permanencia
2 a:_ + _:_ [a,_:_] FP F(a) P(_) Permanencia + Finalización
3 a:_ + a:_ [a,a:_] FF F(a) F(a) Doble Finalización
4 _:a + _:_ [_:_:a_] IP I(a) P(_) Permanencia + Instanciación
5 a:a + _:_ [a,_:a_] PP P(a) P(_) Doble Permanencia
6 _:a + a:_ [_:a:a_] IF I(a) F(a) Instanciación + Finalización

```

Tabla de patrones de transición de SPRT(2S,2D)

1	A:A + A:A	2	1	
2	A:A + B:A	4	2	+1A -1B
3	A:B + A:B	2	2	-2A +2B
4	A:A + A:B	4	2	-1A +1B
5	A:A + B:B	2	2	
6	A:B + B:A	2	2	

Existen en promedio 1.875 símbolos distintos en cada regla del SPRT

E.2.4.4. SPRT 3S2D

Este es el primer sistema en el cual se presentan simultáneamente todos los tipos de transición posibles: Permanencia, Cambio, Instanciación y Finalización, cuando se usan dos símbolos distintos entre si y el símbolo de vacío.

Se realizó edición sobre los resultados solo con fines de ajuste en la incorporación del texto.

Reporte E.6: SPRT 3S2D

SPRT(3S,2D)

Tabla de símbolos = {_,a,b} = 3 símbolos

Dimensiones, simultaneidad, vecindario máximo = 2 = Se combinarán 4 vectores

elementos en cada vector [3 3 3 3] = 81 combinaciones

Existen 45 reglas independientes

Se usan 9 primitivas de transición

_: _ _: a _: b a: _ a: a a: b b: _ b: a b: b

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa	->	Salida/Efecto	Notación
Principal	Celda[0]	{_,a,b} ->	{_,a,b}	{_,a,b:_,a,b}	
Vecina 1	Celda[1]	{_,a,b} ->	{_,a,b}	{_,a,b:_,a,b}	
1 1	_: _ + _: _	[_,_:_,_] PP P(_)	P(_)		Doble Permanencia
2 2	a: _ + _: _	[a,_:_,_] FP F(a)	P(_)	1E-1P 1C 1_-1a	Permanencia + Finalización
3 2	b: _ + _: _	[b,_:_,_] FP F(b)	P(_)	1E-1P 1C 1_-1b	Permanencia + Finalización
4 3	a: _ + a: _	[a,a:_,_] FF F(a)	F(a)	2E-2P 2C 2_-2a	Doble Finalización
5 4	b: _ + a: _	[b,a:_,_] FF F(b)	F(a)	2E-2P 2C 2_-1a-1b	Doble Finalización
6 3	b: _ + b: _	[b,b:_,_] FF F(b)	F(b)	2E-2P 2C 2_-2b	Doble Finalización
7 5	_: a + _: _	[_,_:a,_] IP I(a)	P(_)	-1E 1P 1C -1_ 1a	Permanencia + Instanciación
8 6	a: a + _: _	[a,_:a,_] PP P(a)	P(_)		Doble Permanencia
9 7	b: a + _: _	[b,_:a,_] CP C(b,a)	P(_)	1C 1a-1b	Permanencia + Cambio
10 8	_: a + a: _	[_,a:a,_] IF I(a)	F(a)	2C	Instanciación + Finalización
11 5	a: a + a: _	[a,a:a,_] PF P(a)	F(a)	1E-1P 1C 1_-1a	Permanencia + Finalización
12 9	b: a + a: _	[b,a:a,_] CF C(b,a)	F(a)	1E-1P 2C 1_-1b	Cambio + Finalización
13 9	_: a + b: _	[_,b:a,_] IF I(a)	F(b)	2C 1a-1b	Instanciación + Finalización
14 7	a: a + b: _	[a,b:a,_] PF P(a)	F(b)	1E-1P 1C 1_-1b	Permanencia + Finalización
15 10	b: a + b: _	[b,b:a,_] CF C(b,a)	F(b)	1E-1P 2C 1_ 1a-2b	Cambio + Finalización
16 5	_: b + _: _	[_,_:b,_] IP I(b)	P(_)	-1E 1P 1C -1_ 1b	Permanencia + Instanciación
17 7	a: b + _: _	[a,_:b,_] CP C(a,b)	P(_)	1C -1a 1b	Permanencia + Cambio
18 6	b: b + _: _	[b,_:b,_] PP P(b)	P(_)		Doble Permanencia
19 9	_: b + a: _	[_,a:b,_] IF I(b)	F(a)	2C -1a 1b	Instanciación + Finalización
20 10	a: b + a: _	[a,a:b,_] CF C(a,b)	F(a)	1E-1P 2C 1_-2a 1b	Cambio + Finalización
21 7	b: b + a: _	[b,a:b,_] PF P(b)	F(a)	1E-1P 1C 1_-1a	Permanencia + Finalización
22 8	_: b + b: _	[_,b:b,_] IF I(b)	F(b)	2C	Instanciación + Finalización
23 9	a: b + b: _	[a,b:b,_] CF C(a,b)	F(b)	1E-1P 2C 1_-1a	Cambio + Finalización
24 5	b: b + b: _	[b,b:b,_] PF P(b)	F(b)	1E-1P 1C 1_-1b	Permanencia + Finalización
25 3	_: a + _: a	[_,_:a,a] II I(a)	I(a)	-2E 2P 2C -2_ 2a	Doble Instanciación
26 2	a: a + _: a	[a,_:a,a] PI P(a)	I(a)	-1E 1P 1C -1_ 1a	Permanencia + Instanciación
27 4	b: a + _: a	[b,_:a,a] CI C(b,a)	I(a)	-1E 1P 2C -1_ 2a-1b	Instanciación + Cambio
28 1	a: a + a: a	[a,a:a,a] PP P(a)	P(a)		Doble Permanencia
29 2	b: a + a: a	[b,a:a,a] CP C(b,a)	P(a)	1C 1a-1b	Permanencia + Cambio
30 3	b: a + b: a	[b,b:a,a] CC C(b,a)	C(b,a)	2C 2a-2b	Doble Cambio
31 10	_: b + _: a	[_,_:b,a] II I(b)	I(a)	-2E 2P 2C -2_ 1a 1b	Doble Instanciación
32 9	a: b + _: a	[a,_:b,a] CI C(a,b)	I(a)	-1E 1P 2C -1_ 1b	Instanciación + Cambio
33 7	b: b + _: a	[b,_:b,a] PI P(b)	I(a)	-1E 1P 1C -1_ 1a	Permanencia + Instanciación
34 7	_: b + a: a	[_,a:b,a] IP I(b)	P(a)	-1E 1P 1C -1_ 1b	Permanencia + Instanciación
35 5	a: b + a: a	[a,a:b,a] CP C(a,b)	P(a)	1C -1a 1b	Permanencia + Cambio
36 6	b: b + a: a	[b,a:b,a] PP P(b)	P(a)		Doble Permanencia
37 9	_: b + b: a	[_,b:b,a] IC I(b)	C(b,a)	-1E 1P 2C -1_ 1a	Instanciación + Cambio
38 8	a: b + b: a	[a,b:b,a] CC C(a,b)	C(b,a)	2C	Doble Cambio
39 5	b: b + b: a	[b,b:b,a] PC P(b)	C(b,a)	1C 1a-1b	Permanencia + Cambio
40 3	_: b + _: b	[_,_:b,b] II I(b)	I(b)	-2E 2P 2C -2_ 2b	Doble Instanciación
41 4	a: b + _: b	[a,_:b,b] CI C(a,b)	I(b)	-1E 1P 2C -1_-1a 2b	Instanciación + Cambio
42 2	b: b + _: b	[b,_:b,b] PI P(b)	I(b)	-1E 1P 1C -1_ 1b	Permanencia + Instanciación
43 3	a: b + a: b	[a,a:b,b] CC C(a,b)	C(a,b)	2C -2a 2b	Doble Cambio
44 2	b: b + a: b	[b,a:b,b] PC P(b)	C(a,b)	1C -1a 1b	Permanencia + Cambio
45 1	b: b + b: b	[b,b:b,b] PP P(b)	P(b)		Doble Permanencia

Se realizaron 81 combinaciones, existen 45 reglas independientes = permutaciones de primitivas diferentes

Existen 36 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos
 0_ 0a 0b

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
 0E 0P 1.33C = 66.67% Cambio 0.67I = 33.33% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
 0.67P = 33.33% 0.44I = 22.22% 0.44C = 22.22% 0.44F = 22.22%

Ejemplos de aplicación: Existen 10 patrones de transición

1	_:_ + _:_	[_:_:_]	PP	P(_)	P(_)	Doble Permanencia
2	a:_ + _:_	[a:_:_]	FP	F(a)	P(_)	Permanencia + Finalización
3	a:_ + a:_	[a,a:_:_]	FF	F(a)	F(a)	Doble Finalización
4	b:_ + a:_	[b,a:_:_]	FF	F(b)	F(a)	Doble Finalización
5	_:a + _:_	[_:_:a:_]	IP	I(a)	P(_)	Permanencia + Instanciación
6	a:a + _:_	[a:_:a:_]	PP	P(a)	P(_)	Doble Permanencia
7	b:a + _:_	[b:_:a:_]	CP	C(b,a)	P(_)	Permanencia + Cambio
8	_:a + a:_	[_:a:a:_]	IF	I(a)	F(a)	Instanciación + Finalización
9	b:a + a:_	[b,a:a:_]	CF	C(b,a)	F(a)	Cambio + Finalización
10	b:a + b:_	[b,b:a:_]	CF	C(b,a)	F(b)	Cambio + Finalización

Tabla de patrones de transición de SPRT(3S,2D)

1	A:A + A:A	3	1
2	A:A + B:A	12	2 +1A -1B
3	A:B + A:B	6	2 -2A +2B
4	A:B + C:B	6	3 -1A +2B -1C
5	A:A + A:B	12	2 -1A +1B
6	A:A + B:B	6	2
7	A:A + B:C	12	3 -1B +1C
8	A:B + B:A	6	2
9	A:B + B:C	12	3 -1A +1C
10	A:B + A:C	6	3 -2A +1B +1C

Existen en promedio 2.4074074074074 símbolos distintos en cada regla del SPRT

E.2.4.5. SPRT 3S3D

Se realizó edición sobre los resultados solo con fines de ajuste en la incorporación del texto.

Reporte E.7: SPRT 3S3D

SPRT(3S,3D)

Tabla de símbolos = {_,a,b} = 3 símbolos

Dimensiones, simultaneidad, vecindario máximo = 3 = Se combinarán 6 vectores

elementos en cada vector [3 3 3 3 3] = 729 combinaciones

Existen 165 reglas independientes

Se usan 9 primitivas de transición

_: _ _:a _:b a:_ a:a a:b b:_ b:a b:b

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa	->	Salida/Efecto	Notación
Principal	Celda[0]	{_,a,b}	->	{_,a,b} {_,a,b:_ ,a,b}	
Vecina 1	Celda[1]	{_,a,b}	->	{_,a,b} {_,a,b:_ ,a,b}	
Vecina 2	Celda[2]	{_,a,b}	->	{_,a,b} {_,a,b:_ ,a,b}	
1	1	_: _ + _: _ + _: _	PPP		Triple Permanencia
2	2	a:_ + _: _ + _: _	FPP	1E-1P 1C 1_-1a	Doble Permanencia + Finalización
3	2	b:_ + _: _ + _: _	FPP	1E-1P 1C 1_-1b	Doble Permanencia + Finalización
4	3	a:_ + a:_ + _: _	FFP	2E-2P 2C 2_-2a	Permanencia + Doble Finalización
5	4	b:_ + a:_ + _: _	FFP	2E-2P 2C 2_-1a-1b	Permanencia + Doble Finalización
6	3	b:_ + b:_ + _: _	FFP	2E-2P 2C 2_-2b	Permanencia + Doble Finalización
7	5	a:_ + a:_ + a:_	FFF	3E-3P 3C 3_-3a	Triple Finalización
8	6	b:_ + a:_ + a:_	FFF	3E-3P 3C 3_-2a-1b	Triple Finalización
9	6	b:_ + b:_ + a:_	FFF	3E-3P 3C 3_-1a-2b	Triple Finalización
10	5	b:_ + b:_ + b:_	FFF	3E-3P 3C 3_-3b	Triple Finalización
11	7	_:a + _: _ + _: _	IPP	-1E 1P 1C -1_ 1a	Doble Permanencia + Instanciación
12	8	a:a + a:_ + _: _	PPP		Triple Permanencia
13	9	b:a + _: _ + _: _	CPP	1C 1a-1b	Doble Permanencia + Cambio
14	10	_:a + a:_ + _: _	IFP	2C	Permanencia + Instanciación + Finalización
15	11	a:a + a:_ + _: _	PFP	1E-1P 1C 1_-1a	Doble Permanencia + Finalización
16	12	b:a + a:_ + _: _	CFP	1E-1P 2C 1_-1b	Permanencia + Cambio + Finalización
17	13	_:a + b:_ + _: _	IFP	2C 1a-1b	Permanencia + Instanciación + Finalización
18	14	a:a + b:_ + _: _	PFP	1E-1P 1C 1_-1b	Doble Permanencia + Finalización
19	15	b:a + b:_ + _: _	CFP	1E-1P 2C 1_-1a-2b	Permanencia + Cambio + Finalización
20	16	_:a + a:_ + a:_	IFF	1E-1P 3C 1_-1a	Instanciación + Doble Finalización
21	17	a:a + a:_ + a:_	PFF	2E-2P 2C 2_-2a	Permanencia + Doble Finalización
22	18	b:a + a:_ + a:_	CFF	2E-2P 3C 2_-1a-1b	Cambio + Doble Finalización
23	19	_:a + b:_ + a:_	IFF	1E-1P 3C 1_-1b	Instanciación + Doble Finalización
24	20	a:a + b:_ + a:_	PFF	2E-2P 2C 2_-1a-1b	Permanencia + Doble Finalización
25	21	b:a + b:_ + a:_	CFF	2E-2P 3C 2_-2b	Cambio + Doble Finalización
26	22	_:a + b:_ + b:_	IFF	1E-1P 3C 1_-1a-2b	Instanciación + Doble Finalización
27	23	a:a + b:_ + b:_	PFF	2E-2P 2C 2_-2b	Permanencia + Doble Finalización
28	24	b:a + b:_ + b:_	CFF	2E-2P 3C 2_-1a-3b	Cambio + Doble Finalización
29	7	_:b + _: _ + _: _	IPP	-1E 1P 1C -1_ 1b	Doble Permanencia + Instanciación
30	9	a:b + _: _ + _: _	CPP	1C -1a 1b	Doble Permanencia + Cambio
31	8	b:b + _: _ + _: _	PPP		Triple Permanencia
32	13	_:b + a:_ + _: _	IFP	2C -1a 1b	Permanencia + Instanciación + Finalización
33	15	a:b + a:_ + _: _	CFP	1E-1P 2C 1_-2a 1b	Permanencia + Cambio + Finalización
34	14	b:b + a:_ + _: _	PFP	1E-1P 1C 1_-1a	Doble Permanencia + Finalización
35	10	_:b + b:_ + _: _	IFP	2C	Permanencia + Instanciación + Finalización
36	12	a:b + b:_ + _: _	CFP	1E-1P 2C 1_-1a	Permanencia + Cambio + Finalización
37	11	b:b + b:_ + _: _	PFP	1E-1P 1C 1_-1b	Doble Permanencia + Finalización
38	22	_:b + a:_ + a:_	IFF	1E-1P 3C 1_-2a 1b	Instanciación + Doble Finalización
39	24	a:b + a:_ + a:_	CFF	2E-2P 3C 2_-3a 1b	Cambio + Doble Finalización
40	23	b:b + a:_ + a:_	PFF	2E-2P 2C 2_-2a	Permanencia + Doble Finalización
41	19	_:b + b:_ + a:_	IFF	1E-1P 3C 1_-1a	Instanciación + Doble Finalización
42	21	a:b + b:_ + a:_	CFF	2E-2P 3C 2_-2a	Cambio + Doble Finalización
43	20	b:b + b:_ + a:_	PFF	2E-2P 2C 2_-1a-1b	Permanencia + Doble Finalización
44	16	_:b + b:_ + b:_	IFF	1E-1P 3C 1_-1b	Instanciación + Doble Finalización
45	18	a:b + b:_ + b:_	CFF	2E-2P 3C 2_-1a-1b	Cambio + Doble Finalización
46	17	b:b + b:_ + b:_	PFF	2E-2P 2C 2_-2b	Permanencia + Doble Finalización
47	17	_:a + _:a + _: _	IIP	-2E 2P 2C -2_ 2a	Permanencia + Doble Instanciación
48	11	a:a + _:a + _: _	PIP	-1E 1P 1C -1_ 1a	Doble Permanencia + Instanciación
49	20	b:a + _:a + _: _	CIP	-1E 1P 2C -1_ 2a-1b	Permanencia + Instanciación + Cambio
50	8	a:a + a:a + _: _	PPP		Triple Permanencia
51	14	b:a + a:a + _: _	CPP	1C 1a-1b	Doble Permanencia + Cambio
52	23	b:a + b:a + _: _	CCP	2C 2a-2b	Permanencia + Doble Cambio
53	16	_:a + _:a + a:_	IIF	-1E 1P 3C -1_ 1a	Doble Instanciación + Finalización
54	10	a:a + _:a + a:_	PIF	2C	Permanencia + Instanciación + Finalización
55	19	b:a + _:a + a:_	CIF	3C 1a-1b	Instanciación + Cambio + Finalización

56	7	a:a + a:a + a:_	PPF	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
57	13	b:a + a:a + a:_	CPF	1E-1P 2C	1_-1b	Permanencia + Cambio + Finalización
58	22	b:a + b:a + a:_	CCF	1E-1P 3C	1_ 1a-2b	Doble Cambio + Finalización
59	18	_:a + _:a + b:_	IIF	-1E 1P 3C	-1_ 2a-1b	Doble Instanciación + Finalización
60	12	a:a + _:a + b:_	PIF	2C	1a-1b	Permanencia + Instanciación + Finalización
61	21	b:a + _:a + b:_	CIF	3C	2a-2b	Instanciación + Cambio + Finalización
62	9	a:a + a:a + b:_	PPF	1E-1P 1C	1_-1b	Doble Permanencia + Finalización
63	15	b:a + a:a + b:_	CPF	1E-1P 2C	1_ 1a-2b	Permanencia + Cambio + Finalización
64	24	b:a + b:a + b:_	CCF	1E-1P 3C	1_ 2a-3b	Doble Cambio + Finalización
65	25	_:b + _:a + _:_	IIP	-2E 2P 2C	-2_ 1a 1b	Permanencia + Doble Instanciación
66	26	a:b + _:a + _:_	CIP	-1E 1P 2C	-1_ 1b	Permanencia + Instanciación + Cambio
67	27	b:b + _:a + _:_	PIP	-1E 1P 1C	-1_ 1a	Doble Permanencia + Instanciación
68	27	_:b + a:a + _:_	IPP	-1E 1P 1C	-1_ 1b	Doble Permanencia + Instanciación
69	27	a:b + a:a + _:_	CPP	1C	-1a 1b	Doble Permanencia + Cambio
70	28	b:b + a:a + _:_	PPP			Triple Permanencia
71	26	_:b + b:a + _:_	ICP	-1E 1P 2C	-1_ 1a	Permanencia + Instanciación + Cambio
72	29	a:b + b:a + _:_	CCP	2C		Permanencia + Doble Cambio
73	27	b:b + b:a + _:_	PCP	1C	1a-1b	Doble Permanencia + Cambio
74	30	_:b + _:a + a:_	IIF	-1E 1P 3C	-1_ 1b	Doble Instanciación + Finalización
75	30	a:b + _:a + a:_	CIF	3C	-1a 1b	Instanciación + Cambio + Finalización
76	29	b:b + _:a + a:_	PIF	2C		Permanencia + Instanciación + Finalización
77	26	_:b + a:a + a:_	IPF	2C	-1a 1b	Permanencia + Instanciación + Finalización
78	25	a:b + a:a + a:_	CPF	1E-1P 2C	1_-2a 1b	Permanencia + Cambio + Finalización
79	27	b:b + a:a + a:_	PPF	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
80	31	_:b + b:a + a:_	ICF	3C		Instanciación + Cambio + Finalización
81	30	a:b + b:a + a:_	CCF	1E-1P 3C	1_-1a	Doble Cambio + Finalización
82	26	b:b + b:a + a:_	PCF	1E-1P 2C	1_-1b	Permanencia + Cambio + Finalización
83	30	_:b + _:a + b:_	IIF	-1E 1P 3C	-1_ 1a	Doble Instanciación + Finalización
84	31	a:b + _:a + b:_	CIF	3C		Instanciación + Cambio + Finalización
85	26	b:b + _:a + b:_	PIF	2C	1a-1b	Permanencia + Instanciación + Finalización
86	29	_:b + a:a + b:_	IPF	2C		Permanencia + Instanciación + Finalización
87	26	a:b + a:a + b:_	CPF	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
88	27	b:b + a:a + b:_	PPF	1E-1P 1C	1_-1b	Doble Permanencia + Finalización
89	30	_:b + b:a + b:_	ICF	3C	1a-1b	Instanciación + Cambio + Finalización
90	30	a:b + b:a + b:_	CCF	1E-1P 3C	1_-1b	Doble Cambio + Finalización
91	25	b:b + b:a + b:_	PCF	1E-1P 2C	1_ 1a-2b	Permanencia + Cambio + Finalización
92	17	_:b + _:b + _:_	IIP	-2E 2P 2C	-2_ 2b	Permanencia + Doble Instanciación
93	20	a:b + _:b + _:_	CIP	-1E 1P 2C	-1_-1a 2b	Permanencia + Instanciación + Cambio
94	11	b:b + _:b + _:_	PIP	-1E 1P 1C	-1_ 1b	Doble Permanencia + Instanciación
95	23	a:b + a:b + _:_	CCP	2C	-2a 2b	Permanencia + Doble Cambio
96	14	b:b + a:b + _:_	PCP	1C	-1a 1b	Doble Permanencia + Cambio
97	8	b:b + b:b + _:_	PPP			Triple Permanencia
98	18	_:b + _:b + a:_	IIF	-1E 1P 3C	-1_-1a 2b	Doble Instanciación + Finalización
99	21	a:b + _:b + a:_	CIF	3C	-2a 2b	Instanciación + Cambio + Finalización
100	12	b:b + _:b + a:_	PIF	2C	-1a 1b	Permanencia + Instanciación + Finalización
101	24	a:b + a:b + a:_	CCF	1E-1P 3C	1_-3a 2b	Doble Cambio + Finalización
102	15	b:b + a:b + a:_	PCF	1E-1P 2C	1_-2a 1b	Permanencia + Cambio + Finalización
103	9	b:b + b:b + a:_	PPF	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
104	16	_:b + _:b + b:_	IIF	-1E 1P 3C	-1_ 1b	Doble Instanciación + Finalización
105	19	a:b + _:b + b:_	CIF	3C	-1a 1b	Instanciación + Cambio + Finalización
106	10	b:b + _:b + b:_	PIF	2C		Permanencia + Instanciación + Finalización
107	22	a:b + a:b + b:_	CCF	1E-1P 3C	1_-2a 1b	Doble Cambio + Finalización
108	13	b:b + a:b + b:_	PCF	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
109	7	b:b + b:b + b:_	PPF	1E-1P 1C	1_-1b	Doble Permanencia + Finalización
110	5	_:a + _:a + _:_	III	-3E 3P 3C	-3_ 3a	Triple Instanciación
111	3	a:a + _:a + _:_	PII	-2E 2P 2C	-2_ 2a	Permanencia + Doble Instanciación
112	6	b:a + _:a + _:_	CII	-2E 2P 3C	-2_ 3a-1b	Doble Instanciación + Cambio
113	2	a:a + a:a + _:_	PPI	-1E 1P 1C	-1_ 1a	Doble Permanencia + Instanciación
114	4	b:a + a:a + _:_	CPI	-1E 1P 2C	-1_ 2a-1b	Permanencia + Instanciación + Cambio
115	6	b:a + b:a + _:_	CCI	-1E 1P 3C	-1_ 3a-2b	Instanciación + Doble Cambio
116	1	a:a + a:a + a:_	PPP			Triple Permanencia
117	2	b:a + a:a + a:_	CPP	1C	1a-1b	Doble Permanencia + Cambio
118	3	b:a + b:a + a:_	CCP	2C	2a-2b	Permanencia + Doble Cambio
119	5	b:a + b:a + b:_	CCC	3C	3a-3b	Triple Cambio
120	24	_:b + _:a + _:_	III	-3E 3P 3C	-3_ 2a 1b	Triple Instanciación
121	22	a:b + _:a + _:_	CII	-2E 2P 3C	-2_ 1a 1b	Doble Instanciación + Cambio
122	23	b:b + _:a + _:_	PII	-2E 2P 2C	-2_ 2a	Permanencia + Doble Instanciación
123	15	_:b + a:a + _:_	IPI	-2E 2P 2C	-2_ 1a 1b	Permanencia + Doble Instanciación
124	13	a:b + a:a + _:_	CPI	-1E 1P 2C	-1_ 1b	Permanencia + Instanciación + Cambio
125	14	b:b + a:a + _:_	PPI	-1E 1P 1C	-1_ 1a	Doble Permanencia + Instanciación
126	21	_:b + b:a + _:_	ICI	-2E 2P 3C	-2_ 2a	Doble Instanciación + Cambio
127	19	a:b + b:a + _:_	CCI	-1E 1P 3C	-1_ 1a	Instanciación + Doble Cambio
128	20	b:b + b:a + _:_	PCI	-1E 1P 2C	-1_ 2a-1b	Permanencia + Instanciación + Cambio
129	9	_:b + a:a + a:_	IPP	-1E 1P 1C	-1_ 1b	Doble Permanencia + Instanciación
130	7	a:b + a:a + a:_	CPP	1C	-1a 1b	Doble Permanencia + Cambio
131	8	b:b + a:a + a:_	PPP			Triple Permanencia
132	12	_:b + b:a + a:_	ICP	-1E 1P 2C	-1_ 1a	Permanencia + Instanciación + Cambio

133	10	a:b + b:a + a:a	CCP	2C			Permanencia + Doble Cambio
134	11	b:b + b:a + a:a	PCP	1C	1a-1b		Doble Permanencia + Cambio
135	18	_:b + b:a + b:a	ICC	-1E 1P 3C	-1_	2a-1b	Instanciación + Doble Cambio
136	16	a:b + b:a + b:a	CCC	3C		1a-1b	Triple Cambio
137	17	b:b + b:a + b:a	PCC	2C		2a-2b	Permanencia + Doble Cambio
138	24	_:b + _:b + _:a	III	-3E 3P 3C	-3_	1a 2b	Triple Instanciación
139	21	a:b + _:b + _:a	CII	-2E 2P 3C	-2_	2b	Doble Instanciación + Cambio
140	15	b:b + _:b + _:a	PII	-2E 2P 2C	-2_	1a 1b	Permanencia + Doble Instanciación
141	18	a:b + a:b + _:a	CCI	-1E 1P 3C	-1_-1a	2b	Instanciación + Doble Cambio
142	12	b:b + a:b + _:a	PCI	-1E 1P 2C	-1_	1b	Permanencia + Instanciación + Cambio
143	9	b:b + b:b + _:a	PPI	-1E 1P 1C	-1_	1a	Doble Permanencia + Instanciación
144	23	_:b + _:b + a:a	IIP	-2E 2P 2C	-2_	2b	Permanencia + Doble Instanciación
145	20	a:b + _:b + a:a	CIP	-1E 1P 2C	-1_-1a	2b	Permanencia + Instanciación + Cambio
146	14	b:b + _:b + a:a	PIP	-1E 1P 1C	-1_	1b	Doble Permanencia + Instanciación
147	17	a:b + a:b + a:a	CCP	2C	-2a	2b	Permanencia + Doble Cambio
148	11	b:b + a:b + a:a	PCP	1C	-1a	1b	Doble Permanencia + Cambio
149	8	b:b + b:b + a:a	PPP				Triple Permanencia
150	22	_:b + _:b + b:a	IIC	-2E 2P 3C	-2_	1a 1b	Doble Instanciación + Cambio
151	19	a:b + _:b + b:a	CIC	-1E 1P 3C	-1_	1b	Instanciación + Doble Cambio
152	13	b:b + _:b + b:a	PIC	-1E 1P 2C	-1_	1a	Permanencia + Instanciación + Cambio
153	16	a:b + a:b + b:a	CCC	3C	-1a	1b	Triple Cambio
154	10	b:b + a:b + b:a	PCC	2C			Permanencia + Doble Cambio
155	7	b:b + b:b + b:a	PPC	1C	1a-1b		Doble Permanencia + Cambio
156	5	_:b + _:b + _:b	III	-3E 3P 3C	-3_	3b	Triple Instanciación
157	6	a:b + _:b + _:b	CII	-2E 2P 3C	-2_-1a	3b	Doble Instanciación + Cambio
158	3	b:b + _:b + _:b	PII	-2E 2P 2C	-2_	2b	Permanencia + Doble Instanciación
159	6	a:b + a:b + _:b	CCI	-1E 1P 3C	-1_-2a	3b	Instanciación + Doble Cambio
160	4	b:b + a:b + _:b	PCI	-1E 1P 2C	-1_-1a	2b	Permanencia + Instanciación + Cambio
161	2	b:b + b:b + _:b	PPI	-1E 1P 1C	-1_	1b	Doble Permanencia + Instanciación
162	5	a:b + a:b + a:b	CCC	3C	-3a	3b	Triple Cambio
163	3	b:b + a:b + a:b	PCC	2C	-2a	2b	Permanencia + Doble Cambio
164	2	b:b + b:b + a:b	PPC	1C	-1a	1b	Doble Permanencia + Cambio
165	1	b:b + b:b + b:b	PPP				Triple Permanencia

Se realizaron 729 combinaciones, existen 165 reglas independientes = permutaciones de primitivas diferentes

Existen 564 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos
0_ 0a 0b

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
0E 0P 2.00C = 66.67% Cambio 1.00I = 33.33% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
1.00P = 33.33% 0.67I = 22.22% 0.67C = 22.22% 0.67F = 22.22%

Ejemplos de aplicación: Existen 31 patrones de transición

1	_:_ + _:_ + _:_	[_,_,_,_,_,_,_]	PPP	P(_)	P(_)	P(_)	Triple Permanencia
2	a:_ + a:_ + a:_	[a,_,_,_,_,_,_]	FFP	F(a)	P(_)	P(_)	Doble Permanencia + Finalización
3	a:_ + a:_ + _:_	[a,a,_,_,_,_,_]	FFP	F(a)	F(a)	P(_)	Permanencia + Doble Finalización
4	b:_ + a:_ + _:_	[b,a,_,_,_,_,_]	FFP	F(b)	F(a)	P(_)	Permanencia + Doble Finalización
5	a:_ + a:_ + a:_	[a,a,a,_,_,_,_]	FFF	F(a)	F(a)	F(a)	Triple Finalización
6	b:_ + a:_ + a:_	[b,a,a,_,_,_,_]	FFF	F(b)	F(a)	F(a)	Triple Finalización
7	_:a + _:_ + _:_	[_,_,_,a,_,_,_]	IPP	I(a)	P(_)	P(_)	Doble Permanencia + Instanciación
8	a:a + _:_ + _:_	[a,_,_,a,_,_,_]	PPP	P(a)	P(_)	P(_)	Triple Permanencia
9	b:a + _:_ + _:_	[b,_,_,a,_,_,_]	CPP	C(b,a)	P(_)	P(_)	Doble Permanencia + Cambio
10	_:a + a:_ + _:_	[_,a,_,a,_,_,_]	IFP	I(a)	F(a)	P(_)	Permanencia + Instanciación + Finalización
11	a:a + a:_ + _:_	[a,a,_,a,_,_,_]	PFP	P(a)	F(a)	P(_)	Doble Permanencia + Finalización
12	b:a + a:_ + _:_	[b,a,_,a,_,_,_]	CFP	C(b,a)	F(a)	P(_)	Permanencia + Cambio + Finalización
13	_:a + b:_ + _:_	[_,b,_,a,_,_,_]	IFP	I(a)	F(b)	P(_)	Permanencia + Instanciación + Finalización
14	a:a + b:_ + _:_	[a,b,_,a,_,_,_]	PFP	P(a)	F(b)	P(_)	Doble Permanencia + Finalización
15	b:a + b:_ + _:_	[b,b,_,a,_,_,_]	CFP	C(b,a)	F(b)	P(_)	Permanencia + Cambio + Finalización
16	_:a + a:_ + a:_	[_,a,a,a,_,_,_]	IFF	I(a)	F(a)	F(a)	Instanciación + Doble Finalización
17	a:a + a:_ + a:_	[a,a,a,a,_,_,_]	PFF	P(a)	F(a)	F(a)	Permanencia + Doble Finalización
18	b:a + a:_ + a:_	[b,a,a,a,_,_,_]	CFP	C(b,a)	F(a)	F(a)	Cambio + Doble Finalización
19	_:a + b:_ + a:_	[_,b,a,a,_,_,_]	IFF	I(a)	F(b)	F(a)	Instanciación + Doble Finalización
20	a:a + b:_ + a:_	[a,b,a,a,_,_,_]	PFF	P(a)	F(b)	F(a)	Permanencia + Doble Finalización
21	b:a + b:_ + a:_	[b,b,a,a,_,_,_]	CFP	C(b,a)	F(b)	F(a)	Cambio + Doble Finalización
22	_:a + b:_ + b:_	[_,b,b,a,_,_,_]	IFF	I(a)	F(b)	F(b)	Instanciación + Doble Finalización
23	a:a + b:_ + b:_	[a,b,b,a,_,_,_]	PFF	P(a)	F(b)	F(b)	Permanencia + Doble Finalización
24	b:a + b:_ + b:_	[b,b,b,a,_,_,_]	CFP	C(b,a)	F(b)	F(b)	Cambio + Doble Finalización
25	_:b + _:a + _:_	[_,_,_,b,a,_,_]	IIP	I(b)	I(a)	P(_)	Permanencia + Doble Instanciación
26	a:b + _:a + _:_	[a,_,_,b,a,_,_]	CIP	C(a,b)	I(a)	P(_)	Permanencia + Instanciación + Cambio

27	b:b + _:a + _:_	[b,_,_:b,a,_]	PIP	P(b)	I(a)	P(_)	Doble Permanencia + Instanciación
28	b:b + a:a + _:_	[b,a,_:b,a,_]	PPP	P(b)	P(a)	P(_)	Triple Permanencia
29	a:b + b:a + _:_	[a,b,_:b,a,_]	CCP	C(a,b)	C(b,a)	P(_)	Permanencia + Doble Cambio
30	_:b + _:a + a:_	[_,_,a:b,a,_]	IIF	I(b)	I(a)	F(a)	Doble Instanciación + Finalización
31	_:b + b:a + a:_	[_,b,a:b,a,_]	ICF	I(b)	C(b,a)	F(a)	Instanciación + Cambio + Finalización

Tabla de patrones de transición de SPRT(3S,3D)

1	A:A + A:A + A:A	3	1	
2	A:A + A:A + B:A	18	2	+1A -1B
3	A:A + B:A + B:A	18	2	+2A -2B
4	A:A + B:A + C:A	18	3	+2A -1B -1C
5	A:B + A:B + A:B	6	2	-3A +3B
6	A:B + A:B + C:B	18	3	-2A +3B -1C
7	A:A + A:A + A:B	18	2	-1A +1B
8	A:A + A:A + B:B	18	2	
9	A:A + A:A + B:C	18	3	-1B +1C
10	A:A + A:B + B:A	36	2	
11	A:A + A:B + B:B	36	2	-1A +1B
12	A:A + B:A + C:B	36	3	+1A -1C
13	A:A + A:B + C:A	36	3	+1B -1C
14	A:A + B:A + C:C	36	3	+1A -1B
15	A:A + B:A + B:C	36	3	+1A -2B +1C
16	A:B + A:B + B:A	18	2	-1A +1B
17	A:A + A:B + A:B	18	2	-2A +2B
18	A:B + A:B + C:A	18	3	-1A +2B -1C
19	A:B + B:A + C:A	36	3	+1A -1C
20	A:A + A:B + C:B	36	3	-1A +2B -1C
21	A:B + A:C + B:C	36	3	-2A +2C
22	A:B + A:B + B:C	18	3	-2A +1B +1C
23	A:A + B:C + B:C	18	3	-2B +2C
24	A:B + A:B + A:C	18	3	-3A +2B +1C
25	A:A + A:B + A:C	18	3	-2A +1B +1C
26	A:A + A:B + B:C	36	3	-1A +1C
27	A:A + A:B + C:C	36	3	-1A +1B
28	A:A + B:B + C:C	6	3	
29	A:A + B:C + C:B	18	3	
30	A:B + A:C + B:A	36	3	-1A +1C
31	A:B + B:C + C:A	12	3	

Existen en promedio 2.7366255144033 símbolos distintos en cada regla del SPRT

E.2.4.6. Ejemplo de SPRT de símbolos incompleto: un 4S2D particular

El script GSPRT también puede generar SPRT incompletos, aquellos en los cuales existen restricciones de paso entre los elementos de la tabla de símbolos del sistema. Para configurar uno de estos sistemas en el script, solo hay que establecer las diferencias entre los símbolos a usar en la variable \$z y establecer el valor de dimensión de evaluación deseado. La cantidad de elementos en \$z es siempre par y la dimensión será igual a la mitad del conteo, $\text{count}(\$z)/2$. Se presenta un ejemplo de configuración del Script GSPRT para un SPRT incompleto:

Script E.13: Configuración de GSPRT para un SPRT 4S2D incompleto

```
1 ...
2 //Definición de Tabla de símbolos
3 $o = array('a','b');
4 $p1= array('a','b','c');
5 $p2= array('a','b','d');
6
7 //Configuración del vecindario / dimensiones / simultaneidad
8 //$o:$p1 + $o:$p2 = $o,$o:$p1,$p2
9 //$z es la tabla de combinaciones del sistema
10 $z=array($o,$o,$p1,$p2); //construcción manual en notación de permutación / orden de primitivas
11 $vecindad=count($z)/2; //se requiere calcular la vecindad o dimensión del sistema
12 ...
```

En el ejemplo se configura un SPRT en el cual no hay paso total entre símbolos: los identificadores A y B pueden pasar a ser C y D pero no existen primitivas para retornar C y D a A y B. El sistema es asimétrico, hay diferencias en la cantidad e identidad de los símbolos presentes entre las causas y efectos que describen la tabla de combinaciones del sistema.

Al analizar el SPRT, se obtiene que no hay variación poblacional ni espacial ya que no se define símbolo de celda vacía, todos los símbolos A y B pasan a C o D según corresponda. La tendencia del sistema particular es de remplazo mediante la primitiva de cambio con un 66.67% del total de las operaciones disponibles, esto será así hasta que no existan más A y B en el sistema. El balance global de símbolos es negativo para A y B, positivo para C y D, ya que a largo plazo todas las A y B presentes pasarán a ser C y D.

De la tabla E.1 se observa que una disminución en el número de primitivas de transición reduce la expresividad general del SPRT.¹⁹ El efecto de la cantidad de tipos distintos de símbolos disponibles con los cuales se construyen las primitivas es evidente: Si 4S2D particular usara solo un tipo de símbolo, únicamente sería viable el patrón con #S = 1 y las reglas generables por ese patrón. Para este SPRT existe un máximo de $2D = 2 * 2 = 4$ escalas de alcance de representación de patrones, todas dependientes de las diferencias entre los símbolos usados. Más allá de cuatro símbolos distintos la cantidad de patrones se hace constante, solo es posible discriminar 11 formas de patrones. Esto coincide con los resultados obtenidos en la tabla D.22 en la página 469 del anexo D. Las primitivas o la capacidad disponible de enlace causa / efecto en el sistema de transiciones, establecen los límites de la variedad representacional de diferencias entre símbolos.

¹⁹Todos los identificadores representan elementos distintos entre si. La columna *Uso* contiene el conteo de implementaciones de cada patrón en el sistema y #S cuenta la cantidad de símbolos distintos requeridos en el patrón actual. La sumatoria de valores de *Uso* es igual al total de reglas de transición presentes en el sistema.

Tabla E.1: Efecto de selección de primitivas en un SPRT

SPRT	4S2D general				4S2D particular		
Símbolos = {a,b,c,d}	#S = 4				#S = 4		
Primitivas	#P = 4 ² = 16				#P = 8		
por comprensión dimensional	D ₁	{a,b,c,d:a,b,c,d}			D ₁	{a,b:a,b,c}	
	D ₂	{a,b,c,d:a,b,c,d}			D ₂	{a,b:a,b,d}	
por extensión	a:a a:b a:c a:d				a:a a:b a:c		
	b:a b:b b:c b:d				b:a b:b b:c en D ₁		
	c:a c:b c:c c:d				y		
	d:a d:b d:c d:d				a:a a:b a:d		
	Para D ₁ y D ₂				b:a b:b b:d en D ₂		
Patrones por extensión	Id	Patrón	Uso	#S	Id	Patrón	Uso
	1	A:A + A:A	4	1	1	A:A + A:A	2
	2	A:A + B:A	24	2	2	A:A + B:A	4
	3	A:B + A:B	12	2	3	A:B + A:B	2
	4	A:B + C:B	24	3			
	5	A:A + A:B	24	2	4	A:A + A:B	8
	6	A:A + B:B	12	2	5	A:A + B:B	2
	7	A:A + B:C	48	3	6	A:A + B:C	4
	8	A:B + B:A	12	2	7	A:B + B:A	2
	9	A:B + B:C	48	3	8	A:B + B:C	4
	10	A:B + A:C	24	3	9	A:B + A:C	6
	11	A:B + C:D	24	4	10	A:B + C:D	2
Reglas independientes	136				30		
Total de reglas disponibles	256				36		

Fuente: Autor.

Con respecto a los patrones de transición los sistemas comparados difieren solo en el patrón A:B + C:B, ausente en el 4S2D particular debido a que con el conjunto particular de primitivas establecido²⁰ no es posible construir una regla que describa la relación entre dos símbolos-causas distintos que produzcan un efecto resultante igual pero simultáneamente diferente a las causas relacionadas, p.ej:

$$a:b + d:b = [a,d:b,b]$$

$$b:a + c:a = [b,c:a,a]$$

Por completitud, se incorporan al documento en orden de aparición el SPRT 4S2D incompleto + completo para facilitar el contraste y comparación entre ambos sistemas.

²⁰Aunque las primitivas son a:a, a:b, a:c, a:d y b:a, b:b, b:c, b:d con #P = 8, ellas no aplican simultáneamente en todas las dimensiones de evaluación disponibles.

Reporte E.8: SPRT 4S2D con selección de primitivas

SPRT(4S,2D)

Tabla de símbolos = {a,b,c,d} = 4 símbolos

Dimensiones, simultaneidad, vecindario máximo = 2 = Se combinarán 4 vectores
elementos en cada vector [2 2 3 3] = 36 combinaciones

Este sistema implementa selección de primitivas

Se usan 8 primitivas de transición

a:a a:b a:c b:a b:b b:c a:d b:d

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa	->	Salida/Efecto	Notación
Principal	Celda[0]	{a,b}	->	{a,b,c}	{a,b:a,b,c}
Vecina 1	Celda[1]	{a,b}	->	{a,b,d}	{a,b:a,b,d}

1	1	a:a + a:a	[a,a:a,a]	PP	P(a)	P(a)				Doble Permanencia
2	2	b:a + a:a	[b,a:a,a]	CP	C(b,a)	P(a)	1C	1a-1b		Permanencia + Cambio
3	3	b:a + b:a	[b,b:a,a]	CC	C(b,a)	C(b,a)	2C	2a-2b		Doble Cambio
4	4	a:b + a:a	[a,a:b,a]	CP	C(a,b)	P(a)	1C	-1a 1b		Permanencia + Cambio
5	5	b:b + a:a	[b,a:b,a]	PP	P(b)	P(a)				Doble Permanencia
6	6	a:b + b:a	[a,b:b,a]	CC	C(a,b)	C(b,a)	2C			Doble Cambio
7	4	b:b + b:a	[b,b:b,a]	PC	P(b)	C(b,a)	1C	1a-1b		Permanencia + Cambio
8	4	a:c + a:a	[a,a:c,a]	CP	C(a,c)	P(a)	1C	-1a 1c		Permanencia + Cambio
9	7	b:c + a:a	[b,a:c,a]	CP	C(b,c)	P(a)	1C	-1b 1c		Permanencia + Cambio
10	8	a:c + b:a	[a,b:c,a]	CC	C(a,c)	C(b,a)	2C	-1b 1c		Doble Cambio
11	9	b:c + b:a	[b,b:c,a]	CC	C(b,c)	C(b,a)	2C	1a-2b 1c		Doble Cambio
12	3	a:b + a:b	[a,a:b,b]	CC	C(a,b)	C(a,b)	2C	-2a 2b		Doble Cambio
13	2	b:b + a:b	[b,a:b,b]	PC	P(b)	C(a,b)	1C	-1a 1b		Permanencia + Cambio
14	1	b:b + b:b	[b,b:b,b]	PP	P(b)	P(b)				Doble Permanencia
15	9	a:c + a:b	[a,a:c,b]	CC	C(a,c)	C(a,b)	2C	-2a 1b 1c		Doble Cambio
16	8	b:c + a:b	[b,a:c,b]	CC	C(b,c)	C(a,b)	2C	-1a 1c		Doble Cambio
17	7	a:c + b:b	[a,b:c,b]	CP	C(a,c)	P(b)	1C	-1a 1c		Permanencia + Cambio
18	4	b:c + b:b	[b,b:c,b]	CP	C(b,c)	P(b)	1C	-1b 1c		Permanencia + Cambio
19	4	a:a + a:d	[a,a:a,d]	PC	P(a)	C(a,d)	1C	-1a	1d	Permanencia + Cambio
20	8	b:a + a:d	[b,a:a,d]	CC	C(b,a)	C(a,d)	2C	-1b	1d	Doble Cambio
21	7	a:a + b:d	[a,b:a,d]	PC	P(a)	C(b,d)	1C	-1b	1d	Permanencia + Cambio
22	9	b:a + b:d	[b,b:a,d]	CC	C(b,a)	C(b,d)	2C	1a-2b	1d	Doble Cambio
23	9	a:b + a:d	[a,a:b,d]	CC	C(a,b)	C(a,d)	2C	-2a 1b	1d	Doble Cambio
24	7	b:b + a:d	[b,a:b,d]	PC	P(b)	C(a,d)	1C	-1a	1d	Permanencia + Cambio
25	8	a:b + b:d	[a,b:b,d]	CC	C(a,b)	C(b,d)	2C	-1a	1d	Doble Cambio
26	4	b:b + b:d	[b,b:b,d]	PC	P(b)	C(b,d)	1C	-1b	1d	Permanencia + Cambio
27	9	a:c + a:d	[a,a:c,d]	CC	C(a,c)	C(a,d)	2C	-2a 1c	1d	Doble Cambio
28	10	b:c + a:d	[b,a:c,d]	CC	C(b,c)	C(a,d)	2C	-1a-1b 1c	1d	Doble Cambio
29	10	a:c + b:d	[a,b:c,d]	CC	C(a,c)	C(b,d)	2C	-1a-1b 1c	1d	Doble Cambio
30	9	b:c + b:d	[b,b:c,d]	CC	C(b,c)	C(b,d)	2C	-2b 1c	1d	Doble Cambio

Se realizaron 36 combinaciones, existen 30 reglas independientes = permutaciones de primitivas diferentes

Existen 6 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos

-12a -12b 12c 12d

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)

OE OP 1.33C = 66.67% Cambio 0.67I = 33.33% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización

0.67P = 33.33% 0.00I = 0.00% 1.33C = 66.67% 0.00F = 0.00%

Ejemplos de aplicación: Existen 10 patrones de transición

1	a:a + a:a	[a,a:a,a]	PP	P(a)	P(a)	Doble Permanencia
2	b:a + a:a	[b,a:a,a]	CP	C(b,a)	P(a)	Permanencia + Cambio
3	b:a + b:a	[b,b:a,a]	CC	C(b,a)	C(b,a)	Doble Cambio
4	a:b + a:a	[a,a:b,a]	CP	C(a,b)	P(a)	Permanencia + Cambio
5	b:b + a:a	[b,a:b,a]	PP	P(b)	P(a)	Doble Permanencia
6	a:b + b:a	[a,b:b,a]	CC	C(a,b)	C(b,a)	Doble Cambio
7	b:c + a:a	[b,a:c,a]	CP	C(b,c)	P(a)	Permanencia + Cambio
8	a:c + b:a	[a,b:c,a]	CC	C(a,c)	C(b,a)	Doble Cambio
9	b:c + b:a	[b,b:c,a]	CC	C(b,c)	C(b,a)	Doble Cambio
10	b:c + a:d	[b,a:c,d]	CC	C(b,c)	C(a,d)	Doble Cambio

Tabla de patrones de transición de SPRT(4S,2D) particular de 8 primitivas

1	A:A + A:A	2	1	
2	A:A + B:A	4	2	+1A -1B
3	A:B + A:B	2	2	-2A +2B
4	A:A + A:B	8	2	-1A +1B

```

5  A:A + B:B  2  2
6  A:B + B:A  2  2
7  A:A + B:C  4  3  -1B +1C
8  A:B + B:C  4  3  -1A +1C
9  A:B + A:C  6  3  -2A +1B +1C
10 A:B + C:D  2  4  -1A +1B -1C +1D
Existen en promedio 2.4444444444444444 símbolos distintos en cada regla del SPRT

```


54	7	c:c + b:_	[c,b:c,_]	PF	P(c)	F(b)	1E-1P	1C	1_	-1b	Permanencia + Finalización
55	8	_:c + c:_	[_:c:c,_]	IF	I(c)	F(c)		2C			Instanciación + Finalización
56	9	a:c + c:_	[a,c:c,_]	CF	C(a,c)	F(c)	1E-1P	2C	1_-1a		Cambio + Finalización
57	9	b:c + c:_	[b,c:c,_]	CF	C(b,c)	F(c)	1E-1P	2C	1_	-1b	Cambio + Finalización
58	5	c:c + c:_	[c,c:c,_]	PF	P(c)	F(c)	1E-1P	1C	1_	-1c	Permanencia + Finalización
59	3	_:a + _:a	[_:_:a,a]	II	I(a)	I(a)	-2E	2P	2C	-2_ 2a	Doble Instanciación
60	2	a:a + _:a	[a,_:a,a]	PI	P(a)	I(a)	-1E	1P	1C	-1_ 1a	Permanencia + Instanciación
61	4	b:a + _:a	[b,_:a,a]	CI	C(b,a)	I(a)	-1E	1P	2C	-1_ 2a-1b	Instanciación + Cambio
62	4	c:a + _:a	[c,_:a,a]	CI	C(c,a)	I(a)	-1E	1P	2C	-1_ 2a	-1c Instanciación + Cambio
63	1	a:a + a:a	[a,a:a,a]	PP	P(a)	P(a)					Doble Permanencia
64	2	b:a + a:a	[b,a:a,a]	CP	C(b,a)	P(a)		1C		1a-1b	Permanencia + Cambio
65	2	c:a + a:a	[c,a:a,a]	CP	C(c,a)	P(a)		1C		1a	-1c Permanencia + Cambio
66	3	b:a + b:a	[b,b:a,a]	CC	C(b,a)	C(b,a)		2C		2a-2b	Doble Cambio
67	4	c:a + b:a	[c,b:a,a]	CC	C(c,a)	C(b,a)		2C		2a-1b-1c	Doble Cambio
68	3	c:a + c:a	[c,c:a,a]	CC	C(c,a)	C(c,a)		2C		2a	-2c Doble Cambio
69	10	_:b + _:a	[_:_:b,a]	II	I(b)	I(a)	-2E	2P	2C	-2_ 1a 1b	Doble Instanciación
70	9	a:b + _:a	[a,_:b,a]	CI	C(a,b)	I(a)	-1E	1P	2C	-1_ 1b	Instanciación + Cambio
71	7	b:b + _:a	[b,_:b,a]	PI	P(b)	I(a)	-1E	1P	1C	-1_ 1a	Permanencia + Instanciación
72	11	c:b + _:a	[c,_:b,a]	CI	C(c,b)	I(a)	-1E	1P	2C	-1_ 1a 1b-1c	Instanciación + Cambio
73	7	_:b + a:a	[_:a:b,a]	IP	I(b)	P(a)	-1E	1P	1C	-1_ 1b	Permanencia + Instanciación
74	5	a:b + a:a	[a,a:b,a]	CP	C(a,b)	P(a)		1C		-1a 1b	Permanencia + Cambio
75	6	b:b + a:a	[b,a:b,a]	PP	P(b)	P(a)					Doble Permanencia
76	7	c:b + a:a	[c,a:b,a]	CP	C(c,b)	P(a)		1C		1b-1c	Permanencia + Cambio
77	9	_:b + b:a	[_:b:b,a]	IC	I(b)	C(b,a)	-1E	1P	2C	-1_ 1a	Instanciación + Cambio
78	8	a:b + b:a	[a,b:b,a]	CC	C(a,b)	C(b,a)		2C			Doble Cambio
79	5	b:b + b:a	[b,b:b,a]	PC	P(b)	C(b,a)		1C		1a-1b	Permanencia + Cambio
80	9	c:b + b:a	[c,b:b,a]	CC	C(c,b)	C(b,a)		2C		1a	-1c Doble Cambio
81	11	_:b + c:a	[_:c:b,a]	IC	I(b)	C(c,a)	-1E	1P	2C	-1_ 1a 1b-1c	Instanciación + Cambio
82	9	a:b + c:a	[a,c:b,a]	CC	C(a,b)	C(c,a)		2C		1b-1c	Doble Cambio
83	7	b:b + c:a	[b,c:b,a]	PC	P(b)	C(c,a)		1C		1a	-1c Permanencia + Cambio
84	10	c:b + c:a	[c,c:b,a]	CC	C(c,b)	C(c,a)		2C		1a 1b-2c	Doble Cambio
85	10	_:c + _:a	[_:_:c,a]	II	I(c)	I(a)	-2E	2P	2C	-2_ 1a	1c Doble Instanciación
86	9	a:c + _:a	[a,_:c,a]	CI	C(a,c)	I(a)	-1E	1P	2C	-1_ 1c	Instanciación + Cambio
87	11	b:c + _:a	[b,_:c,a]	CI	C(b,c)	I(a)	-1E	1P	2C	-1_ 1a-1b 1c	Instanciación + Cambio
88	7	c:c + _:a	[c,_:c,a]	PI	P(c)	I(a)	-1E	1P	1C	-1_ 1a	Permanencia + Instanciación
89	7	_:c + a:a	[_:a:c,a]	IP	I(c)	P(a)	-1E	1P	1C	-1_ 1c	Permanencia + Instanciación
90	5	a:c + a:a	[a,a:c,a]	CP	C(a,c)	P(a)		1C		-1a 1c	Permanencia + Cambio
91	7	b:c + a:a	[b,a:c,a]	CP	C(b,c)	P(a)		1C		-1b 1c	Permanencia + Cambio
92	6	c:c + a:a	[c,a:c,a]	PP	P(c)	P(a)					Doble Permanencia
93	11	_:c + b:a	[_:b:c,a]	IC	I(c)	C(b,a)	-1E	1P	2C	-1_ 1a-1b 1c	Instanciación + Cambio
94	9	a:c + b:a	[a,b:c,a]	CC	C(a,c)	C(b,a)		2C		-1b 1c	Doble Cambio
95	10	b:c + b:a	[b,b:c,a]	CC	C(b,c)	C(b,a)		2C		1a-2b 1c	Doble Cambio
96	7	c:c + b:a	[c,b:c,a]	PC	P(c)	C(b,a)		1C		1a-1b	Permanencia + Cambio
97	9	_:c + c:a	[_:c:c,a]	IC	I(c)	C(c,a)	-1E	1P	2C	-1_ 1a	Instanciación + Cambio
98	8	a:c + c:a	[a,c:c,a]	CC	C(a,c)	C(c,a)		2C			Doble Cambio
99	9	b:c + c:a	[b,c:c,a]	CC	C(b,c)	C(c,a)		2C		1a-1b	Doble Cambio
100	5	c:c + c:a	[c,c:c,a]	PC	P(c)	C(c,a)		1C		1a	-1c Permanencia + Cambio
101	3	_:b + _:b	[_:_:b,b]	II	I(b)	I(b)	-2E	2P	2C	-2_ 2b	Doble Instanciación
102	4	a:b + _:b	[a,_:b,b]	CI	C(a,b)	I(b)	-1E	1P	2C	-1_ -1a 2b	Instanciación + Cambio
103	2	b:b + _:b	[b,_:b,b]	PI	P(b)	I(b)	-1E	1P	1C	-1_ 1b	Permanencia + Instanciación
104	4	c:b + _:b	[c,_:b,b]	CI	C(c,b)	I(b)	-1E	1P	2C	-1_ 2b-1c	Instanciación + Cambio
105	3	a:b + a:b	[a,a:b,b]	CC	C(a,b)	C(a,b)		2C		-2a 2b	Doble Cambio
106	2	b:b + a:b	[b,a:b,b]	PC	P(b)	C(a,b)		1C		-1a 1b	Permanencia + Cambio
107	4	c:b + a:b	[c,a:b,b]	CC	C(c,b)	C(a,b)		2C		-1a 2b-1c	Doble Cambio
108	1	b:b + b:b	[b,b:b,b]	PP	P(b)	P(b)					Doble Permanencia
109	2	c:b + b:b	[c,b:b,b]	CP	C(c,b)	P(b)		1C		1b-1c	Permanencia + Cambio
110	3	c:b + c:b	[c,c:b,b]	CC	C(c,b)	C(c,b)		2C		2b-2c	Doble Cambio
111	10	_:c + _:b	[_:_:c,b]	II	I(c)	I(b)	-2E	2P	2C	-2_ 1b 1c	Doble Instanciación
112	11	a:c + _:b	[a,_:c,b]	CI	C(a,c)	I(b)	-1E	1P	2C	-1_ -1a 1b 1c	Instanciación + Cambio
113	9	b:c + _:b	[b,_:c,b]	CI	C(b,c)	I(b)	-1E	1P	2C	-1_ 1c	Instanciación + Cambio
114	7	c:c + _:b	[c,_:c,b]	PI	P(c)	I(b)	-1E	1P	1C	-1_ 1b	Permanencia + Instanciación
115	11	_:c + a:b	[_:a:c,b]	IC	I(c)	C(a,b)	-1E	1P	2C	-1_ -1a 1b 1c	Instanciación + Cambio
116	10	a:c + a:b	[a,a:c,b]	CC	C(a,c)	C(a,b)		2C		-2a 1b 1c	Doble Cambio
117	9	b:c + a:b	[b,a:c,b]	CC	C(b,c)	C(a,b)		2C		-1a 1c	Doble Cambio
118	7	c:c + a:b	[c,a:c,b]	PC	P(c)	C(a,b)		1C		-1a 1b	Permanencia + Cambio
119	7	_:c + b:b	[_:b:c,b]	IP	I(c)	P(b)	-1E	1P	1C	-1_ 1c	Permanencia + Instanciación
120	7	a:c + b:b	[a,b:c,b]	CP	C(a,c)	P(b)		1C		-1a 1c	Permanencia + Cambio
121	5	b:c + b:b	[b,b:c,b]	CP	C(b,c)	P(b)		1C		-1b 1c	Permanencia + Cambio
122	6	c:c + b:b	[c,b:c,b]	PP	P(c)	P(b)					Doble Permanencia
123	9	_:c + c:b	[_:c:c,b]	IC	I(c)	C(c,b)	-1E	1P	2C	-1_ 1b	Instanciación + Cambio
124	9	a:c + c:b	[a,c:c,b]	CC	C(a,c)	C(c,b)		2C		-1a 1b	Doble Cambio
125	8	b:c + c:b	[b,c:c,b]	CC	C(b,c)	C(c,b)		2C			Doble Cambio
126	5	c:c + c:b	[c,c:c,b]	PC	P(c)	C(c,b)		1C		1b-1c	Permanencia + Cambio
127	3	_:c + _:c	[_:_:c,c]	II	I(c)	I(c)	-2E	2P	2C	-2_ 2c	Doble Instanciación
128	4	a:c + _:c	[a,_:c,c]	CI	C(a,c)	I(c)	-1E	1P	2C	-1_ -1a 2c	Instanciación + Cambio
129	4	b:c + _:c	[b,_:c,c]	CI	C(b,c)	I(c)	-1E	1P	2C	-1_ -1b 2c	Instanciación + Cambio
130	2	c:c + _:c	[c,_:c,c]	PI	P(c)	I(c)	-1E	1P	1C	-1_ 1c	Permanencia + Instanciación

```

131 3   a:c + a:c [a,a:c,c] CC C(a,c) C(a,c)      2C   -2a   2c Doble Cambio
132 4   b:c + a:c [b,a:c,c] CC C(b,c) C(a,c)      2C   -1a-1b 2c Doble Cambio
133 2   c:c + a:c [c,a:c,c] PC P(c)   C(a,c)      1C   -1a   1c Permanencia + Cambio
134 3   b:c + b:c [b,b:c,c] CC C(b,c) C(b,c)      2C       -2b 2c Doble Cambio
135 2   c:c + b:c [c,b:c,c] PC P(c)   C(b,c)      1C       -1b 1c Permanencia + Cambio
136 1   c:c + c:c [c,c:c,c] PP P(c)   P(c)                Doble Permanencia
Se realizaron 256 combinaciones, existen 136 reglas independientes = permutaciones de primitivas
diferentes

```

Existen 120 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos

0_ 0a 0b 0c

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)

0E 0P 1.50C = 75.00% Cambio 0.50I = 25.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización

0.50P = 25.00% 0.38I = 18.75% 0.75C = 37.50% 0.38F = 18.75%

Ejemplos de aplicación: Existen 11 patrones de transición

```

1  _:_ + _:_ [_:_:_] PP P(_) P(_) Doble Permanencia
2  a:_ + _:_ [a,_:_] FP F(a) P(_) Permanencia + Finalización
3  a:_ + a:_ [a,a:_] FF F(a) F(a) Doble Finalización
4  b:_ + a:_ [b,a:_] FF F(b) F(a) Doble Finalización
5  _:a + _:_ [_:_:a] IP I(a) P(_) Permanencia + Instanciación
6  a:a + _:_ [a,_:a] PP P(a) P(_) Doble Permanencia
7  b:a + _:_ [b,_:a] CP C(b,a) P(_) Permanencia + Cambio
8  _:a + a:_ [_:a:a] IF I(a) F(a) Instanciación + Finalización
9  b:a + a:_ [b,a:a] CF C(b,a) F(a) Cambio + Finalización
10 b:a + b:_ [b,b:a] CF C(b,a) F(b) Cambio + Finalización
11 c:a + b:_ [c,b:a] CF C(c,a) F(b) Cambio + Finalización

```

Tabla de patrones de transición de SPRT(4S,2D)

```

1  A:A + A:A 4 1
2  A:A + B:A 24 2 +1A -1B
3  A:B + A:B 12 2 -2A +2B
4  A:B + C:B 24 3 -1A +2B -1C
5  A:A + A:B 24 2 -1A +1B
6  A:A + B:B 12 2
7  A:A + B:C 48 3 -1B +1C
8  A:B + B:A 12 2
9  A:B + B:C 48 3 -1A +1C
10 A:B + A:C 24 3 -2A +1B +1C
11 A:B + C:D 24 4 -1A +1B -1C +1D

```

Existen en promedio 2.734375 símbolos distintos en cada regla del SPRT

E.2.4.8. SPRT 4S3D

El listado E.10 contiene la descripción del menor SPRT que comprende estructuralmente a Homos 1.0. El sistema se implementa mediante tres dimensiones de evaluación y hace uso de cuatro símbolos, el vacío y otros tres identificadores distintos entre si. Estos son los requisitos mínimos necesarios para formalizar el sistema de transiciones descrito en la MBOR y las variantes presentes en la herramienta software.

Observando el sistema construido, la cantidad real de transiciones que es posible representar con las dimensiones y diferencias entre símbolos requeridos es claramente superior a lo expuesto por la MBOR. El SPRT 4S3D incluye todos los sistemas descritos por $nSmD$ para $n \leq 4$ y $m \leq 3$. Para ver el aporte total del 4S3D, comparar los valores obtenidos en el listado E.10 con las descripciones de transición de la MBOR de la tabla D.37 en la página 494 y las descripciones calculadas para los $nSmD$ incorporados en este documento.

Figura E.4: Sección de la Matriz SPRT comprendida por 4S3D

$$M_{4S3D} = \begin{pmatrix} 1S1D & 1S2D & 1S3D \\ 2S1D & 2S2D & 2S3D \\ 3S1D & 3S2D & 3S3D \\ 4S1D & 4S2D & 4S3D \end{pmatrix}$$

El SPRT 4S3D contiene por comprensión a los demás sistemas de menos símbolos y dimensiones mostrados.

Fuente: Autor.

Los resultados presentados han sido editados:

- Se eliminaron las columnas identificador de acción de primitivas y notación de permutación de primitivas en la tabla de reglas independientes.
- No se presentó la lista de reglas repetidas del sistema.

Los cambios realizados obedecen al propósito de facilitar la inclusión y presentación de los resultados obtenidos por el script. El SPRT 4S3D inalterado está en `contenido/anexos/scripts/SPRT/nSmD/4s3d.txt`

Reporte E.10: SPRT 4S3D

SPRT(4S,3D)

Tabla de símbolos = {_,a,b,c} = 4 símbolos

Dimensiones, simultaneidad, vecindario máximo = 3 = Se combinarán 6 vectores

elementos en cada vector [4 4 4 4 4] = 4096 combinaciones

Existen 816 reglas independientes

Se usan 16 primitivas de transición

_: _	_: a	_: b	_: c	a: _	a: a	a: b	a: c
b: _	b: a	b: b	b: c	c: _	c: a	c: b	c: c

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa	->	Salida/Efecto	Notación
Principal	Celda[0]	{_,a,b,c}	->	{_,a,b,c}	{_,a,b,c:_,a,b,c}
Vecina 1	Celda[1]	{_,a,b,c}	->	{_,a,b,c}	{_,a,b,c:_,a,b,c}
Vecina 2	Celda[2]	{_,a,b,c}	->	{_,a,b,c}	{_,a,b,c:_,a,b,c}

1	1	_: _ + _: _ + _: _	PPP			Triple Permanencia
2	2	a: _ + _: _ + _: _	FPP	1E-1P 1C	1_ -1a	Doble Permanencia + Finalización
3	2	b: _ + _: _ + _: _	FPP	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
4	2	c: _ + _: _ + _: _	FPP	1E-1P 1C	1_ -1c	Doble Permanencia + Finalización
5	3	a: _ + a: _ + _: _	FFP	2E-2P 2C	2_ -2a	Permanencia + Doble Finalización
6	4	b: _ + a: _ + _: _	FFP	2E-2P 2C	2_ -1a-1b	Permanencia + Doble Finalización
7	4	c: _ + a: _ + _: _	FFP	2E-2P 2C	2_ -1a -1c	Permanencia + Doble Finalización
8	3	b: _ + b: _ + _: _	FFP	2E-2P 2C	2_ -2b	Permanencia + Doble Finalización
9	4	c: _ + b: _ + _: _	FFP	2E-2P 2C	2_ -1b-1c	Permanencia + Doble Finalización
10	3	c: _ + c: _ + _: _	FFP	2E-2P 2C	2_ -2c	Permanencia + Doble Finalización
11	5	a: _ + a: _ + a: _	FFF	3E-3P 3C	3_ -3a	Triple Finalización
12	6	b: _ + a: _ + a: _	FFF	3E-3P 3C	3_ -2a-1b	Triple Finalización
13	6	c: _ + a: _ + a: _	FFF	3E-3P 3C	3_ -2a -1c	Triple Finalización
14	6	b: _ + b: _ + a: _	FFF	3E-3P 3C	3_ -1a-2b	Triple Finalización
15	7	c: _ + b: _ + a: _	FFF	3E-3P 3C	3_ -1a-1b-1c	Triple Finalización
16	6	c: _ + c: _ + a: _	FFF	3E-3P 3C	3_ -1a -2c	Triple Finalización
17	5	b: _ + b: _ + b: _	FFF	3E-3P 3C	3_ -3b	Triple Finalización
18	6	c: _ + b: _ + b: _	FFF	3E-3P 3C	3_ -2b-1c	Triple Finalización
19	6	c: _ + c: _ + b: _	FFF	3E-3P 3C	3_ -1b-2c	Triple Finalización
20	5	c: _ + c: _ + c: _	FFF	3E-3P 3C	3_ -3c	Triple Finalización
21	8	_: a + _: _ + _: _	IPP	-1E 1P 1C	-1_ 1a	Doble Permanencia + Instanciación
22	9	a: a + _: _ + _: _	PPP			Triple Permanencia
23	10	b: a + _: _ + _: _	CPP		1C 1a-1b	Doble Permanencia + Cambio
24	10	c: a + _: _ + _: _	CPP		1C 1a -1c	Doble Permanencia + Cambio
25	11	_: a + a: _ + _: _	IFP		2C	Permanencia + Instanciación + Finalización
26	12	a: a + a: _ + _: _	PFP	1E-1P 1C	1_ -1a	Doble Permanencia + Finalización
27	13	b: a + a: _ + _: _	CFP	1E-1P 2C	1_ -1b	Permanencia + Cambio + Finalización
28	13	c: a + a: _ + _: _	CFP	1E-1P 2C	1_ -1c	Permanencia + Cambio + Finalización
29	14	_: a + b: _ + _: _	IFP		2C 1a-1b	Permanencia + Instanciación + Finalización
30	15	a: a + b: _ + _: _	PFP	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
31	16	b: a + b: _ + _: _	CFP	1E-1P 2C	1_ 1a-2b	Permanencia + Cambio + Finalización
32	17	c: a + b: _ + _: _	CFP	1E-1P 2C	1_ 1a-1b-1c	Permanencia + Cambio + Finalización
33	14	_: a + c: _ + _: _	IFP		2C 1a -1c	Permanencia + Instanciación + Finalización
34	15	a: a + c: _ + _: _	PFP	1E-1P 1C	1_ -1c	Doble Permanencia + Finalización
35	17	b: a + c: _ + _: _	CFP	1E-1P 2C	1_ 1a-1b-1c	Permanencia + Cambio + Finalización
36	16	c: a + c: _ + _: _	CFP	1E-1P 2C	1_ 1a -2c	Permanencia + Cambio + Finalización
37	18	_: a + a: _ + a: _	IFF	1E-1P 3C	1_ -1a	Instanciación + Doble Finalización
38	19	a: a + a: _ + a: _	PFF	2E-2P 2C	2_ -2a	Permanencia + Doble Finalización
39	20	b: a + a: _ + a: _	CFF	2E-2P 3C	2_ -1a-1b	Cambio + Doble Finalización
40	20	c: a + a: _ + a: _	CFF	2E-2P 3C	2_ -1a -1c	Cambio + Doble Finalización
41	21	_: a + b: _ + a: _	IFF	1E-1P 3C	1_ -1b	Instanciación + Doble Finalización
42	22	a: a + b: _ + a: _	PFF	2E-2P 2C	2_ -1a-1b	Permanencia + Doble Finalización
43	23	b: a + b: _ + a: _	CFF	2E-2P 3C	2_ -2b	Cambio + Doble Finalización
44	24	c: a + b: _ + a: _	CFF	2E-2P 3C	2_ -1b-1c	Cambio + Doble Finalización
45	21	_: a + c: _ + a: _	IFF	1E-1P 3C	1_ -1c	Instanciación + Doble Finalización
46	22	a: a + c: _ + a: _	PFF	2E-2P 2C	2_ -1a -1c	Permanencia + Doble Finalización
47	24	b: a + c: _ + a: _	CFF	2E-2P 3C	2_ -1b-1c	Cambio + Doble Finalización
48	23	c: a + c: _ + a: _	CFF	2E-2P 3C	2_ -2c	Cambio + Doble Finalización
49	25	_: a + b: _ + b: _	IFF	1E-1P 3C	1_ 1a-2b	Instanciación + Doble Finalización
50	26	a: a + b: _ + b: _	PFF	2E-2P 2C	2_ -2b	Permanencia + Doble Finalización
51	27	b: a + b: _ + b: _	CFF	2E-2P 3C	2_ 1a-3b	Cambio + Doble Finalización
52	28	c: a + b: _ + b: _	CFF	2E-2P 3C	2_ 1a-2b-1c	Cambio + Doble Finalización
53	29	_: a + c: _ + b: _	IFF	1E-1P 3C	1_ 1a-1b-1c	Instanciación + Doble Finalización
54	30	a: a + c: _ + b: _	PFF	2E-2P 2C	2_ -1b-1c	Permanencia + Doble Finalización
55	31	b: a + c: _ + b: _	CFF	2E-2P 3C	2_ 1a-2b-1c	Cambio + Doble Finalización
56	31	c: a + c: _ + b: _	CFF	2E-2P 3C	2_ 1a-1b-2c	Cambio + Doble Finalización
57	25	_: a + c: _ + c: _	IFF	1E-1P 3C	1_ 1a -2c	Instanciación + Doble Finalización
58	26	a: a + c: _ + c: _	PFF	2E-2P 2C	2_ -2c	Permanencia + Doble Finalización
59	28	b: a + c: _ + c: _	CFF	2E-2P 3C	2_ 1a-1b-2c	Cambio + Doble Finalización
60	27	c: a + c: _ + c: _	CFF	2E-2P 3C	2_ 1a -3c	Cambio + Doble Finalización

61	8	_ : b + _ : _ + _ : _	IPP	-1E 1P 1C	-1_ 1b	Doble Permanencia + Instanciación
62	10	a : b + _ : _ + _ : _	CPP	1C	-1a 1b	Doble Permanencia + Cambio
63	9	b : b + _ : _ + _ : _	PPP			Triple Permanencia
64	10	c : b + _ : _ + _ : _	CPP	1C	1b-1c	Doble Permanencia + Cambio
65	14	_ : b + a : _ + _ : _	IFP	2C	-1a 1b	Permanencia + Instanciación + Finalización
66	16	a : b + a : _ + _ : _	CFP	1E-1P 2C	1_-2a 1b	Permanencia + Cambio + Finalización
67	15	b : b + a : _ + _ : _	PFP	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
68	17	c : b + a : _ + _ : _	CFP	1E-1P 2C	1_-1a 1b-1c	Permanencia + Cambio + Finalización
69	11	_ : b + b : _ + _ : _	IFP	2C		Permanencia + Instanciación + Finalización
70	13	a : b + b : _ + _ : _	CFP	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
71	12	b : b + b : _ + _ : _	PFP	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
72	13	c : b + b : _ + _ : _	CFP	1E-1P 2C	1_ -1c	Permanencia + Cambio + Finalización
73	14	_ : b + c : _ + _ : _	IFP	2C	1b-1c	Permanencia + Instanciación + Finalización
74	17	a : b + c : _ + _ : _	CFP	1E-1P 2C	1_-1a 1b-1c	Permanencia + Cambio + Finalización
75	15	b : b + c : _ + _ : _	PFP	1E-1P 1C	1_ -1c	Doble Permanencia + Finalización
76	16	c : b + c : _ + _ : _	CFP	1E-1P 2C	1_ 1b-2c	Permanencia + Cambio + Finalización
77	25	_ : b + a : _ + a : _	IFF	1E-1P 3C	1_-2a 1b	Instanciación + Doble Finalización
78	27	a : b + a : _ + a : _	CFF	2E-2P 3C	2_-3a 1b	Cambio + Doble Finalización
79	26	b : b + a : _ + a : _	PFF	2E-2P 2C	2_-2a	Permanencia + Doble Finalización
80	28	c : b + a : _ + a : _	CFF	2E-2P 3C	2_-2a 1b-1c	Cambio + Doble Finalización
81	21	_ : b + b : _ + a : _	IFF	1E-1P 3C	1_-1a	Instanciación + Doble Finalización
82	23	a : b + b : _ + a : _	CFF	2E-2P 3C	2_-2a	Cambio + Doble Finalización
83	22	b : b + b : _ + a : _	PFF	2E-2P 2C	2_-1a-1b	Permanencia + Doble Finalización
84	24	c : b + b : _ + a : _	CFF	2E-2P 3C	2_-1a -1c	Cambio + Doble Finalización
85	29	_ : b + c : _ + a : _	IFF	1E-1P 3C	1_-1a 1b-1c	Instanciación + Doble Finalización
86	31	a : b + c : _ + a : _	CFF	2E-2P 3C	2_-2a 1b-1c	Cambio + Doble Finalización
87	30	b : b + c : _ + a : _	PFF	2E-2P 2C	2_-1a -1c	Permanencia + Doble Finalización
88	31	c : b + c : _ + a : _	CFF	2E-2P 3C	2_-1a 1b-2c	Cambio + Doble Finalización
89	18	_ : b + b : _ + b : _	IFF	1E-1P 3C	1_ -1b	Instanciación + Doble Finalización
90	20	a : b + b : _ + b : _	CFF	2E-2P 3C	2_-1a-1b	Cambio + Doble Finalización
91	19	b : b + b : _ + b : _	PFF	2E-2P 2C	2_ -2b	Permanencia + Doble Finalización
92	20	c : b + b : _ + b : _	CFF	2E-2P 3C	2_ -1b-1c	Cambio + Doble Finalización
93	21	_ : b + c : _ + b : _	IFF	1E-1P 3C	1_ -1c	Instanciación + Doble Finalización
94	24	a : b + c : _ + b : _	CFF	2E-2P 3C	2_-1a -1c	Cambio + Doble Finalización
95	22	b : b + c : _ + b : _	PFF	2E-2P 2C	2_ -1b-1c	Permanencia + Doble Finalización
96	23	c : b + c : _ + b : _	CFF	2E-2P 3C	2_ -2c	Cambio + Doble Finalización
97	25	_ : b + c : _ + c : _	IFF	1E-1P 3C	1_ 1b-2c	Instanciación + Doble Finalización
98	28	a : b + c : _ + c : _	CFF	2E-2P 3C	2_-1a 1b-2c	Cambio + Doble Finalización
99	26	b : b + c : _ + c : _	PFF	2E-2P 2C	2_ -2c	Permanencia + Doble Finalización
100	27	c : b + c : _ + c : _	CFF	2E-2P 3C	2_ 1b-3c	Cambio + Doble Finalización
101	8	_ : c + _ : _ + _ : _	IPP	-1E 1P 1C	-1_ 1c	Doble Permanencia + Instanciación
102	10	a : c + _ : _ + _ : _	CPP	1C	-1a 1c	Doble Permanencia + Cambio
103	10	b : c + _ : _ + _ : _	CPP	1C	-1b 1c	Doble Permanencia + Cambio
104	9	c : c + _ : _ + _ : _	PPP			Triple Permanencia
105	14	_ : c + a : _ + _ : _	IFP	2C	-1a 1c	Permanencia + Instanciación + Finalización
106	16	a : c + a : _ + _ : _	CFP	1E-1P 2C	1_-2a 1c	Permanencia + Cambio + Finalización
107	17	b : c + a : _ + _ : _	CFP	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
108	15	c : c + a : _ + _ : _	PFP	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
109	14	_ : c + b : _ + _ : _	IFP	2C	-1b 1c	Permanencia + Instanciación + Finalización
110	17	a : c + b : _ + _ : _	CFP	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
111	16	b : c + b : _ + _ : _	CFP	1E-1P 2C	1_ -2b 1c	Permanencia + Cambio + Finalización
112	15	c : c + b : _ + _ : _	PFP	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
113	11	_ : c + c : _ + _ : _	IFP	2C		Permanencia + Instanciación + Finalización
114	13	a : c + c : _ + _ : _	CFP	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
115	13	b : c + c : _ + _ : _	CFP	1E-1P 2C	1_ -1b	Permanencia + Cambio + Finalización
116	12	c : c + c : _ + _ : _	PFP	1E-1P 1C	1_ -1c	Doble Permanencia + Finalización
117	25	_ : c + a : _ + a : _	IFF	1E-1P 3C	1_-2a 1c	Instanciación + Doble Finalización
118	27	a : c + a : _ + a : _	CFF	2E-2P 3C	2_-3a 1c	Cambio + Doble Finalización
119	28	b : c + a : _ + a : _	CFF	2E-2P 3C	2_-2a-1b 1c	Cambio + Doble Finalización
120	26	c : c + a : _ + a : _	PFF	2E-2P 2C	2_-2a	Permanencia + Doble Finalización
121	29	_ : c + b : _ + a : _	IFF	1E-1P 3C	1_-1a-1b 1c	Instanciación + Doble Finalización
122	31	a : c + b : _ + a : _	CFF	2E-2P 3C	2_-2a-1b 1c	Cambio + Doble Finalización
123	31	b : c + b : _ + a : _	CFF	2E-2P 3C	2_-1a-2b 1c	Cambio + Doble Finalización
124	30	c : c + b : _ + a : _	PFF	2E-2P 2C	2_-1a-1b	Permanencia + Doble Finalización
125	21	_ : c + c : _ + a : _	IFF	1E-1P 3C	1_-1a	Instanciación + Doble Finalización
126	23	a : c + c : _ + a : _	CFF	2E-2P 3C	2_-2a	Cambio + Doble Finalización
127	24	b : c + c : _ + a : _	CFF	2E-2P 3C	2_-1a-1b	Cambio + Doble Finalización
128	22	c : c + c : _ + a : _	PFF	2E-2P 2C	2_-1a -1c	Permanencia + Doble Finalización
129	25	_ : c + b : _ + b : _	IFF	1E-1P 3C	1_ -2b 1c	Instanciación + Doble Finalización
130	28	a : c + b : _ + b : _	CFF	2E-2P 3C	2_-1a-2b 1c	Cambio + Doble Finalización
131	27	b : c + b : _ + b : _	CFF	2E-2P 3C	2_ -3b 1c	Cambio + Doble Finalización
132	26	c : c + b : _ + b : _	PFF	2E-2P 2C	2_ -2b	Permanencia + Doble Finalización
133	21	_ : c + c : _ + b : _	IFF	1E-1P 3C	1_ -1b	Instanciación + Doble Finalización
134	24	a : c + c : _ + b : _	CFF	2E-2P 3C	2_-1a-1b	Cambio + Doble Finalización
135	23	b : c + c : _ + b : _	CFF	2E-2P 3C	2_ -2b	Cambio + Doble Finalización
136	22	c : c + c : _ + b : _	PFF	2E-2P 2C	2_ -1b-1c	Permanencia + Doble Finalización
137	18	_ : c + c : _ + c : _	IFF	1E-1P 3C	1_ -1c	Instanciación + Doble Finalización

138	20	a:c + c:_ + c:_	CFF	2E-2P	3C	2_-1a	-1c	Cambio + Doble Finalización	
139	20	b:c + c:_ + c:_	CFF	2E-2P	3C	2_-	-1b-1c	Cambio + Doble Finalización	
140	19	c:c + c:_ + c:_	PFF	2E-2P	2C	2_-	-2c	Permanencia + Doble Finalización	
141	19	_:a + _:a + _:_	IIP	-2E 2P	2C	-2_	2a	Permanencia + Doble Instanciación	
142	12	a:a + _:a + _:_	PIP	-1E 1P	1C	-1_	1a	Doble Permanencia + Instanciación	
143	22	b:a + _:a + _:_	CIP	-1E 1P	2C	-1_	2a-1b	Permanencia + Instanciación + Cambio	
144	22	c:a + _:a + _:_	CIP	-1E 1P	2C	-1_	2a	-1c	Permanencia + Instanciación + Cambio
145	9	a:a + a:a + _:_	PPP					Triple Permanencia	
146	15	b:a + a:a + _:_	CPP		1C		1a-1b	Doble Permanencia + Cambio	
147	15	c:a + a:a + _:_	CPP		1C		1a	-1c	Doble Permanencia + Cambio
148	26	b:a + b:a + _:_	CCP		2C		2a-2b	Permanencia + Doble Cambio	
149	30	c:a + b:a + _:_	CCP		2C		2a-1b-1c	Permanencia + Doble Cambio	
150	26	c:a + c:a + _:_	CCP		2C		2a	-2c	Permanencia + Doble Cambio
151	18	_:a + _:a + a:_	IIF	-1E 1P	3C	-1_	1a	Doble Instanciación + Finalización	
152	11	a:a + _:a + a:_	PIF		2C			Permanencia + Instanciación + Finalización	
153	21	b:a + _:a + a:_	CIF		3C		1a-1b	Instanciación + Cambio + Finalización	
154	21	c:a + _:a + a:_	CIF		3C		1a	-1c	Instanciación + Cambio + Finalización
155	8	a:a + a:a + a:_	PPF	1E-1P	1C	1_-	1a	Doble Permanencia + Finalización	
156	14	b:a + a:a + a:_	CPF	1E-1P	2C	1_-	-1b	Permanencia + Cambio + Finalización	
157	14	c:a + a:a + a:_	CPF	1E-1P	2C	1_-	-1c	Permanencia + Cambio + Finalización	
158	25	b:a + b:a + a:_	CCF	1E-1P	3C	1_-	1a-2b	Doble Cambio + Finalización	
159	29	c:a + b:a + a:_	CCF	1E-1P	3C	1_-	1a-1b-1c	Doble Cambio + Finalización	
160	25	c:a + c:a + a:_	CCF	1E-1P	3C	1_-	1a	-2c	Doble Cambio + Finalización
161	20	_:a + _:a + b:_	IIF	-1E 1P	3C	-1_	2a-1b	Doble Instanciación + Finalización	
162	13	a:a + _:a + b:_	PIF		2C		1a-1b	Permanencia + Instanciación + Finalización	
163	23	b:a + _:a + b:_	CIF		3C		2a-2b	Instanciación + Cambio + Finalización	
164	24	c:a + _:a + b:_	CIF		3C		2a-1b-1c	Instanciación + Cambio + Finalización	
165	10	a:a + a:a + b:_	PPF	1E-1P	1C	1_-	-1b	Doble Permanencia + Finalización	
166	16	b:a + a:a + b:_	CPF	1E-1P	2C	1_-	1a-2b	Permanencia + Cambio + Finalización	
167	17	c:a + a:a + b:_	CPF	1E-1P	2C	1_-	1a-1b-1c	Permanencia + Cambio + Finalización	
168	27	b:a + b:a + b:_	CCF	1E-1P	3C	1_-	2a-3b	Doble Cambio + Finalización	
169	31	c:a + b:a + b:_	CCF	1E-1P	3C	1_-	2a-2b-1c	Doble Cambio + Finalización	
170	28	c:a + c:a + b:_	CCF	1E-1P	3C	1_-	2a-1b-2c	Doble Cambio + Finalización	
171	20	_:a + _:a + c:_	IIF	-1E 1P	3C	-1_	2a	-1c	Doble Instanciación + Finalización
172	13	a:a + _:a + c:_	PIF		2C		1a	-1c	Permanencia + Instanciación + Finalización
173	24	b:a + _:a + c:_	CIF		3C		2a-1b-1c	Instanciación + Cambio + Finalización	
174	23	c:a + _:a + c:_	CIF		3C		2a	-2c	Instanciación + Cambio + Finalización
175	10	a:a + a:a + c:_	PPF	1E-1P	1C	1_-	-1c	Doble Permanencia + Finalización	
176	17	b:a + a:a + c:_	CPF	1E-1P	2C	1_-	1a-1b-1c	Permanencia + Cambio + Finalización	
177	16	c:a + a:a + c:_	CPF	1E-1P	2C	1_-	1a	-2c	Permanencia + Cambio + Finalización
178	28	b:a + b:a + c:_	CCF	1E-1P	3C	1_-	2a-2b-1c	Doble Cambio + Finalización	
179	31	c:a + b:a + c:_	CCF	1E-1P	3C	1_-	2a-1b-2c	Doble Cambio + Finalización	
180	27	c:a + c:a + c:_	CCF	1E-1P	3C	1_-	2a	-3c	Doble Cambio + Finalización
181	32	_:b + _:a + _:_	IIP	-2E 2P	2C	-2_	1a 1b	Permanencia + Doble Instanciación	
182	33	a:b + _:a + _:_	CIP	-1E 1P	2C	-1_	-1b	Permanencia + Instanciación + Cambio	
183	34	b:b + _:a + _:_	PIP	-1E 1P	1C	-1_	1a	Doble Permanencia + Instanciación	
184	35	c:b + _:a + _:_	CIP	-1E 1P	2C	-1_	1a 1b-1c	Permanencia + Instanciación + Cambio	
185	34	_:b + a:a + _:_	IPP	-1E 1P	1C	-1_	1b	Doble Permanencia + Instanciación	
186	34	a:b + a:a + _:_	CPP		1C		-1a 1b	Doble Permanencia + Cambio	
187	36	b:b + a:a + _:_	PPP					Triple Permanencia	
188	37	c:b + a:a + _:_	CPP		1C		1b-1c	Doble Permanencia + Cambio	
189	33	_:b + b:a + _:_	ICP	-1E 1P	2C	-1_	1a	Permanencia + Instanciación + Cambio	
190	38	a:b + b:a + _:_	CCP		2C			Permanencia + Doble Cambio	
191	34	b:b + b:a + _:_	PCP		1C		1a-1b	Doble Permanencia + Cambio	
192	39	c:b + b:a + _:_	CCP		2C		1a	-1c	Permanencia + Doble Cambio
193	35	_:b + c:a + _:_	ICP	-1E 1P	2C	-1_	1a 1b-1c	Permanencia + Instanciación + Cambio	
194	39	a:b + c:a + _:_	CCP		2C		1b-1c	Permanencia + Doble Cambio	
195	37	b:b + c:a + _:_	PCP		1C		1a	-1c	Doble Permanencia + Cambio
196	40	c:b + c:a + _:_	CCP		2C		1a 1b-2c	Permanencia + Doble Cambio	
197	41	_:b + _:a + a:_	IIF	-1E 1P	3C	-1_	1b	Doble Instanciación + Finalización	
198	41	a:b + _:a + a:_	CIF		3C		-1a 1b	Instanciación + Cambio + Finalización	
199	38	b:b + _:a + a:_	PIF		2C			Permanencia + Instanciación + Finalización	
200	42	c:b + _:a + a:_	CIF		3C		1b-1c	Instanciación + Cambio + Finalización	
201	33	_:b + a:a + a:_	IPF		2C		-1a 1b	Permanencia + Instanciación + Finalización	
202	32	a:b + a:a + a:_	CPF	1E-1P	2C	1_-	2a 1b	Permanencia + Cambio + Finalización	
203	34	b:b + a:a + a:_	PPF	1E-1P	1C	1_-	-1a	Doble Permanencia + Finalización	
204	35	c:b + a:a + a:_	CPF	1E-1P	2C	1_-	1a 1b-1c	Permanencia + Cambio + Finalización	
205	43	_:b + b:a + a:_	ICF		3C			Instanciación + Cambio + Finalización	
206	41	a:b + b:a + a:_	CCF	1E-1P	3C	1_-	1a	Doble Cambio + Finalización	
207	33	b:b + b:a + a:_	PCF	1E-1P	2C	1_-	-1b	Permanencia + Cambio + Finalización	
208	44	c:b + b:a + a:_	CCF	1E-1P	3C	1_-	-1c	Doble Cambio + Finalización	
209	44	_:b + c:a + a:_	ICF		3C		1b-1c	Instanciación + Cambio + Finalización	
210	45	a:b + c:a + a:_	CCF	1E-1P	3C	1_-	1a 1b-1c	Doble Cambio + Finalización	
211	39	b:b + c:a + a:_	PCF	1E-1P	2C	1_-	-1c	Permanencia + Cambio + Finalización	
212	46	c:b + c:a + a:_	CCF	1E-1P	3C	1_-	1b-2c	Doble Cambio + Finalización	
213	41	_:b + _:a + b:_	IIF	-1E 1P	3C	-1_	1a	Doble Instanciación + Finalización	
214	43	a:b + _:a + b:_	CIF		3C			Instanciación + Cambio + Finalización	

215	33	b:b + _:a + b:_	PIF	2C	1a-1b	Permanencia + Instanciación + Finalización
216	44	c:b + _:a + b:_	CIF	3C	1a -1c	Instanciación + Cambio + Finalización
217	38	_:b + a:a + b:_	IPF	2C		Permanencia + Instanciación + Finalización
218	33	a:b + a:a + b:_	CPF	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
219	34	b:b + a:a + b:_	PPF	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
220	39	c:b + a:a + b:_	CPF	1E-1P 2C	1_ -1c	Permanencia + Cambio + Finalización
221	41	_:b + b:a + b:_	ICF	3C	1a-1b	Instanciación + Cambio + Finalización
222	41	a:b + b:a + b:_	CCF	1E-1P 3C	1_ -1b	Doble Cambio + Finalización
223	32	b:b + b:a + b:_	PCF	1E-1P 2C	1_ 1a-2b	Permanencia + Cambio + Finalización
224	45	c:b + b:a + b:_	CCF	1E-1P 3C	1_ 1a-1b-1c	Doble Cambio + Finalización
225	42	_:b + c:a + b:_	ICF	3C	1a -1c	Instanciación + Cambio + Finalización
226	44	a:b + c:a + b:_	CCF	1E-1P 3C	1_ -1c	Doble Cambio + Finalización
227	35	b:b + c:a + b:_	PCF	1E-1P 2C	1_ 1a-1b-1c	Permanencia + Cambio + Finalización
228	46	c:b + c:a + b:_	CCF	1E-1P 3C	1_ 1a -2c	Doble Cambio + Finalización
229	45	_:b + _:a + c:_	IIF	-1E 1P 3C	-1_ 1a 1b-1c	Doble Instanciación + Finalización
230	44	a:b + _:a + c:_	CIF	3C	1b-1c	Instanciación + Cambio + Finalización
231	39	b:b + _:a + c:_	PIF	2C	1a -1c	Permanencia + Instanciación + Finalización
232	46	c:b + _:a + c:_	CIF	3C	1a 1b-2c	Instanciación + Cambio + Finalización
233	39	_:b + a:a + c:_	IPF	2C		Permanencia + Instanciación + Finalización
234	35	a:b + a:a + c:_	CPF	1E-1P 2C	1_-1a 1b-1c	Permanencia + Cambio + Finalización
235	37	b:b + a:a + c:_	PPF	1E-1P 1C	1_ -1c	Doble Permanencia + Finalización
236	40	c:b + a:a + c:_	CPF	1E-1P 2C	1_ 1b-2c	Permanencia + Cambio + Finalización
237	44	_:b + b:a + c:_	ICF	3C	1a -1c	Instanciación + Cambio + Finalización
238	42	a:b + b:a + c:_	CCF	1E-1P 3C	1_ -1c	Doble Cambio + Finalización
239	35	b:b + b:a + c:_	PCF	1E-1P 2C	1_ 1a-1b-1c	Permanencia + Cambio + Finalización
240	46	c:b + b:a + c:_	CCF	1E-1P 3C	1_ 1a -2c	Doble Cambio + Finalización
241	46	_:b + c:a + c:_	ICF	3C	1a 1b-2c	Instanciación + Cambio + Finalización
242	46	a:b + c:a + c:_	CCF	1E-1P 3C	1_ 1b-2c	Doble Cambio + Finalización
243	40	b:b + c:a + c:_	PCF	1E-1P 2C	1_ 1a -2c	Permanencia + Cambio + Finalización
244	47	c:b + c:a + c:_	CCF	1E-1P 3C	1_ 1a 1b-3c	Doble Cambio + Finalización
245	32	_:c + _:a + _:_	IIP	-2E 2P 2C	-2_ 1a 1c	Permanencia + Doble Instanciación
246	33	a:c + _:a + _:_	CIP	-1E 1P 2C	-1_ 1c	Permanencia + Instanciación + Cambio
247	35	b:c + _:a + _:_	CIP	-1E 1P 2C	-1_ 1a-1b 1c	Permanencia + Instanciación + Cambio
248	34	c:c + _:a + _:_	PIP	-1E 1P 1C	-1_ 1a	Doble Permanencia + Instanciación
249	34	_:c + a:a + _:_	IPP	-1E 1P 1C	-1_ 1c	Doble Permanencia + Instanciación
250	34	a:c + a:a + _:_	CPP	1C	-1a 1c	Doble Permanencia + Cambio
251	37	b:c + a:a + _:_	CPP	1C	-1b 1c	Doble Permanencia + Cambio
252	36	c:c + a:a + _:_	PPP			Triple Permanencia
253	35	_:c + b:a + _:_	ICP	-1E 1P 2C	-1_ 1a-1b 1c	Permanencia + Instanciación + Cambio
254	39	a:c + b:a + _:_	CCP	2C	-1b 1c	Permanencia + Doble Cambio
255	40	b:c + b:a + _:_	CCP	2C	1a-2b 1c	Permanencia + Doble Cambio
256	37	c:c + b:a + _:_	PCP	1C	1a-1b	Doble Permanencia + Cambio
257	33	_:c + c:a + _:_	ICP	-1E 1P 2C	-1_ 1a	Permanencia + Instanciación + Cambio
258	38	a:c + c:a + _:_	CCP	2C		Permanencia + Doble Cambio
259	39	b:c + c:a + _:_	CCP	2C	1a-1b	Permanencia + Doble Cambio
260	34	c:c + c:a + _:_	PCP	1C	1a -1c	Doble Permanencia + Cambio
261	41	_:c + _:a + a:_	IIF	-1E 1P 3C	-1_ 1c	Doble Instanciación + Finalización
262	41	a:c + _:a + a:_	CIF	3C	-1a 1c	Instanciación + Cambio + Finalización
263	42	b:c + _:a + a:_	CIF	3C	-1b 1c	Instanciación + Cambio + Finalización
264	38	c:c + _:a + a:_	PIF	2C		Permanencia + Instanciación + Finalización
265	33	_:c + a:a + a:_	IPF	2C	-1a 1c	Permanencia + Instanciación + Finalización
266	32	a:c + a:a + a:_	CPF	1E-1P 2C	1_-2a 1c	Permanencia + Cambio + Finalización
267	35	b:c + a:a + a:_	CPF	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
268	34	c:c + a:a + a:_	PPF	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
269	44	_:c + b:a + a:_	ICF	3C	-1b 1c	Instanciación + Cambio + Finalización
270	45	a:c + b:a + a:_	CCF	1E-1P 3C	1_-1a-1b 1c	Doble Cambio + Finalización
271	46	b:c + b:a + a:_	CCF	1E-1P 3C	1_ -2b 1c	Doble Cambio + Finalización
272	39	c:c + b:a + a:_	PCF	1E-1P 2C	1_ -1b	Permanencia + Cambio + Finalización
273	43	_:c + c:a + a:_	ICF	3C		Instanciación + Cambio + Finalización
274	41	a:c + c:a + a:_	CCF	1E-1P 3C	1_-1a	Doble Cambio + Finalización
275	44	b:c + c:a + a:_	CCF	1E-1P 3C	1_ -1b	Doble Cambio + Finalización
276	33	c:c + c:a + a:_	PCF	1E-1P 2C	1_ -1c	Permanencia + Cambio + Finalización
277	45	_:c + _:a + b:_	IIF	-1E 1P 3C	-1_ 1a-1b 1c	Doble Instanciación + Finalización
278	44	a:c + _:a + b:_	CIF	3C	-1b 1c	Instanciación + Cambio + Finalización
279	46	b:c + _:a + b:_	CIF	3C	1a-2b 1c	Instanciación + Cambio + Finalización
280	39	c:c + _:a + b:_	PIF	2C	1a-1b	Permanencia + Instanciación + Finalización
281	39	_:c + a:a + b:_	IPF	2C	-1b 1c	Permanencia + Instanciación + Finalización
282	35	a:c + a:a + b:_	CPF	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
283	40	b:c + a:a + b:_	CPF	1E-1P 2C	1_ -2b 1c	Permanencia + Cambio + Finalización
284	37	c:c + a:a + b:_	PPF	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
285	46	_:c + b:a + b:_	ICF	3C	1a-2b 1c	Instanciación + Cambio + Finalización
286	46	a:c + b:a + b:_	CCF	1E-1P 3C	1_ -2b 1c	Doble Cambio + Finalización
287	47	b:c + b:a + b:_	CCF	1E-1P 3C	1_ 1a-3b 1c	Doble Cambio + Finalización
288	40	c:c + b:a + b:_	PCF	1E-1P 2C	1_ 1a-2b	Permanencia + Cambio + Finalización
289	44	_:c + c:a + b:_	ICF	3C	1a-1b	Instanciación + Cambio + Finalización
290	42	a:c + c:a + b:_	CCF	1E-1P 3C	1_ -1b	Doble Cambio + Finalización
291	46	b:c + c:a + b:_	CCF	1E-1P 3C	1_ 1a-2b	Doble Cambio + Finalización

292	35	c:c + c:a + b:_	PCF	1E-1P	2C	1_	1a-1b-1c	Permanencia + Cambio + Finalización
293	41	_:c + _:a + c:_	IIF	-1E 1P	3C	-1_	1a	Doble Instanciación + Finalización
294	43	a:c + _:a + c:_	CIF		3C			Instanciación + Cambio + Finalización
295	44	b:c + _:a + c:_	CIF		3C		1a-1b	Instanciación + Cambio + Finalización
296	33	c:c + _:a + c:_	PIF		2C		1a -1c	Permanencia + Instanciación + Finalización
297	38	_:c + a:a + c:_	IPF		2C			Permanencia + Instanciación + Finalización
298	33	a:c + a:a + c:_	CPF	1E-1P	2C	1_	-1a	Permanencia + Cambio + Finalización
299	39	b:c + a:a + c:_	CPF	1E-1P	2C	1_	-1b	Permanencia + Cambio + Finalización
300	34	c:c + a:a + c:_	PPF	1E-1P	1C	1_	-1c	Doble Permanencia + Finalización
301	42	_:c + b:a + c:_	ICF		3C		1a-1b	Instanciación + Cambio + Finalización
302	44	a:c + b:a + c:_	CCF	1E-1P	3C	1_	-1b	Doble Cambio + Finalización
303	46	b:c + b:a + c:_	CCF	1E-1P	3C	1_	1a-2b	Doble Cambio + Finalización
304	35	c:c + b:a + c:_	PCF	1E-1P	2C	1_	1a-1b-1c	Permanencia + Cambio + Finalización
305	41	_:c + c:a + c:_	ICF		3C		1a -1c	Instanciación + Cambio + Finalización
306	41	a:c + c:a + c:_	CCF	1E-1P	3C	1_	-1c	Doble Cambio + Finalización
307	45	b:c + c:a + c:_	CCF	1E-1P	3C	1_	1a-1b-1c	Doble Cambio + Finalización
308	32	c:c + c:a + c:_	PCF	1E-1P	2C	1_	1a -2c	Permanencia + Cambio + Finalización
309	19	_:b + _:b + _:_	IIP	-2E 2P	2C	-2_	2b	Permanencia + Doble Instanciación
310	22	a:b + _:b + _:_	CIP	-1E 1P	2C	-1_	-1a 2b	Permanencia + Instanciación + Cambio
311	12	b:b + _:b + _:_	PIP	-1E 1P	1C	-1_	1b	Doble Permanencia + Instanciación
312	22	c:b + _:b + _:_	CIP	-1E 1P	2C	-1_	2b-1c	Permanencia + Instanciación + Cambio
313	26	a:b + a:b + _:_	CCP		2C		-2a 2b	Permanencia + Doble Cambio
314	15	b:b + a:b + _:_	PCP		1C		-1a 1b	Doble Permanencia + Cambio
315	30	c:b + a:b + _:_	CCP		2C		-1a 2b-1c	Permanencia + Doble Cambio
316	9	b:b + b:b + _:_	PPP					Triple Permanencia
317	15	c:b + b:b + _:_	CPP		1C		1b-1c	Doble Permanencia + Cambio
318	26	c:b + c:b + _:_	CCP		2C		2b-2c	Permanencia + Doble Cambio
319	20	_:b + _:b + a:_	IIF	-1E 1P	3C	-1_	-1a 2b	Doble Instanciación + Finalización
320	23	a:b + _:b + a:_	CIF		3C		-2a 2b	Instanciación + Cambio + Finalización
321	13	b:b + _:b + a:_	PIF		2C		-1a 1b	Permanencia + Instanciación + Finalización
322	24	c:b + _:b + a:_	CIF		3C		-1a 2b-1c	Instanciación + Cambio + Finalización
323	27	a:b + a:b + a:_	CCF	1E-1P	3C	1_	-3a 2b	Doble Cambio + Finalización
324	16	b:b + a:b + a:_	PCF	1E-1P	2C	1_	-2a 1b	Permanencia + Cambio + Finalización
325	31	c:b + a:b + a:_	CCF	1E-1P	3C	1_	-2a 2b-1c	Doble Cambio + Finalización
326	10	b:b + b:b + a:_	PPF	1E-1P	1C	1_	-1a	Doble Permanencia + Finalización
327	17	c:b + b:b + a:_	CPF	1E-1P	2C	1_	-1a 1b-1c	Permanencia + Cambio + Finalización
328	28	c:b + c:b + a:_	CCF	1E-1P	3C	1_	-1a 2b-2c	Doble Cambio + Finalización
329	18	_:b + _:b + b:_	IIF	-1E 1P	3C	-1_	1b	Doble Instanciación + Finalización
330	21	a:b + _:b + b:_	CIF		3C		-1a 1b	Instanciación + Cambio + Finalización
331	11	b:b + _:b + b:_	PIF		2C			Permanencia + Instanciación + Finalización
332	21	c:b + _:b + b:_	CIF		3C		1b-1c	Instanciación + Cambio + Finalización
333	25	a:b + a:b + b:_	CCF	1E-1P	3C	1_	-2a 1b	Doble Cambio + Finalización
334	14	b:b + a:b + b:_	PCF	1E-1P	2C	1_	-1a	Permanencia + Cambio + Finalización
335	29	c:b + a:b + b:_	CCF	1E-1P	3C	1_	-1a 1b-1c	Doble Cambio + Finalización
336	8	b:b + b:b + b:_	PPF	1E-1P	1C	1_	-1b	Doble Permanencia + Finalización
337	14	c:b + b:b + b:_	CPF	1E-1P	2C	1_	-1c	Permanencia + Cambio + Finalización
338	25	c:b + c:b + b:_	CCF	1E-1P	3C	1_	1b-2c	Doble Cambio + Finalización
339	20	_:b + _:b + c:_	IIF	-1E 1P	3C	-1_	2b-1c	Doble Instanciación + Finalización
340	24	a:b + _:b + c:_	CIF		3C		-1a 2b-1c	Instanciación + Cambio + Finalización
341	13	b:b + _:b + c:_	PIF		2C		1b-1c	Permanencia + Instanciación + Finalización
342	23	c:b + _:b + c:_	CIF		3C		2b-2c	Instanciación + Cambio + Finalización
343	28	a:b + a:b + c:_	CCF	1E-1P	3C	1_	-2a 2b-1c	Doble Cambio + Finalización
344	17	b:b + a:b + c:_	PCF	1E-1P	2C	1_	-1a 1b-1c	Permanencia + Cambio + Finalización
345	31	c:b + a:b + c:_	CCF	1E-1P	3C	1_	-1a 2b-2c	Doble Cambio + Finalización
346	10	b:b + b:b + c:_	PPF	1E-1P	1C	1_	-1c	Doble Permanencia + Finalización
347	16	c:b + b:b + c:_	CPF	1E-1P	2C	1_	1b-2c	Permanencia + Cambio + Finalización
348	27	c:b + c:b + c:_	CCF	1E-1P	3C	1_	2b-3c	Doble Cambio + Finalización
349	32	_:c + _:b + _:_	IIP	-2E 2P	2C	-2_	1b 1c	Permanencia + Doble Instanciación
350	35	a:c + _:b + _:_	CIP	-1E 1P	2C	-1_	-1a 1b 1c	Permanencia + Instanciación + Cambio
351	33	b:c + _:b + _:_	CIP	-1E 1P	2C	-1_	1c	Permanencia + Instanciación + Cambio
352	34	c:c + _:b + _:_	PIP	-1E 1P	1C	-1_	1b	Doble Permanencia + Instanciación
353	35	_:c + a:b + _:_	ICP	-1E 1P	2C	-1_	-1a 1b 1c	Permanencia + Instanciación + Cambio
354	40	a:c + a:b + _:_	CCP		2C		-2a 1b 1c	Permanencia + Doble Cambio
355	39	b:c + a:b + _:_	CCP		2C		-1a 1c	Permanencia + Doble Cambio
356	37	c:c + a:b + _:_	PCP		1C		-1a 1b	Doble Permanencia + Cambio
357	34	_:c + b:b + _:_	IPP	-1E 1P	1C	-1_	1c	Doble Permanencia + Instanciación
358	37	a:c + b:b + _:_	CPP		1C		-1a 1c	Doble Permanencia + Cambio
359	34	b:c + b:b + _:_	CPP		1C		-1b 1c	Doble Permanencia + Cambio
360	36	c:c + b:b + _:_	PPP					Triple Permanencia
361	33	_:c + c:b + _:_	ICP	-1E 1P	2C	-1_	1b	Permanencia + Instanciación + Cambio
362	39	a:c + c:b + _:_	CCP		2C		-1a 1b	Permanencia + Doble Cambio
363	38	b:c + c:b + _:_	CCP		2C			Permanencia + Doble Cambio
364	34	c:c + c:b + _:_	PCP		1C		1b-1c	Doble Permanencia + Cambio
365	45	_:c + _:b + a:_	IIF	-1E 1P	3C	-1_	-1a 1b 1c	Doble Instanciación + Finalización
366	46	a:c + _:b + a:_	CIF		3C		-2a 1b 1c	Instanciación + Cambio + Finalización
367	44	b:c + _:b + a:_	CIF		3C		-1a 1c	Instanciación + Cambio + Finalización
368	39	c:c + _:b + a:_	PIF		2C		-1a 1b	Permanencia + Instanciación + Finalización

369	46	_ : c + a : b + a : _	ICF	3C	-2a 1b 1c	Instanciación + Cambio + Finalización
370	47	a : c + a : b + a : _	CCF	1E-1P 3C	1_-3a 1b 1c	Doble Cambio + Finalización
371	46	b : c + a : b + a : _	CCF	1E-1P 3C	1_-2a 1c	Doble Cambio + Finalización
372	40	c : c + a : b + a : _	PCF	1E-1P 2C	1_-2a 1b	Permanencia + Cambio + Finalización
373	39	_ : c + b : b + a : _	IPF	2C	-1a 1c	Permanencia + Instanciación + Finalización
374	40	a : c + b : b + a : _	CPF	1E-1P 2C	1_-2a 1c	Permanencia + Cambio + Finalización
375	35	b : c + b : b + a : _	CPF	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
376	37	c : c + b : b + a : _	PPF	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
377	44	_ : c + c : b + a : _	ICF	3C	-1a 1b	Instanciación + Cambio + Finalización
378	46	a : c + c : b + a : _	CCF	1E-1P 3C	1_-2a 1b	Doble Cambio + Finalización
379	42	b : c + c : b + a : _	CCF	1E-1P 3C	1_-1a	Doble Cambio + Finalización
380	35	c : c + c : b + a : _	PCF	1E-1P 2C	1_-1a 1b-1c	Permanencia + Cambio + Finalización
381	41	_ : c + _ : b + b : _	IIF	-1E 1P 3C	-1_ 1c	Doble Instanciación + Finalización
382	42	a : c + _ : b + b : _	CIF	3C	-1a 1c	Instanciación + Cambio + Finalización
383	41	b : c + _ : b + b : _	CIF	3C	-1b 1c	Instanciación + Cambio + Finalización
384	38	c : c + _ : b + b : _	PIF	2C		Permanencia + Instanciación + Finalización
385	44	_ : c + a : b + b : _	ICF	3C	-1a 1c	Instanciación + Cambio + Finalización
386	46	a : c + a : b + b : _	CCF	1E-1P 3C	1_-2a 1c	Doble Cambio + Finalización
387	45	b : c + a : b + b : _	CCF	1E-1P 3C	1_-1a-1b 1c	Doble Cambio + Finalización
388	39	c : c + a : b + b : _	PCF	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
389	33	_ : c + b : b + b : _	IPF	2C	-1b 1c	Permanencia + Instanciación + Finalización
390	35	a : c + b : b + b : _	CPF	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
391	32	b : c + b : b + b : _	CPF	1E-1P 2C	1_ -2b 1c	Permanencia + Cambio + Finalización
392	34	c : c + b : b + b : _	PPF	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
393	43	_ : c + c : b + b : _	ICF	3C		Instanciación + Cambio + Finalización
394	44	a : c + c : b + b : _	CCF	1E-1P 3C	1_-1a	Doble Cambio + Finalización
395	41	b : c + c : b + b : _	CCF	1E-1P 3C	1_ -1b	Doble Cambio + Finalización
396	33	c : c + c : b + b : _	PCF	1E-1P 2C	1_ -1c	Permanencia + Cambio + Finalización
397	41	_ : c + _ : b + c : _	IIF	-1E 1P 3C	-1_ 1b	Doble Instanciación + Finalización
398	44	a : c + _ : b + c : _	CIF	3C	-1a 1b	Instanciación + Cambio + Finalización
399	43	b : c + _ : b + c : _	CIF	3C		Instanciación + Cambio + Finalización
400	33	c : c + _ : b + c : _	PIF	2C	1b-1c	Permanencia + Instanciación + Finalización
401	42	_ : c + a : b + c : _	ICF	3C	-1a 1b	Instanciación + Cambio + Finalización
402	46	a : c + a : b + c : _	CCF	1E-1P 3C	1_-2a 1b	Doble Cambio + Finalización
403	44	b : c + a : b + c : _	CCF	1E-1P 3C	1_-1a	Doble Cambio + Finalización
404	35	c : c + a : b + c : _	PCF	1E-1P 2C	1_-1a 1b-1c	Permanencia + Cambio + Finalización
405	38	_ : c + b : b + c : _	IPF	2C		Permanencia + Instanciación + Finalización
406	39	a : c + b : b + c : _	CPF	1E-1P 2C	1_-1a	Permanencia + Cambio + Finalización
407	33	b : c + b : b + c : _	CPF	1E-1P 2C	1_ -1b	Permanencia + Cambio + Finalización
408	34	c : c + b : b + c : _	PPF	1E-1P 1C	1_ -1c	Doble Permanencia + Finalización
409	41	_ : c + c : b + c : _	ICF	3C	1b-1c	Instanciación + Cambio + Finalización
410	45	a : c + c : b + c : _	CCF	1E-1P 3C	1_-1a 1b-1c	Doble Cambio + Finalización
411	41	b : c + c : b + c : _	CCF	1E-1P 3C	1_ -1c	Doble Cambio + Finalización
412	32	c : c + c : b + c : _	PCF	1E-1P 2C	1_ 1b-2c	Permanencia + Cambio + Finalización
413	19	_ : c + _ : c + _ : _	IIP	-2E 2P 2C	-2_ 2c	Permanencia + Doble Instanciación
414	22	a : c + _ : c + _ : _	CIP	-1E 1P 2C	-1_ -1a 2c	Permanencia + Instanciación + Cambio
415	22	b : c + _ : c + _ : _	CIP	-1E 1P 2C	-1_ -1b 2c	Permanencia + Instanciación + Cambio
416	12	c : c + _ : c + _ : _	PIP	-1E 1P 1C	-1_ 1c	Doble Permanencia + Instanciación
417	26	a : c + a : c + _ : _	CCP	2C	-2a 2c	Permanencia + Doble Cambio
418	30	b : c + a : c + _ : _	CCP	2C	-1a-1b 2c	Permanencia + Doble Cambio
419	15	c : c + a : c + _ : _	PCP	1C	-1a 1c	Doble Permanencia + Cambio
420	26	b : c + b : c + _ : _	CCP	2C	-2b 2c	Permanencia + Doble Cambio
421	15	c : c + b : c + _ : _	PCP	1C	-1b 1c	Doble Permanencia + Cambio
422	9	c : c + c : c + _ : _	PPP			Triple Permanencia
423	20	_ : c + _ : c + a : _	IIF	-1E 1P 3C	-1_ -1a 2c	Doble Instanciación + Finalización
424	23	a : c + _ : c + a : _	CIF	3C	-2a 2c	Instanciación + Cambio + Finalización
425	24	b : c + _ : c + a : _	CIF	3C	-1a-1b 2c	Instanciación + Cambio + Finalización
426	13	c : c + _ : c + a : _	PIF	2C	-1a 1c	Permanencia + Instanciación + Finalización
427	27	a : c + a : c + a : _	CCF	1E-1P 3C	1_-3a 2c	Doble Cambio + Finalización
428	31	b : c + a : c + a : _	CCF	1E-1P 3C	1_-2a-1b 2c	Doble Cambio + Finalización
429	16	c : c + a : c + a : _	PCF	1E-1P 2C	1_-2a 1c	Permanencia + Cambio + Finalización
430	28	b : c + b : c + a : _	CCF	1E-1P 3C	1_-1a-2b 2c	Doble Cambio + Finalización
431	17	c : c + b : c + a : _	PCF	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
432	10	c : c + c : c + a : _	PPF	1E-1P 1C	1_-1a	Doble Permanencia + Finalización
433	20	_ : c + _ : c + b : _	IIF	-1E 1P 3C	-1_ -1b 2c	Doble Instanciación + Finalización
434	24	a : c + _ : c + b : _	CIF	3C	-1a-1b 2c	Instanciación + Cambio + Finalización
435	23	b : c + _ : c + b : _	CIF	3C	-2b 2c	Instanciación + Cambio + Finalización
436	13	c : c + _ : c + b : _	PIF	2C	-1b 1c	Permanencia + Instanciación + Finalización
437	28	a : c + a : c + b : _	CCF	1E-1P 3C	1_-2a-1b 2c	Doble Cambio + Finalización
438	31	b : c + a : c + b : _	CCF	1E-1P 3C	1_-1a-2b 2c	Doble Cambio + Finalización
439	17	c : c + a : c + b : _	PCF	1E-1P 2C	1_-1a-1b 1c	Permanencia + Cambio + Finalización
440	27	b : c + b : c + b : _	CCF	1E-1P 3C	1_ -3b 2c	Doble Cambio + Finalización
441	16	c : c + b : c + b : _	PCF	1E-1P 2C	1_ -2b 1c	Permanencia + Cambio + Finalización
442	10	c : c + c : c + b : _	PPF	1E-1P 1C	1_ -1b	Doble Permanencia + Finalización
443	18	_ : c + _ : c + c : _	IIF	-1E 1P 3C	-1_ 1c	Doble Instanciación + Finalización
444	21	a : c + _ : c + c : _	CIF	3C	-1a 1c	Instanciación + Cambio + Finalización
445	21	b : c + _ : c + c : _	CIF	3C	-1b 1c	Instanciación + Cambio + Finalización

446	11	c:c + _:c + c:_	PIF	2C			Permanencia + Instanciación + Finalización
447	25	a:c + a:c + c:_	CCF	1E-1P 3C	1_-2a	1c	Doble Cambio + Finalización
448	29	b:c + a:c + c:_	CCF	1E-1P 3C	1_-1a-1b	1c	Doble Cambio + Finalización
449	14	c:c + a:c + c:_	PCF	1E-1P 2C	1_-1a		Permanencia + Cambio + Finalización
450	25	b:c + b:c + c:_	CCF	1E-1P 3C	1_-2b	1c	Doble Cambio + Finalización
451	14	c:c + b:c + c:_	PCF	1E-1P 2C	1_-1b		Permanencia + Cambio + Finalización
452	8	c:c + c:c + c:_	PPF	1E-1P 1C	1_-1c		Doble Permanencia + Finalización
453	5	_:a + _:a + _:a	III	-3E 3P 3C	-3_ 3a		Triple Instanciación
454	3	a:a + _:a + _:a	PII	-2E 2P 2C	-2_ 2a		Permanencia + Doble Instanciación
455	6	b:a + _:a + _:a	CII	-2E 2P 3C	-2_ 3a-1b		Doble Instanciación + Cambio
456	6	c:a + _:a + _:a	CII	-2E 2P 3C	-2_ 3a	-1c	Doble Instanciación + Cambio
457	2	a:a + a:a + _:a	PPI	-1E 1P 1C	-1_ 1a		Doble Permanencia + Instanciación
458	4	b:a + a:a + _:a	CPI	-1E 1P 2C	-1_ 2a-1b		Permanencia + Instanciación + Cambio
459	4	c:a + a:a + _:a	CPI	-1E 1P 2C	-1_ 2a	-1c	Permanencia + Instanciación + Cambio
460	6	b:a + b:a + _:a	CCI	-1E 1P 3C	-1_ 3a-2b		Instanciación + Doble Cambio
461	7	c:a + b:a + _:a	CCI	-1E 1P 3C	-1_ 3a-1b-1c		Instanciación + Doble Cambio
462	6	c:a + c:a + _:a	CCI	-1E 1P 3C	-1_ 3a	-2c	Instanciación + Doble Cambio
463	1	a:a + a:a + a:a	PPP				Triple Permanencia
464	2	b:a + a:a + a:a	CPP	1C	1a-1b		Doble Permanencia + Cambio
465	2	c:a + a:a + a:a	CPP	1C	1a	-1c	Doble Permanencia + Cambio
466	3	b:a + b:a + a:a	CCP	2C	2a-2b		Permanencia + Doble Cambio
467	4	c:a + b:a + a:a	CCP	2C	2a-1b-1c		Permanencia + Doble Cambio
468	3	c:a + c:a + a:a	CCP	2C	2a	-2c	Permanencia + Doble Cambio
469	5	b:a + b:a + b:a	CCC	3C	3a-3b		Triple Cambio
470	6	c:a + b:a + b:a	CCC	3C	3a-2b-1c		Triple Cambio
471	6	c:a + c:a + b:a	CCC	3C	3a-1b-2c		Triple Cambio
472	5	c:a + c:a + c:a	CCC	3C	3a	-3c	Triple Cambio
473	27	_:b + _:a + _:a	III	-3E 3P 3C	-3_ 2a 1b		Triple Instanciación
474	25	a:b + _:a + _:a	CII	-2E 2P 3C	-2_ 1a 1b		Doble Instanciación + Cambio
475	26	b:b + _:a + _:a	PII	-2E 2P 2C	-2_ 2a		Permanencia + Doble Instanciación
476	28	c:b + _:a + _:a	CII	-2E 2P 3C	-2_ 2a 1b-1c		Doble Instanciación + Cambio
477	16	_:b + a:a + _:a	IPI	-2E 2P 2C	-2_ 1a 1b		Permanencia + Doble Instanciación
478	14	a:b + a:a + _:a	CPI	-1E 1P 2C	-1_ 1b		Permanencia + Instanciación + Cambio
479	15	b:b + a:a + _:a	PPI	-1E 1P 1C	-1_ 1a		Doble Permanencia + Instanciación
480	17	c:b + a:a + _:a	CPI	-1E 1P 2C	-1_ 1a 1b-1c		Permanencia + Instanciación + Cambio
481	23	_:b + b:a + _:a	ICI	-2E 2P 3C	-2_ 2a		Doble Instanciación + Cambio
482	21	a:b + b:a + _:a	CCI	-1E 1P 3C	-1_ 1a		Instanciación + Doble Cambio
483	22	b:b + b:a + _:a	PCI	-1E 1P 2C	-1_ 2a-1b		Permanencia + Instanciación + Cambio
484	24	c:b + b:a + _:a	CCI	-1E 1P 3C	-1_ 2a	-1c	Instanciación + Doble Cambio
485	31	_:b + c:a + _:a	ICI	-2E 2P 3C	-2_ 2a 1b-1c		Doble Instanciación + Cambio
486	29	a:b + c:a + _:a	CCI	-1E 1P 3C	-1_ 1a 1b-1c		Instanciación + Doble Cambio
487	30	b:b + c:a + _:a	PCI	-1E 1P 2C	-1_ 2a	-1c	Permanencia + Instanciación + Cambio
488	31	c:b + c:a + _:a	CCI	-1E 1P 3C	-1_ 2a 1b-2c		Instanciación + Doble Cambio
489	10	_:b + a:a + a:a	IPP	-1E 1P 1C	-1_ 1b		Doble Permanencia + Instanciación
490	8	a:b + a:a + a:a	CPP	1C	-1a 1b		Doble Permanencia + Cambio
491	9	b:b + a:a + a:a	PPP				Triple Permanencia
492	10	c:b + a:a + a:a	CPP	1C	1b-1c		Doble Permanencia + Cambio
493	13	_:b + b:a + a:a	ICP	-1E 1P 2C	-1_ 1a		Permanencia + Instanciación + Cambio
494	11	a:b + b:a + a:a	CCP	2C			Permanencia + Doble Cambio
495	12	b:b + b:a + a:a	PCP	1C	1a-1b		Doble Permanencia + Cambio
496	13	c:b + b:a + a:a	CCP	2C	1a	-1c	Permanencia + Doble Cambio
497	17	_:b + c:a + a:a	ICP	-1E 1P 2C	-1_ 1a 1b-1c		Permanencia + Instanciación + Cambio
498	14	a:b + c:a + a:a	CCP	2C	1b-1c		Permanencia + Doble Cambio
499	15	b:b + c:a + a:a	PCP	1C	1a	-1c	Doble Permanencia + Cambio
500	16	c:b + c:a + a:a	CCP	2C	1a 1b-2c		Permanencia + Doble Cambio
501	20	_:b + b:a + b:a	ICC	-1E 1P 3C	-1_ 2a-1b		Instanciación + Doble Cambio
502	18	a:b + b:a + b:a	CCC	3C	1a-1b		Triple Cambio
503	19	b:b + b:a + b:a	PCC	2C	2a-2b		Permanencia + Doble Cambio
504	20	c:b + b:a + b:a	CCC	3C	2a-1b-1c		Triple Cambio
505	24	_:b + c:a + b:a	ICC	-1E 1P 3C	-1_ 2a	-1c	Instanciación + Doble Cambio
506	21	a:b + c:a + b:a	CCC	3C	1a	-1c	Triple Cambio
507	22	b:b + c:a + b:a	PCC	2C	2a-1b-1c		Permanencia + Doble Cambio
508	23	c:b + c:a + b:a	CCC	3C	2a	-2c	Triple Cambio
509	28	_:b + c:a + c:a	ICC	-1E 1P 3C	-1_ 2a 1b-2c		Instanciación + Doble Cambio
510	25	a:b + c:a + c:a	CCC	3C	1a 1b-2c		Triple Cambio
511	26	b:b + c:a + c:a	PCC	2C	2a	-2c	Permanencia + Doble Cambio
512	27	c:b + c:a + c:a	CCC	3C	2a 1b-3c		Triple Cambio
513	27	_:c + _:a + _:a	III	-3E 3P 3C	-3_ 2a	1c	Triple Instanciación
514	25	a:c + _:a + _:a	CII	-2E 2P 3C	-2_ 1a	1c	Doble Instanciación + Cambio
515	28	b:c + _:a + _:a	CII	-2E 2P 3C	-2_ 2a-1b	1c	Doble Instanciación + Cambio
516	26	c:c + _:a + _:a	PII	-2E 2P 2C	-2_ 2a		Permanencia + Doble Instanciación
517	16	_:c + a:a + _:a	IPI	-2E 2P 2C	-2_ 1a	1c	Permanencia + Doble Instanciación
518	14	a:c + a:a + _:a	CPI	-1E 1P 2C	-1_ 1c		Permanencia + Instanciación + Cambio
519	17	b:c + a:a + _:a	CPI	-1E 1P 2C	-1_ 1a-1b	1c	Permanencia + Instanciación + Cambio
520	15	c:c + a:a + _:a	PPI	-1E 1P 1C	-1_ 1a		Doble Permanencia + Instanciación
521	31	_:c + b:a + _:a	ICI	-2E 2P 3C	-2_ 2a-1b	1c	Doble Instanciación + Cambio
522	29	a:c + b:a + _:a	CCI	-1E 1P 3C	-1_ 1a-1b	1c	Instanciación + Doble Cambio

523	31	b:c + b:a + _:a	CCI	-1E 1P 3C	-1_ 2a-2b 1c	Instanciación + Doble Cambio
524	30	c:c + b:a + _:a	PCI	-1E 1P 2C	-1_ 2a-1b	Permanencia + Instanciación + Cambio
525	23	_:c + c:a + _:a	ICI	-2E 2P 3C	-2_ 2a	Doble Instanciación + Cambio
526	21	a:c + c:a + _:a	CCI	-1E 1P 3C	-1_ 1a	Instanciación + Doble Cambio
527	24	b:c + c:a + _:a	CCI	-1E 1P 3C	-1_ 2a-1b	Instanciación + Doble Cambio
528	22	c:c + c:a + _:a	PCI	-1E 1P 2C	-1_ 2a -1c	Permanencia + Instanciación + Cambio
529	10	_:c + a:a + a:a	IPP	-1E 1P 1C	-1_ 1c	Doble Permanencia + Instanciación
530	8	a:c + a:a + a:a	CPP	1C	-1a 1c	Doble Permanencia + Cambio
531	10	b:c + a:a + a:a	CPP	1C	-1b 1c	Doble Permanencia + Cambio
532	9	c:c + a:a + a:a	PPP			Triple Permanencia
533	17	_:c + b:a + a:a	ICP	-1E 1P 2C	-1_ 1a-1b 1c	Permanencia + Instanciación + Cambio
534	14	a:c + b:a + a:a	CCP	2C	-1b 1c	Permanencia + Doble Cambio
535	16	b:c + b:a + a:a	CCP	2C	1a-2b 1c	Permanencia + Doble Cambio
536	15	c:c + b:a + a:a	PCP	1C	1a-1b	Doble Permanencia + Cambio
537	13	_:c + c:a + a:a	ICP	-1E 1P 2C	-1_ 1a	Permanencia + Instanciación + Cambio
538	11	a:c + c:a + a:a	CCP	2C		Permanencia + Doble Cambio
539	13	b:c + c:a + a:a	CCP	2C	1a-1b	Permanencia + Doble Cambio
540	12	c:c + c:a + a:a	PCP	1C	1a -1c	Doble Permanencia + Cambio
541	28	_:c + b:a + b:a	ICC	-1E 1P 3C	-1_ 2a-2b 1c	Instanciación + Doble Cambio
542	25	a:c + b:a + b:a	CCC	3C	1a-2b 1c	Triple Cambio
543	27	b:c + b:a + b:a	CCC	3C	2a-3b 1c	Triple Cambio
544	26	c:c + b:a + b:a	PCC	2C	2a-2b	Permanencia + Doble Cambio
545	24	_:c + c:a + b:a	ICC	-1E 1P 3C	-1_ 2a-1b	Instanciación + Doble Cambio
546	21	a:c + c:a + b:a	CCC	3C	1a-1b	Triple Cambio
547	23	b:c + c:a + b:a	CCC	3C	2a-2b	Triple Cambio
548	22	c:c + c:a + b:a	PCC	2C	2a-1b-1c	Permanencia + Doble Cambio
549	20	_:c + c:a + c:a	ICC	-1E 1P 3C	-1_ 2a -1c	Instanciación + Doble Cambio
550	18	a:c + c:a + c:a	CCC	3C	1a -1c	Triple Cambio
551	20	b:c + c:a + c:a	CCC	3C	2a-1b-1c	Triple Cambio
552	19	c:c + c:a + c:a	PCC	2C	2a -2c	Permanencia + Doble Cambio
553	27	_:b + _:b + _:a	III	-3E 3P 3C	-3_ 1a 2b	Triple Instanciación
554	23	a:b + _:b + _:a	CII	-2E 2P 3C	-2_ 2b	Doble Instanciación + Cambio
555	16	b:b + _:b + _:a	PII	-2E 2P 2C	-2_ 1a 1b	Permanencia + Doble Instanciación
556	31	c:b + _:b + _:a	CII	-2E 2P 3C	-2_ 1a 2b-1c	Doble Instanciación + Cambio
557	20	a:b + a:b + _:a	CCI	-1E 1P 3C	-1_ -1a 2b	Instanciación + Doble Cambio
558	13	b:b + a:b + _:a	PCI	-1E 1P 2C	-1_ 1b	Permanencia + Instanciación + Cambio
559	24	c:b + a:b + _:a	CCI	-1E 1P 3C	-1_ 2b-1c	Instanciación + Doble Cambio
560	10	b:b + b:b + _:a	PPI	-1E 1P 1C	-1_ 1a	Doble Permanencia + Instanciación
561	17	c:b + b:b + _:a	CPI	-1E 1P 2C	-1_ 1a 1b-1c	Permanencia + Instanciación + Cambio
562	28	c:b + c:b + _:a	CCI	-1E 1P 3C	-1_ 1a 2b-2c	Instanciación + Doble Cambio
563	26	_:b + _:b + a:a	IIP	-2E 2P 2C	-2_ 2b	Permanencia + Doble Instanciación
564	22	a:b + _:b + a:a	CIP	-1E 1P 2C	-1_ -1a 2b	Permanencia + Instanciación + Cambio
565	15	b:b + _:b + a:a	PIP	-1E 1P 1C	-1_ 1b	Doble Permanencia + Instanciación
566	30	c:b + _:b + a:a	CIP	-1E 1P 2C	-1_ 2b-1c	Permanencia + Instanciación + Cambio
567	19	a:b + a:b + a:a	CCP	2C	-2a 2b	Permanencia + Doble Cambio
568	12	b:b + a:b + a:a	PCP	1C	-1a 1b	Doble Permanencia + Cambio
569	22	c:b + a:b + a:a	CCP	2C	-1a 2b-1c	Permanencia + Doble Cambio
570	9	b:b + b:b + a:a	PPP			Triple Permanencia
571	15	c:b + b:b + a:a	CPP	1C	1b-1c	Doble Permanencia + Cambio
572	26	c:b + c:b + a:a	CCP	2C	2b-2c	Permanencia + Doble Cambio
573	25	_:b + _:b + b:a	IIC	-2E 2P 3C	-2_ 1a 1b	Doble Instanciación + Cambio
574	21	a:b + _:b + b:a	CIC	-1E 1P 3C	-1_ 1b	Instanciación + Doble Cambio
575	14	b:b + _:b + b:a	PIC	-1E 1P 2C	-1_ 1a	Permanencia + Instanciación + Cambio
576	29	c:b + _:b + b:a	CIC	-1E 1P 3C	-1_ 1a 1b-1c	Instanciación + Doble Cambio
577	18	a:b + a:b + b:a	CCC	3C	-1a 1b	Triple Cambio
578	11	b:b + a:b + b:a	PCC	2C		Permanencia + Doble Cambio
579	21	c:b + a:b + b:a	CCC	3C	1b-1c	Triple Cambio
580	8	b:b + b:b + b:a	PPC	1C	1a-1b	Doble Permanencia + Cambio
581	14	c:b + b:b + b:a	CPC	2C	1a -1c	Permanencia + Doble Cambio
582	25	c:b + c:b + b:a	CCC	3C	1a 1b-2c	Triple Cambio
583	28	_:b + _:b + c:a	IIC	-2E 2P 3C	-2_ 1a 2b-1c	Doble Instanciación + Cambio
584	24	a:b + _:b + c:a	CIC	-1E 1P 3C	-1_ 2b-1c	Instanciación + Doble Cambio
585	17	b:b + _:b + c:a	PIC	-1E 1P 2C	-1_ 1a 1b-1c	Permanencia + Instanciación + Cambio
586	31	c:b + _:b + c:a	CIC	-1E 1P 3C	-1_ 1a 2b-2c	Instanciación + Doble Cambio
587	20	a:b + a:b + c:a	CCC	3C	-1a 2b-1c	Triple Cambio
588	13	b:b + a:b + c:a	PCC	2C	1b-1c	Permanencia + Doble Cambio
589	23	c:b + a:b + c:a	CCC	3C	2b-2c	Triple Cambio
590	10	b:b + b:b + c:a	PPC	1C	1a -1c	Doble Permanencia + Cambio
591	16	c:b + b:b + c:a	CPC	2C	1a 1b-2c	Permanencia + Doble Cambio
592	27	c:b + c:b + c:a	CCC	3C	1a 2b-3c	Triple Cambio
593	47	_:c + _:b + _:a	III	-3E 3P 3C	-3_ 1a 1b 1c	Triple Instanciación
594	46	a:c + _:b + _:a	CII	-2E 2P 3C	-2_ 1b 1c	Doble Instanciación + Cambio
595	46	b:c + _:b + _:a	CII	-2E 2P 3C	-2_ 1a 1c	Doble Instanciación + Cambio
596	40	c:c + _:b + _:a	PII	-2E 2P 2C	-2_ 1a 1b	Permanencia + Doble Instanciación
597	46	_:c + a:b + _:a	ICI	-2E 2P 3C	-2_ 1b 1c	Doble Instanciación + Cambio
598	45	a:c + a:b + _:a	CCI	-1E 1P 3C	-1_ -1a 1b 1c	Instanciación + Doble Cambio
599	44	b:c + a:b + _:a	CCI	-1E 1P 3C	-1_ 1c	Instanciación + Doble Cambio

600	39	c:c + a:b + _:a	PCI	-1E	1P	2C	-1_	1b	Permanencia + Instanciación + Cambio
601	40	_:c + b:b + _:a	IPI	-2E	2P	2C	-2_	1a 1c	Permanencia + Doble Instanciación
602	39	a:c + b:b + _:a	CPI	-1E	1P	2C	-1_	1c	Permanencia + Instanciación + Cambio
603	35	b:c + b:b + _:a	CPI	-1E	1P	2C	-1_	1a-1b 1c	Permanencia + Instanciación + Cambio
604	37	c:c + b:b + _:a	PPI	-1E	1P	1C	-1_	1a	Doble Permanencia + Instanciación
605	46	_:c + c:b + _:a	ICI	-2E	2P	3C	-2_	1a 1b	Doble Instanciación + Cambio
606	44	a:c + c:b + _:a	CCI	-1E	1P	3C	-1_	1b	Instanciación + Doble Cambio
607	42	b:c + c:b + _:a	CCI	-1E	1P	3C	-1_	1a	Instanciación + Doble Cambio
608	35	c:c + c:b + _:a	PCI	-1E	1P	2C	-1_	1a 1b-1c	Permanencia + Instanciación + Cambio
609	40	_:c + _:b + a:a	IIP	-2E	2P	2C	-2_	1b 1c	Permanencia + Doble Instanciación
610	35	a:c + _:b + a:a	CIP	-1E	1P	2C	-1_	1a 1b 1c	Permanencia + Instanciación + Cambio
611	39	b:c + _:b + a:a	CIP	-1E	1P	2C	-1_	1c	Permanencia + Instanciación + Cambio
612	37	c:c + _:b + a:a	PIP	-1E	1P	1C	-1_	1b	Doble Permanencia + Instanciación
613	35	_:c + a:b + a:a	ICP	-1E	1P	2C	-1_	1a 1b 1c	Permanencia + Instanciación + Cambio
614	32	a:c + a:b + a:a	CCP			2C	-2a	1b 1c	Permanencia + Doble Cambio
615	33	b:c + a:b + a:a	CCP			2C	-1a	1c	Permanencia + Doble Cambio
616	34	c:c + a:b + a:a	PCP			1C	-1a	1b	Doble Permanencia + Cambio
617	37	_:c + b:b + a:a	IPP	-1E	1P	1C	-1_	1c	Doble Permanencia + Instanciación
618	34	a:c + b:b + a:a	CPP			1C	-1a	1c	Doble Permanencia + Cambio
619	34	b:c + b:b + a:a	CPP			1C		-1b 1c	Doble Permanencia + Cambio
620	36	c:c + b:b + a:a	PPP						Triple Permanencia
621	39	_:c + c:b + a:a	ICP	-1E	1P	2C	-1_	1b	Permanencia + Instanciación + Cambio
622	33	a:c + c:b + a:a	CCP			2C	-1a	1b	Permanencia + Doble Cambio
623	38	b:c + c:b + a:a	CCP			2C			Permanencia + Doble Cambio
624	34	c:c + c:b + a:a	PCP			1C		1b-1c	Doble Permanencia + Cambio
625	46	_:c + _:b + b:a	IIC	-2E	2P	3C	-2_	1a 1c	Doble Instanciación + Cambio
626	44	a:c + _:b + b:a	CIC	-1E	1P	3C	-1_	1c	Instanciación + Doble Cambio
627	45	b:c + _:b + b:a	CIC	-1E	1P	3C	-1_	1a-1b 1c	Instanciación + Doble Cambio
628	39	c:c + _:b + b:a	PIC	-1E	1P	2C	-1_	1a	Permanencia + Instanciación + Cambio
629	42	_:c + a:b + b:a	ICC	-1E	1P	3C	-1_	1c	Instanciación + Doble Cambio
630	41	a:c + a:b + b:a	CCC			3C	-1a	1c	Triple Cambio
631	41	b:c + a:b + b:a	CCC			3C		-1b 1c	Triple Cambio
632	38	c:c + a:b + b:a	PCC			2C			Permanencia + Doble Cambio
633	35	_:c + b:b + b:a	IPC	-1E	1P	2C	-1_	1a-1b 1c	Permanencia + Instanciación + Cambio
634	33	a:c + b:b + b:a	CPC			2C		-1b 1c	Permanencia + Doble Cambio
635	32	b:c + b:b + b:a	CPC			2C		1a-2b 1c	Permanencia + Doble Cambio
636	34	c:c + b:b + b:a	PPC			1C		1a-1b	Doble Permanencia + Cambio
637	44	_:c + c:b + b:a	ICC	-1E	1P	3C	-1_	1a	Instanciación + Doble Cambio
638	43	a:c + c:b + b:a	CCC			3C			Triple Cambio
639	41	b:c + c:b + b:a	CCC			3C		1a-1b	Triple Cambio
640	33	c:c + c:b + b:a	PCC			2C		1a -1c	Permanencia + Doble Cambio
641	46	_:c + _:b + c:a	IIC	-2E	2P	3C	-2_	1a 1b	Doble Instanciación + Cambio
642	42	a:c + _:b + c:a	CIC	-1E	1P	3C	-1_	1b	Instanciación + Doble Cambio
643	44	b:c + _:b + c:a	CIC	-1E	1P	3C	-1_	1a	Instanciación + Doble Cambio
644	35	c:c + _:b + c:a	PIC	-1E	1P	2C	-1_	1a 1b-1c	Permanencia + Instanciación + Cambio
645	44	_:c + a:b + c:a	ICC	-1E	1P	3C	-1_	1b	Instanciación + Doble Cambio
646	41	a:c + a:b + c:a	CCC			3C	-1a	1b	Triple Cambio
647	43	b:c + a:b + c:a	CCC			3C			Triple Cambio
648	33	c:c + a:b + c:a	PCC			2C		1b-1c	Permanencia + Doble Cambio
649	39	_:c + b:b + c:a	IPC	-1E	1P	2C	-1_	1a	Permanencia + Instanciación + Cambio
650	38	a:c + b:b + c:a	CPC			2C			Permanencia + Doble Cambio
651	33	b:c + b:b + c:a	CPC			2C		1a-1b	Permanencia + Doble Cambio
652	34	c:c + b:b + c:a	PPC			1C		1a -1c	Doble Permanencia + Cambio
653	45	_:c + c:b + c:a	ICC	-1E	1P	3C	-1_	1a 1b-1c	Instanciación + Doble Cambio
654	41	a:c + c:b + c:a	CCC			3C		1b-1c	Triple Cambio
655	41	b:c + c:b + c:a	CCC			3C		1a -1c	Triple Cambio
656	32	c:c + c:b + c:a	PCC			2C		1a 1b-2c	Permanencia + Doble Cambio
657	27	_:c + _:c + _:a	III	-3E	3P	3C	-3_	1a 2c	Triple Instanciación
658	23	a:c + _:c + _:a	CII	-2E	2P	3C	-2_	2c	Doble Instanciación + Cambio
659	31	b:c + _:c + _:a	CII	-2E	2P	3C	-2_	1a-1b 2c	Doble Instanciación + Cambio
660	16	c:c + _:c + _:a	PII	-2E	2P	2C	-2_	1a 1c	Permanencia + Doble Instanciación
661	20	a:c + a:c + _:a	CCI	-1E	1P	3C	-1_	-1a 2c	Instanciación + Doble Cambio
662	24	b:c + a:c + _:a	CCI	-1E	1P	3C	-1_	-1b 2c	Instanciación + Doble Cambio
663	13	c:c + a:c + _:a	PCI	-1E	1P	2C	-1_	1c	Permanencia + Instanciación + Cambio
664	28	b:c + b:c + _:a	CCI	-1E	1P	3C	-1_	1a-2b 2c	Instanciación + Doble Cambio
665	17	c:c + b:c + _:a	PCI	-1E	1P	2C	-1_	1a-1b 1c	Permanencia + Instanciación + Cambio
666	10	c:c + c:c + _:a	PPI	-1E	1P	1C	-1_	1a	Doble Permanencia + Instanciación
667	26	_:c + _:c + a:a	IIP	-2E	2P	2C	-2_	2c	Permanencia + Doble Instanciación
668	22	a:c + _:c + a:a	CIP	-1E	1P	2C	-1_	-1a 2c	Permanencia + Instanciación + Cambio
669	30	b:c + _:c + a:a	CIP	-1E	1P	2C	-1_	-1b 2c	Permanencia + Instanciación + Cambio
670	15	c:c + _:c + a:a	PIP	-1E	1P	1C	-1_	1c	Doble Permanencia + Instanciación
671	19	a:c + a:c + a:a	CCP			2C	-2a	2c	Permanencia + Doble Cambio
672	22	b:c + a:c + a:a	CCP			2C	-1a-1b	2c	Permanencia + Doble Cambio
673	12	c:c + a:c + a:a	PCP			1C	-1a	1c	Doble Permanencia + Cambio
674	26	b:c + b:c + a:a	CCP			2C		-2b 2c	Permanencia + Doble Cambio
675	15	c:c + b:c + a:a	PCP			1C		-1b 1c	Doble Permanencia + Cambio
676	9	c:c + c:c + a:a	PPP						Triple Permanencia

677	28	_ :c + _ :c + b:a	IIC	-2E	2P	3C	-2_	1a-1b	2c	Doble	Instanciación	+ Cambio
678	24	a:c + _ :c + b:a	CIC	-1E	1P	3C	-1_	-1b	2c	Instanciación	+ Doble Cambio	
679	31	b:c + _ :c + b:a	CIC	-1E	1P	3C	-1_	1a-2b	2c	Instanciación	+ Doble Cambio	
680	17	c:c + _ :c + b:a	PIC	-1E	1P	2C	-1_	1a-1b	1c	Permanencia	+ Instanciación + Cambio	
681	20	a:c + a:c + b:a	CCC			3C		-1a-1b	2c	Triple Cambio		
682	23	b:c + a:c + b:a	CCC			3C		-2b	2c	Triple Cambio		
683	13	c:c + a:c + b:a	PCC			2C		-1b	1c	Permanencia	+ Doble Cambio	
684	27	b:c + b:c + b:a	CCC			3C		1a-3b	2c	Triple Cambio		
685	16	c:c + b:c + b:a	PCC			2C		1a-2b	1c	Permanencia	+ Doble Cambio	
686	10	c:c + c:c + b:a	PPC			1C		1a-1b		Doble Permanencia	+ Cambio	
687	25	_ :c + _ :c + c:a	IIC	-2E	2P	3C	-2_	1a	1c	Doble	Instanciación + Cambio	
688	21	a:c + _ :c + c:a	CIC	-1E	1P	3C	-1_		1c	Instanciación	+ Doble Cambio	
689	29	b:c + _ :c + c:a	CIC	-1E	1P	3C	-1_	1a-1b	1c	Instanciación	+ Doble Cambio	
690	14	c:c + _ :c + c:a	PIC	-1E	1P	2C	-1_	1a		Permanencia	+ Instanciación + Cambio	
691	18	a:c + a:c + c:a	CCC			3C		-1a	1c	Triple Cambio		
692	21	b:c + a:c + c:a	CCC			3C		-1b	1c	Triple Cambio		
693	11	c:c + a:c + c:a	PCC			2C				Permanencia	+ Doble Cambio	
694	25	b:c + b:c + c:a	CCC			3C		1a-2b	1c	Triple Cambio		
695	14	c:c + b:c + c:a	PCC			2C		1a-1b		Permanencia	+ Doble Cambio	
696	8	c:c + c:c + c:a	PPC			1C		1a	-1c	Doble Permanencia	+ Cambio	
697	5	_ :b + _ :b + _ :b	III	-3E	3P	3C	-3_		3b	Triple	Instanciación	
698	6	a:b + _ :b + _ :b	CII	-2E	2P	3C	-2_	-1a	3b	Doble	Instanciación + Cambio	
699	3	b:b + _ :b + _ :b	PII	-2E	2P	2C	-2_		2b	Permanencia	+ Doble Instanciación	
700	6	c:b + _ :b + _ :b	CII	-2E	2P	3C	-2_		3b-1c	Doble	Instanciación + Cambio	
701	6	a:b + a:b + _ :b	CCI	-1E	1P	3C	-1_	-2a	3b	Instanciación	+ Doble Cambio	
702	4	b:b + a:b + _ :b	PCI	-1E	1P	2C	-1_	-1a	2b	Permanencia	+ Instanciación + Cambio	
703	7	c:b + a:b + _ :b	CCI	-1E	1P	3C	-1_	-1a	3b-1c	Instanciación	+ Doble Cambio	
704	2	b:b + b:b + _ :b	PPI	-1E	1P	1C	-1_		1b	Doble Permanencia	+ Instanciación	
705	4	c:b + b:b + _ :b	CPI	-1E	1P	2C	-1_		2b-1c	Permanencia	+ Instanciación + Cambio	
706	6	c:b + c:b + _ :b	CCI	-1E	1P	3C	-1_		3b-2c	Instanciación	+ Doble Cambio	
707	5	a:b + a:b + a:b	CCC			3C		-3a	3b	Triple Cambio		
708	3	b:b + a:b + a:b	PCC			2C		-2a	2b	Permanencia	+ Doble Cambio	
709	6	c:b + a:b + a:b	CCC			3C		-2a	3b-1c	Triple Cambio		
710	2	b:b + b:b + a:b	PPC			1C		-1a	1b	Doble Permanencia	+ Cambio	
711	4	c:b + b:b + a:b	CPC			2C		-1a	2b-1c	Permanencia	+ Doble Cambio	
712	6	c:b + c:b + a:b	CCC			3C		-1a	3b-2c	Triple Cambio		
713	1	b:b + b:b + b:b	PPP							Triple Permanencia		
714	2	c:b + b:b + b:b	CPP			1C		1b-1c		Doble Permanencia	+ Cambio	
715	3	c:b + c:b + b:b	CCP			2C		2b-2c		Permanencia	+ Doble Cambio	
716	5	c:b + c:b + c:b	CCC			3C		3b-3c		Triple Cambio		
717	27	_ :c + _ :b + _ :b	III	-3E	3P	3C	-3_		2b 1c	Triple	Instanciación	
718	28	a:c + _ :b + _ :b	CII	-2E	2P	3C	-2_	-1a	2b 1c	Doble	Instanciación + Cambio	
719	25	b:c + _ :b + _ :b	CII	-2E	2P	3C	-2_		1b 1c	Doble	Instanciación + Cambio	
720	26	c:c + _ :b + _ :b	PII	-2E	2P	2C	-2_		2b	Permanencia	+ Doble Instanciación	
721	31	_ :c + a:b + _ :b	ICI	-2E	2P	3C	-2_	-1a	2b 1c	Doble	Instanciación + Cambio	
722	31	a:c + a:b + _ :b	CCI	-1E	1P	3C	-1_	-2a	2b 1c	Instanciación	+ Doble Cambio	
723	29	b:c + a:b + _ :b	CCI	-1E	1P	3C	-1_	-1a	1b 1c	Instanciación	+ Doble Cambio	
724	30	c:c + a:b + _ :b	PCI	-1E	1P	2C	-1_	-1a	2b	Permanencia	+ Instanciación + Cambio	
725	16	_ :c + b:b + _ :b	IPI	-2E	2P	2C	-2_		1b 1c	Permanencia	+ Doble Instanciación	
726	17	a:c + b:b + _ :b	CPI	-1E	1P	2C	-1_	-1a	1b 1c	Permanencia	+ Instanciación + Cambio	
727	14	b:c + b:b + _ :b	CPI	-1E	1P	2C	-1_		1c	Permanencia	+ Instanciación + Cambio	
728	15	c:c + b:b + _ :b	PPI	-1E	1P	1C	-1_		1b	Doble Permanencia	+ Instanciación	
729	23	_ :c + c:b + _ :b	ICI	-2E	2P	3C	-2_		2b	Doble	Instanciación + Cambio	
730	24	a:c + c:b + _ :b	CCI	-1E	1P	3C	-1_	-1a	2b	Instanciación	+ Doble Cambio	
731	21	b:c + c:b + _ :b	CCI	-1E	1P	3C	-1_		1b	Instanciación	+ Doble Cambio	
732	22	c:c + c:b + _ :b	PCI	-1E	1P	2C	-1_		2b-1c	Permanencia	+ Instanciación + Cambio	
733	28	_ :c + a:b + a:b	ICC	-1E	1P	3C	-1_	-2a	2b 1c	Instanciación	+ Doble Cambio	
734	27	a:c + a:b + a:b	CCC			3C		-3a	2b 1c	Triple Cambio		
735	25	b:c + a:b + a:b	CCC			3C		-2a	1b 1c	Triple Cambio		
736	26	c:c + a:b + a:b	PCC			2C		-2a	2b	Permanencia	+ Doble Cambio	
737	17	_ :c + b:b + a:b	IPC	-1E	1P	2C	-1_	-1a	1b 1c	Permanencia	+ Instanciación + Cambio	
738	16	a:c + b:b + a:b	CPC			2C		-2a	1b 1c	Permanencia	+ Doble Cambio	
739	14	b:c + b:b + a:b	CPC			2C		-1a	1c	Permanencia	+ Doble Cambio	
740	15	c:c + b:b + a:b	PPC			1C		-1a	1b	Doble Permanencia	+ Cambio	
741	24	_ :c + c:b + a:b	ICC	-1E	1P	3C	-1_	-1a	2b	Instanciación	+ Doble Cambio	
742	23	a:c + c:b + a:b	CCC			3C		-2a	2b	Triple Cambio		
743	21	b:c + c:b + a:b	CCC			3C		-1a	1b	Triple Cambio		
744	22	c:c + c:b + a:b	PCC			2C		-1a	2b-1c	Permanencia	+ Doble Cambio	
745	10	_ :c + b:b + b:b	IPP	-1E	1P	1C	-1_		1c	Doble Permanencia	+ Instanciación	
746	10	a:c + b:b + b:b	CPP			1C		-1a	1c	Doble Permanencia	+ Cambio	
747	8	b:c + b:b + b:b	CPP			1C		-1b	1c	Doble Permanencia	+ Cambio	
748	9	c:c + b:b + b:b	PPP							Triple Permanencia		
749	13	_ :c + c:b + b:b	ICP	-1E	1P	2C	-1_		1b	Permanencia	+ Instanciación + Cambio	
750	13	a:c + c:b + b:b	CCP			2C		-1a	1b	Permanencia	+ Doble Cambio	
751	11	b:c + c:b + b:b	CCP			2C				Permanencia	+ Doble Cambio	
752	12	c:c + c:b + b:b	PCP			1C		1b-1c		Doble Permanencia	+ Cambio	
753	20	_ :c + c:b + c:b	ICC	-1E	1P	3C	-1_		2b-1c	Instanciación	+ Doble Cambio	

754	20	a:c + c:b + c:b	CCC	3C	-1a 2b-1c	Triple Cambio
755	18	b:c + c:b + c:b	CCC	3C	1b-1c	Triple Cambio
756	19	c:c + c:b + c:b	PCC	2C	2b-2c	Permanencia + Doble Cambio
757	27	_:c + _:c + _:b	III	-3E 3P 3C	-3_ 1b 2c	Triple Instanciación
758	31	a:c + _:c + _:b	CII	-2E 2P 3C	-2_-1a 1b 2c	Doble Instanciación + Cambio
759	23	b:c + _:c + _:b	CII	-2E 2P 3C	-2_ 2c	Doble Instanciación + Cambio
760	16	c:c + _:c + _:b	PII	-2E 2P 2C	-2_ 1b 1c	Permanencia + Doble Instanciación
761	28	a:c + a:c + _:b	CCI	-1E 1P 3C	-1_-2a 1b 2c	Instanciación + Doble Cambio
762	24	b:c + a:c + _:b	CCI	-1E 1P 3C	-1_-1a 2c	Instanciación + Doble Cambio
763	17	c:c + a:c + _:b	PCI	-1E 1P 2C	-1_-1a 1b 1c	Permanencia + Instanciación + Cambio
764	20	b:c + b:c + _:b	CCI	-1E 1P 3C	-1_ -1b 2c	Instanciación + Doble Cambio
765	13	c:c + b:c + _:b	PCI	-1E 1P 2C	-1_ 1c	Permanencia + Instanciación + Cambio
766	10	c:c + c:c + _:b	PPI	-1E 1P 1C	-1_ 1b	Doble Permanencia + Instanciación
767	28	_:c + _:c + a:b	IIC	-2E 2P 3C	-2_-1a 1b 2c	Doble Instanciación + Cambio
768	31	a:c + _:c + a:b	CIC	-1E 1P 3C	-1_-2a 1b 2c	Instanciación + Doble Cambio
769	24	b:c + _:c + a:b	CIC	-1E 1P 3C	-1_-1a 2c	Instanciación + Doble Cambio
770	17	c:c + _:c + a:b	PIC	-1E 1P 2C	-1_-1a 1b 1c	Permanencia + Instanciación + Cambio
771	27	a:c + a:c + a:b	CCC	3C	-3a 1b 2c	Triple Cambio
772	23	b:c + a:c + a:b	CCC	3C	-2a 2c	Triple Cambio
773	16	c:c + a:c + a:b	PCC	2C	-2a 1b 1c	Permanencia + Doble Cambio
774	20	b:c + b:c + a:b	CCC	3C	-1a-1b 2c	Triple Cambio
775	13	c:c + b:c + a:b	PCC	2C	-1a 1c	Permanencia + Doble Cambio
776	10	c:c + c:c + a:b	PPC	1C	-1a 1b	Doble Permanencia + Cambio
777	26	_:c + _:c + b:b	IIP	-2E 2P 2C	-2_ 2c	Permanencia + Doble Instanciación
778	30	a:c + _:c + b:b	CIP	-1E 1P 2C	-1_-1a 2c	Permanencia + Instanciación + Cambio
779	22	b:c + _:c + b:b	CIP	-1E 1P 2C	-1_ -1b 2c	Permanencia + Instanciación + Cambio
780	15	c:c + _:c + b:b	PIP	-1E 1P 1C	-1_ 1c	Doble Permanencia + Instanciación
781	26	a:c + a:c + b:b	CCP	2C	-2a 2c	Permanencia + Doble Cambio
782	22	b:c + a:c + b:b	CCP	2C	-1a-1b 2c	Permanencia + Doble Cambio
783	15	c:c + a:c + b:b	PCP	1C	-1a 1c	Doble Permanencia + Cambio
784	19	b:c + b:c + b:b	CCP	2C	-2b 2c	Permanencia + Doble Cambio
785	12	c:c + b:c + b:b	PCP	1C	-1b 1c	Doble Permanencia + Cambio
786	9	c:c + c:c + b:b	PPP			Triple Permanencia
787	25	_:c + _:c + c:b	IIC	-2E 2P 3C	-2_ 1b 1c	Doble Instanciación + Cambio
788	29	a:c + _:c + c:b	CIC	-1E 1P 3C	-1_-1a 1b 1c	Instanciación + Doble Cambio
789	21	b:c + _:c + c:b	CIC	-1E 1P 3C	-1_ 1c	Instanciación + Doble Cambio
790	14	c:c + _:c + c:b	PIC	-1E 1P 2C	-1_ 1b	Permanencia + Instanciación + Cambio
791	25	a:c + a:c + c:b	CCC	3C	-2a 1b 1c	Triple Cambio
792	21	b:c + a:c + c:b	CCC	3C	-1a 1c	Triple Cambio
793	14	c:c + a:c + c:b	PCC	2C	-1a 1b	Permanencia + Doble Cambio
794	18	b:c + b:c + c:b	CCC	3C	-1b 1c	Triple Cambio
795	11	c:c + b:c + c:b	PCC	2C		Permanencia + Doble Cambio
796	8	c:c + c:c + c:b	PPC	1C	1b-1c	Doble Permanencia + Cambio
797	5	_:c + _:c + _:c	III	-3E 3P 3C	-3_ 3c	Triple Instanciación
798	6	a:c + _:c + _:c	CII	-2E 2P 3C	-2_-1a 3c	Doble Instanciación + Cambio
799	6	b:c + _:c + _:c	CII	-2E 2P 3C	-2_ -1b 3c	Doble Instanciación + Cambio
800	3	c:c + _:c + _:c	PII	-2E 2P 2C	-2_ 2c	Permanencia + Doble Instanciación
801	6	a:c + a:c + _:c	CCI	-1E 1P 3C	-1_-2a 3c	Instanciación + Doble Cambio
802	7	b:c + a:c + _:c	CCI	-1E 1P 3C	-1_-1a-1b 3c	Instanciación + Doble Cambio
803	4	c:c + a:c + _:c	PCI	-1E 1P 2C	-1_-1a 2c	Permanencia + Instanciación + Cambio
804	6	b:c + b:c + _:c	CCI	-1E 1P 3C	-1_ -2b 3c	Instanciación + Doble Cambio
805	4	c:c + b:c + _:c	PCI	-1E 1P 2C	-1_ -1b 2c	Permanencia + Instanciación + Cambio
806	2	c:c + c:c + _:c	PPI	-1E 1P 1C	-1_ 1c	Doble Permanencia + Instanciación
807	5	a:c + a:c + a:c	CCC	3C	-3a 3c	Triple Cambio
808	6	b:c + a:c + a:c	CCC	3C	-2a-1b 3c	Triple Cambio
809	3	c:c + a:c + a:c	PCC	2C	-2a 2c	Permanencia + Doble Cambio
810	6	b:c + b:c + a:c	CCC	3C	-1a-2b 3c	Triple Cambio
811	4	c:c + b:c + a:c	PCC	2C	-1a-1b 2c	Permanencia + Doble Cambio
812	2	c:c + c:c + a:c	PPC	1C	-1a 1c	Doble Permanencia + Cambio
813	5	b:c + b:c + b:c	CCC	3C	-3b 3c	Triple Cambio
814	3	c:c + b:c + b:c	PCC	2C	-2b 2c	Permanencia + Doble Cambio
815	2	c:c + c:c + b:c	PPC	1C	-1b 1c	Doble Permanencia + Cambio
816	1	c:c + c:c + c:c	PPP			Triple Permanencia

Se realizaron 4096 combinaciones, existen 816 reglas independientes = permutaciones de primitivas diferentes

Existen 3280 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos
0_ 0a 0b 0c

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
0E 0P 2.25C = 75.00% Cambio 0.75I = 25.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
0.75P = 25.00% 0.56I = 18.75% 1.12C = 37.50% 0.56F = 18.75%

Ejemplos de aplicación: Existen 47 patrones de transición

1	_:_ + _:_ + _:_	[_,_,_:_,_,_] PPP	P(_)	P(_)	P(_)	Triple Permanencia
2	a:_ + _:_ + _:_	[a,_,_:_,_,_] FFP	F(a)	P(_)	P(_)	Doble Permanencia + Finalización
3	a:_ + a:_ + _:_	[a,a,_,_:_,_] FFP	F(a)	F(a)	P(_)	Permanencia + Doble Finalización
4	b:_ + a:_ + _:_	[b,a,_,_:_,_] FFP	F(b)	F(a)	P(_)	Permanencia + Doble Finalización
5	a:_ + a:_ + a:_	[a,a,a,_,_:_,_] FFF	F(a)	F(a)	F(a)	Triple Finalización
6	b:_ + a:_ + a:_	[b,a,a,_,_:_,_] FFF	F(b)	F(a)	F(a)	Triple Finalización
7	c:_ + b:_ + a:_	[c,b,a,_,_:_,_] FFF	F(c)	F(b)	F(a)	Triple Finalización
8	_:_ + _:_ + _:_	[_,_,_:a,_,_,_] IPP	I(a)	P(_)	P(_)	Doble Permanencia + Instanciación
9	a:a + _:_ + _:_	[a,_,_:a,_,_,_] PPP	P(a)	P(_)	P(_)	Triple Permanencia
10	b:a + _:_ + _:_	[b,_,_:a,_,_,_] CPP	C(b,a)	P(_)	P(_)	Doble Permanencia + Cambio
11	_:_ + a:_ + _:_	[_,a,_,_:a,_,_,_] IFP	I(a)	F(a)	P(_)	Permanencia + Instanciación + Finalización
12	a:a + a:_ + _:_	[a,a,_,_:a,_,_,_] PFP	P(a)	F(a)	P(_)	Doble Permanencia + Finalización
13	b:a + a:_ + _:_	[b,a,_,_:a,_,_,_] CFP	C(b,a)	F(a)	P(_)	Permanencia + Cambio + Finalización
14	_:_ + b:_ + _:_	[_,b,_,_:a,_,_,_] IFP	I(a)	F(b)	P(_)	Permanencia + Instanciación + Finalización
15	a:a + b:_ + _:_	[a,b,_,_:a,_,_,_] PFP	P(a)	F(b)	P(_)	Doble Permanencia + Finalización
16	b:a + b:_ + _:_	[b,b,_,_:a,_,_,_] CFP	C(b,a)	F(b)	P(_)	Permanencia + Cambio + Finalización
17	c:a + b:_ + _:_	[c,b,_,_:a,_,_,_] CFP	C(c,a)	F(b)	P(_)	Permanencia + Cambio + Finalización
18	_:_ + a:_ + a:_	[_,a,a,_,_:_,_] IFF	I(a)	F(a)	F(a)	Instanciación + Doble Finalización
19	a:a + a:_ + a:_	[a,a,a,_,_:_,_] PFF	P(a)	F(a)	F(a)	Permanencia + Doble Finalización
20	b:a + a:_ + a:_	[b,a,a,_,_:_,_] CFF	C(b,a)	F(a)	F(a)	Cambio + Doble Finalización
21	_:_ + b:_ + a:_	[_,b,a,_,_:_,_] IFF	I(a)	F(b)	F(a)	Instanciación + Doble Finalización
22	a:a + b:_ + a:_	[a,b,a,_,_:_,_] PFF	P(a)	F(b)	F(a)	Permanencia + Doble Finalización
23	b:a + b:_ + a:_	[b,b,a,_,_:_,_] CFF	C(b,a)	F(b)	F(a)	Cambio + Doble Finalización
24	c:a + b:_ + a:_	[c,b,a,_,_:_,_] CFF	C(c,a)	F(b)	F(a)	Cambio + Doble Finalización
25	_:_ + b:_ + b:_	[_,b,b,_,_:_,_] IFF	I(a)	F(b)	F(b)	Instanciación + Doble Finalización
26	a:a + b:_ + b:_	[a,b,b,_,_:_,_] PFF	P(a)	F(b)	F(b)	Permanencia + Doble Finalización
27	b:a + b:_ + b:_	[b,b,b,_,_:_,_] CFF	C(b,a)	F(b)	F(b)	Cambio + Doble Finalización
28	c:a + b:_ + b:_	[c,b,b,_,_:_,_] CFF	C(c,a)	F(b)	F(b)	Cambio + Doble Finalización
29	_:_ + c:_ + b:_	[_,c,b,_,_:_,_] IFF	I(a)	F(c)	F(b)	Instanciación + Doble Finalización
30	a:a + c:_ + b:_	[a,c,b,_,_:_,_] PFF	P(a)	F(c)	F(b)	Permanencia + Doble Finalización
31	b:a + c:_ + b:_	[b,c,b,_,_:_,_] CFF	C(b,a)	F(c)	F(b)	Cambio + Doble Finalización
32	_:_ + _:_ + _:_	[_,_,_:b,a,_,_] IIP	I(b)	I(a)	P(_)	Permanencia + Doble Instanciación
33	a:b + _:_ + _:_	[a,_,_:b,a,_,_] CIP	C(a,b)	I(a)	P(_)	Permanencia + Instanciación + Cambio
34	b:b + _:_ + _:_	[b,_,_:b,a,_,_] PIP	P(b)	I(a)	P(_)	Doble Permanencia + Instanciación
35	c:b + _:_ + _:_	[c,_,_:b,a,_,_] CIP	C(c,b)	I(a)	P(_)	Permanencia + Instanciación + Cambio
36	b:b + a:a + _:_	[b,a,_,_:b,a,_,_] PPP	P(b)	P(a)	P(_)	Triple Permanencia
37	c:b + a:a + _:_	[c,a,_,_:b,a,_,_] CPP	C(c,b)	P(a)	P(_)	Doble Permanencia + Cambio
38	a:b + b:a + _:_	[a,b,_,_:b,a,_,_] CCP	C(a,b)	C(b,a)	P(_)	Permanencia + Doble Cambio
39	c:b + b:a + _:_	[c,b,_,_:b,a,_,_] CCP	C(c,b)	C(b,a)	P(_)	Permanencia + Doble Cambio
40	c:b + c:a + _:_	[c,c,_,_:b,a,_,_] CCP	C(c,b)	C(c,a)	P(_)	Permanencia + Doble Cambio
41	_:_ + _:_ + a:_	[_,_,_:a,b,a,_,_] IIF	I(b)	I(a)	F(a)	Doble Instanciación + Finalización
42	c:b + _:_ + a:_	[c,_,_:a,b,a,_,_] CIF	C(c,b)	I(a)	F(a)	Instanciación + Cambio + Finalización
43	_:_ + b:a + a:_	[_,b,a,_,_:b,a,_,_] ICF	I(b)	C(b,a)	F(a)	Instanciación + Cambio + Finalización
44	a:b + b:a + a:_	[c,b,a,_,_:b,a,_,_] CCF	C(c,b)	C(b,a)	F(a)	Doble Cambio + Finalización
45	a:b + c:a + a:_	[a,c,a,_,_:b,a,_,_] CCF	C(a,b)	C(c,a)	F(a)	Doble Cambio + Finalización
46	c:b + c:a + a:_	[c,c,a,_,_:b,a,_,_] CCF	C(c,b)	C(c,a)	F(a)	Doble Cambio + Finalización
47	c:b + c:a + c:_	[c,c,c,_,_:b,a,_,_] CCF	C(c,b)	C(c,a)	F(c)	Doble Cambio + Finalización

Tabla de patrones de transición de SPRT(4S,3D)

1	A:A + A:A + A:A	4	1
2	A:A + A:A + B:A	36	2 +1A -1B
3	A:A + B:A + B:A	36	2 +2A -2B
4	A:A + B:A + C:A	72	3 +2A -1B -1C
5	A:B + A:B + A:B	12	2 -3A +3B
6	A:B + A:B + C:B	72	3 -2A +3B -1C
7	A:B + C:B + D:B	24	4 -1A +3B -1C -1D
8	A:A + A:A + A:B	36	2 -1A +1B
9	A:A + A:A + B:B	36	2
10	A:A + A:A + B:C	72	3 -1B +1C
11	A:A + A:B + B:A	72	2
12	A:A + A:B + B:B	72	2 -1A +1B
13	A:A + B:A + C:B	144	3 +1A -1C
14	A:A + A:B + C:A	144	3 +1B -1C
15	A:A + B:A + C:C	144	3 +1A -1B
16	A:A + B:A + B:C	144	3 +1A -2B +1C
17	A:A + B:A + C:D	144	4 +1A -1B -1C +1D
18	A:B + A:B + B:A	36	2 -1A +1B
19	A:A + A:B + A:B	36	2 -2A +2B
20	A:B + A:B + C:A	72	3 -1A +2B -1C
21	A:B + B:A + C:A	144	3 +1A -1C
22	A:A + A:B + C:B	144	3 -1A +2B -1C
23	A:B + A:C + B:C	144	3 -2A +2C
24	A:B + B:C + D:C	144	4 -1A +2C -1D

25	A:B + A:B + B:C	72	3	-2A +1B +1C
26	A:A + B:C + B:C	72	3	-2B +2C
27	A:B + A:B + A:C	72	3	-3A +2B +1C
28	A:B + A:B + C:D	72	4	-2A +2B -1C +1D
29	A:B + B:C + D:B	72	4	-1A +1B +1C -1D
30	A:A + B:C + D:C	72	4	-1B +2C -1D
31	A:B + A:C + D:B	144	4	-2A +2B +1C -1D
32	A:A + A:B + A:C	72	3	-2A +1B +1C
33	A:A + A:B + B:C	144	3	-1A +1C
34	A:A + A:B + C:C	144	3	-1A +1B
35	A:A + A:B + C:D	144	4	-1A +1B -1C +1D
36	A:A + B:B + C:C	24	3	
37	A:A + B:B + C:D	72	4	-1C +1D
38	A:A + B:C + C:B	72	3	
39	A:A + B:C + C:D	144	4	-1B +1D
40	A:A + B:C + B:D	72	4	-2B +1C +1D
41	A:B + A:C + B:A	144	3	-1A +1C
42	A:B + B:A + C:D	72	4	-1C +1D
43	A:B + B:C + C:A	48	3	
44	A:B + B:C + C:D	144	4	-1A +1D
45	A:B + A:C + D:A	72	4	-1A +1B +1C -1D
46	A:B + A:C + B:D	144	4	-2A +1C +1D
47	A:B + A:C + A:D	24	4	-3A +1B +1C +1D

Existen en promedio 3.2880859375 símbolos distintos en cada regla del SPRT

E.2.5. SPRT nRD: Sistemas de referencias de símbolos

Para generar con el script GSPRT los sistemas nRD se establece en la tabla de combinación principal del sistema \$z un único símbolo causa por cada dimensión de evaluación a realizar, seguido de los símbolos efecto deseados. Un nRD completo define en sus efectos de transición en cada dimensión de evaluación solo un identificador de cada uno de los símbolos causas disponibles en el sistema, de otro modo se configurará un sistema incompleto.

La siguiente porción de código ilustra formas de configuración de GSPRT para la implementación de sistemas nRD:

Script E.14: Configuración de GSPRT para un SPRT de referencias

```
1 ...
2 //////////////////////////////////////
3 // Representación de SPRT nRD y configuración de tablas de símbolos
4 //////////////////////////////////////
5
6 //a1) Una definición manual de Tabla de símbolos para un nRD completo
7 $o1 = array('A');
8 $o2 = array('B');
9 $o3 = array('C');
10 $o4 = array('D');
11 $p = array('A','B','C','D'); //paso completo entre símbolos
12
13 //a2) Una definición manual de Tabla de símbolos para un nRD incompleto
14 $o1 = array('A');
15 $o2 = array('B');
16 $o3 = array('C');
17 $o4 = array('D');
18 $p = array('A','B','C'); //no hay paso a la referencia 'D'
19
20 //a3) Una definición manual de Tabla de símbolos para un nRD completo redundante
21 $o1 = array('A');
22 $o2 = array('B');
23 $o3 = array('C');
24 $o4 = array('D');
25 $p = array('A','A','B','C','D'); //hay paso a mas de una referencia 'A' en el sistema
26
27 //construcción efectiva de cualquiera de los sistemas a1, a2, a3
28 $z=array($o1,$o2,$o3,$o4,$p,$p,$p,$p);
29 $vecindad=count($z)/2;
30
31 //b1) Una definición automática de Tabla de símbolos para un sistema de referencias
32 //construcción de un sistema de referencias completo de dimensión específica
33 $z=array(); //inicio de la tabla de combinaciones de símbolos
34 $cr_ref=array();
35 $dimensiones=4;
36 for($i=1;$i<=$dimensiones;$i++){ //adición de referencias únicas
37 // $ref=chr(64+$i); //usando identificadores alfabéticos
38 $ref=$i; //usando identificadores numéricos
39 $z[]=array($ref);
40 $cr_ref[]=$ref;
41 }
42 for($i=1;$i<=$dimensiones;$i++){ //adición de combinaciones con repetición de referencias
43 $z[]=$cr_ref;
44 }
45 $vecindad=$dimensiones; //vecindad=count($z)/2; //tamaño de la vecindad
46 ...
```

E.2.5.1. SPRT 1RD

Una interpretación matricial de estos resultados se presenta en la tabla D.24 en la página 472. En general representa la idempotencia o permanencia de referencias en el sistema, en semejanza al 1S1D presentado en el apartado E.2.4.1 en la página 523. La diferencia con este sistema está en que explícitamente se asegura que la instancia es la misma durante cada paso de transición.

Reporte E.11: SPRT 1RD

```
SPRT(1S,1D)
Tabla de símbolos = {1} = 1 símbolos
Dimensiones, simultaneidad, vecindario máximo = 1 = Se combinarán 2 vectores
elementos en cada vector [ 1 1 ] = 1 combinaciones
Existen 1 reglas independientes

Se usan 1 primitivas de transición
1:1

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario  Dimensión  Entrada/Causa  ->  Salida/Efecto  Notación
Principal             Celda[0]      {1}           ->  {1}           {1:1}

1 1 1:1 [1:1] P P(1)      Permanencia
Se realizaron 1 combinaciones, existen 1 reglas independientes = permutaciones de primitivas
diferentes

Existen 0 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
01

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
OE      OP      0.00C = 0.00% Cambio      1.00I = 100.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
1.00P = 100.00% 0.00I = 0.00%      0.00C = 0.00%      0.00F = 0.00%

Ejemplos de aplicación: Existen 1 patrones de transición
1 1:1 [1:1] P P(1)      Permanencia

Tabla de patrones de transición de SPRT(1S,1D)
1 A:A 1 1
Existen en promedio 1 símbolos distintos en cada regla del SPRT
```

E.2.5.2. SPRT 2RD

Una interpretación matricial de estos resultados se presenta en la tabla D.24 en la página 472. Desde este sistema en adelante es posible realizar el intercambio simultáneo o swap de referencias, que es la forma más simple de rotación. También se establece el fenómeno de la dominación de referencias.

Reporte E.12: SPRT 2RD

```
SPRT(2S,2D)
Tabla de símbolos = {1,2} = 2 símbolos
Dimensiones, simultaneidad, vecindario máximo = 2 = Se combinarán 4 vectores
elementos en cada vector [ 1 1 2 2 ] = 4 combinaciones
Este sistema implementa selección de primitivas

Se usan 4 primitivas de transición
1:1      1:2      2:1      2:2

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario  Dimensión  Entrada/Causa  ->  Salida/Efecto  Notación
Principal            Celda[0]      {1}            ->  {1,2}        {1:1,2}
Vecina 1             Celda[1]      {2}            ->  {1,2}        {2:1,2}

1 1 1:1 + 2:1 [1,2:1,1] PC P(1)  C(2,1) 1C 11-12 Permanencia + Cambio
2 2 1:2 + 2:1 [1,2:2,1] CC C(1,2) C(2,1) 2C          Doble Cambio
3 3 1:1 + 2:2 [1,2:1,2] PP P(1)  P(2)          Doble Permanencia
4 1 1:2 + 2:2 [1,2:2,2] CP C(1,2) P(2)  1C -11 12 Permanencia + Cambio
Se realizaron 4 combinaciones, existen 4 reglas independientes = permutaciones de primitivas
diferentes

Existen 0 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
  01  02

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
  0E   0P   1.00C = 50.00% Cambio   1.00I = 50.00% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
  1.00P = 50.00%  0.00I = 0.00%   1.00C = 50.00%  0.00F = 0.00%

Ejemplos de aplicación: Existen 3 patrones de transición
1 1:1 + 2:1 [1,2:1,1] PC P(1)  C(2,1) Permanencia + Cambio
2 1:2 + 2:1 [1,2:2,1] CC C(1,2) C(2,1) Doble Cambio
3 1:1 + 2:2 [1,2:1,2] PP P(1)  P(2)  Doble Permanencia

Tabla de patrones de transición de SPRT(2S,2D) particular de 4 primitivas
1 A:A + B:A 2 2 +1A -1B
2 A:B + B:A 1 2
3 A:A + B:B 1 2
Existen en promedio 2 símbolos distintos en cada regla del SPRT
```

E.2.5.3. SPRT 3RD

Una interpretación matricial de estos resultados se presenta en la tabla D.26 en la página 475. Desde este sistema en adelante es posible representar el intercambio múltiple o rotación de referencias. Adicionalmente aparecen las filas: procesos o cambios simultáneos entre referencias, son una forma de rotación inconclusa en la cual se consumen y producen las referencias que aparecen en los extremos de la estructura.

Figura E.5: Vector de SPRT nRD comprendidos por 3RD

$$M_{nRD} = \begin{pmatrix} 1RD & 2RD & 3RD \end{pmatrix}$$

Fuente: Autor.

Reporte E.13: SPRT 3RD

SPRT (3S,3D)

Tabla de símbolos = {1,2,3} = 3 símbolos

Dimensiones, simultaneidad, vecindario máximo = 3 = Se combinarán 6 vectores

elementos en cada vector [1 1 1 3 3 3] = 27 combinaciones

Este sistema implementa selección de primitivas

Se usan 9 primitivas de transición

1:1 1:2 1:3 2:1 2:2 2:3 3:1 3:2 3:3

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa	->	Salida/Efecto	Notación
Principal	Celda[0]	{1}	->	{1,2,3}	{1:1,2,3}
Vecina 1	Celda[1]	{2}	->	{1,2,3}	{2:1,2,3}
Vecina 2	Celda[2]	{3}	->	{1,2,3}	{3:1,2,3}

1	1	1:1 + 2:1 + 3:1	[1,2,3:1,1,1]	PCC	P(1)	C(2,1)	C(3,1)	2C	21-12-13	Permanencia + Doble Cambio
2	2	1:2 + 2:1 + 3:1	[1,2,3:2,1,1]	CCC	C(1,2)	C(2,1)	C(3,1)	3C	11 -13	Triple Cambio
3	2	1:3 + 2:1 + 3:1	[1,2,3:3,1,1]	CCC	C(1,3)	C(2,1)	C(3,1)	3C	11-12	Triple Cambio
4	3	1:1 + 2:2 + 3:1	[1,2,3:1,2,1]	PPC	P(1)	P(2)	C(3,1)	1C	11 -13	Doble Permanencia + Cambio
5	4	1:2 + 2:2 + 3:1	[1,2,3:2,2,1]	CPC	C(1,2)	P(2)	C(3,1)	2C	12-13	Permanencia + Doble Cambio
6	5	1:3 + 2:2 + 3:1	[1,2,3:3,2,1]	CPC	C(1,3)	P(2)	C(3,1)	2C		Permanencia + Doble Cambio
7	4	1:1 + 2:3 + 3:1	[1,2,3:1,3,1]	PCC	P(1)	C(2,3)	C(3,1)	2C	11-12	Permanencia + Doble Cambio
8	6	1:2 + 2:3 + 3:1	[1,2,3:2,3,1]	CCC	C(1,2)	C(2,3)	C(3,1)	3C		Triple Cambio
9	2	1:3 + 2:3 + 3:1	[1,2,3:3,3,1]	CCC	C(1,3)	C(2,3)	C(3,1)	3C	-12 13	Triple Cambio
10	4	1:1 + 2:1 + 3:2	[1,2,3:1,1,2]	PCC	P(1)	C(2,1)	C(3,2)	2C	11 -13	Permanencia + Doble Cambio
11	2	1:2 + 2:1 + 3:2	[1,2,3:2,1,2]	CCC	C(1,2)	C(2,1)	C(3,2)	3C	12-13	Triple Cambio
12	6	1:3 + 2:1 + 3:2	[1,2,3:3,1,2]	CCC	C(1,3)	C(2,1)	C(3,2)	3C		Triple Cambio
13	3	1:1 + 2:2 + 3:2	[1,2,3:1,2,2]	PPC	P(1)	P(2)	C(3,2)	1C	12-13	Doble Permanencia + Cambio
14	1	1:2 + 2:2 + 3:2	[1,2,3:2,2,2]	CPC	C(1,2)	P(2)	C(3,2)	2C	-11 22-13	Permanencia + Doble Cambio
15	4	1:3 + 2:2 + 3:2	[1,2,3:3,2,2]	CPC	C(1,3)	P(2)	C(3,2)	2C	-11 12	Permanencia + Doble Cambio
16	5	1:1 + 2:3 + 3:2	[1,2,3:1,3,2]	PCC	P(1)	C(2,3)	C(3,2)	2C		Permanencia + Doble Cambio
17	2	1:2 + 2:3 + 3:2	[1,2,3:2,3,2]	CCC	C(1,2)	C(2,3)	C(3,2)	3C	-11 12	Triple Cambio
18	2	1:3 + 2:3 + 3:2	[1,2,3:3,3,2]	CCC	C(1,3)	C(2,3)	C(3,2)	3C	-11 13	Triple Cambio
19	3	1:1 + 2:1 + 3:3	[1,2,3:1,1,3]	PCP	P(1)	C(2,1)	P(3)	1C	11-12	Doble Permanencia + Cambio
20	5	1:2 + 2:1 + 3:3	[1,2,3:2,1,3]	CCP	C(1,2)	C(2,1)	P(3)	2C		Permanencia + Doble Cambio
21	4	1:3 + 2:1 + 3:3	[1,2,3:3,1,3]	CCP	C(1,3)	C(2,1)	P(3)	2C	-12 13	Permanencia + Doble Cambio
22	7	1:1 + 2:2 + 3:3	[1,2,3:1,2,3]	PPP	P(1)	P(2)	P(3)			Triple Permanencia
23	3	1:2 + 2:2 + 3:3	[1,2,3:2,2,3]	CPP	C(1,2)	P(2)	P(3)	1C	-11 12	Doble Permanencia + Cambio
24	3	1:3 + 2:2 + 3:3	[1,2,3:3,2,3]	CPP	C(1,3)	P(2)	P(3)	1C	-11 13	Doble Permanencia + Cambio

```

25 3 1:1 + 2:3 + 3:3 [1,2,3:1,3,3] PCP P(1) C(2,3) P(3) 1C -12 13 Doble Permanencia +
    Cambio
26 4 1:2 + 2:3 + 3:3 [1,2,3:2,3,3] CCP C(1,2) C(2,3) P(3) 2C -11 13 Permanencia + Doble
    Cambio
27 1 1:3 + 2:3 + 3:3 [1,2,3:3,3,3] CCP C(1,3) C(2,3) P(3) 2C -11-12 23 Permanencia + Doble
    Cambio
Se realizaron 27 combinaciones, existen 27 reglas independientes = permutaciones de primitivas
diferentes

```

Existen 0 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos
01 02 03

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
0E 0P 2.00C = 66.67% Cambio 1.00I = 33.33% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
1.00P = 33.33% 0.00I = 0.00% 2.00C = 66.67% 0.00F = 0.00%

Ejemplos de aplicación: Existen 7 patrones de transición

```

1 1:1 + 2:1 + 3:1 [1,2,3:1,1,1] PCC P(1) C(2,1) C(3,1) Permanencia + Doble Cambio
2 1:2 + 2:1 + 3:1 [1,2,3:2,1,1] CCC C(1,2) C(2,1) C(3,1) Triple Cambio
3 1:1 + 2:2 + 3:1 [1,2,3:1,2,1] PPC P(1) P(2) C(3,1) Doble Permanencia + Cambio
4 1:2 + 2:2 + 3:1 [1,2,3:2,2,1] CPC C(1,2) P(2) C(3,1) Permanencia + Doble Cambio
5 1:3 + 2:2 + 3:1 [1,2,3:3,2,1] CPC C(1,3) P(2) C(3,1) Permanencia + Doble Cambio
6 1:2 + 2:3 + 3:1 [1,2,3:2,3,1] CCC C(1,2) C(2,3) C(3,1) Triple Cambio
7 1:1 + 2:2 + 3:3 [1,2,3:1,2,3] PPP P(1) P(2) P(3) Triple Permanencia

```

Tabla de patrones de transición de SPRT(3S,3D) particular de 9 primitivas

```

1 A:A + B:A + C:A 3 3 +2A -1B -1C
2 A:B + B:A + C:A 6 3 +1A -1C
3 A:A + B:A + C:C 6 3 +1A -1B
4 A:A + B:A + C:B 6 3 +1A -1C
5 A:A + B:C + C:B 3 3
6 A:B + B:C + C:A 2 3
7 A:A + B:B + C:C 1 3

```

Existen en promedio 3 símbolos distintos en cada regla del SPRT

Más allá del 3RD La cantidad de combinaciones de operaciones de las permanencias, dominaciones, intercambios y rotaciones de referencias en las reglas de transición irán en un progresivo aumento y complejidad; a largo plazo se presenta la realización de todas las formas de combinación posibles de las funciones características de los nRD en las reglas de transición del sistema.

Las permanencias permitirán la implementación de funciones en las cuales se requiere de la presencia explícita de cierto tipo de referencias específicas para la realización de la transición, en semejanza con la metáfora de alteración descrita por la MBOR, P.ej:

$R_i = 1:1 + 2:3 + 3:4 + 4:2$, rotación de 2,3,4 en presencia de 1, con $R_i \in 4RD$.

$R_j = 1:2 + 2:3 + 3:4 + 4:4 + 5:5$, una fila o rotación incompleta de tres etapas que toma a 1 y produce 4 en presencia de 4 y 5, con $R_j \in 5RD$.

$R_k = 1:2 + 2:3 + 3:4 + 4:1 + 5:5 + 6:7 + 7:6 + 8:8 + 9:8 + 10:8 + 11:8$, una rotación de las referencias 1,2,3,4, un intercambio entre 6 y 7 junto con la dominación de 8 de las demás dimensiones restantes en presencia una referencia estable 5 y 8; con $R_k \in 11RD$.

E.2.6. SPRT 2mSmD: Límites de representación de patrones de transición

Al incrementar el número de símbolos en un SPRT, aumenta el número de reglas de transición y de reglas de transición independientes que es posible expresar en ese sistema. La cantidad máxima de metasímbolos distintos que puede usarse en una regla de transición la establece la cantidad de dimensiones de evaluación que se requieran para su construcción. Una vez que no hay más metasímbolos distintos entre sí que agregar a la expresión de transición se encuentra el límite de representación de los patrones de transición; a partir de kSmD con $k \geq 2m$ no se generan nuevos tipos de patrones de transición. Esta es la razón que hace interesantes a los sistemas 2mSmD.

Los sistemas 2S1D y 4S2D pertenecen a esta clase de sistemas y han sido presentados previamente en los reportes E.3 en la página 525 y E.9 en la página 539. En los siguientes reportes se hace énfasis solo en los patrones de transición presentes en cada sistema.

E.2.6.1. SPRT 6S3D

Este sistema contiene el máximo de patrones de transición que es posible representar con tres dimensiones de evaluación. En la práctica, indica el máximo teórico del potencial de representación de transiciones posible en Homos 1.0. 4S3D es mucho menor que 6S3D, hay muchas expresiones de transición útiles que aún no se incorporan a los procesos de modelado en MBOR y Homos 1.0.

Reporte E.14: SPRT 6S3D

```
SPRT (6S,3D)
Tabla de símbolos = {_,a,b,c,d,e} = 6 símbolos
Dimensiones, simultaneidad, vecindario máximo = 3 = Se combinarán 6 vectores
elementos en cada vector [ 6 6 6 6 6 6 ] = 46656 combinaciones
Existen 8436 reglas independientes

Se usan 36 primitivas de transición
_: _   _: a   _: b   _: c   _: d   _: e   a: _   a: a   a: b   a: c   a: d   a: e
b: _   b: a   b: b   b: c   b: d   b: e   c: _   c: a   c: b   c: c   c: d   c: e
d: _   d: a   d: b   d: c   d: d   d: e   e: _   e: a   e: b   e: c   e: d   e: e

Construcción de reglas mediante configuración de símbolos en vecindad
Celda del vecindario   Dimensión   Entrada/Causa   ->   Salida/Efecto   Notación
Principal              Celda[0]       {_,a,b,c,d,e}   ->   {_,a,b,c,d,e}   {_,a,b,c,d,e:_,a,b,
c,d,e}
Vecina 1                Celda[1]       {_,a,b,c,d,e}   ->   {_,a,b,c,d,e}   {_,a,b,c,d,e:_,a,b,
c,d,e}
Vecina 2                Celda[2]       {_,a,b,c,d,e}   ->   {_,a,b,c,d,e}   {_,a,b,c,d,e:_,a,b,
c,d,e}

Se realizaron 46656 combinaciones, existen 8436 reglas independientes = permutaciones de primitivas
diferentes

Existen 38220 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT
* Balance global de símbolos
  0_   0a   0b   0c   0d   0e

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)
  0E      0P      2.50C = 83.33% Cambio      0.50I = 16.67% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización
  0.50P = 16.67%   0.42I = 13.89%   1.67C = 55.56%   0.42F = 13.89%

Ejemplos de aplicación: Existen 52 patrones de transición

Tabla de patrones de transición de SPRT(6S,3D)
1      A:A + A:A + A:A      6      1
```

2	A:A + A:A + B:A	90	2	+1A -1B	
3	A:A + B:A + B:A	90	2	+2A -2B	
4	A:A + B:A + C:A	360	3	+2A -1B -1C	
5	A:B + A:B + A:B	30	2	-3A +3B	
6	A:B + A:B + C:B	360	3	-2A +3B -1C	
7	A:B + C:B + D:B	360	4	-1A +3B -1C -1D	
8	A:A + A:A + A:B	90	2	-1A +1B	
9	A:A + A:A + B:B	90	2		
10	A:A + A:A + B:C	360	3	-1B +1C	
11	A:A + A:B + B:A	180	2		
12	A:A + A:B + B:B	180	2	-1A +1B	
13	A:A + B:A + C:B	720	3	+1A -1C	
14	A:A + A:B + C:A	720	3	+1B -1C	
15	A:A + B:A + C:C	720	3	+1A -1B	
16	A:A + B:A + B:C	720	3	+1A -2B +1C	
17	A:A + B:A + C:D	2160	4	+1A -1B -1C +1D	
18	A:B + A:B + B:A	90	2	-1A +1B	
19	A:A + A:B + A:B	90	2	-2A +2B	
20	A:B + A:B + C:A	360	3	-1A +2B -1C	
21	A:B + B:A + C:A	720	3	+1A -1C	
22	A:A + A:B + C:B	720	3	-1A +2B -1C	
23	A:B + A:C + B:C	720	3	-2A +2C	
24	A:B + B:C + D:C	2160	4	-1A +2C -1D	
25	A:B + A:B + B:C	360	3	-2A +1B +1C	
26	A:A + B:C + B:C	360	3	-2B +2C	
27	A:B + A:B + A:C	360	3	-3A +2B +1C	
28	A:B + A:B + C:D	1080	4	-2A +2B -1C +1D	
29	A:B + B:C + D:B	1080	4	-1A +1B +1C -1D	
30	A:A + B:C + D:C	1080	4	-1B +2C -1D	
31	A:B + A:C + D:B	2160	4	-2A +2B +1C -1D	
32	A:B + C:B + D:E	2160	5	-1A +2B -1C -1D +1E	
33	A:A + A:B + A:C	360	3	-2A +1B +1C	
34	A:A + A:B + B:C	720	3	-1A +1C	
35	A:A + A:B + C:C	720	3	-1A +1B	
36	A:A + A:B + C:D	2160	4	-1A +1B -1C +1D	
37	A:A + B:B + C:C	120	3		
38	A:A + B:B + C:D	1080	4	-1C +1D	
39	A:A + B:C + C:B	360	3		
40	A:A + B:C + C:D	2160	4	-1B +1D	
41	A:A + B:C + B:D	1080	4	-2B +1C +1D	
42	A:A + B:C + D:E	2160	5	-1B +1C -1D +1E	
43	A:B + A:C + B:A	720	3	-1A +1C	
44	A:B + B:A + C:D	1080	4	-1C +1D	
45	A:B + B:C + C:A	240	3		
46	A:B + B:C + C:D	2160	4	-1A +1D	
47	A:B + A:C + D:A	1080	4	-1A +1B +1C -1D	
48	A:B + A:C + B:D	2160	4	-2A +1C +1D	
49	A:B + B:C + D:E	4320	5	-1A +1C -1D +1E	
50	A:B + A:C + A:D	360	4	-3A +1B +1C +1D	
51	A:B + A:C + D:E	2160	5	-2A +1B +1C -1D +1E	
52	A:B + C:D + E:F	720	6	-1A +1B -1C +1D -1E +1F	

Existen en promedio 3.9906121399177 simbolos distintos en cada regla del SPRT

E.2.6.2. SPRT 8S4D

En este sistema se presenta el siguiente nivel de complejidad, al adicionar de una nueva dimensión de evaluación al sistema de reglas de transición.

Aparecen nuevos tipos de patrones, que en realidad son formas mixtas los de patrones encontrados en el 6D3D y sistemas anteriores. Esta tendencia de mezclas y combinaciones de formas de transición se repite en nSmD mayores hasta el infinito.

Reporte E.15: SPRT 8S4D

SPRT(8S,4D)

Tabla de símbolos = {_,a,b,c,d,e,f,g} = 8 símbolos
Dimensiones, simultaneidad, vecindario máximo = 4 = Se combinarán 8 vectores
elementos en cada vector [8 8 8 8 8 8 8 8] = 16777216 combinaciones
Existen 766480 reglas independientes

Se usan 64 primitivas de transición

:	_:a	_:b	_:c	_:d	_:e	_:f	_:g
a:_	a:a	a:b	a:c	a:d	a:e	a:f	a:g
b:_	b:a	b:b	b:c	b:d	b:e	b:f	b:g
c:_	c:a	c:b	c:c	c:d	c:e	c:f	c:g
d:_	d:a	d:b	d:c	d:d	d:e	d:f	d:g
e:_	e:a	e:b	e:c	e:d	e:e	e:f	e:g
f:_	f:a	f:b	f:c	f:d	f:e	f:f	f:g
g:_	g:a	g:b	g:c	g:d	g:e	g:f	g:g

Construcción de reglas mediante configuración de símbolos en vecindad

Celda del vecindario	Dimensión	Entrada/Causa->Salida/Efecto	Notación
Principal Celda[0]	{_,a,b,c,d,e,f,g}	->{_,a,b,c,d,e,f,g}	{_,a,b,c,d,e,f,g:_a,b,c,d,e,f,g}
Vecina 1 Celda[1]	{_,a,b,c,d,e,f,g}	->{_,a,b,c,d,e,f,g}	{_,a,b,c,d,e,f,g:_a,b,c,d,e,f,g}
Vecina 2 Celda[2]	{_,a,b,c,d,e,f,g}	->{_,a,b,c,d,e,f,g}	{_,a,b,c,d,e,f,g:_a,b,c,d,e,f,g}
Vecina 3 Celda[3]	{_,a,b,c,d,e,f,g}	->{_,a,b,c,d,e,f,g}	{_,a,b,c,d,e,f,g:_a,b,c,d,e,f,g}

Se realizaron 16777216 combinaciones, existen 766480 reglas independientes = permutaciones de primitivas diferentes

Existen 16010736 reglas repetidas en el sistema: diferencias en orden situación / primitivas

Estadísticas del SPRT

* Balance global de símbolos

0_ 0a 0b 0c 0d 0e 0f 0g

* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)

0E 0P 3.50C = 87.50% Cambio 0.50I = 12.50% Inercia

* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y Finalización

0.50P = 12.50% 0.44I = 10.94% 2.62C = 65.62% 0.44F = 10.94%

Ejemplos de aplicación: Existen 296 patrones de transición

Tabla de patrones de transición de SPRT(8S,4D)

1	A:A + A:A + A:A + A:A	8	1
2	A:A + A:A + A:A + B:A	224	2 +1A -1B
3	A:A + A:A + B:A + B:A	336	2 +2A -2B
4	A:A + A:A + B:A + C:A	2016	3 +2A -1B -1C
5	A:A + B:A + B:A + B:A	224	2 +3A -3B
6	A:A + B:A + B:A + C:A	4032	3 +3A -2B -1C
7	A:A + B:A + C:A + D:A	6720	4 +3A -1B -1C -1D
8	A:B + A:B + A:B + A:B	56	2 -4A +4B
9	A:B + A:B + A:B + C:B	1344	3 -3A +4B -1C
10	A:B + A:B + C:B + C:B	1008	3 -2A +4B -2C
11	A:B + A:B + C:B + D:B	10080	4 -2A +4B -1C -1D
12	A:B + C:B + D:B + E:B	6720	5 -1A +4B -1C -1D -1E
13	A:A + A:A + A:A + A:B	224	2 -1A +1B
14	A:A + A:A + A:A + B:B	224	2
15	A:A + A:A + A:A + B:C	1344	3 -1B +1C
16	A:A + A:A + A:B + B:A	672	2
17	A:A + A:A + B:A + B:B	672	2 +1A -1B
18	A:A + A:A + B:A + C:B	4032	3 +1A -1C
19	A:A + A:A + A:B + C:A	4032	3 +1B -1C
20	A:A + A:A + B:A + C:C	4032	3 +1A -1B
21	A:A + A:A + B:A + B:C	4032	3 +1A -2B +1C

22	A:A + A:A + B:A + C:D	20160	4	+1A -1B -1C +1D
23	A:A + A:B + B:A + B:A	672	2	+1A -1B
24	A:A + A:B + A:B + B:B	672	2	-2A +2B
25	A:A + B:A + B:A + C:B	4032	3	+2A -1B -1C
26	A:A + A:B + B:A + C:A	8064	3	+1A -1C
27	A:A + A:B + B:B + C:B	8064	3	-1A +2B -1C
28	A:A + B:A + B:C + C:A	8064	3	+2A -2B
29	A:A + B:A + C:A + D:B	40320	4	+2A -1C -1D
30	A:A + A:B + C:A + C:A	4032	3	+1A +1B -2C
31	A:A + B:A + B:A + C:C	4032	3	+2A -2B
32	A:A + B:A + B:A + B:C	4032	3	+2A -3B +1C
33	A:A + B:A + B:A + C:D	20160	4	+2A -2B -1C +1D
34	A:A + A:B + C:A + D:A	20160	4	+1A +1B -1C -1D
35	A:A + B:A + C:A + D:D	20160	4	+2A -1B -1C
36	A:A + B:A + B:C + D:A	40320	4	+2A -2B +1C -1D
37	A:A + B:A + C:A + D:E	80640	5	+2A -1B -1C -1D +1E
38	A:B + A:B + A:B + B:A	224	2	-2A +2B
39	A:A + A:B + A:B + A:B	224	2	-3A +3B
40	A:B + A:B + A:B + C:A	1344	3	-2A +3B -1C
41	A:B + A:B + B:A + C:B	4032	3	-1A +2B -1C
42	A:A + A:B + A:B + C:B	4032	3	-2A +3B -1C
43	A:B + A:B + C:A + C:B	4032	3	-1A +3B -2C
44	A:B + A:B + C:A + D:B	20160	4	-1A +3B -1C -1D
45	A:B + A:B + B:C + C:B	4032	3	-2A +2B
46	A:A + A:B + C:B + C:B	4032	3	-1A +3B -2C
47	A:B + A:B + A:C + C:B	4032	3	-3A +3B
48	A:B + A:B + C:B + D:C	20160	4	-2A +3B -1D
49	A:B + B:A + C:A + D:A	20160	4	+2A -1C -1D
50	A:A + A:B + C:B + D:B	20160	4	-1A +3B -1C -1D
51	A:B + A:C + B:C + D:C	40320	4	-2A +3C -1D
52	A:B + B:C + D:C + E:C	80640	5	-1A +3C -1D -1E
53	A:B + A:B + A:B + B:C	1344	3	-3A +2B +1C
54	A:A + B:C + B:C + B:C	1344	3	-3B +3C
55	A:B + A:B + A:B + A:C	1344	3	-4A +3B +1C
56	A:B + A:B + A:B + C:D	6720	4	-3A +3B -1C +1D
57	A:B + A:B + B:C + D:B	20160	4	-2A +2B +1C -1D
58	A:A + B:C + B:C + D:C	20160	4	-2B +3C -1D
59	A:B + A:B + A:C + D:B	20160	4	-3A +3B +1C -1D
60	A:B + A:B + C:B + C:D	20160	4	-2A +3B -2C +1D
61	A:B + A:B + C:B + D:E	80640	5	-2A +3B -1C -1D +1E
62	A:B + B:C + D:B + E:B	26880	5	-1A +2B +1C -1D -1E
63	A:A + B:C + D:C + E:C	26880	5	-1B +3C -1D -1E
64	A:B + A:C + D:B + E:B	80640	5	-2A +3B +1C -1D -1E
65	A:B + C:B + D:B + E:F	80640	6	-1A +3B -1C -1D -1E +1F
66	A:A + A:A + A:B + A:B	336	2	-2A +2B
67	A:A + A:A + A:B + B:B	672	2	-1A +1B
68	A:A + A:A + A:B + C:B	4032	3	-1A +2B -1C
69	A:A + A:A + B:B + B:B	168	2	
70	A:A + A:A + B:B + C:B	4032	3	+1B -1C
71	A:A + A:A + B:C + B:C	2016	3	-2B +2C
72	A:A + A:A + B:C + D:C	10080	4	-1B +2C -1D
73	A:A + A:B + A:B + B:A	672	2	-1A +1B
74	A:A + A:B + B:A + B:B	672	2	
75	A:A + A:B + B:A + C:B	8064	3	+1B -1C
76	A:A + A:B + B:B + C:A	8064	3	+1B -1C
77	A:A + B:A + C:B + C:B	4032	3	+1A +1B -2C
78	A:A + B:A + C:B + D:B	20160	4	+1A +1B -1C -1D
79	A:A + A:B + A:B + C:A	4032	3	-1A +2B -1C
80	A:A + A:B + C:A + C:B	8064	3	+2B -2C
81	A:A + A:B + C:A + D:B	40320	4	+2B -1C -1D
82	A:A + B:A + B:C + C:C	4032	3	+1A -2B +1C
83	A:A + B:A + C:C + D:C	20160	4	+1A -1B +1C -1D
84	A:A + B:A + B:C + B:C	4032	3	+1A -3B +2C
85	A:A + B:A + B:C + D:C	40320	4	+1A -2B +2C -1D
86	A:A + B:A + C:D + C:D	20160	4	+1A -1B -2C +2D
87	A:A + B:A + C:D + E:D	80640	5	+1A -1B -1C +2D -1E
88	A:B + A:B + B:A + B:A	168	2	
89	A:B + A:B + B:A + C:A	4032	3	+1B -1C
90	A:B + A:B + B:C + B:C	2016	3	-2A +2C
91	A:B + A:B + C:A + D:A	10080	4	+2B -1C -1D
92	A:B + A:C + B:C + C:B	4032	3	-2A +1B +1C
93	A:B + B:A + C:A + D:B	20160	4	+1A +1B -1C -1D
94	A:B + A:B + A:C + B:C	4032	3	-3A +1B +2C
95	A:B + A:C + B:C + D:B	40320	4	-2A +1B +2C -1D
96	A:B + A:B + B:C + D:C	20160	4	-2A +1B +2C -1D
97	A:B + B:C + D:B + E:C	80640	5	-1A +1B +2C -1D -1E
98	A:B + A:B + A:C + A:C	1008	3	-4A +2B +2C

99	A:B + A:B + A:C + D:C	20160	4	-3A +2B +2C -1D
100	A:B + A:B + C:D + C:D	5040	4	-2A +2B -2C +2D
101	A:B + A:B + C:D + E:D	40320	5	-2A +2B -1C +2D -1E
102	A:B + A:C + D:B + D:C	10080	4	-2A +2B +2C -2D
103	A:B + A:C + D:B + E:C	80640	5	-2A +2B +2C -1D -1E
104	A:B + C:B + D:E + F:E	60480	6	-1A +2B -1C -1D +2E -1F
105	A:A + A:A + A:B + A:C	2016	3	-2A +1B +1C
106	A:A + A:A + A:B + B:C	4032	3	-1A +1C
107	A:A + A:A + A:B + C:C	4032	3	-1A +1B
108	A:A + A:A + A:B + C:D	20160	4	-1A +1B -1C +1D
109	A:A + A:A + B:B + B:C	4032	3	-1B +1C
110	A:A + A:A + B:B + C:C	2016	3	
111	A:A + A:A + B:B + C:D	20160	4	-1C +1D
112	A:A + A:A + B:C + C:B	2016	3	
113	A:A + A:A + B:C + C:D	20160	4	-1B +1D
114	A:A + A:A + B:C + B:D	10080	4	-2B +1C +1D
115	A:A + A:A + B:C + D:E	40320	5	-1B +1C -1D +1E
116	A:A + A:B + A:C + B:A	8064	3	-1A +1C
117	A:A + A:B + B:A + B:C	8064	3	-1B +1C
118	A:A + A:B + B:A + C:C	8064	3	
119	A:A + A:B + B:A + C:D	40320	4	-1C +1D
120	A:A + A:B + B:B + B:C	8064	3	-1A +1C
121	A:A + A:B + A:C + B:B	8064	3	-2A +1B +1C
122	A:A + A:B + B:B + C:C	8064	3	-1A +1B
123	A:A + A:B + B:B + C:D	40320	4	-1A +1B -1C +1D
124	A:A + A:B + B:C + C:A	8064	3	
125	A:A + B:A + B:C + C:B	8064	3	+1A -1B
126	A:A + A:B + B:C + C:C	8064	3	-1A +1C
127	A:A + B:A + C:B + D:C	40320	4	+1A -1D
128	A:A + A:B + C:A + D:C	40320	4	+1B -1D
129	A:A + B:A + B:C + D:B	40320	4	+1A -1B +1C -1D
130	A:A + B:A + C:B + D:D	40320	4	+1A -1C
131	A:A + B:A + C:B + C:D	40320	4	+1A -2C +1D
132	A:A + B:A + C:B + D:E	161280	5	+1A -1C -1D +1E
133	A:A + A:B + A:C + D:A	20160	4	-1A +1B +1C -1D
134	A:A + A:B + B:C + D:A	40320	4	+1C -1D
135	A:A + A:B + C:A + D:D	40320	4	+1B -1C
136	A:A + A:B + C:A + C:D	40320	4	+1B -2C +1D
137	A:A + A:B + C:A + D:E	161280	5	+1B -1C -1D +1E
138	A:A + A:B + C:C + D:C	40320	4	-1A +1B +1C -1D
139	A:A + B:A + C:C + D:D	20160	4	+1A -1B
140	A:A + B:A + B:C + D:D	40320	4	+1A -2B +1C
141	A:A + B:A + C:C + D:E	161280	5	+1A -1B -1D +1E
142	A:A + B:A + C:D + D:C	20160	4	+1A -1B
143	A:A + B:A + B:C + C:D	40320	4	+1A -2B +1D
144	A:A + B:A + C:D + D:E	161280	5	+1A -1B -1C +1E
145	A:A + B:A + B:C + B:D	20160	4	+1A -3B +1C +1D
146	A:A + B:A + B:C + D:E	161280	5	+1A -2B +1C -1D +1E
147	A:A + B:A + C:D + C:E	80640	5	+1A -1B -2C +1D +1E
148	A:A + B:A + C:D + E:F	241920	6	+1A -1B -1C +1D -1E +1F
149	A:B + A:B + B:A + B:C	4032	3	-1A +1C
150	A:B + A:B + A:C + B:A	4032	3	-2A +1B +1C
151	A:A + B:C + B:C + C:B	4032	3	-1B +1C
152	A:B + A:B + B:A + C:D	20160	4	-1A +1B -1C +1D
153	A:A + A:B + A:B + B:C	4032	3	-2A +1B +1C
154	A:A + A:B + A:B + A:C	4032	3	-3A +2B +1C
155	A:A + A:B + A:B + C:C	4032	3	-2A +2B
156	A:A + A:B + A:B + C:D	20160	4	-2A +2B -1C +1D
157	A:B + A:B + B:C + C:A	4032	3	-1A +1B
158	A:B + A:B + A:C + C:A	4032	3	-2A +2B
159	A:A + A:B + B:C + B:C	4032	3	-1A -1B +2C
160	A:B + A:B + C:A + D:C	20160	4	-1A +2B -1D
161	A:B + A:B + B:C + D:A	20160	4	-1A +1B +1C -1D
162	A:B + A:B + A:C + D:A	20160	4	-2A +2B +1C -1D
163	A:A + B:C + B:C + D:B	20160	4	-1B +2C -1D
164	A:B + A:B + C:A + C:D	20160	4	-1A +2B -2C +1D
165	A:B + A:B + C:A + D:E	80640	5	-1A +2B -1C -1D +1E
166	A:B + A:C + B:A + C:A	4032	3	
167	A:B + A:C + B:A + C:B	8064	3	-1A +1B
168	A:A + A:B + B:C + C:B	8064	3	-1A +1B
169	A:B + B:A + C:A + D:C	40320	4	+1A -1D
170	A:A + A:B + A:C + B:C	8064	3	-2A +2C
171	A:A + A:B + C:B + C:C	4032	3	-1A +2B -1C
172	A:A + A:B + C:B + D:C	40320	4	-1A +2B -1D
173	A:B + A:C + B:A + B:C	4032	3	-1A -1B +2C
174	A:B + A:C + B:C + D:A	40320	4	-1A +2C -1D
175	A:B + A:C + B:D + C:D	20160	4	-2A +2D

176	A:B + B:C + D:C + E:D	80640	5	-1A +2C -1E
177	A:B + A:C + B:A + D:A	40320	4	+1C -1D
178	A:B + A:C + B:A + D:B	40320	4	-1A +1B +1C -1D
179	A:A + B:C + C:B + D:B	40320	4	+1B -1D
180	A:B + A:C + B:D + D:B	40320	4	-2A +1B +1C
181	A:B + B:A + C:A + D:E	161280	5	+1A -1C -1D +1E
182	A:A + A:B + B:C + D:B	40320	4	-1A +1B +1C -1D
183	A:A + A:B + A:C + D:B	40320	4	-2A +2B +1C -1D
184	A:A + A:B + C:B + D:D	40320	4	-1A +2B -1C
185	A:A + A:B + C:B + C:D	40320	4	-1A +2B -2C +1D
186	A:A + A:B + C:B + D:E	161280	5	-1A +2B -1C -1D +1E
187	A:B + B:C + C:A + D:A	40320	4	+1A -1D
188	A:B + A:C + B:A + D:C	40320	4	-1A +2C -1D
189	A:A + A:B + B:C + D:C	40320	4	-1A +2C -1D
190	A:B + A:C + B:D + D:C	40320	4	-2A +2C
191	A:B + B:C + C:D + E:D	161280	5	-1A +2D -1E
192	A:B + A:C + B:C + C:D	40320	4	-2A +1C +1D
193	A:B + A:C + B:C + B:D	40320	4	-2A -1B +2C +1D
194	A:A + B:C + B:D + C:D	40320	4	-2B +2D
195	A:B + A:C + A:D + B:C	40320	4	-3A +2C +1D
196	A:B + A:C + B:C + D:E	161280	5	-2A +2C -1D +1E
197	A:B + B:C + C:D + E:C	161280	5	-1A +1C +1D -1E
198	A:B + A:C + D:A + E:B	161280	5	-1A +2B +1C -1D -1E
199	A:A + B:C + C:D + E:D	161280	5	-1B +2D -1E
200	A:B + A:C + D:B + E:D	161280	5	-2A +2B +1C -1E
201	A:B + A:C + B:D + E:D	161280	5	-2A +1C +2D -1E
202	A:B + B:C + D:C + E:F	483840	6	-1A +2C -1D -1E +1F
203	A:B + A:B + B:C + B:D	10080	4	-2A +1C +1D
204	A:B + A:B + B:C + C:D	20160	4	-2A +1B +1D
205	A:A + B:C + B:C + C:D	20160	4	-2B +1C +1D
206	A:B + A:B + A:C + B:D	20160	4	-3A +1B +1C +1D
207	A:B + A:B + B:C + D:E	80640	5	-2A +1B +1C -1D +1E
208	A:A + A:B + C:D + C:D	20160	4	-1A +1B -2C +2D
209	A:A + B:B + C:D + C:D	10080	4	-2C +2D
210	A:A + B:C + B:C + B:D	20160	4	-3B +2C +1D
211	A:A + B:C + B:C + D:E	80640	5	-2B +2C -1D +1E
212	A:B + A:B + C:D + D:C	10080	4	-2A +2B
213	A:B + A:B + A:C + C:D	20160	4	-3A +2B +1D
214	A:B + A:B + C:D + D:E	80640	5	-2A +2B -1C +1E
215	A:B + A:B + A:C + A:D	10080	4	-4A +2B +1C +1D
216	A:B + A:B + A:C + D:E	80640	5	-3A +2B +1C -1D +1E
217	A:B + A:B + C:D + C:E	40320	5	-2A +2B -2C +1D +1E
218	A:B + A:B + C:D + E:F	120960	6	-2A +2B -1C +1D -1E +1F
219	A:B + A:C + D:A + E:A	40320	5	+1B +1C -1D -1E
220	A:B + B:C + C:D + E:B	80640	5	-1A +1B +1D -1E
221	A:A + B:C + C:D + E:C	80640	5	-1B +1C +1D -1E
222	A:B + A:C + B:D + E:B	161280	5	-2A +1B +1C +1D -1E
223	A:B + B:C + D:B + E:F	241920	6	-1A +1B +1C -1D -1E +1F
224	A:A + A:B + C:D + E:D	80640	5	-1A +1B -1C +2D -1E
225	A:A + B:B + C:D + E:D	40320	5	-1C +2D -1E
226	A:A + B:C + B:D + E:C	161280	5	-2B +2C +1D -1E
227	A:A + B:C + D:C + E:F	241920	6	-1B +2C -1D -1E +1F
228	A:B + B:A + C:D + E:D	40320	5	-1C +2D -1E
229	A:B + A:C + B:D + E:C	161280	5	-2A +2C +1D -1E
230	A:B + B:C + D:E + F:E	241920	6	-1A +1C -1D +2E -1F
231	A:B + A:C + A:D + E:B	80640	5	-3A +2B +1C +1D -1E
232	A:B + A:C + D:B + D:E	80640	5	-2A +2B +1C -2D +1E
233	A:B + A:C + D:B + E:F	483840	6	-2A +2B +1C -1D -1E +1F
234	A:B + A:C + D:E + F:E	120960	6	-2A +1B +1C -1D +2E -1F
235	A:B + C:B + D:E + F:G	241920	7	-1A +2B -1C -1D +1E -1F +1G
236	A:A + A:B + A:C + A:D	6720	4	-3A +1B +1C +1D
237	A:A + A:B + A:C + B:D	40320	4	-2A +1C +1D
238	A:A + A:B + A:C + D:D	20160	4	-2A +1B +1C
239	A:A + A:B + A:C + D:E	80640	5	-2A +1B +1C -1D +1E
240	A:A + A:B + B:C + B:D	20160	4	-1A -1B +1C +1D
241	A:A + A:B + B:C + C:D	40320	4	-1A +1D
242	A:A + A:B + B:C + D:D	40320	4	-1A +1C
243	A:A + A:B + B:C + D:E	161280	5	-1A +1C -1D +1E
244	A:A + A:B + C:C + C:D	20160	4	-1A +1B -1C +1D
245	A:A + A:B + C:C + D:D	20160	4	-1A +1B
246	A:A + A:B + C:C + D:E	161280	5	-1A +1B -1D +1E
247	A:A + A:B + C:D + D:C	20160	4	-1A +1B
248	A:A + A:B + C:D + D:E	161280	5	-1A +1B -1C +1E
249	A:A + A:B + C:D + C:E	80640	5	-1A +1B -2C +1D +1E
250	A:A + A:B + C:D + E:F	241920	6	-1A +1B -1C +1D -1E +1F
251	A:A + B:B + C:C + D:D	1680	4	
252	A:A + B:B + C:C + D:E	26880	5	-1D +1E

253	A:A + B:B + C:D + D:C	10080	4	
254	A:A + B:B + C:D + D:E	80640	5	-1C +1E
255	A:A + B:B + C:D + C:E	40320	5	-2C +1D +1E
256	A:A + B:B + C:D + E:F	120960	6	-1C +1D -1E +1F
257	A:A + B:C + B:D + C:B	40320	4	-1B +1D
258	A:A + B:C + C:B + D:E	80640	5	-1D +1E
259	A:A + B:C + C:D + D:B	13440	4	
260	A:A + B:C + C:D + D:E	161280	5	-1B +1E
261	A:A + B:C + B:D + E:B	80640	5	-1B +1C +1D -1E
262	A:A + B:C + B:D + C:E	161280	5	-2B +1D +1E
263	A:A + B:C + C:D + E:F	483840	6	-1B +1D -1E +1F
264	A:A + B:C + B:D + B:E	26880	5	-3B +1C +1D +1E
265	A:A + B:C + B:D + E:F	241920	6	-2B +1C +1D -1E +1F
266	A:A + B:C + D:E + F:G	161280	7	-1B +1C -1D +1E -1F +1G
267	A:B + A:C + A:D + B:A	20160	4	-2A +1C +1D
268	A:B + A:C + B:A + B:D	20160	4	-1A -1B +1C +1D
269	A:B + A:C + B:A + C:D	40320	4	-1A +1D
270	A:B + A:C + B:A + D:E	161280	5	-1A +1C -1D +1E
271	A:B + B:A + C:D + D:C	5040	4	
272	A:B + B:A + C:D + D:E	80640	5	-1C +1E
273	A:B + A:C + D:E + E:D	40320	5	-2A +1B +1C
274	A:B + B:A + C:D + E:F	120960	6	-1C +1D -1E +1F
275	A:B + A:C + B:D + D:A	40320	4	-1A +1C
276	A:B + B:C + C:A + D:E	53760	5	-1D +1E
277	A:B + B:C + C:D + D:A	10080	4	
278	A:B + B:C + C:D + D:E	161280	5	-1A +1E
279	A:B + A:C + D:A + E:D	80640	5	-1A +1B +1C -1E
280	A:B + A:C + B:D + E:A	161280	5	-1A +1C +1D -1E
281	A:B + A:C + B:D + D:E	161280	5	-2A +1C +1E
282	A:B + B:C + C:D + E:F	483840	6	-1A +1D -1E +1F
283	A:B + A:C + A:D + E:A	26880	5	-2A +1B +1C +1D -1E
284	A:B + A:C + B:D + B:E	80640	5	-2A -1B +1C +1D +1E
285	A:B + A:C + D:A + E:F	241920	6	-1A +1B +1C -1D -1E +1F
286	A:B + A:C + B:D + C:E	80640	5	-2A +1D +1E
287	A:B + B:C + D:E + E:F	241920	6	-1A +1C -1D +1F
288	A:B + A:C + A:D + B:E	80640	5	-3A +1C +1D +1E
289	A:B + A:C + B:D + E:F	483840	6	-2A +1C +1D -1E +1F
290	A:B + A:C + D:E + E:F	241920	6	-2A +1B +1C -1D +1F
291	A:B + B:C + D:E + F:G	483840	7	-1A +1C -1D +1E -1F +1G
292	A:B + A:C + A:D + A:E	6720	5	-4A +1B +1C +1D +1E
293	A:B + A:C + A:D + E:F	80640	6	-3A +1B +1C +1D -1E +1F
294	A:B + A:C + D:E + D:F	60480	6	-2A +1B +1C -2D +1E +1F
295	A:B + A:C + D:E + F:G	241920	7	-2A +1B +1C -1D +1E -1F +1G
296	A:B + C:D + E:F + G:H	40320	8	-1A +1B -1C +1D -1E +1F -1G +1H

Existen en promedio 5.2511286735535 símbolos distintos en cada regla del SPRT

Mas allá de 8S4D

En la actual implementación del script GSPRT, explorar más allá del 8S4D es costoso computacionalmente hablando. Una mejora a este problema es la implementación del script en forma compilada, por ejemplo en Delphi o C++, pero no es una solución definitiva ya que la naturaleza exponencial y factorial del algoritmo garantiza que siempre habrá un sistema para el que ya no sea práctico intentar su construcción.

E.2.7. Código fuente del script GSPRT

Esta es la descripción básica de la estructura interna y funcionamiento del script:

- Secciones de configuración y estado inicial del generador:
 - Sección de configuración del script, asignación de la tabla de símbolos y la tabla de combinación principal del sistema \$z.
 - Sección de selección de opciones de presentación de listados disponibles en el script:
 - \$swpri: Determinar si reportar o no la tabla de primitivas de transición del sistema.
 - \$swimp: Establecer si presentar o no el listado de reglas independientes.
 - \$swpre: Configurar la presentación de las estructuras repetidas: ocultar, integrar al listado de reglas independientes o mostrar por separado.
 - \$swppt: Establecer si calcular o no los patrones de transición presentes en el SPRT.
 - \$swept: Determinar si se presentan los ejemplos de uso de patrones de transición.

Por defecto se muestran las estadísticas globales de los SPRT.

- Determinar el tipo de sistema configurado en \$z, si es completo o no.
 - Determinar los símbolos del sistema y construir la tabla de símbolos: internamente los símbolos se manejan como identificadores numéricos, esto reduce el uso de memoria, facilita labores de identificación, manejo y posterior búsqueda de símbolos.
 - Iniciar en cero los contadores poblacionales para cada uno de los símbolos establecidos y los demás elementos de estadísticas del script.
 - Construcción del encabezado del reporte.
 - Calcular de la cantidad de combinaciones con repetición a realizar por el script.
 - Construir de la tabla de primitivas del sistema.
 - Construir la representación texto de la configuración del sistema según los símbolos presentes en la tabla de combinaciones del sistema.
-
- Ciclo principal de cálculo:
 - Ciclo de generación de la combinación / regla actual mediante variación con repeticiones según el valor establecido en el contador de secuencia de reglas \$g.
 - Ciclo de construcción de la representación interna de la última regla generada \$regla, en notación de combinación y permutación de primitivas.
 - Ciclo de determinación mecánica del significado de cada combinación, según las operaciones fundamentales de Idempotencia o Permanencia, Instanciación, Transformación o Cambio y Terminación.
 - Búsqueda de las reglas de transición independientes del sistema:

- Construir una representación de símbolos para las reglas: establecer un identificador para la regla mediante la ordenación de los símbolos que componen la regla de transición según su estructura.
- Determinar si ya se ha incorporado el identificador de regla: Si la representación obtenida no existe entonces añadirla a las reglas independientes del sistema. El proceso de comparaciones se implementa mediante un algoritmo de búsqueda en índices.
- Búsqueda de los patrones de transición del sistema:
 - ◊ Determinar la cantidad de símbolos distintos presentes en la regla actual, discriminar entre símbolos para establecer el identificador de diferencias.
 - ◊ Generar mediante permutaciones del identificador de diferencias la menor expresión de patrón de transición que represente la regla actual. La cantidad de ciclos requerida para realizar este paso es igual al factorial del número de símbolos distintos presentes en la regla de transición. Ver tablas D.11 y D.12 en la página 455.
 - ◊ Ordenar de menor a mayor las representaciones de patrones obtenidas, determinar la menor representación posible.
 - ◊ Comparar la menor representación de patrón encontrada con los patrones ya almacenados durante la ejecución del algoritmo. Si la representación actual no existe, añadirla a los patrones del sistema. El proceso de comparaciones se implementa mediante un algoritmo de búsqueda en índices.

Anidar la búsqueda de patrones dentro de la búsqueda de reglas de transición independiente reduce el total de cálculos requeridos para la generación del SPRT. La búsqueda de patrones se ejecuta solo si la regla en evaluación es independiente.

- Construir el reporte de resultados según los datos calculados y la configuración del script establecida en las variables \$sw.
- Incrementar el contador de secuencia de regla de transición en proceso \$g.
- Finalizar con la presentación de las estructuras obtenidas luego del proceso de cálculo realizado según las opciones establecidas en la sección de configuración.

Ejecución del script y límites de cálculo

Aunque con el script GSPRT sea posible generar los SPRT completos según una configuración arbitraria de símbolos y dimensiones, existen límites para su uso derivados de la naturaleza del proceso de construcción de las reglas y de la búsqueda de patrones de transición. Existen cantidades de S y D para los cuales no es prudente esperar por los resultados.

La descripción fundamental del algoritmo principal es un ciclo dentro de otro, una *burbuja*. La cantidad de ciclos a realizar por la capa externa es igual al total de reglas de transición a generar por el algoritmo. Aunque existen varios ciclos internos, solo la búsqueda de patrones representa

un verdadero lastre, ya que requiere para su funcionamiento de la realización de operaciones de permutación.

En general, cuando se construye un SPRT, el ciclo externo crece exponencialmente en función de la cantidad de símbolos y de dimensiones de evaluación requeridas y el ciclo interno de búsqueda de patrones es factorial con respecto a la cantidad de símbolos distintos ds presentes en cada regla de transición del sistema.

La actual implementación del algoritmo de construcción de los SPRT ha sido optimizada para ejecutar la menor cantidad posible de búsquedas de patrones de transición. El total de búsquedas de patrones a realizar es igual a la cantidad de reglas de transición independientes presentes en el sistema. Durante la construcción del algoritmo se observó que es posible reducir aun más la cantidad de búsquedas de patrones y de reglas de transición independientes mediante combinaciones dirigidas²¹, se deja esta labor a las siguientes iteraciones de investigación sobre el tema.

Total de ciclos requeridos para la construcción de un SPRT de referencias - nRD

Para calcular la cantidad de ciclos necesaria para la construcción de un SPRT completo de referencias, basta con saber que existe un símbolo o referencia de causa único y distinto de las demás disponibles por cada dimensión de evaluación requerida por el sistema. Esto hace que el promedio de símbolos distintos presente en cada expresión de regla de transición sea constante e igual al número de dimensiones existentes, $ds = R = S = D$. Ya que para estos sistemas la cantidad de reglas de transición es igual al total de reglas independientes entonces el total de ciclos requeridos para la construcción de un nRD es:²²

$$Total\ ciclos\ en\ un\ SPRT_{nRD}\ Completo = D^D D! \quad (E.1)$$

Total de ciclos requeridos para la construcción de un SPRT de Símbolos - nSmD

El total de reglas de transición requerido para generar en un nSmD es:

$$Total\ de\ reglas\ transición = TRT_{nSmD} = \begin{cases} S^{2D} & \text{si } nSmD \text{ es un SPRT Completo, } P = S^2 \\ P^D & \text{si } nSmD \text{ es un SPRT Incompleto, } P \neq S^2 \end{cases} \quad (E.2)$$

Para cualquier regla de transición de un SPRT, los valores límite de las diferencias entre símbolos depende de la cantidad de símbolos y el total de dimensiones disponibles así:

²¹Crear solo cierto tipo de combinaciones. El enfoque actual es crear todas las posibles combinaciones.

²²OEIS A061711 = 1, 8, 162, 6144, 375000, 33592320, 4150656720, 676457349120, 140587147048320, 3628800000000000, 11388728893445164800, 4270826380475341209600, 1886009588552176549862400, 968725766854884321342259200, ... etc. [Slo11m]

$$\overline{ds_{aprox}} = \begin{cases} S & si S \leq D \\ D & si D < S < 2D \\ 2D & si S \geq 2D \end{cases} \quad (E.3)$$

- Si el número de símbolos es menor que el total de dimensiones, solo se puede presentar hasta S símbolos distintos por regla.
- Si el número de símbolos es mayor o igual que el doble de dimensiones disponibles, entonces es posible acomodar simultáneamente símbolos diferentes en las causas y efectos de las reglas de transición del sistema, esto corresponde a una cantidad igual al doble de la cantidad de dimensiones de evaluación presentes, 2D.
- Si el número de símbolos está comprendido entre D y 2D, su valor depende enteramente de la frecuencia de patrones y la cantidad particular de símbolos presente en cada patrón. Usar a D como valor promedio hace que se estime el límite de la menor cantidad de permutaciones posibles.

Los valores S y 2D del promedio de símbolos distintos corresponden con la estimación del valor máximo que es posible alcanzar por $\overline{ds_{aprox}}$ en cualquier nSmD.

Tabla E.2: Matriz de máximos de permutación en SPRT de símbolos

$\overline{ds_{aprox}}$		Dimensiones								
		1	2	3	4	5	6	7	8	...
S	1	s	s	s	s	s	s	s	s	
	2	2d	s	s	s	s	s	s	s	
	3	2d	$\bar{d}$	s	s	s	s	s	s	
	4	2d	2d	$\bar{d}$	s	s	s	s	s	
	5	2d	2d	$\bar{d}$	$\bar{d}$	s	s	s	s	
	6	2d	2d	2d	$\bar{d}$	$\bar{d}$	s	s	s	s
	7	2d	2d	2d	$\bar{d}$	$\bar{d}$	$\bar{d}$	s	s	
	8	2d	2d	2d	2d	$\bar{d}$	$\bar{d}$	$\bar{d}$	s	
	9	2d	2d	2d	2d	$\bar{d}$	$\bar{d}$	$\bar{d}$	$\bar{d}$	
	10	2d	2d	2d	2d	2d	$\bar{d}$	$\bar{d}$	$\bar{d}$	
	$\vdots$				2d					$\zeta x^{\cdot}$

Guía de tendencia de máximos de permutación en SPRT. Al aumentar el numero de símbolos se esperan 2d permutaciones, al aumentar el número de dimensiones se esperan s permutaciones. Se encuentra que al aumentar simultáneamente la cantidad de símbolos y dimensiones, el total de permutaciones esta comprendido entre $2\bar{d} < x \leq \bar{d}$.

Fuente: Autor.

La tabla E.2 presenta la distribución de $\overline{ds_{aprox}}$ presente en la Matriz SPRT según (E.3). Una aproximación en términos de (E.3) al número total de ciclos a realizar en el cálculo de un SPRT es:

$$Aproximación Total de ciclos SPRT_{nSmD} \approx TRT_{nSmD} \cdot \overline{ds_{aprox}}! \quad (E.4)$$

La exactitud de estas aproximaciones se pierde rápidamente para grandes valores de S, P y D cuando $D < S < 2D$. Asumiendo una distribución uniforme de $\overline{ds}$ por ciclo con $\overline{ds} \in \mathbb{R}$ y $\Gamma(n+1) = n!$, entonces aproximar el total de ciclos a realizar para el cálculo de un SPRT es:

$$\text{Aproximación ciclos de cálculo en SPRT} \approx TRT_{nSmD} \cdot \Gamma\left(\omega\left(\overline{ds} + 1\right)\right) \quad (E.5)$$

Lamentablemente, se desconoce la forma de la función de ajuste ω para E.5 con la cual se determine la interacción entre patrones y símbolos de un SPRT. La cantidad exacta de ciclos requerida para la construcción de un nSmD se describe naturalmente en función de la frecuencia de los patrones y el total de símbolos distintos de cada patrón de transición presente en el sistema así:

- Se definen las siguientes funciones:

$$\text{Frecuencia patrón}_i = FP_i \quad (E.6)$$

$$\text{Total símbolos distintos en patrón}_i = dsP_i \quad (E.7)$$

- En términos de estas expresiones se establece que:

$$\text{Total ciclos de SPRT } nSmD = \sum (FP_i \cdot (dsP_i)!) \quad (E.8)$$

- A partir de (E.6) y (E.7), el promedio de símbolos distintos en un SPRT según la distribución de símbolos del sistema es:

$$\text{Total Símbolos en patrones de } nSmD = TSPT_{nSmD} = \sum (FP_i \cdot dsP_i) \quad (E.9)$$

- De donde se determina que:

$$\text{Promedio símbolos distintos en SPRT}_{nSmD} = \overline{ds} = \frac{TSPT_{nSmD}}{TRT_{nSmD}} \quad (E.10)$$

Las ecuaciones (E.8) y (E.10) son aplicables a todos los SPRT, sin importar su configuración o tipo. El valor de (E.10) se reporta en cada ejecución del script GSPRT. Al igual que con ω en (E.5), no se conoce una fórmula explícita para la frecuencia de patrones y el cálculo del total de patrones de transición presente en un SPRT. Como se observa en (E.8) el cómputo del total de diferencias entre símbolos de un patrón depende de su configuración particular.

Notas de implementación

La función `pc_next_permutation` incorporada en el cuerpo del script GSPRT ha sido tomada de:

`http://docstore.mik.ua/oreilly/webprog/pcook/ch04\_26.htm`

El enlace hace referencia al capítulo 4.26. *Finding All Permutations of an Array* de la primera edición del libro de PHP Cookbook de David Sklar y Adam Trachtenberg, editado por O'Reilly en 2002. En [ST02] se describe la mejor solución disponible para la implementación de una rutina en PHP que permita generar permutaciones sin recursión²³. El código de la función ha sido modificado según necesidades. La experiencia de construcción del algoritmo permite asegurar que la búsqueda en índices mediante Tries²⁴ o diccionarios, constituye una solución óptima para el problema de la identificación rápida de patrones y reglas de transición.

El código fuente del script GSPRT incluido en este documento está configurado por defecto con el SPRT 4S3D, el cual contiene el conjunto de todas las transiciones disponibles y posibles presentes en Homos 1.0.

²³El uso dado a este código es académico, está dentro de los lineamientos descritos en: *O'Reilly Policy on Re-Use of Code Examples from Books*. Ver http://openp2p.com/pub/a/oreilly/ask_tim/2001/codepolicy.html

²⁴Estructura de datos de árbol ordenado usada para almacenar vectores o *arrays* asociativos.

Script E.15: Generador de Sistemas de Producción de Reglas de Transición

```

1 //Generador de sistemas de producción de reglas de transición
2 set_time_limit(0); //elimina el límite de tiempo de ejecución de los phplets
3 $phplet_dir_ = "c:\_phplets"; //directorio de trabajo
4 $phplet_slf_ = "GSPRT.php"; //nombre archivo
5 $phplet_rows_ = 30;
6
7 //***** Memoria maxima a usar por el script *****
8 ini_set("memory_limit","1024M"); //por defecto = 32MB.
9
10 //***** Sección de configuración del script *****
11 /*
12 Configuración del SPRT
13 Definición de Tabla de símbolos, script configurado para SPRT 4S3D completo
14 */
15 $o = array('_', 'a', 'b', 'c');
16
17 /*
18 Configuración del vecindario / dimensiones / simultaneidad
19 estructura de $z
20 c1,c2..cn = causas y e1,e2..en = efectos
21 z = c1,c2,..,cN,e1,e2,..,eN = siempre en pares, en notación de simultaneidad y orden
22 */
23 $vecindad = 3; for($i=0;$i<$vecindad;$i++) { $z[]=$o;$z[]=$o; } //creación de parejas en vecindad
24
25 //***** Opciones de configuración del script GSPRT *****
26 /*
27 Tabla de primitivas de transición
28 0 = no reportar
29 1 = incluir
30 */
31 $swpri=1;
32
33 /*
34 Productos del script
35 0 = imprimir resultados = presentar el resultado de opciones de listados de reglas
36 1 = reportar solo las cantidades generales
37 */
38 $swimp=01;
39
40 /*
41 Presentar estructuras repetidas de reglas de transición:
42 0 = no listar
43 1 = integradas dentro del listado de reglas independientes
44 2 = como una lista separada
45 */
46 $swpre=0;
47
48 /*
49 Cálculo de patrones de transición:
50 0 = generar solo las reglas del SPRT
51 1 = generar reglas + patrones = mayor duración de cálculos
52 */
53 $swppt=1;
54
55 /*
56 Presentar ejemplos de aplicación de patrones de transición:
57 0 = no presentar
58 1 = listar ejemplos de cada patron de transición del sistema
59 */
60 $swept=1;
61
62 /*
63 ***** Notas de uso: *****
64 El caracter '_' (underscore) es usado en este script como identificador / símbolo de vacio.
65
66 opciones de construcción:
67 1)SPRT completo: Todas las causas y efectos del sistema usan los mismos símbolos.
68 Solo hay que establecer el tamaño de la vecindad y los símbolos a ser usados en $o = tabla de
    símbolos
69 Vecindad = D = Dimensiones, simultaneidad, complejidad espacial máxima del sistema
70 Símbolos = S = Cantidad de símbolos distintos presentes en la tabla
71 Primitivas = P = Cantidad de primitivas de transición armables con la tabla de símbolos
72 Total de primitivas en el sistema = (S^2) = P
73 Total de reglas en el sistema = (S^2)^D = P^D = S^(2D)
74 Total de reglas independientes = (S+D)!*S/((S+D)*S!*D!)
75 Ejemplo:

```

```

76
77 $o = {A,B}; //S = 2
78 $vecindad = 3; //D = 3 = dimensión, complejidad, simultaneidad del sistema
79 for($i=0;$i<$vecindad;$i++) {$z[]=$o;$z[]=$o;} //creación de parejas en vecindad, $z siempre es par
80
81 //c = causa | e = efecto c1: e1 + c2: e2 = c1,c2: e1, e2
82 lo expuesto equivale en notación de reglas = $o:$p1 + $o:$p2 = $o,$o:$p1,$p2
83 la tabla de símbolos del SPRT será = {A,B} = 2 símbolos
84 4 primitivas, 64 reglas con 20 independientes y 44 dependientes, 10 patrones de transición
85
86 2)SPRT incompleto: Las causas y efectos del sistema usan diferentes combinaciones de símbolos entre
87 si.
88 se establece una configuración manual del sistema, hay que declarar las tablas auxiliares de
89 símbolos y ubicarlas en $z.
90 la tabla final de símbolos es la colección de todos los identificadores distintos presentes en $z
91 la cantidad de primitivas ya no es S^2. La fórmula general para el cálculo de la cantidad de reglas
92 independientes es diferente y requiere realizarse por separado
93 Ejemplo:
94
95 $o = array('A','B');
96 $p1= array('A','B','C');
97 $p2= array('A','B','D');
98 $z=array($o,$o,$p1,$p2); //construcción efectiva
99 $vecindad=count($z)/2; //se requiere calcular la vecindad o dimensión del sistema
100
101 //c = causa | e = efecto c1, c2 : e1, e2 = c2, c1: e2, e1
102 lo expuesto equivale en notación de reglas = $o:$p1 + $o:$p2 = $o, $o :$p1,$p2 = $o, $o:$p2,$p1
103 la tabla de símbolos del SPRT será = {A,B,C,D} = 4 símbolos
104 8 primitivas, 36 reglas con 30 independientes y 6 dependientes, 10 patrones de transición
105 */
106
107 //***** Cuerpo principal del script *****
108
109 $simbolo_vacio = '_'; //símbolo de vacío o espacio libre en el sistema
110 $ituple=array('','','Doble','Triple','Cuádruple','Quíntuple','Séxtuple','Séptuple','Óctuple','
111 Nónuple','Décuple','Ondécuple','Dodécuple','Tridécuple');
112
113 //determinar si el SPRT usa un conjunto completo de primitivas o no
114 //normalizar el contenido de símbolos de $zr
115 $zr=$z;
116 foreach($zr as $id=>$zi) sort($zr[$id]);
117
118 //determinar si en cada causa y efecto del sistema se maneja la misma configuración de símbolos
119 $sw_completo=1;
120 foreach($zr as $zi)
121     if($zi<>$zr[0]) {
122         $sw_completo=0; //en este sistema P <>S^2 = incompleto
123         break;
124     }
125
126 //tabla de símbolos del sistema
127 $ts=array();
128 foreach($z as $zi)
129     foreach($zi as $zj)
130         if(count($ts)==0) $ts[]=$zj;
131         else {
132             $sw=0;
133             foreach($ts as $tsi){
134                 if($tsi==$zj) { $sw=1; break; }
135             }
136             if ($sw==0) $ts[]=$zj;
137         }
138 sort($ts);
139
140 $id_ts=array();
141 foreach($ts as $n=>$id) $id_ts[$id]=$n;
142
143 //conversión de ids de símbolos en números
144 $nz=array();
145 $cn=0;
146 foreach($z as $zi){
147     foreach($zi as $zj){
148         $nz[$cn]=$id_ts[$zj];
149     }; $cn++;
150 }
151
152 $tabla_simbolos="";

```

```

149 //contador para cada símbolo presente en el sistema
150 $ks = array();
151 $id_vacio=-1;
152
153 foreach($ts as $id=>$n){
154     $ks[]=0; //0,1,2.. símbolos
155     if($tabla_simbolos<>"") $tabla_simbolos.=",";
156     $tabla_simbolos.=$n;
157     if($n==$simbolo_vacio) $id_vacio=$id;
158 }
159 //iniciar en ceros
160 $bs = $ks; //balance global de símbolos
161 $k0 = $ks; //conteo local de símbolos
162
163 //balance de espacio, población y cambio
164 $ep=array('E'=>0,'P'=>0,'C'=>0); //ce,cp,cc
165
166 //determinar si se tiene en cuenta los vacios/espacio en los cálculos
167 if($id_vacio<>-1) $dif_en_espacio=1; else $dif_en_espacio=0;
168
169 //tipificación de transiciones
170 $np['P']='Permanencia'; //idempotencia
171 $np['I']='Instanciación'; //creación
172 $np['C']='Cambio'; //transformación / remplazo
173 $np['F']='Finalización'; //terminación
174
175 $rp=array();
176 foreach($np as $id=>$n){
177     $nd[$id] = array(); //constructor de estructura de regla
178     $rp[$id] = 0; //balance local de acción de primitivas en regla
179     $rt[$id] = 0; //balance global de acción de primitivas
180 }
181 //reiniciar el conteo
182 $w0 = $nd;
183 $w1 = $rp;
184
185 $nru=array();
186 $lzrc=''; //lista de reglas repetidas
187 $lzpt=''; //lista de patrones de transición
188
189 $zmax = count($z); //número de vectores a manejar iniciando en cero
190 $med = $zmax / 2 - 1;
191
192 //encabezado del reporte
193 $id_sprt="SPRT(" . count($ts) . "S," . $vecindad . "D)"; //identificador del sistema de producción de
    reglas actual
194 echo " $id_sprt<br>";
195 echo "Tabla de símbolos = {" . $tabla_simbolos . "} = " . count($ts) . " símbolos<br>";
196 echo "Dimensiones, simultaneidad, vecindario máximo = $vecindad = Se combinarán $zmax vectores<br>"
    ;
197 $g = array_fill(0,$zmax, 0); //contador de secuencias
198 echo "elementos en cada vector [ ";
199 $comb = 1; //contador del número total de permutaciones de primitivas = cantidad de combinaciones
    de símbolos
200 foreach($z as $n){
201     $temp = count($n);
202     echo $temp . " ";
203     $h[] = $temp; //máximo de conteo en cada vector
204     $comb = $comb * $temp;
205 };
206 echo "]" = $comb combinaciones<br>";
207
208 //cálculo del factorial de un número
209 function fac($a){
210     if($a>1) $f=$a*fac($a-1); else $f=1;
211     return $f;
212 }
213
214 if($sw_completo==1){ // es P = S^2 ?
215     $n_ = count($ts);
216     $n_*= $n_;
217     // $m_=$vecindad-1;
218     // $rind = $n_*fac($n_+$m_)/(fac($n_)*fac($m_)*($m_+1)); //eq. original
219
220     $m_=$vecindad;
221     $nm=$n_+$m_;
222     $rind = $n_*fac($nm)/(fac($n_)*fac($m_)*$nm); //eq. final

```

```

223
224     echo "Existen $rind reglas independientes<br>\n";
225 } else {
226     echo "Este sistema implementa selección de primitivas<br>\n"; // P <> S^2
227     $rind=$comb;
228 }
229
230 $ncmb = strlen($comb); //cantidad de dígitos en combinaciones de reglas
231 $nsep = strlen($rind); //cantidad de dígitos en reglas independientes
232 $ndimc = strlen($vecindad); //cantidad de dígitos en acción de regla = dimensión
233 $ndim = $ndimc+1;
234 $spad=""; $spadep = str_pad($spad,$ndim+1," ");
235 $spad=""; $spadc = str_pad($spad,$ndim," ");
236
237 //construcción de la tabla de primitivas del sistema
238 if($swpri){
239     $tp=array();
240     for($i=0;$i<=$med;$i++){
241         foreach($z[$i] as $iv=>$v){
242             foreach($z[$i+$med+1] as $iw=>$w){
243                 $key="$v:$w";
244                 if(count($tp)>0) {
245                     if(!array_key_exists($key,$tp)) $tp[$key]=0;
246                 } else {
247                     $tp[$key]=0;
248                 };
249             }
250         }
251     }
252     $npt=count($tp);
253     echo "Se usan $npt primitivas de transición<br>";
254     foreach($tp as $itp=>$tpi) echo $itp."\t";
255     echo "<br>\n";
256 };
257
258 echo "Construcción de reglas mediante configuración de símbolos en vecindad<br>";
259 echo "Celda del vecindario\tDimensión\tEntrada/Causa\t->\tSalida/Efecto\tNotación<br>";
260 $cru="Principal\t\t";
261 for($i=0;$i<=$med;$i++){
262     $ncru="";
263     if ($i>0) $cru.="Vecina $i\t\t";
264     $cru.="Celda[$i]\t{";
265     foreach($z[$i] as $im=>$w){
266         if ($im>0) {$cru.=","; $ncru.=",";}
267         $cru.=$w;
268         $ncru.=$w;
269     }
270     $ncru.=",";
271     $cru.="}\t->\t{";
272     foreach($z[$i+$med+1] as $im=>$w){
273         if ($im>0) {$cru.=","; $ncru.=",";}
274         $cru.=$w;
275         $ncru.=$w;
276     }
277     $cru.="}\t{".$ncru."}<br>";
278 }
279 echo "$cru<br>";
280
281 /*
282 Función no recursiva para la realización de permutaciones de elementos en un array
283 Obtenida de http://docstore.mik.ua/oreilly/webprog/pcook/ch04_26.htm
284 Versión web del código presentado en el capítulo 4.26. "Finding All Permutations of an Array" del
285 libro PHP Cookbook de David Sklar y Adam Trachtenberg, Primera edición. 2002.
286 Copyright 2003 O'Reilly & Associates.
287 El código ha sido modificado según necesidades
288 */
289 function pc_next_permutation($p, $size) {
290     // slide down the array looking for where we're smaller than the next guy
291     // for ($i = $size - 1; $p[$i] >= $p[$i+1]; --$i) { } //línea original
292     for ($i = $size - 1; ($i>-1) and ($p[$i] >= $p[$i+1]) ; --$i) { } //modificación, evita el
293         notice del $p[-1]
294
295     // if this doesn't occur, we've finished our permutations
296     // the array is reversed: (1, 2, 3, 4) => (4, 3, 2, 1)
297     if ($i == -1) { return false; }

```

```

298 // slide down the array looking for a bigger number than what we found before
299 for ($j = $size; $p[$j] <= $p[$i]; --$j) { }
300
301 // swap them
302 $tmp = $p[$i]; $p[$i] = $p[$j]; $p[$j] = $tmp;
303
304 // now reverse the elements in between by swapping the ends
305 for (++$i, $j = $size; $i < $j; ++$i, --$j) {
306     $tmp = $p[$i]; $p[$i] = $p[$j]; $p[$j] = $tmp;
307 }
308 return $p;
309 }
310
311 $sep = '';
312 $zlim=$zmax-1;
313
314 $fin=0;
315 $ord=0;
316 $okv=0;
317 $hm=0;
318
319 $total_patrones=0;
320
321 $ci=0;
322
323 $id_regla=0;
324
325 $sprt_patron=array();
326 $fast_patron=array();
327
328 $cn_pt=array();
329 $rt_pt=array();
330
331 for($ii=0;$ii<$comb;$ii++){ //ciclo principal de construcción de reglas mediante combinación de
    símbolos
332     $hi=0;
333
334     //variación con repeticiones implementada mediante incremento ordenado de secuencias
335     while($g[$hi]==$h[$hi]){ $g[$hi]=0; $hi++; $g[$hi]++; $ci++; };
336
337     if ($ci>$zlim) break; //fin de ciclo de cálculos , máximo de cambios posibles alcanzado
338     $ci=0;
339     $ord++;
340     $prc="";
341     for($jj=0;$jj<=$zmax-1;$jj++){ //ciclo de presentación de combinación
342         switch($jj){
343             case $med : $sep='.'; break; //símbolo de transición
344             case $zlim: $sep=''; break; //sin coma al final
345             default: $sep=','; break; //la coma es el separador de identificadores
346         };
347         $prc.= $z[$jj][$g[$jj]].$sep; //valor $jj de combinación según identificadores de
            símbolos
348     } //fin de bloque de construcción de combinación
349
350     //contadores texto y numéricos
351     $enu=''; //listado de funciones primitivas
352     $eni=''; //funcion primitiva
353     // $ndi=''; //relación de objetos en primitiva
354     $id=''; //id de combinación de tipo de primitiva
355     $cp=0; //de población
356     $ce=0; //de espacios usados
357     $cc=0; //de cambios
358     $ks=$k0; //contador de cambios poblacionales en clase por acción de primitiva en regla
359     $rn=""; //id de primitiva
360     $rp=$w1;
361
362     $pri = "";
363     $regla=array();
364
365     //ciclo separado de determinación mecánica de significado de combinación de cambio
366     for($jj=0;$jj<=$med;$jj++){ //ciclo de presentación de combinación
367         $causa = $jj;
368         $efect = $jj+$med+1;
369
370         $ant = $g[$causa]; //Situación causa
371         $sig = $g[$efect]; //Situación efecto
372

```



```

373     $sant = $z[$causa][$sant]; //símbolos
374     $ssig = $z[$efect][$sig];
375
376     $iant = $nz[$causa][$sant]; //id de símbolo
377     $isig = $nz[$efect][$sig];
378
379     if($pri<>"") $pri.=' + ';
380     $pri.= $sant.':'. $ssig;
381
382     if($swpnt==1) $regla[]=array($iant,$isig); //regla como id/símbolos
383
384     switch($sant){
385         case $ssig: //es igual al siguiente? si $sant persiste, es idempotente en
este paso
386             $eni="P($sant) "; //persistencia
387             $ndi=$iant;
388             $rn = 'P';
389             break;
390         case $simbolo_vacio: // $sant es vacío?
391             if($ssig!=$simbolo_vacio){ //si $sant <> $sig entonces se crea/
instancia $sig
392                 $eni="I($ssig) "; //instanciación / creación
393                 $ndi=$isig;
394                 $rn = 'I';
395                 $ce--;
396                 $cp++;
397                 $cc++;
398
399                 $bs[$iant]--; //vacío
400                 $bs[$isig]++;
401
402                 $ks[$iant]--; //vacío
403                 $ks[$isig]++;
404             }
405             break;
406         default: // $sant es diferente de $sig y no es vacío
407             if($ssig==$simbolo_vacio) {
408                 $eni="F($sant) "; //terminación / finalización
409                 $ndi=$iant;
410                 $ce++;
411                 $cp--;
412                 $cc++;
413                 $rn = 'F';
414             } else {
415                 $eni="C($sant,$ssig) "; //cambio / transformación
416                 $ndi="$iant-$isig";
417                 $rn = 'C';
418                 $cc++;
419             }
420             $bs[$iant]--;
421             $bs[$isig]++;
422
423             $ks[$iant]--;
424             $ks[$isig]++;
425             break;
426     };
427     $rp[$rn]++; //cambio local primitivas en regla
428     $rt[$rn]++; //cambio global de primitiva
429     $enu=$eni; //cambio / transformación
430     $id=$rn;
431     $nd[$rn][]=$ndi; //vector de configuración de estructura de regla
432 } //fin de lectura de combinación
433
434 //estadísticas de espacio, población y cambio
435 $ep["E"]+=$ce;
436 $ep["P"]+=$cp;
437 $ep["C"]+=$cc;
438
439 $prc = "$pri [". $prc. " "; //nomenclatura de la regla en forma de combinación y permutación
de primitivas
440
441 //normalización de estructura de las reglas generadas
442 //construcción del índice rápido de estructura de regla
443 $ndx="";
444 foreach($nd as $n0){
445     if($ndx!="") $ndx.="|";
446     if(count($n0)>0){

```

```

447             sort($n0); //ordenar si es necesario
448             foreach($n0 as $nj=>$n1) {
449                 if($nj>0) $ndx.=",";
450                 $ndx.=$n1;
451             }
452         } else {
453             $ndx.=" ";
454         };
455     }
456
457     //determinar si la regla generada existe o no
458     $sw=0;
459     if(count($nru)>0)
460         if (array_key_exists($ndx,$nru)){ //la llave existe
461             $sw=1;
462             if($swimp==0){
463                 $zrc="$prc = R".($nru[$ndx])."<br>"; //la combinación actual repite
464                 a $ndx
465                 if ($swpre>0){
466                     if ($swpre==1){
467                         echo $zrc;
468                     } else { //swpre=2
469                         if ($swpre==2) $lzrc.=$zrc;
470                     };
471                 };
472             }
473         }
474
475     //la regla actual no se repite , es independiente
476     if($sw==0) {
477         //
478         //////////////////////////////////////
479
480         //búsqueda de patrones de transición del SPRT en conjunto de reglas independientes
481         $sw_patron=0;
482         if ($swpnt==1){
483             $patron=array();
484             $min=array();
485             $tsl=array();
486             foreach($regla as $ri)
487                 foreach($ri as $rj)
488                     if(count($tsl)==0) $tsl[]=$rj;
489                     else {
490                         $sw=0;
491                         foreach($tsl as $ki){
492                             if($ki==$rj) { $sw=1; break; }
493                         }
494                         if ($sw==0) $tsl[]=$rj;
495                     }
496             }
497             sort($tsl);
498
499             $its=array();
500             $kdo=array();
501             foreach($tsl as $n=>$ij) {
502                 $its[$ij]=$n; //relación entre símbolos y tabla de permutaciones
503                 $kdo[]=$n; //0,1,2,3.. tabla de permutaciones
504             }
505             $size = count($kdo) - 1;
506
507             //lista las posibles combinaciones y busca la menor secuencia posible
508             foreach($regla as $idn) $min[]=array($kdo[$its[$idn[0]]],$kdo[$its[$idn
509                 [1]]]); //primera permutación
510             sort($min);
511             $patron[]=$min;
512
513             //la búsqueda de la menor expresión posible es realmente lenta , depende de
514             factorial(size)
515             //se recomienda encontrar un nuevo método para su generación
516             while ($kdo = pc_next_permutation($kdo, $size)){ //siguientes combinaciones
517                 $min=array();
518                 foreach($regla as $idn) $min[]=array($kdo[$its[$idn[0]]],$kdo[$its[
519                     $idn[1]]]); //permutaciones
520                 sort($min);
521                 $patron[]=$min;
522             }
523             sort($patron); //ordenar , el primer patrón es la descripción mínima de la
524             combinación actual de símbolos

```

```

517
518 $key_patron="";foreach($patron[0] as $p0) $key_patron.=$p0[0].$p0[1]; //
    clave mínima actual
519
520 //almacenar patrones distintos entre si
521 //sw_patron=0;
522 if(count($sprt_patron)>0){ //búsqueda inmediata de la llave actual
523     if (array_key_exists($key_patron,$fast_patron)){ //la llave existe
524         $sw_patron=1;
525         $id_pat=$fast_patron[$key_patron];
526     } else { //la llave no existe
527         $sprt_patron[]=$patron[0]; //patrón distinto
528         $id_pat=count($sprt_patron);
529         $fast_patron[$key_patron]=$id_pat;
530     }
531 } else { //primer patrón
532     $sprt_patron[]=$patron[0]; //primer patrón del sistema
533     $id_pat=count($sprt_patron);
534     $fast_patron[$key_patron]=$id_pat; //llave y conteo
535 }
536 if($sw_patron==0) $cn_pt[$id_pat-1]=0; //iniciar el contador de patrones
537 }
538 //
    //////////////////////////////////////
539 if ($swpnt==1) $rt_pt[$ndx]=$id_pat-1; //vector de asociacion entre reglas y
    patrones, usado en $cn_pt
540
541 $okv++;
542 $nru[$ndx]=$okv;
543 //conteo de patrones encontrados en el SPRT
544 if($sw_patron==0) {
545     $total_patrones++;
546     $cpt=$total_patrones;
547 } else $cpt=$id_pat;
548
549 if($swimp==0){
550     if($swpnt==1) $cpt=sprintf("%-". $nsep. "s",$cpt);
551     else $cpt="";
552
553     $id_regla++;
554     $tal="$prc $id $enu";
555     //combinación, id de primitivas, notación funcional de primitivas
556     echo sprintf("%-". $nsep. "s",$id_regla)." $cpt $tal";
557
558     if($dif_en_espacio){
559         if($ce!=0) printf("%". $ndim. "sE",$ce); else echo $spadep; // +/-
            cambios en espacio
560         if($cp!=0) printf("%". $ndim. "sP",$cp); else echo $spadep; // +/-
            cambios en población
561         echo " ";
562     };
563     if($cc!=0) printf("%". $ndimc. "sC",$cc); else echo $spadc; // >=0
            cambios en la regla
564     echo " ";
565
566     foreach($ts as $ni=>$n) {
567         $kn = $ks[$ni];
568         if ($kn!=0) printf("%". $ndim. "s%s",$kn,$n); //total de cambios
            realizados por la regla
569         else echo $spadep;
570     }
571
572     //descripción general de la combinación de primitivas que conforman la
        regla
573     $sr="";
574     $cie=count($rp);
575     foreach($rp as $ie=>$n) {
576         if (($sr!="")and($n>0)) $sr.=" + "; //espaciado y separación de
            términos
577         if ($n>1) $sr.= $ituple[$n].' '; //facilidad para numeración de
            elementos
578         if ($n>0) $sr.=$np[$ie]; //nombre de primitiva
579     }
580     echo " $sr<br>";
581
582     //descripción asociada de cada patrón de transición encontrado

```

```

583             if ($swppt==1) if($sw_patron==0) $lzp="cpt $tal$sr<br>";
584         }
585     }
586     if ($swppt==1) $cn_pt[$rt_pt[$ndx]]++; //contador de patrones de transicion del sistema
587     $nd=$w0; //reiniciar estructura de regla
588     $g[0]++; //incremento de secuencia del conteo principal generador de las combinaciones de
        reglas
589 };
590
591 //***** análisis + resultados del SPRT *****
592
593 echo "Se realizaron $ord combinaciones, existen $okv reglas independientes = permutaciones de
    primitivas diferentes<br>";
594 echo "<br>Existen ".$($ord-$okv)." reglas repetidas en el sistema: diferencias en orden situación /
    primitivas<br>";
595 if ($swpre==2) echo $lzrc; //lista de reglas repetidas
596 echo "<br>Estadísticas del SPRT";
597 echo "<br>* Balance global de símbolos<br>";
598 foreach($ts as $ni=>$n) printf("%".($ndim+2)."s%s", $bs[$ni], $n); //total de cambios realizados por
    la regla
599 echo "<br>";
600
601 echo "<br>* Balance global de Espacio, Población, Cambio(reemplazo) e Inercia(idempotencia)<br>";
602 $ni='E'; printf("%".($ndim+1)."s%s", $sep[$ni], $ni);
603 $ni='P'; printf("%".($ndim+1)."s%s", $sep[$ni], $ni);
604
605 $cambio = ($sep['C']/$ord); //sumatoria de cambio / total de reglas del sistema <= dimensiones del
    sistema
606 $pje_cambio = 100*$cambio/$vecindad;
607
608 $ni='C'; printf("\t%.2f%s = %.2f%% Cambio", $cambio, $ni, $pje_cambio);
609     ; printf("\t%.2fI = %.2f%% Inercia", $vecindad-$cambio, (100-$pje_cambio));
610 echo "<br>";
611
612 echo "<br>* Balance global de acción de primitivas Permanencia, Instanciación, Cambio y
    Finalización<br>";
613 foreach($rt as $ni=>$n) {
614     $cambio = ($n/$ord); //sumatoria de cambio / total de reglas del sistema <= dimensiones del
        sistema
615     $pje_cambio = 100*$cambio/$vecindad;
616     printf("%.2f%s = %.2f%%\t", $cambio, $ni, $pje_cambio);
617 }
618 echo "<br>";
619
620 //presentación de patrones de transición del SPRT
621 if ($swppt==1){
622     //usar letras mayusculas como identificadores en patrones de transicion. Hasta 29 simbolos
        disponibles
623     $tso=array();
624     foreach($ts as $its=>$tsi) {
625         $tso[]=chr(65+$its);
626         $kso[]=0;
627     };
628     $ts=$tso; //bloquear para usar solo los simbolos definidos
629
630     $c_pt=0;
631     $d_pt=0;
632
633     if ($swept==1) echo "<br>Ejemplos de aplicación: Existen $total_patrones patrones de
        transición<br>$lzp";
634
635     if($sw_completo==0) $txt_prt="particular de $npt primitivas"; else $txt_prt="";
636     echo "<br>Tabla de patrones de transición de $id_sprt $txt_prt<br>";
637     foreach($sprt_patron as $jd=>$sp){
638         $ks=$kso; //iniciar el contador de aporte diferencial del patron
639         $dpt=array(); //iniciar el contador de simbolos
640         echo sprintf("%-". $nsep."s", $jd+1)." "; //id de patrón
641         foreach($sp as $id=>$s){ //patrón de transición
642             if($id>0) echo " + ";
643             echo $ts[$s[0]].":".$ts[$s[1]];
644             $dpt[$s[0]]=0; $dpt[$s[1]]=0; //registrar la diferencia entre símbolos
645
646             $ks[$s[0]]--; //incremento neto //calculo de aporte diferencial del patron
                de transicion
647             $ks[$s[1]]++; //decremento neto
648         }
649         $cdpt=count($dpt);

```

```

650     $d_pt+=$cdpt;
651     echo " ".sprintf("%".$ncmb."s",$cn_pt [$jd]); //uso de patrón en el sistema
652     echo " ".sprintf("%".$ndim."s",$cdpt)." "; //conteo de simbolos distintos en el
        patrón
653     $c_pt+=$cn_pt[$jd]*$cdpt;
654
655     foreach($ts as $ni=>$n) {
656         $kn = $ks[$ni];
657         if ($kn!=0) printf(" %".$ndim."d%s",$kn,$n); //aporte diferencial de
            primitiva en patron
658     }
659     echo "<br>";
660 }
661 echo "Existen en promedio ".$c_pt/$ord." símbolos distintos en cada regla del SPRT";
662 }

```

E.3. Prueba de uso de la interfaz OLE de Homos 1.0

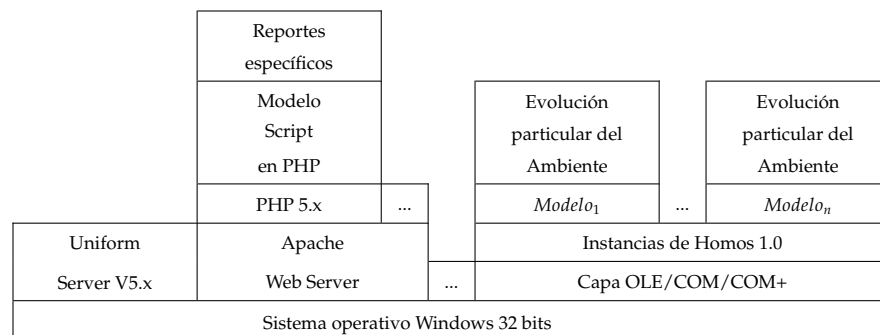
En esta tesis se comprobó exitosamente la funcionalidad de la interfaz OLE de la herramienta Homos 1.0. Los scripts contenidos en este apartado permitieron el redescubrimiento del servidor de aplicaciones; dieron soporte a ideas que serán elementos centrales de una siguiente versión de la MBOR y de una futura aplicación que la soporte plenamente.

Los modelos-script presentados en el apartado E.4 en la página 588 constituyen ejemplos prácticos de uso de la herramienta software Homos. Los scripts de las secciones E.4.1 y E.4.2 realizan controles sobre la configuración y ejecución de las reglas de transición que no es posible realizar directamente con las funciones previstas en la interfaz gráfica de herramienta software. El script de la sección E.4.3 implementa control simultáneo de dos instancias de Homos 1.0. Los modelos incluidos en E.4.4 y E.4.5 hacen uso de nuevas funciones incorporadas en la interfaz OLE. Las novedades surgieron luego del proceso de análisis realizado sobre la interfaz, se desarrollaron como parte de una actividad interna de mantenimiento realizada por este proyecto sobre el sistema de automatización de Homos 1.0. Cada script incorporado en este apartado presenta:

- Una tabla que describe el modelo de transición presente en cada ejemplo.
- Una tabla que describe la caracterización en diferencias de las reglas de transición del modelo.
- Una representación transparente de la estructura de transiciones presente en el modelo.
- Uno o más ejemplos de representación gráfica del diagrama BOR.
- Una imagen del comportamiento poblacional obtenido de la ejecución del script.

El código fuente de los scripts, el proyecto *.prh generado por la automatización y el resultado texto obtenido de la simulación se encuentran en la carpeta contenido/anexos/scripts/OLE/ del CD-ROM que acompaña este proyecto.

Figura E.6: Homos 1.0 como servidor de aplicaciones



Desde un script con acceso a funciones COM es posible controlar una o más instancias de Homos 1.0, manejando solo un modelo BOR por cada instancia de automatización. El control global de ejecución y el compartir datos entre modelos se realiza a nivel del script. Homos 1.0 es una aplicación windows de 32 bits.

Fuente: Autor.

E.3.1. Revisión y mantenimiento de la Interfaz OLE

Se listan a continuación los cambios realizados por este proyecto a la interfaz OLE de Homos 1.0. El propósito de los cambios se realizaron fue:

- Corregir los errores y deficiencias más importantes detectados durante la etapa de análisis, Ver página 124. Esto fue necesario para facilitar el proceso de construcción de los distintos scripts.
- Mejorar la experiencia de construcción de modelos BOR mediante programación usando la interfaz OLE
- Probar con ejemplos las capacidades de la interfaz como medio para la creación de modelos.

Aparecen registrados en este apartado solo los cambios realizados. Las demás funciones previstas en la interfaz permanecen inalteradas, corresponden con los detalles descritos la sección A.5.7 en la página 184.

La aplicación ha sido recompilada con Delphi 3, que es lenguaje de programación con el cual se desarrolló originalmente la herramienta software. Se realizaron varios intentos de portar la aplicación a una versión más reciente como Delphi 7, pero debido a problemas de compatibilidad de objetos, específicamente con el objeto Teechart, a los distintos elementos encontrados en el análisis y por estar fuera del alcance de los objetivos de este proyecto de grado, no se llegó a feliz término con ese proceso.

Esta investigación participa de los trabajos de reparación de Homos 1.0 mediante el reporte de los resultados obtenidos del análisis y los cambios realizados a la interfaz OLE, pero propone que el trabajo global de mantenimiento y actualización pueda ser llevado a cabo en propiedad por un próximo trabajo de grado.

E.3.2. Ajuste del parámetro probabilidad de las reglas existentes en Homos 1.0

Ahora es posible asignar directamente valores de probabilidad flotantes en la definición de cada una de las reglas de transición:

1. `procedure ReglaCrearDecadencia(
 const Nombre: WideString; Probabilidad: Double; const ClaseObjeto, Descripcion:
 WideString);safecall;`
2. `procedure ReglaCrearMovimiento(
 const Nombre: WideString; Probabilidad: Double; Prioridad, Direccion, NumeroCeldas:
 Integer;
 const ClaseObjeto, Descripcion: WideString); safecall;`
3. `procedure ReglaCrearExpansion(
 const Nombre: WideString; Probabilidad: Double; Prioridad, Direccion: Integer;
 const ClaseObjeto, Descripcion: WideString); safecall;`
4. `procedure ReglaCrearRetraccion(
 const Nombre: WideString; Probabilidad: Double; Prioridad, Direccion: Integer;
 const ClaseObjeto, Descripcion: WideString); safecall;`
5. `procedure ReglaCrearNeutralizacion(
 const Nombre: WideString; Probabilidad: Double; Prioridad: Integer;
 const ClaseObjeto1, ClaseObjeto2, Descripcion: WideString); safecall;`
6. `procedure ReglaCrearTransformacion(
 const Nombre: WideString; Probabilidad: Double; Prioridad: Integer;
 const ClaseObjeto1, ClaseObjeto2, Descripcion: WideString); safecall;`
7. `procedure ReglaCrearEliminacion(
 const Nombre: WideString; Probabilidad: Double; Prioridad: Integer;
 const ClaseObjeto1, ClaseObjeto2, Descripcion: WideString); safecall;`
8. `procedure ReglaCrearAlteracion(
 const Nombre: WideString; Probabilidad: Double; Prioridad: Integer;
 const ClaseObjeto1, ClaseObjeto2, ClaseObjeto3, Descripcion: WideString); safecall;`
9. `procedure ReglaCrearReproduccion(
 const Nombre: WideString; Probabilidad: Double; Prioridad: Integer;
 const ClaseObjeto1, ClaseObjeto2, ClaseObjeto3, Descripcion: WideString); safecall;`

Adicionalmente se corrigió un error interno en la asignación de las prioridades de ejecución de las reglas Expansión, Retracción y Transformación.

E.3.3. Cambios realizados a las funciones presentes en la interfaz OLE

Las siguientes funciones se adicionaron al conjunto original previsto en la interfaz, debido a que facilitan el desarrollo y ejecución de modelos mediante scripts. Constituyen un aporte de mantenimiento y soporte a Homos 1.0 por parte de este trabajo de grado.

1. function ReglaDarPrioridad(const Nombre: WideString): Integer; safecall;

Retorna el valor de la prioridad asignada a la regla identificada con Nombre.

2. procedure ReglaAsignarPrioridad(const Nombre: WideString; Prioridad: Integer); safecall;

Establece el valor de la prioridad de ejecución de la regla referenciada por Nombre. Este método hace posible que se pueda implementar durante la simulación reglas de transición con valores de prioridad variables, los cambios a estos valores se realizan mediante llamados regulares durante la etapa de simulación del modelo.

Para realizar un cambio exitoso del valor de probabilidad este debe estar entre 0 y 255, en otro caso se ignora la petición. La función permite alterar la forma en la que el simulador selecciona las reglas de transición del modelo, ya que permite separarlas por orden de importancia de ejecución.

3. function SimulacionPasoActual: Integer; safecall;

Retorna el valor del paso actual de simulación. Este dato se obtiene del tamaño de la tabla de datos interna del simulador. Si la tabla de datos no se ha establecido, devuelve -1.

4. function EstadoSimulacion: Integer; safecall;

Retorna el estado interno del simulador:

Valor	Significado operativo
-1	No hay proyecto cargado ni simulación en proceso.
0	Simulación terminada.
1	Simulación en curso, ejecutándose.
2	Simulación en pausa.
3	Indicador de estado desconocido del simulador, posible error interno.

Esta función permite un seguimiento preciso del desarrollo de simulación, ya no son necesarios controles adicionales a nivel del script como los presentados en la sección A.5.9.3 en la página 197.

5. Como consecuencia del análisis, la función cerrar de la interfaz OLE fue eliminada del repertorio de funciones disponibles de la interfaz OLE.

La presencia en la interfaz de las funciones SimulacionPasoActual y EstadoSimulacion simplifican el desarrollo de formas de control estructuradas y simples de usar en los scripts.

E.4. Ejemplos de completitud en modelado con la MBOR

En los distintos ejemplos incorporados en esta seccion se establece una integracion entre las formas de representacion existentes de la MBOR y las generadas durante este proyecto.

Se presenta en la tabla E.3 una relación de las representaciones disponibles de las que se hizo uso en los ejemplos que componen este apartado. En la sección E.6.3 en la página 638 se da un ejemplo de aplicacion de la caracterizacion en diferencias de los modelos de cristalización y predador presa de 7 reglas de transicion.

La idea de la integración de propuestas es permitir una mejor comprensión del fenomeno a representar por la metodologia de modelado. El sentido de completitud a usar en este documento esta relacionado con disponer de utiles de modelado que permitan realizar una labor satisfactoria de desarrollo de modelos.

Tabla E.3: Componentes y representaciones disponibles en ejemplos de Modelado BOR

Elementos originales	Representaciones propuestas	Detalle
Modelo Prosa	Representación tabular del modelo BOR	Descripción del modelo. Modelo según las reglas de transición existentes.
	Representación gráfica de transiciones	Representación iconográfica de las reglas de transición del modelo. Fácil de leer.
	Caracterización en diferencias del modelo BOR	Soporte matemático complementario al modelo de automata previsto en Homos 1.0.
	Representación transparente del modelo	Unión entre descripción matemática y función de las expresiones de transición. Precursor de los diagramas BOR.
Diagrama BOR	Diagrama BOR simbólico, pictográfico y mixto	Propuesta de remplazo, mejoras en la representación de transiciones.
Ejemplo de resultados de simulación		Ambiente, gráficas de comparación, vistas, resultados en archivos texto y representación binaria de un proyecto.
Curvas poblacionales		Valores poblacionales en función del tiempo.
	Modelos Script (en PHP)	Representación del modelo en forma no tradicional, flexible.

E.4.1. Ejemplo de modelo con regla de transición de probabilidad variable

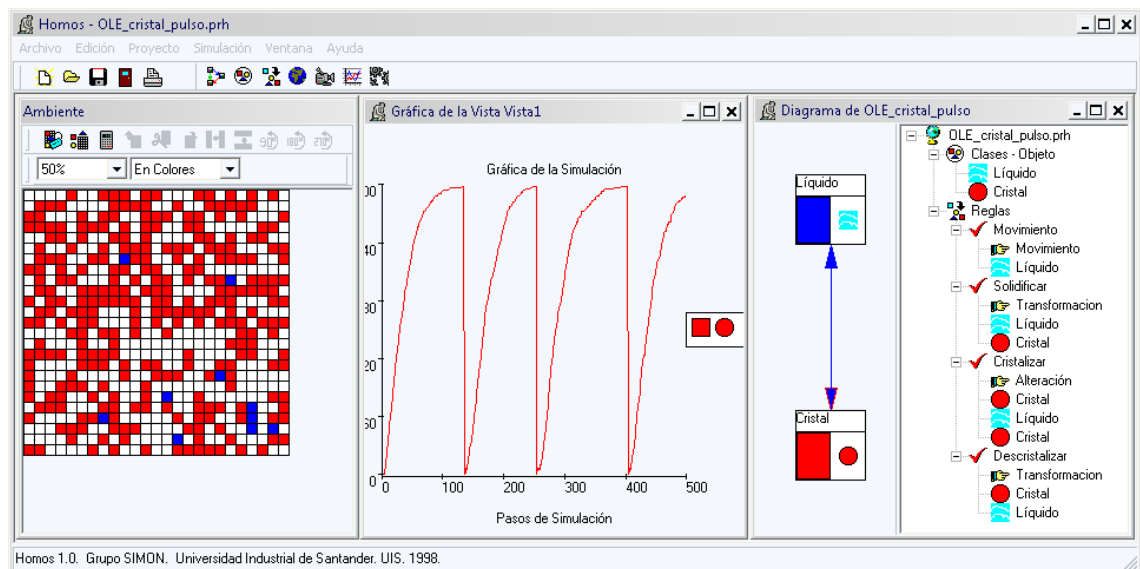
El script OLE_EJ_cristales_pulso.php recrea un modelo en el cual se caracteriza un material X que inicialmente se presenta en estado líquido y que paulatinamente se va cristalizando. En estado líquido las partículas del material pueden moverse libremente en todas direcciones, las partículas pueden cambiar aleatoriamente a estado sólido y en este último estado producir cristalización en las partículas vecinas.²⁵

Cuando se detecta la cristalización total del sistema, se inicia un pulso de descristalización que cambia todas las partículas de sólido a líquido, reiniciando el ciclo.

El estado de la variable \$actualizar_regla determina si se activa o no la regla \$r4, encargada del paso de las partículas de sólido a líquido.

En OLE_EJ_cristales_pulso.txt se presenta un ejemplo de resultado de ejecución.

Figura E.7: Resultados de simulación de OLE_EJ_cristales_pulso.php



Nótese las diferencias entre la representación del diagrama BOR según Homos 1.0 versus la figura E.8 y la representación transparente E.11 en la página 592.

Fuente: Autor.

²⁵ Descripción en prosa del modelo.

Tabla E.4: Modelo BOR de OLE_EJ_cristales_pulso.php

Símbolos	Regla	Notación	Metáfora descriptiva	%	\$	Evaluación de celdas vecinas
1 Sustancia X en estado Líquido	r_1	$[1, -: , 1]$	Movimiento del líquido Movimiento	10	1	Toda dirección disponible distancia de paso = 2
s Sustancia X en estado Sólido	r_2	$[1 : s]$	Transformar de líquido a sólido Solidificar	1	1	Ninguna
- Vacío	r_3	$[s, 1 : s, s]$	Cambio por Alteración Solidificar el líquido circundante Cristalizar	5	1	La del objeto relacionado 1
	r_4	$(1=0, 100\%, 0\%)$ $[s : 1]$	Cambio por transformación Transformar de cristal a líquido Descristalizar	0	1	Ninguna
Metasímbolos				#		Distribución
? Desconocido	s_1	$[? : _]$	Desconocido como espacio vacío en ambiente	∞		Uniforme
	s_2	$[_ : 1]$	Inserción de líquido en espacios vacíos del ambiente	300		Aleatoria
	s_3	$[_ : s]$	Inserción de Sólido en espacios vacíos del ambiente	0		Aleatoria

El modelo se implementa en la herramienta usando un ambiente de 25×25 celdas de borde cíclico.
Las reglas de transición han sido expresadas en notación de permutación. % y \$ son abreviaturas de Probabilidad y Prioridad de ejecución. # representa una función de conteo.
El movimiento es una transición de referencia, $[1, -: , 1]$ se interpreta como $[1, _ : \&2, \&4]$.
 r_4 es el pulso de regeneración y no es ejecutable inicialmente. El script implementa un pulso de un paso de duración.
Este modelo está contenido dentro de los SPRT 3S2D y 2RD.
Fuente: Autor.

Tabla E.5: Caracterización en diferencias de OLE_EJ_cristales_pulso.php

Id	Regla	Acción bruta	Acción neta	Actividad	Aporte neto de primitivas	Población	Actividad poblacional
1	[1, -, -, 1]	-1 - - + - +1	0	$n_1 R_1$	0	L	$n_4 - n_2 - n_3$
2	[1 : s]	-1 + s	-1 + s	$n_2 R_2$	$-n_2 l + n_2 s$	S	$n_2 + n_3 - n_4$
3	[s, 1 : s, s]	-s -1 + s + s	-1 + s	$n_3 R_3$	$-n_3 l + n_3 s$	-	0
4	[s : 1]	-s +1	-s +1	$n_4 R_4$	$-n_4 s + n_4 l$		

Tr = -1 +s

Tabla E.6: Actividad de primitivas y patrones presentes en OLE_EJ_cristales_pulso.php

Id	Primitiva	Actividad	Tipo	Id	Patrón	Actividad
1	1 : -	n_1	c	1	A, B : B, A	n_1
2	- : 1	n_1	c	2	A, B : A, A	n_3
3	1 : s	$n_2 + n_3$	c	3	A : B	$n_2 + n_4$
4	s : 1	n_4	c			
5	s : s	n_3	p			

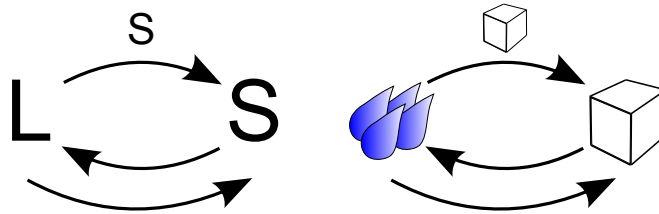
Tp = 0 $\Sigma = \frac{4}{5}c + \frac{1}{5}p$

El modelo presenta transición incompleta entre símbolos. Usa explícitamente 5 de 9 primitivas de transiciones posibles. Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k . p y c son abreviaturas de Permanencia y Cambio. Tr y Tp son abreviaturas de tendencia de reglas y tendencia de primitivas del sistema. Fuente: Autor.

Representación transparente del modelo de transiciones presente en `OLE_EJ_cristales_pulso.php`.
En la ecuación (E.11) se agrupan las representaciones de transición por su similaridad estructural.

$$\begin{aligned}
 \text{Modelo} &= \sum n_i R_i \\
 &= n_1 R_1 + n_2 R_2 + n_3 R_3 + n_4 R_4 \\
 &= n_1 \left[\begin{array}{c} l \rightarrow - \\ - \quad l \end{array} \right] + \\
 &\quad + n_3 \left[l \xrightarrow{s} s \right] + \\
 &\quad + n_2 [l \rightarrow s] + n_4 [s \rightarrow l]
 \end{aligned} \tag{E.11}$$

Figura E.8: Representación de diagrama BOR del modelo `OLE_EJ_cristales_pulso.php`



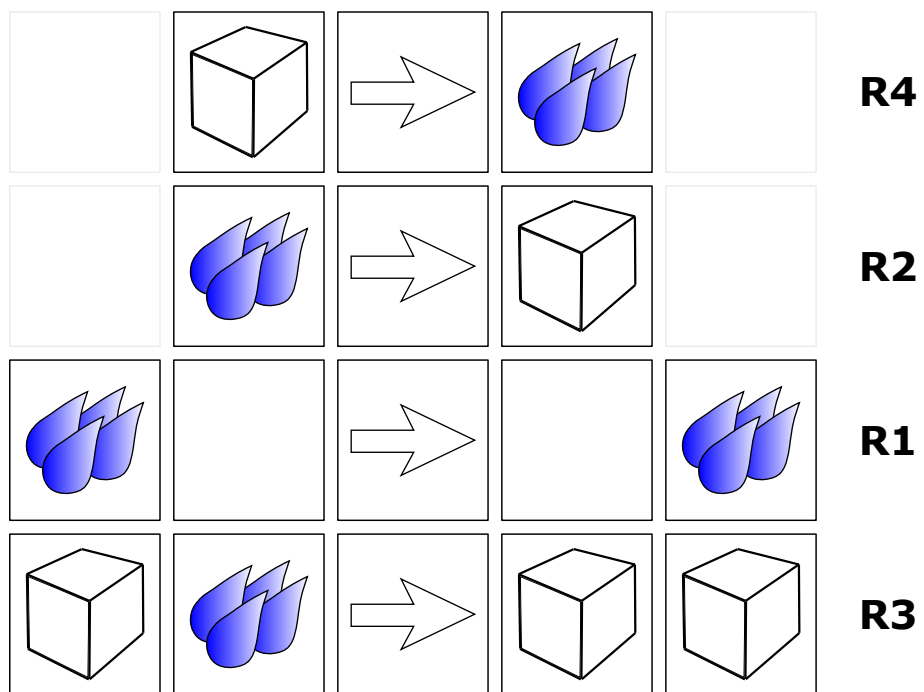
Se presentan de izquierda a derecha dos formas de representación gráfica del diagrama BOR

- Usando solo los símbolos o identificadores de clase.
- Reemplazo de símbolos por imágenes o pictogramas.

Fuente: Autor.

Figura E.9: Representación gráfica de transiciones de OLE_EJ_cristales_pulso.php

Transiciones de cristalización



Fuente: Autor.

Script E.16: Un modelo de crecimiento de cristales con pulso de regeneración poblacional

```
1 //Modelo de cristalización y descristalización
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = 'c:\_phplets\OLE'; //directorio por defecto
4 $phplet_slf_ = 'OLE_EJ_cristales_pulso.php'; //nombre archivo
5 // $phplet_lib_ = ''; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 echo "Crear Enlace<br>";
9 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
10
11 echo "Configurar el proyecto<br>";
12 $h1->proyetonuevo('C:\Program Files\UIS\Homos\OLE_cristal_pulso.prh');
13 $h1->visible=false;
14
15 echo "Crear elementos del modelo<br>";
16 $obj1="Líquido";
17 $obj2="Cristal";
18 $h1->claseobjetocrear($obj1,"C:\Program Files\UIS\Homos\Recursos\agua.bmp",hexdec(strrev("0000FF")))
19     ,"Elemento en estado líquido");
20 $h1->claseobjetocrear($obj2,"C:\Program Files\UIS\Homos\Recursos\bola.bmp",hexdec(strrev("FF0000")))
21     ,"Elemento en estado sólido, cristalizado");
22
23 echo "Crear clase<br>";
24 $r1="Movimiento";
25 $r2="Solidificar";
26 $r3="Cristalizar";
27 $r4="Descristalizar";
28 $h1->reglacrearmovimiento ($r1,10,1,255,2,$obj1, "Movimiento, control de encuentros
29     entre partículas de líquido y cristales");
30 $h1->reglacreartransformacion($r2,1,1, $obj1,$obj2, "Solidificación aleatoria de una
31     partícula de sustancia");
32 $h1->reglacrearalteracion ($r3,5,1, $obj2,$obj1,$obj2,"Acción cristalización sobre el
33     líquido");
34 $h1->reglacreartransformacion($r4,0,1, $obj2,$obj1, "Pulso de descristalización");
35
36 echo "Configuración del ambiente<br>";
37 $ciclico=1;
38 $finito=2;
39 $infinito=3;
40 $h1->numerofilas=25;
41 $h1->numerocolumnas=25;
42 $h1->tipolimites=$ciclico;
43
44 echo "Crear Escenario<br>";
45
46 $numpasos=PHP_INT_MAX; //maxima cantidad de ciclos posibles
47 $max_ciclos_solidificacion=3; //cantidad de ciclos de solidificacion a ser simulados
48 $contador_solidificacion=0; //contador de solidificaciones
49
50 $h1->escenariocrear("Recipiente",$numpasos);
51 $h1->escenarioseleccionar(0);
52 $h1->escenarioseleccionar(0);
53 $h1->objetosinsertar($obj1,"Aleatorio",300,0,0);
54 $h1->colorceldas=hexdec("000000");
55
56 echo "Actualizar el ambiente<br>";
57 $h1->ambienteactualizar;
58 $h1->visible=true;
59
60 $actualizar_regla=0; //variable de control de pulso de regeneracion
61
62 echo "Ciclo de simulación<br>";
63 $pasofinal=$numpasos-1;
64 $ciclo=0;
65 $pospaso=0;
66 $ultimodato = $h1->objetosdarcantidad($pasofinal);
67 while($ultimodato[0]==-1){ //simular hasta terminar
68     if ($ciclo==0) { //iniciar/continuar la simulación
69         $h1->simulacioniniciar(false);
70     } else $h1->simulacionparar;
71     $ciclo++;
72     //tiempo de simulación por ciclo, pausa entre evaluaciones
73     usleep(62500); //1/16 seg | 1 seg = 1000000 microsegundos
74     $h1->simulacionparar; //detener
75 }
76 //determinar el paso actual
```



```

72     for($j=$pospaso;$j<$pasofinal;$j++){
73         $vec=$h1->objetosdarcantidad($j); //obtener los valores de paso indicados
74         if ($vec[0]==-1) break;
75     }
76     $pospaso=$j; //ultimo indice conocido
77
78 //realizar la operaciones con los datos de simulacion obtenidos
79     if ($j>0){
80         $actualdato = $h1->objetosdarcantidad($j-1); //obtener los datos del paso actual
81         if ($actualdato[0]==0){
82             $h1->reglaasignarprobabilidad($r4,100); //pulso de descristalizacion
83             $actualizar_regla=1;
84             $contador_solidificacion++;
85         }
86         else
87             if($actualizar_regla){
88                 $h1->reglaasignarprobabilidad($r4,0 ); //normalizacion del sistema
89                 $actualizar_regla=0;
90             }
91     }
92     $ultimodato = $h1->objetosdarcantidad($pasofinal);
93
94 //determinar el fin de ciclos de solidificion
95     if($contador_solidificacion>=$max_ciclos_solidificacion){
96         $numpasos=$ciclo; //determinar la cantidad de ciclos de simulacion realizados
97         break;
98     }; //terminar el ciclo de simulacion principal
99 }
100 echo "Se realizaron $ciclo ciclos de evaluación del estado de la simulación<br>\n";
101 $h1->simulacionterminar; //finalizar la simulacion
102 usleep(100000); //tiempo para cierre
103
104 echo "Presentación de resultados<br>";
105 echo "<table>\n<tr><td>Paso</td>";
106 $numclases=$h1->cuantasclasesobjeto;
107 for($i=0;$i<$numclases;$i++) echo "<td>".$h1->claseobjeto($i)."</td>";
108 echo "\n";
109 for($paso=0;$paso<$numpasos;$paso++){
110     $resultados = $h1->objetosdarcantidad($paso);
111     echo "<tr><td>$paso</td>\n";
112     foreach($resultados as $item) echo "<td>$item</td>\n";
113     echo "</tr>\n";
114 }
115 echo "</table>\n";
116
117 echo "Cierre del enlace y terminación del script<br>\n";
118 $h1->proyectocerrar(true);

```

E.4.2. Ejemplo de modelo con regla de transición de probabilidad variable y retardo en cambio de información

El script `OLE_EJ_din_voto_retardo.php` define un modelo de voto en el cual dos clases V1 y V2 se alteran mutuamente entre si. Inicialmente se define una probabilidad de 10 % en la capacidad de cambio de opinión mutuo entre objetos.²⁶

El elemento de retardo implementado en el modelo se establece mediante una variable que controla el valor de la probabilidad de influencia de cambio de opinión entre las clases. A partir de 100 pasos o más de simulación comienza a llegar la información administrada por el retardo a las probabilidades de cambio de cada clase. Los valores realimentados de probabilidad son simétricos, respectivamente $p\%$ y $(1-p)\%$ para cada clase.

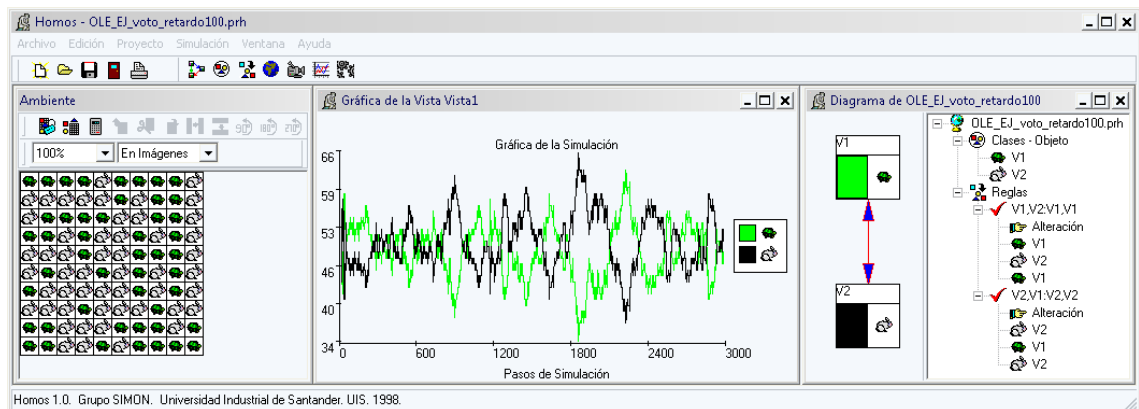
La variable `$rett` determina la amplitud del retardo, `$retardo` es el contenedor de los datos.

Al finalizar la ejecución del script se reporta:

- Los últimos valores almacenados dentro de la estructura de retardo.
- La cantidad de pasos realizados y el número de objetos de cada clase presente durante la simulación.

En `OLE_EJ_din_voto_retardo.txt` se presenta un ejemplo de resultado de ejecución.

Figura E.10: Resultados de simulación de `OLE_EJ_din_voto_retardo.php`



Nótese las diferencias entre la representación del diagrama BOR según Homos 1.0 versus la figura E.11 y la representación transparente E.12 en la página 599.

Fuente: Autor.

²⁶Descripción en prosa del modelo.

Tabla E.7: Modelo BOR de OLE_EJ_din_voto_retardo.php

Símbolos	Regla	Notación	Metáfora descriptiva	%	\$	Evaluación de celdas vecinas
a	r_1	$[a, b : a, a]$	a Altera a b Cambio de opinión asistida de b a a	10	1	La del objeto relacionado b
b	r_2	$[b, a : b, b]$	b Altera a a Cambio de opinión asistida por de a a b	10	1	La del objeto relacionado a
Metasímbolos				#		Distribución
p	s_1	$[p : a]$	Persona de voto a	50		Aleatoria
	s_2	$[p : b]$	Persona de voto b	50		Aleatoria

El modelo se implementa en la herramienta usando un ambiente de 10×10 celdas de borde cíclico, se insertan 50a y 50b asegurándose de no permitir espacios vacíos. Las reglas de transición han sido expresadas en notación de permutación. % y \$ son abreviaturas de Probabilidad y Prioridad de ejecución. # representa una función de conteo. Este modelo está contenido dentro del SPRT 2S2D. No presenta relaciones de movimiento explícitas.

Fuente: Autor.

Tabla E.8: Caracterización en diferencias de OLE_EJ_din_voto_retardo.php

Id	Regla	Acción bruta	Acción neta	Actividad	Aporte neto de primitivas	Población	Actividad poblacional
1	$[a, b : a, a]$	$-a \ -b \ +a \ +a$	$-b \ +a$	$n_1 R_1$	$-n_1 b + n_1 a$	A	$n_1 - n_2$
2	$[b, a : b, b]$	$-b \ -a \ +b \ +b$	$-a \ +b$	$n_2 R_2$	$-n_2 a + n_2 b$	B	$n_2 - n_1$
Tr = 0							

Tabla E.9: Actividad de primitivas y patrones presentes en OLE_EJ_din_voto_retardo.php

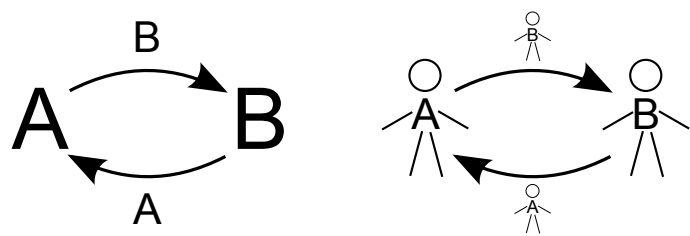
Id	Primitiva	Actividad	Tipo	Id	Patrón	Actividad
1	a : a	n_1	p	1	A, B : A, A	$n_1 + n_2$
2	a : b	n_2	c			
3	b : a	n_1	c			
4	b : b	n_2	p			
Tp = 0			$\Sigma = \frac{1}{2}c + \frac{1}{2}p$			

El modelo presenta transición completa entre símbolos, usa todas las primitivas de transiciones posibles. Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k . p y c son abreviaturas de Permanencia y Cambio. Tr y Tp son abreviaturas de tendencia de reglas y tendencia de primitivas del sistema. Fuente: Autor.

Representación transparente del modelo de transiciones presente en OLE_EJ_din_voto_retardo. En la ecuación (E.12) se agrupan las representaciones de transición por su similitud estructural.

$$\begin{aligned}
 Modelo &= \sum n_i R_i \\
 &= n_1 R_1 + n_2 R_2 \\
 &= n_1 \left[b \overset{a}{\rightarrow} a \right] + n_2 \left[a \overset{b}{\rightarrow} b \right]
 \end{aligned}
 \tag{E.12}$$

Figura E.11: Representación de diagrama BOR del modelo OLE_EJ_din_voto_retardo.php



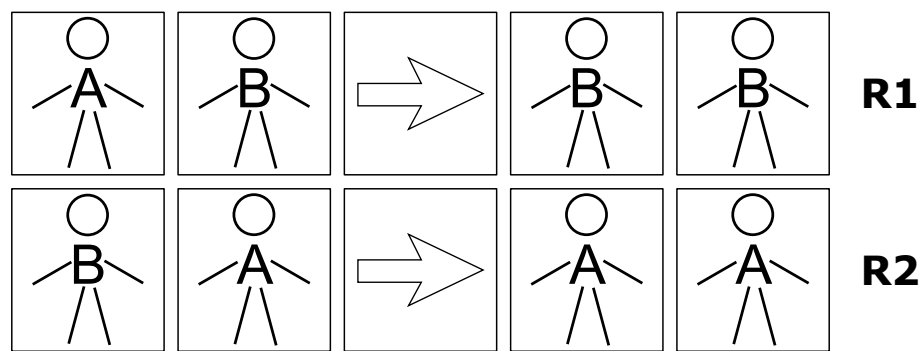
Se presentan de izquierda a derecha dos formas de representación gráfica del diagrama BOR

- Usando solo los símbolos o identificadores de clase.
- Reemplazo de símbolos por imágenes o pictogramas.

Fuente: Autor.

Figura E.12: Representación gráfica de transiciones en OLE_EJ_din_voto_retardo.php

Transiciones de voto



Fuente: Autor.

Script E.17: Un modelo de voto con retardo

```
1 //Modelo de voto con cambio de opinión con retardo
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = 'c:\_phplets\OLE'; //directorio por defecto
4 $phplet_slf_ = "OLE_EJ_din_voto_retardo.php"; //nombre archivo
5 // $phplet_lib_ = ''; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 $rett = 100; //valor de retraso del retardo implementado en el modelo actual
9
10 echo "Modelo de voto<br>"; //ml
11 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
12 $h1->proyectonuevo('C:\Program Files\UIS\Homos\OLE_EJ_voto_retardo'. $rett.'.prh');
13 $h1->visible=false;
14 echo "Crear clases-objeto<br>";
15 $o1="V1";
16 $o2="V2";
17 $h1->claseobjetocrear($o1,"C:\Program Files\UIS\Homos\Recursos\Tortuga.bmp",hexdec(strrev("00FF00"))
18     ),"Votante 1");
19 $h1->claseobjetocrear($o2,"C:\Program Files\UIS\Homos\Recursos\Conejo.bmp",hexdec(strrev("000000"))
20     ),"Votante 2");
21 echo "Crear reglas<br>";
22 $r1="V1,V2:V1,V1";
23 $r2="V2,V1:V2,V2";
24 //reglas de cambio de opinion de los votantes, influencias entre votantes
25 $h1->reglacrearalteracion($r1,10,1,$o1,$o2,$o1,$r1, Un votante de opinion V1 hace que un opositor
26     cambie de parecer, de V2 a V1");
27 $h1->reglacrearalteracion($r2,10,1,$o2,$o1,$o2,$r2, Un votante de opinion V2 hace que un opositor
28     cambie de parecer, de V1 a V2");
29
30 echo "Configuración del ambiente<br>"; $ciclico=1; $finito=2; $infinito=3;
31 $nfilas=10; $ncmnas=10;
32 $nceldas=$nfilas*$ncmnas;
33 $h1->numerofilas=$nfilas;
34 $h1->numerocolumnas=$ncmnas;
35 $h1->tipolimites=$ciclico;
36
37 echo "Crear Escenario<br>";
38 $numpasos=3000; //duracion de la simulación
39 $h1->escenariocrear("ambiente político",$numpasos);
40 $h1->escenarioseleccionar(0);
41 $h1->objetosinsertar($o1,"Aleatorio",$nceldas/2,0,0);
42 $h1->objetosinsertar($o2,"Aleatorio",$nceldas/2,0,0);
43 $h1->colorceldas=hexdec("CCCCC"); //gris
44 echo "Actualizar el ambiente<br>"; $h1->ambienteactualizar; $h1->visible=true;
45
46 echo "\nCiclo de simulación<br>";
47 $ciclo=0;
48 $pasofinal=$numpasos-1;
49 $pospaso=0;
50 $ultimodato = $h1->objetosdarcantidad($pasofinal);
51
52 //vector con valores de retardo: $retardo
53 $reti=0; //posicion dentro del vector de valores de retardo
54
55 $h1->simulacioniniciar(false);
56 while($ultimodato[0]==-1){ //simular hasta llegar al paso final
57     $h1->simulacionparar; //pausa a la simulacion actual
58
59     //determinar el paso actual en ml
60     for($j=$pospaso;$j<$pasofinal;$j++){
61         $vec=$h1->objetosdarcantidad($j); //obtener los valores de paso indicados
62         if ($vec[0]==-1) break;
63     }
64     $pospaso=$j; //Último indice de paso conocido en h1
65
66     if ($j>0){
67         $actualdato = $h1->objetosdarcantidad($j-1); //obtener los datos del paso actual
68         //si se ha definido un valor de retardo entonces usar el codigo para ese proposito
69         if($rett>0){
70             if($ciclo<$rett+1){
71                 $retardo[] = $actualdato[0]/$nceldas; //llenar el vector de retardo
72             } else {
73                 if($reti>$rett) $reti=0; //posicionamiento ciclico dentro de
74                     retardo;
```

```

71 //probabilidad de alteracion en funcion de la poblacion contraria
72                                     $h1->reglaasignarprobabilidad($r1,1-$retardo[$reti]);
73                                     $h1->reglaasignarprobabilidad($r2, $retardo[$reti]);
74 //actualizacion del valor actual de poblacion
75                                     $retardo[$reti]=$actualdato[0]/$nceldas;
76 //puntero de avance en la tabla de retardo
77                                     $reti++;
78                                     }
79                                     }
80                                     }
81                                     $ciclo++;
82                                     $ultimodato = $h1->objetosdarcantidad($pasofinal);
83                                     $h1->simulacionparar; //continuar la simulacion
84 }
85 //vector con los ultimos valores de retardo calculados
86 print_r($retardo);
87
88 //finalización de simulaciones
89 $h1->simulacionterminar;
90 usleep(100000); //tiempo para cierre
91
92 echo "Se realizaron $ciclo ciclos de evaluación del estado de simulación del modelo 1<br>";
93
94 echo "\nPresentación de resultados del modelo 1<br>";
95 echo "<table>\n<tr><td>Paso</td>";
96 $numclases=$h1->cuantasclasesobjeto;
97 for($i=0;$i<$numclases;$i++) echo "<td>".$h1->claseobjeto($i)."</td>";
98 echo "\n";
99 for($paso=0;$paso<$numpasos;$paso++){
100     $resultados = $h1->objetosdarcantidad($paso);
101     echo "<tr><td>$paso</td>\n";
102     foreach($resultados as $item) echo "<td>$item</td>\n";
103     echo "</tr>\n";
104 }
105 echo "</table>\n";
106
107 echo "Cierre del enlace y terminación del script<br>\n";
108 $h1->proyectoguardar; //llamada explicita al proceso de guardado del modelo
109 $h1->proyectocerrar(false); //cerrar y salir
110
111 $h1=null;
112 unset($h1);

```

E.4.3. Ejemplo de control simultáneo de dos modelos

El script `OLE_EJ_pulso_m1_m2.php` define y relaciona dos modelos identificados como \$h1 y \$h2 que implementan la dinámica de hostilidad de una población bacteriana según las condiciones presentes del ambiente.

- El modelo \$h1 implementa un conjunto de bacterias en una probeta. Solo hay bacterias de un mismo tipo en la probeta: pueden moverse, reproducirse y morir. Las bacterias tienen normalmente una tasa de mortalidad de 50 % y un porcentaje de movilidad del 10 %. Cuando se superan las 400 bacterias en la probeta se inicia entre ellas una etapa de hostilidad: un aumento notable en el movimiento y una incremento en la posibilidad de muerte de cada bacteria.²⁷
- El modelo \$h2 representa un factor ambiental que aplica sobre las bacterias en la probeta, por ejemplo temperatura, humedad o luz. En este caso se implementa un modelo de voto en el cual existen dos objetos E1 y E2. Para generar una amplia variabilidad en los valores obtenidos, los objetos implementados tienen movimiento, pueden transformarse libremente de una clase a otra y pueden alterarse mutuamente entre sí.²⁸

La interacción prevista entre modelos es la siguiente: en \$h1 la probabilidad de reproducción de las bacterias en la probeta se hace dependiente de la cantidad de objetos E1 presentes en \$h2. Según sean los valores obtenidos en \$h2 se inducirá o no hostilidad en la población de bacterias.

Al finalizar la ejecución del script se reporta:

- Una lista de sucesos de simulación: presentación de la secuencia de pasos o momentos en los cuales se dio un evento de hostilidad entre bacterias.
- La cantidad de pasos realizados y el total de bacterias presente en cada paso durante todo el proceso de simulación.

En `OLE_EJ_pulso_m1_m2.txt` se presenta un ejemplo de resultado de ejecución.

²⁷ Descripción en prosa del primer modelo.

²⁸ Descripción en prosa del segundo modelo.

Tabla E.10: Modelo BOR de OLE_EJ_pulso_m1_m2.php - Bacterias en probeta

Símbolos	Regla	Notación	Metáfora	%	\$	Evaluación de celdas vecinas
b	r_1	$[b, _ : _, b]$	Movimiento bacteria Bacteria.Mover	10	1	Toda dirección disponible distancia de paso = 2
–	r_2	$[b, _ : b, b]$	Reproducción por expansión Bacteria.Reproducir	0	1	Toda dirección disponible
	r_3	$[b, b : b, _]$	Retracción bacterial Violencia interbacterial Bacteria.Agredir	\$heb	1	Toda dirección disponible
Metasímbolos				#		Distribución
? Desconocido	s_1	$[? : _]$	Desconocido como espacio vacío en ambiente	∞		Uniforme
	s_2	$[_ : b]$	Inserción de b en espacios vacíos del ambiente	5		Aleatoria

El modelo se implementa en la herramienta usando un ambiente de 50×50 celdas de borde cíclico, se ubican aleatoriamente 5 bacterias.
Las reglas de transición han sido expresadas en notación de permutación. % y \$ son abreviaturas de Probabilidad y Prioridad de ejecución. # representa una función de conteo.
Si las condiciones ambientales son favorables a las bacterias, se permite la ejecución de r_2 . La agresividad entre bacterias es dinámica, r_3 se ejecuta según el valor de probabilidad establecido en \$heb.
El movimiento es una transición de referencia, $[b, _ : _, b]$ se interpreta como $[b, _ : \&2, \&4]$.
Este modelo está contenido dentro de los SPRT 2S2D y 2RD.
Fuente: Autor.

Tabla E.11: Caracterización en diferencias del modelo #1, Bacterias en probeta

Id	Regla	Acción bruta	Acción neta	Actividad	Aporte neto de primitivas	Población	Actividad poblacional
1	$[b, -: -, b]$	-b - ₋ + ₋ +b	0	$n_1 R_1$	0	B	$n_2 - n_3$
2	$[b, -: b, b]$	-b - ₋ +b +b	- ₋ +b	$n_2 R_2$	$-n_{2-} + n_2 b$	-	$n_3 - n_2$
3	$[b, b: b, -]$	-b -b +b + ₋	-b + ₋	$n_3 R_3$	$-n_3 b + n_{3-}$		

Tr = 0

Tabla E.12: Actividad de primitivas y patrones presentes en Bacterias en probeta

Id	Primitiva	Actividad	Tipo	Id	Patrón	Actividad
1	-:b	$n_1 + n_2$	c	1	A, B: B, A	n_1
2	b:-	$n_3 + n_1$	c	2	A, B: A, A	n_2
3	b:b	$n_2 + n_3$	p	3	B, B: B, A	n_3
Tp = 0			$\Sigma = \frac{2}{3}c + \frac{1}{3}p$			

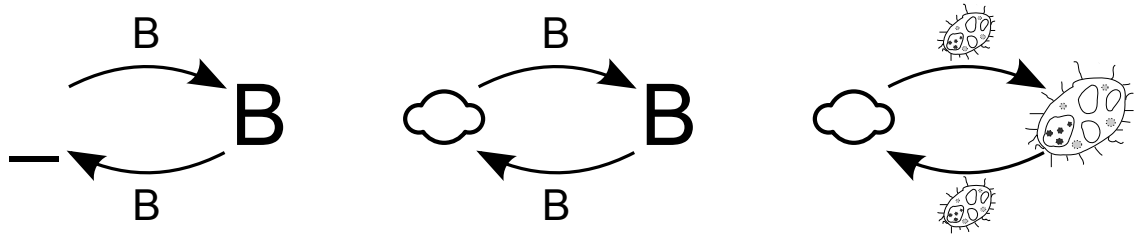
El modelo presenta transición incompleta entre símbolos. Usa explícitamente 3 de 4 primitivas de transiciones posibles. Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k . p y c son abreviaturas de Permanencia y Cambio. Tr y Tp son abreviaturas de tendencia de reglas y tendencia de primitivas del sistema. Fuente: Autor.

Representación transparente de modelo #1 de OLE_EJ_pulso_m1_m2.php.

En la ecuación (E.13) se agrupan las representaciones de transición por su similaridad estructural.

$$\begin{aligned}
 \text{Modelo}_1 &= \sum n_i R_i \\
 &= n_1 R_1 + n_2 R_2 + n_3 R_3 \\
 &= n_1 \left[\begin{array}{c} b \rightarrow - \\ - \rightarrow b \end{array} \right] + \\
 &\quad + n_2 \left[- \xrightarrow{b} b \right] + n_3 \left[b \xrightarrow{b} - \right]
 \end{aligned} \tag{E.13}$$

Figura E.13: Representación de diagrama BOR del modelo #1, Bacterias en probeta



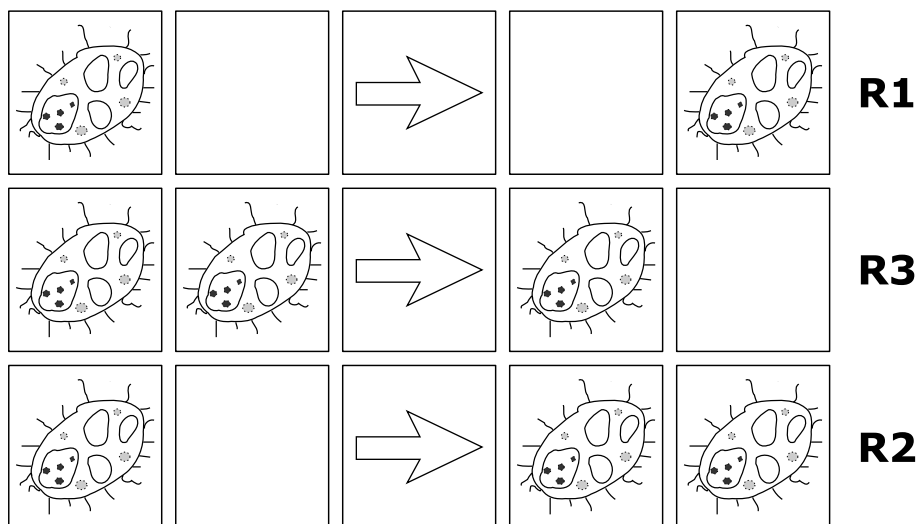
Se presentan de izquierda a derecha tres formas de representación gráfica del diagrama BOR

- Usando solo los símbolos o identificadores de clase.
- Mixto entre símbolos y nube que representa el espacio.
- Reemplazo de símbolos por imágenes o pictogramas.

Fuente: Autor.

Figura E.14: Representación gráfica de transiciones de Bacterias en probeta

Transiciones hostilidad bacteriana



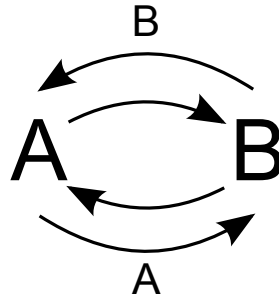
Fuente: Autor.

Representación transparente de modelo #2 de OLE_EJ_pulso_m1_m2.php.

En la ecuación (E.14) se agrupan las representaciones de transición por su similaridad estructural.

$$\begin{aligned}
 Modelo_2 &= \sum n_i R_i \\
 &= n_1 R_1 + n_2 R_2 + n_3 R_3 + n_4 R_4 + n_5 R_5 + n_6 R_6 \\
 &= n_3 \begin{bmatrix} a & \rightarrow & - \\ - & & a \end{bmatrix} + n_4 \begin{bmatrix} b & \rightarrow & - \\ - & & b \end{bmatrix} + \\
 &\quad + n_1 \left[a \xrightarrow{b} b \right] + n_2 \left[b \xrightarrow{a} a \right] + \\
 &\quad + n_5 [b \rightarrow a] + n_6 [a \rightarrow b]
 \end{aligned} \tag{E.14}$$

Figura E.15: Representación de diagrama BOR del modelo #2, Factor ambiental



Debido a que no se establecieron elementos gráficos con los cuales representar las clases del modelo, solo se muestra el diagrama de relación entre símbolos.

Fuente: Autor.

Tabla E.13: Modelo BOR de 0LE_EJ_pulso_m1_m2.php - Factor ambiental

Símbolos	Regla	Notación	Metáfora	%	\$	Evaluación de celdas vecinas
a	r_1	$[b, a : b, b]$	Alteración de a en b en presencia de b	10	1	La del objeto relacionado a
b	r_2	$[a, b : a, a]$	Alteración de b en a en presencia de a	10	1	La del objeto relacionado b
-	r_3	$[a, - : -, a]$	Movimiento de a	100	2	Toda dirección disponible distancia de paso = 2
	r_4	$[b, - : -, b]$	Movimiento de b	100	2	Toda dirección disponible distancia de paso = 2
	r_5	$[b : a]$	b se transforma en a Generación emergente de a	1	1	Ninguna
	r_6	$[a : b]$	a se transforma en b Generación emergente de b	1	1	Ninguna

Metasímbolos		#				Distribución
?	Desconocido	s_1	$[?:\_]$	Desconocido como espacio vacío en ambiente	∞	Uniforme
		s_2	$[\_ : a]$	Insertión de a en espacios vacíos del ambiente	50	Aleatoria
		s_3	$[\_ : b]$	Insertión de b en espacios vacíos del ambiente	50	Aleatoria

El modelo se implementa en la herramienta usando un ambiente de 15×15 celdas de borde cíclico, se insertan aleatoriamente 50a y 50b. Las reglas de transición han sido expresadas en notación de permutación. % y \$ son abreviaturas de Probabilidad y Prioridad de ejecución. # representa una función de conteo. El modelo da mayor prioridad a las operaciones de cambio que a las de movimiento. El propósito es generar variaciones rápidas entre los estados a y b. El movimiento es una transición de referencia, $[a, - : -, a]$ y $[b, - : -, b]$ se interpretan como $[a, - : &2, &1]$ y $[b, - : &2, &1]$. Este modelo está contenido dentro de los SPRT 3S2D y 2RD.

Fuente: Autor.

Tabla E.14: Caracterización en diferencias del modelo #2, Factor ambiental

Id	Regla	Acción bruta	Acción neta	Actividad	Aporte neto de primitivas	Población	Actividad poblacional
1	[b, a : b, b]	-b -a +b +b	-a +b	$n_1 R_1$	$-n_1 a + n_1 b$	A	$n_2 + n_5 - n_1 - n_6$
2	[a, b : a, a]	-a -b +a +a	-b +a	$n_2 R_2$	$-n_2 b + n_2 a$	B	$n_1 + n_6 - n_2 - n_5$
3	[a, _ : _, a]	-a -_ +_ +a	0	$n_3 R_3$	0	-	0
4	[b, _ : _, b]	-b -_ +_ +b	0	$n_4 R_4$	0		
5	[b : a]	-b +a	-b +a	$n_5 R_5$	$-n_5 b + n_5 a$		
6	[a : b]	-a +b	-a +b	$n_6 R_6$	$-n_6 a + n_6 b$		

Tr = 0

Tabla E.15: Actividad de primitivas y patrones presentes en Factor ambiental

Id	Primitiva	Actividad	Tipo	Id	Patrón	Actividad
1	_ : a	n_3	c	1	A, B : A, A	$n_1 + n_2$
2	a : _	n_3	c	2	A, B : B, A	$n_3 + n_4$
3	_ : b	n_4	c	3	A : B	$n_5 + n_6$
4	b : _	n_4	c			
5	a : b	$n_1 + n_6$	c			
6	b : a	$n_2 + n_5$	c			
7	a : a	n_2	p			
8	b : b	n_1	p			

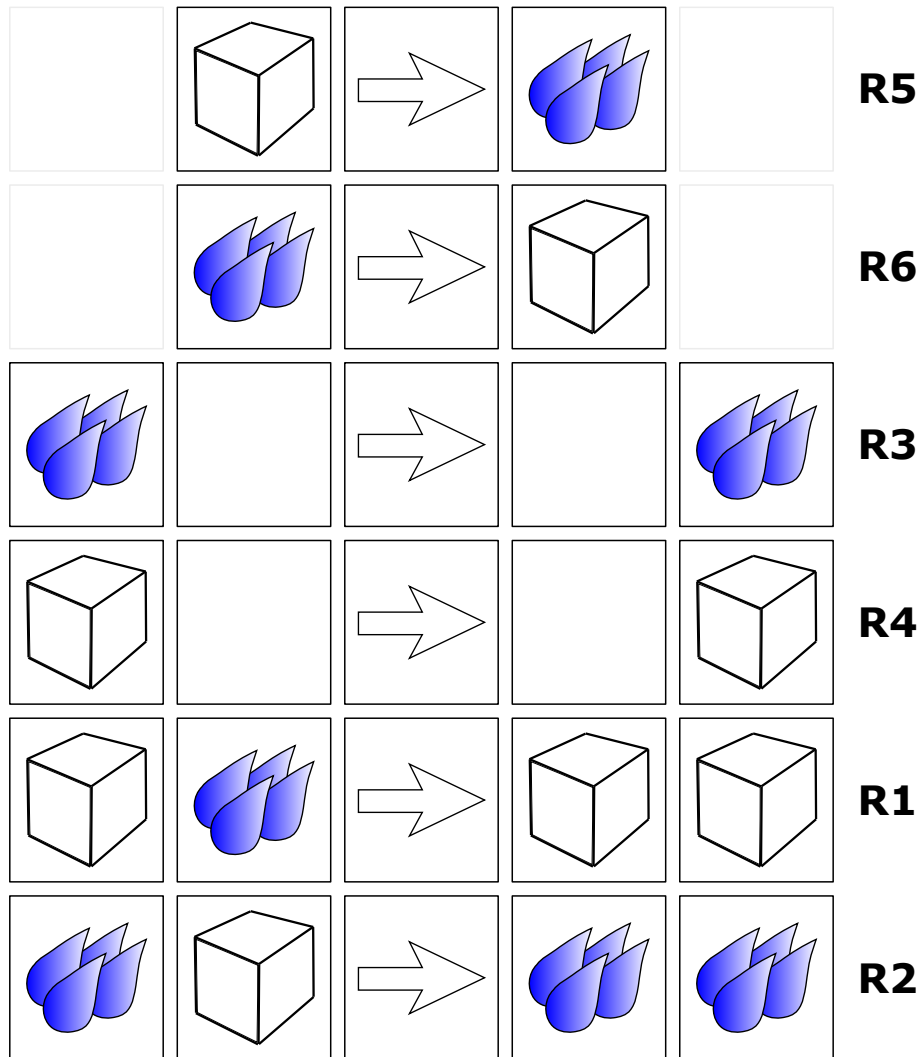
Tp = 0 $\Sigma = \frac{3}{4}c + \frac{1}{4}p$

El modelo presenta transición incompleta entre símbolos. Usa explícitamente 8 de 9 primitivas de transiciones posibles. Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k . Tr y Tp son abreviaturas de tendencia de reglas y tendencia de primitivas del sistema p y c son abreviaturas de Permanencia y Cambio.

Fuente: Autor.

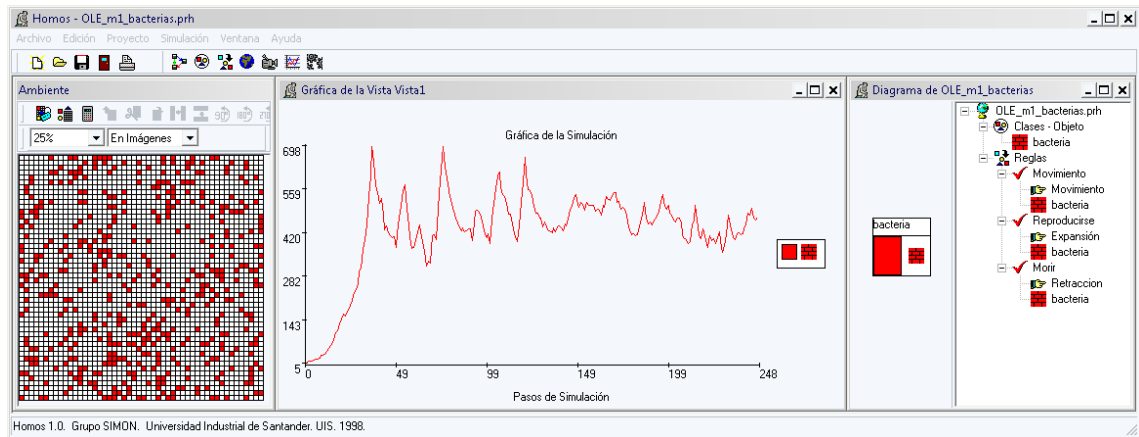
Figura E.16: Representación gráfica de transiciones de Factor ambiental

Transiciones factor ambiental



Fuente: Autor.

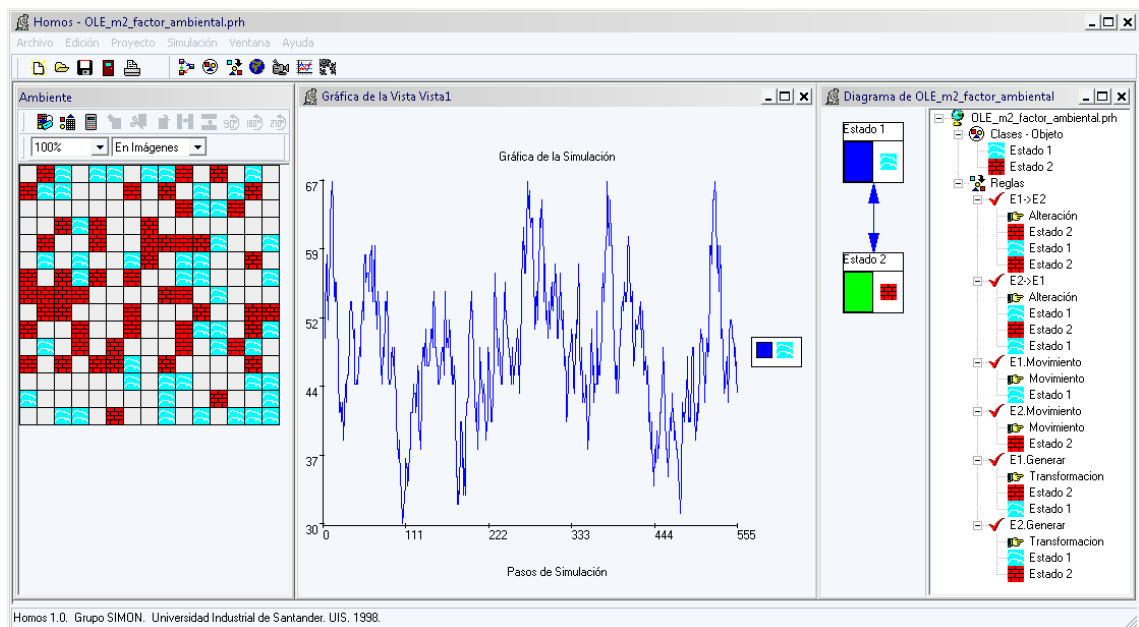
Figura E.17: Población de bacterias en OLE_EJ_pulso_m1_m2.php



Nótese las diferencias entre la representación del diagrama BOR según Homos 1.0 versus la figura E.13 y la representación transparente E.13 en la página 605.

Fuente: Autor.

Figura E.18: Factor ambiental en OLE_EJ_pulso_m1_m2.php



Nótese las diferencias entre la representación del diagrama BOR según Homos 1.0 versus la figura E.15 y la representación transparente E.14 en la página 607.

Fuente: Autor.

Script E.18: Ejemplo de manejo paralelo de 2 modelos

```

1 //Ejemplo:Conexión entre dos modelos, uno provee datos a otro
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = 'c:\_phplets\OLE'; //directorio por defecto
4 $phplet_slf_ = 'OLE_EJ_pulso_m1_m2.php'; //nombre archivo
5 //$phplet_lib_ = ''; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 echo "Ejemplo:Conexión entre dos modelos distintos, un modelo provee datos a otro<br>\n";
9 echo "Modelo 1:bacterias en una probeta<br>"; //m1
10 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
11 $h1->proyectonuevo('C:\Program Files\UIS\Homos\OLE_m1_bacterias.prh');
12 $h1->visible=false;
13 echo "Crear clases-objeto del modelo 1<br>";
14 $obj1="bacteria";
15 $h1->claseobjetocrear($obj1,"C:\Program Files\UIS\Homos\Recursos\ladrillo.bmp",hexdec(strrev("FF0000")),"Bacteria");
16 echo "Crear reglas<br>";
17 $r1="Movimiento";
18 $r2="Reproducirse";
19 $r3="Morir";
20 $heb=50; //hostilidad entre bacterias
21 $h1->reglacreararmovimiento($r1,10 ,1,255,2,$obj1,"Movimiento de la bacteria");
22 $h1->reglacrearretraccion($r2,0 ,1,255 ,,$obj1,"Orden de aumento de la cantidad de bacterias");
23 $h1->reglacrearretraccion($r3,$heb,1,255 ,,$obj1,"Orden de reducción de la cantidad de bacterias");
24 echo "Configuración del ambiente<br>"; $ciclico=1; $finito=2; $infinito=3;
25 $h1->numerofilas=50;
26 $h1->numerocolumnas=50;
27 $h1->tipolimites=$ciclico;
28 echo "Crear Escenario<br>";
29 $numpasos1=250;
30 $h1->escenariocrear("Recipiente Normal",$numpasos1);
31 $h1->escenarioseleccionar(0);
32 $h1->objetosinsertar($obj1,"Aleatorio",5,0,0);
33 $h1->colorceldas=hexdec("000000");
34 //echo "Actualizar el ambiente<br>"; $h1->ambienteactualizar; $h1->visible=true;
35
36 echo "\nModelo 2:factor ambiental sobre bacterias<br>"; //m2
37 $h2 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
38 $h2->proyectonuevo('C:\Program Files\UIS\Homos\OLE_m2_factor_ambiental.prh');
39 $h2->visible=false;
40 echo "Crear clases-objeto del modelo 2<br>";
41 $og1="Estado 1";
42 $og2="Estado 2";
43 $h2->claseobjetocrear($og1,"C:\Program Files\UIS\Homos\Recursos\agua.bmp",hexdec(strrev("0000FF")),
44 "Estado 1");
45 $h2->claseobjetocrear($og2,"C:\Program Files\UIS\Homos\Recursos\ladrillo.bmp",hexdec(strrev("00FF00")),
46 "Estado 2");
47 echo "Crear reglas<br>";
48 //adaptación de un modelo de dinámica del voto
49 $rg1="E1->E2";
50 $rg2="E2->E1";
51 $rg3="E1.Movimiento";
52 $rg4="E2.Movimiento";
53 //previsión para la generación del objetos en el estado contrario, por si alguno desaparece durante
54 la simulación
55 $rg5="E1.Generar";
56 $rg6="E2.Generar";
57 $h2->reglacrearalteracion ($rg1,10 ,1, $og2,$og1,$og2,"Paso de E1 a E2");
58 $h2->reglacrearalteracion ($rg2,10 ,1, $og1,$og2,$og1,"Paso de E2 a E1");
59 $h2->reglacreararmovimiento ($rg3,100,2,255,2,$og1, "Movimiento del objeto E1");
60 $h2->reglacreararmovimiento ($rg4,100,2,255,2,$og2, "Movimiento del objeto E2");
61 $h2->reglacreartransformacion($rg5,1 ,1, $og2,$og1, "Generar un E1 de E2");
62 $h2->reglacreartransformacion($rg6,1 ,1, $og1,$og2, "Generar un E2 de E1");
63 echo "Configuración del ambiente<br>";
64 $h2->numerofilas=15;
65 $h2->numerocolumnas=15;
66 $h2->tipolimites=$ciclico;
67 echo "Crear Escenario<br>";
68 $factor=8; //multiplicador del tiempo de simulación de m2 + exceso por imprevistos
69 $numpasos2=$numpasos1*$factor; //se establece un valor mayor que el definido en la simulación
70 destino ml
71 $h2->escenariocrear("Recipiente",$numpasos2);
72 $h2->escenarioseleccionar(0);
73 $h2->escenarioseleccionar(0);
74 $h2->objetosinsertar($og1,"Aleatorio",50,0,0);
75 $h2->objetosinsertar($og2,"Aleatorio",50,0,0);

```

```

72 $h2->colorceldas=hexdec("EEEEEE");
73 //echo "Actualizar el ambiente<br>"; $h2->ambienteactualizar; $h2->visible=true;
74
75 echo "\nCiclo de simulación<br>";
76 $pasofinal1=$numpasos1-1;
77 $pasofinal2=$numpasos2-1;
78 $ciclo=0;
79 $pospaso1=0;
80 $pospaso2=0;
81 $actualizar_regla=0;
82 $ultimodato = $h1->objetosdarcantidad($pasofinal1);
83
84 //avance sincronizado en ambos modelos
85 while($ultimodato[0]==-1){ //simular hasta terminar
86     if ($ciclo==0) { //iniciar/continuar la simulación
87         $h2->simulacioniniciar(false);
88         $h2->simulacionparar; //detener h2
89         $h1->simulacioniniciar(false);
90         $h1->simulacionparar; //detener h1
91     } else {
92         $h2->simulacionparar; //iniciar h2
93         $h2->simulacionparar; //detener h2
94         $h1->simulacionparar; //iniciar h1
95         $h1->simulacionparar; //detener h1
96     }
97     $ciclo++;
98
99 //determinar el paso actual en m1
100 for($j=$pospaso1;$j<$pasofinal1;$j++){
101     $vec=$h1->objetosdarcantidad($j); //obtener los valores de paso indicados
102     if ($vec[0]==-1) break;
103 }
104 $pospaso1=$j; //Último índice de paso conocido en h1
105
106 //determinar el paso actual en m2
107 for($k=$pospaso2;$k<$pasofinal2;$k++){
108     $vec=$h2->objetosdarcantidad($k); //obtener los valores de paso indicados
109     if ($vec[0]==-1) break;
110 }
111 $pospaso2=$k; //Último índice de paso conocido en h2
112
113 //realizar la operaciones con los datos de simulación obtenidos
114 if ($k>0){
115     $actualdato = $h2->objetosdarcantidad($k-1); //obtener los datos del paso actual
116 //alterar la probabilidad de reproducción de las bacterias en función de la población de El del
    modelo 2
117     $h1->reglaasignarprobabilidad($r2,$actualdato[0]);
118 }
119 //bloque de operaciones sobre m1
120 if ($j>0){
121     $actualdato = $h1->objetosdarcantidad($j-1); //obtener los datos del paso actual
122     $dato=$actualdato[0];
123 //definición de condiciones de aplicación del pulso $heh en el modelo 1
124 if ($dato>=400){
125     if($actualizar_regla==0){
126         $h1->reglaasignarprobabilidad($r1,100); //movilidad máxima
127         $h1->reglaasignarprobabilidad($r3,100); //hostilidad máxima
128         $actualizar_regla=1;
129         $suceso[]="Hostilidad máxima en $j";
130     }
131 }
132 else
133     if($actualizar_regla){
134         $h1->reglaasignarprobabilidad($r1,10 ); //movilidad normal
135         $h1->reglaasignarprobabilidad($r3,$heh); //normalización de $heh
136         $actualizar_regla=0;
137         $suceso[]="Hostilidad normal en $j";
138     }
139 }
140 $ultimodato = $h1->objetosdarcantidad($pasofinal1);
141 }
142 echo "Se realizaron $ciclo ciclos de evaluación del modelo 1<br>";
143 if(count($suceso)>0){
144     echo "Sucesos de simulación:<br>";
145     foreach($suceso as $item) echo "$item<br>";
146 }
147

```

```

148 //finalización de simulaciones
149 $h1->simulacionterminar;
150 $h2->simulacionterminar;
151 usleep(100000); //tiempo para cierre
152
153 echo "\nPresentación de resultados del modelo 1<br>";
154 echo "<table>\n<tr><td>Paso</td>";
155 $numclases=$h1->cuantasclassesobjeto;
156 for($i=0;$i<$numclases;$i++) echo "<td>".$h1->claseobjeto($i). "</td>";
157 echo "\n";
158 for($paso=0;$paso<$numpasos1;$paso++){
159     $resultados = $h1->objetosdarcantidad($paso);
160     echo "<tr><td>$paso</td>\n";
161     foreach($resultados as $item) echo "<td>$item</td>\n";
162     echo "</tr>\n";
163 }
164 echo "</table>\n";
165
166 echo "Cierre del enlace y terminación del script<br>\n";
167 $h1->proyectoguardar;
168 $h1->proyectocerrar(false);
169 $h2->proyectoguardar;
170 $h2->proyectocerrar(false);

```

E.4.4. Recreación del modelo Predador - Presa presente en el manual de usuario de Homos 1.0

Este modelo se describe en detalle en el manual de Homos 1.0, en el apéndice C, *Cómo crear un modelo en Homos paso a paso*, en las páginas 82-86²⁹. El modelo tomado tal cual está planteado en ese documento se expresa según la notación de transición desarrollada en la actual investigación en la tabla E.16.

El modelo pertenece a un SPRT 3S2D que referencialmente pertenece a un 2RD. Las descripciones dadas en la columna Metáfora de la tabla E.16 coinciden con los elementos existentes en las reglas de transición de Homos 1.0.

Durante la simulación se presenta la curva característica del modelo predador presa, en una versión bastante inestable que pronto llega a poblaciones en ceros. Se requieren de varios intentos para obtener un comportamiento similar al ilustrado en la figura E.24 en la página 628. Esto es así ya que depende de los valores de aleatoriedad y de la ubicación momentánea de los objetos presentes en el ambiente. Si los objetos tienden a distribuirse uniformemente en el espacio, se produce una terminación rápida de las poblaciones presentes en el modelo. En la figura E.21 en la página 620 se muestra una ejecución con largo periodo de estabilidad.

En este modelo la muerte de los objetos tiene la mayor prioridad, seguida de la reproducción y el movimiento. A partir de la curva poblacional se observa que los lobos tienden a vivir más que los conejos, esto se debe a que la probabilidad de reproducción de los lobos es mucho mayor que su decadencia, lo suficientemente alta como para mantener arriba esa población y asegurar su permanencia en la simulación por más tiempo.

El recuadro de vista mostrado en las figuras E.21 y E.24 fue manualmente agregado posteriormente a la ejecución del modelo, se realizó con propósitos de representación gráfica del comportamiento poblacional. La organización automática de las vistas no es directamente manipulable desde la interfaz OLE. Si se observa el código fuente del script, el ciclo del proceso de simulación se ha simplificado. Este es un ejemplo de mejora en el proceso de creación de modelos con la interfaz de automatización.

²⁹ Anexo 4 de la tesis de Homos 1.0.

Tabla E.16: Modelo BOR de OLE_predador_presa_6rt.php

Símbolos	Regla	Notación	Metáfora descriptiva	%	\$	Evaluación de celdas vecinas
c	Conejo	$[c, _ : _, c]$	Movimiento del conejo	100	2	Toda dirección disponible
1	Lobo	$[c, _ : c, c]$	Conejo.Mover			distancia de paso = 2
			Reproducción por expansión	10	1	Toda dirección disponible
-	Vacío	$[c : _]$	Conejo.Reproducir			
			Decadencia del conejo	0.83	0	Ninguna
			Conejo.Morir			
		$[1, _ : _, 1]$	Movimiento del lobo	100	2	Toda dirección disponible
			Lobo.Mover			distancia de paso = 2
		$[1, c : 1, 1]$	Reproducción por alteración	20	1	La del objeto relacionado c
			Lobo.Reproducir			
		$[1 : _]$	Decadencia del lobo	4.5	0	Ninguna
			Lobo.Morir			

Metasímbolos						
?	Desconocido	s_1	$[?:\_]$	Desconocido como espacio vacío en ambiente	∞	Uniforme
		s_2 <td>$[_:c]$<td>Insertión inicial de conejos en espacios vacíos del ambiente<td>100<td>Aleatoria</td></td></td></td>	$[_:c]$ <td>Insertión inicial de conejos en espacios vacíos del ambiente<td>100<td>Aleatoria</td></td></td>	Insertión inicial de conejos en espacios vacíos del ambiente <td>100<td>Aleatoria</td></td>	100 <td>Aleatoria</td>	Aleatoria
		s_3 <td>$[_:1]$<td>Insertión inicial de lobos en espacios vacíos del ambiente<td>10<td>Aleatoria</td></td></td></td>	$[_:1]$ <td>Insertión inicial de lobos en espacios vacíos del ambiente<td>10<td>Aleatoria</td></td></td>	Insertión inicial de lobos en espacios vacíos del ambiente <td>10<td>Aleatoria</td></td>	10 <td>Aleatoria</td>	Aleatoria

El modelo se implementa en la herramienta usando un ambiente de 25×50 celdas de borde cíclico, se insertan 100 conejos y 10 lobos.
Las reglas de transición han sido expresadas en notación de permutación. % y \$ son abreviaturas de Probabilidad y Prioridad de ejecución. # representa una función de conteo.
El movimiento es una transición de referencia, $[c, _ : _, c]$ y $[1, _ : _, 1]$ se interpretan como $[c, _ : \&2, \&1]$ y $[1, _ : \&2, \&1]$.
Este modelo está contenido dentro de los SPRT 3S2D y 2RD.
Fuente: Autor.

Tabla E.17: Caracterización en diferencias de OLE_predador_presa_6rt.php

Id	Regla	Acción bruta	Acción neta	Actividad	Aporte neto de primitivas	Población	Actividad poblacional
1	$c, -: -, c$	$-c \quad -_+ \quad +c$	0	n_1R_1	0	C	$n_2 - n_3 - n_5$
2	$c, -: c, c$	$-c \quad -_+ \quad +c \quad +c$	$-_+ \quad +c$	n_2R_2	$-n_{2-} + n_{2c}$	L	$n_5 - n_6$
3	$c : -_+$	$-c \quad +_+$	$-c \quad +_+$	n_3R_3	$-n_{3c} + n_{3-}$	-	$n_3 + n_6 - n_2$
4	$l, -: -, l$	$-l \quad -_+ \quad +_+ \quad +l$	0	n_4R_4	0		
5	$l, c : l, l$	$-l \quad -c \quad +l \quad +l$	$-c \quad +l$	n_5R_5	$-n_{5c} + n_{5l}$		
6	$l : -_+$	$-l \quad +_+$	$-l \quad +_+$	n_6R_6	$-n_{6l} + n_{6-}$		

Tr = -c +_+

Tabla E.18: Actividad de primitivas y patrones presentes en OLE_predador_presa_6rt.php

Id	Primitiva	Actividad	Tipo	Id	Patrón	Actividad
1	$- : c$	$n_1 + n_2$	c	1	A, B : B, A	$n_1 + n_4$
2	$- : l$	n_4	c	2	A, B : A, A	$n_2 + n_5$
3	$c : -_+$	$n_1 + n_3$	c	3	A : B	$n_3 + n_6$
4	$l : -_+$	$n_4 + n_6$	c			
5	$c : l$	n_5	c			
6	$c : c$	n_2	p			
7	$l : l$	n_5	p			
Tp = -c +l			$\Sigma = \frac{5}{7}c + \frac{2}{7}p$			

El modelo presenta transición incompleta entre símbolos. Usa explícitamente 7 de 9 primitivas de transiciones posibles. Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k . p y c son abreviaturas de Permanencia y Cambio. Tr y Tp son abreviaturas de tendencia de reglas y tendencia de primitivas del sistema

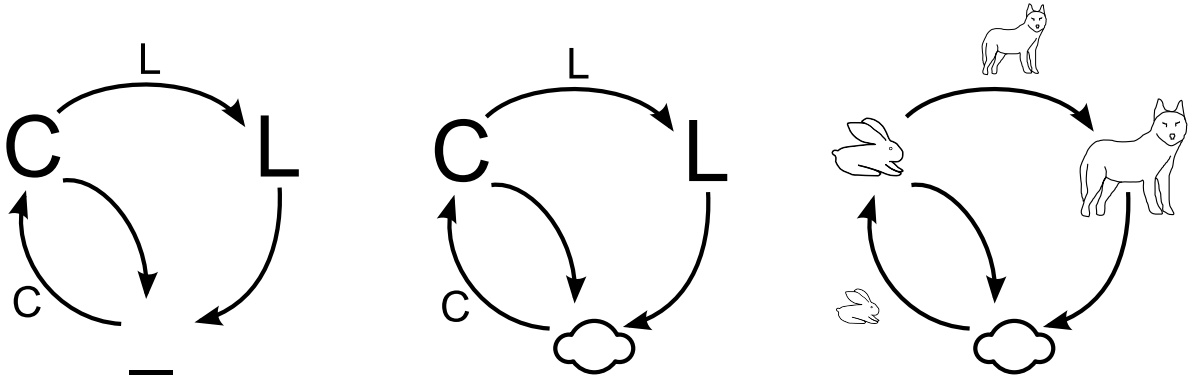
Fuente: Autor.

Representación transparente de 0LE_predador_presa_6rt.php.

En la ecuación (E.15) se agrupan las representaciones de transición por su similaridad estructural.

$$\begin{aligned}
 \text{Modelo} &= \sum n_i R_i \\
 &= n_1 R_1 + n_2 R_2 + n_3 R_3 + n_4 R_4 + n_5 R_5 + n_6 R_6 \\
 &= n_1 \begin{bmatrix} c & \rightarrow & - \\ - & & c \end{bmatrix} + n_4 \begin{bmatrix} l & \rightarrow & - \\ - & & l \end{bmatrix} + \\
 &\quad + n_2 \left[- \xrightarrow{c} c \right] + n_5 \left[c \xrightarrow{l} l \right] + \\
 &\quad + n_3 \left[c \rightarrow - \right] + n_6 \left[l \rightarrow - \right]
 \end{aligned} \tag{E.15}$$

Figura E.19: Representación de diagrama BOR del modelo predador presa de seis reglas



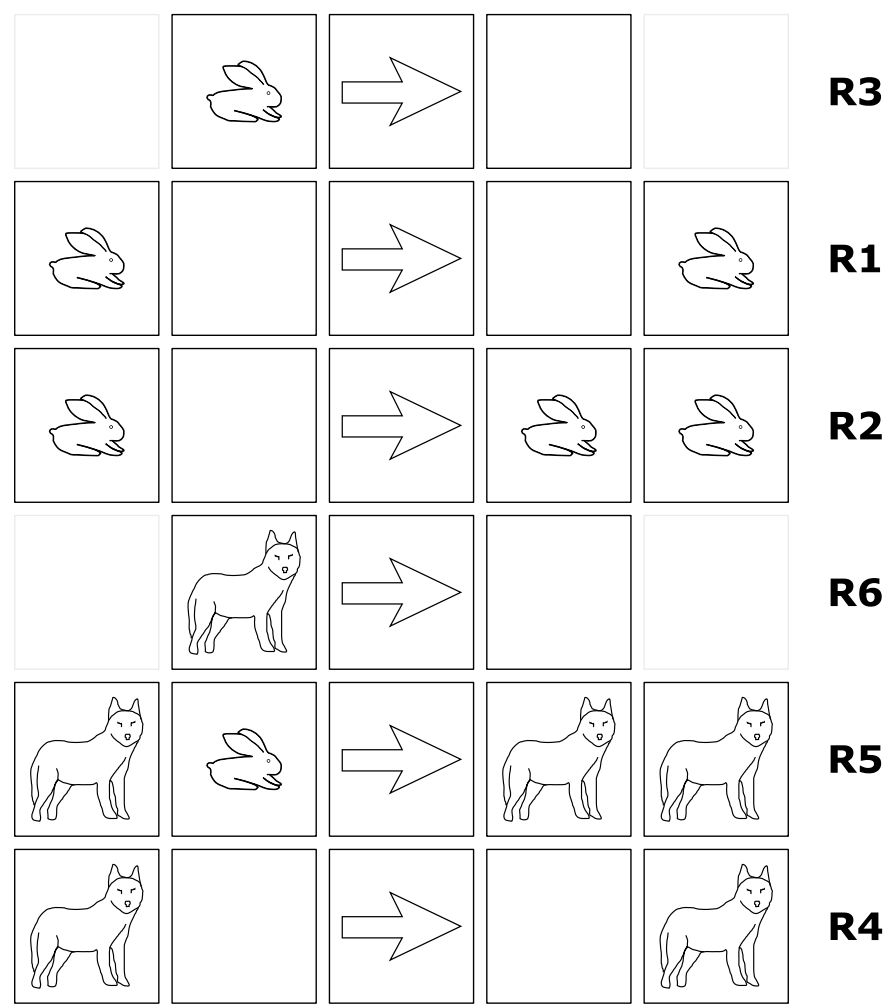
Se presentan de izquierda a derecha tres formas de representación gráfica del diagrama BOR

- Usando solo los símbolos o identificadores de clase.
- Mixto entre símbolos y nube que representa el espacio.
- Reemplazo de símbolos por imágenes o pictogramas.

Fuente: Autor.

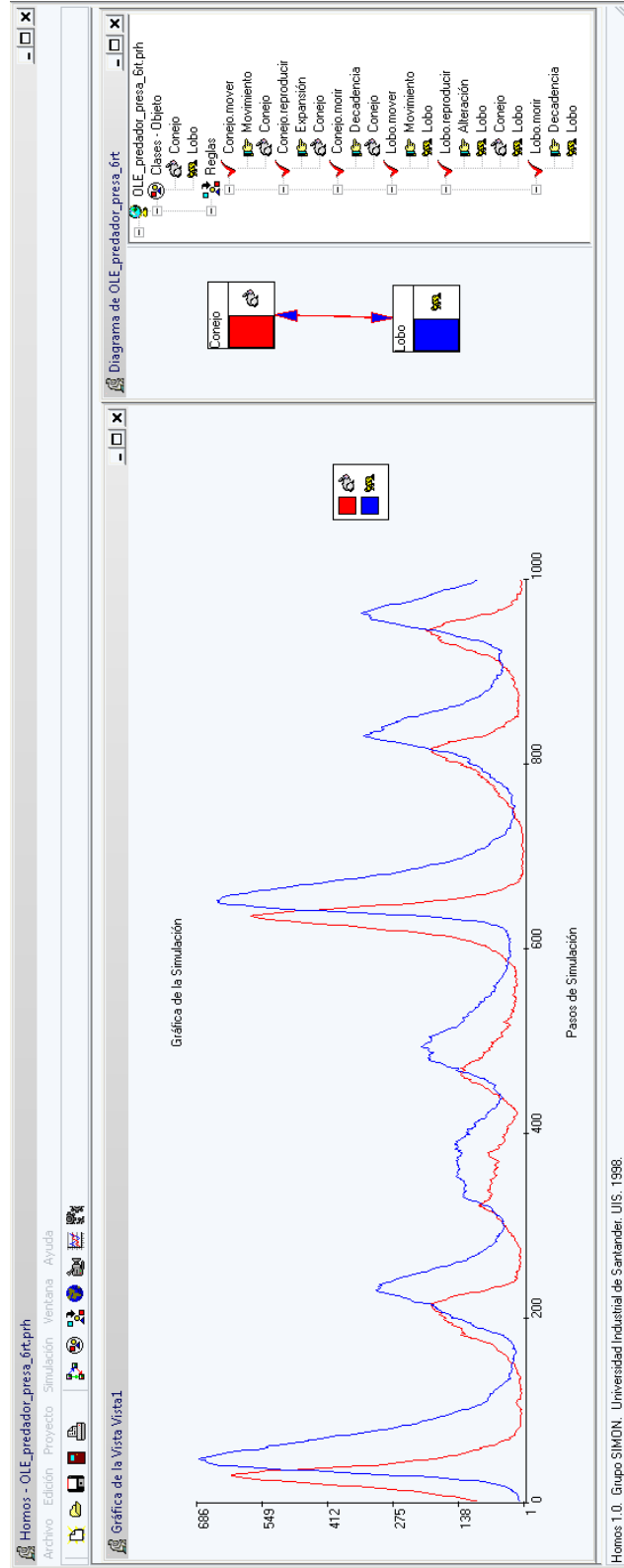
Figura E.20: Representación gráfica de transiciones de 0LE_predador_presa_6rt.php

Transiciones Predador presa de 6 reglas



Fuente: Autor.

Figura E.21: Resultados de simulación de OLE_predador_presa_6rt. .php



Nótese las diferencias entre la representación del diagrama BOR según Homos 1.0 versus la figura E.19 y la representación transparente (E.15).

Fuente: Autor:

Script E.19: Recreación del modelo Predador-Presa

```

1 //descripción del script
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = "c:/_phplets"; //directorio por defecto
4 $phplet_slf_ = "OLE_predador_presa_6rt.php"; //nombre archivo
5 //$phplet_lib_ = ""; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 ///////////////////////////////////////////////////////////////////
9 //Realización del modelo predador-presa lobos y conejos presentado
10 //en el apéndice C del manual de usuario de Homos 1.0, páginas 84 y 85.
11 ///////////////////////////////////////////////////////////////////
12
13 //enlace con la interfaz OLE de Homos 1.0
14 $id=0;
15 echo "Etapa inicial: Crear Enlace<br>\n";
16 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
17
18 //enlace con la interfaz OLE de Homos 1.0
19 echo "Etapa ".$id++." : Crear elementos del modelo<br>\n";
20 $h1->visible=false;
21
22 //crear el archivo del proyecto
23 $h1->proyectonuevo('C:\Program Files\UIS\Homos\OLE_predador_presa_6rt.prh');
24
25 //cualquier error en la entrada de datos se traduce en una terminacion inmediata de la ejecucion de
    la interfaz
26
27 //nombre de clases-objeto
28 $c="Conejo";
29 $l="Lobo";
30
31 //creacion de las clases objeto del modelo
32 //hexdec(strrev("color_en_HEXadecimal")) rojo = FF0000 azul = 0000FF
33 $h1->claseobjetocrear($c,"C:\Program Files\UIS\Homos\Recursos\Conejo.bmp",hexdec(strrev("FF0000")),
    $c);
34 $h1->claseobjetocrear($l,"C:\Program Files\UIS\Homos\Recursos\Tigre.bmp",hexdec(strrev("0000FF")),
    $l);
35
36 //creacion de las reglas de la clase conejo y lobo
37 echo "Etapa ".$id++." : Crear clase $c<br>\n";
38
39 //movimiento(nombre,probabilidad,prioridad,direccion,numero_celdas_avance,claseobjeto1,descripcion)
40 $h1->reglacrear movimiento("$c.mover",100,2,255,2,$c,"El conejo se mueve en cualquier dirección");
41
42 //expansion(nombre,probabilidad,prioridad,direccion,claseobjeto1,descripcion)
43 $h1->reglacrear expansion("$c.reproducir",10,1,255,$c,"El $c se reproduce a una tasa estandar de
    crecimiento de población");
44
45 //decadencia(nombre,probabilidad,claseobjeto,descripcion)
46 $h1->reglacrear decadencia("$c.morir",0.83,$c,"El $c muere a los 4 meses"); //por defecto prioridad
    = 0
47
48 echo "Etapa ".$id++." : Crear clase $l<br>\n";
49 //movimiento(nombre,probabilidad,prioridad,direccion,numero_celdas_avance,claseobjeto1,descripcion)
50 $h1->reglacrear movimiento("$l.mover",100,2,255,2,$l,"El $l se mueve en cualquier dirección");
51
52 //alteracion(nombre,probabilidad,prioridad,direccion,claseobjeto1,descripcion)
53 $h1->reglacrear alteracion("$l.reproducir",20,1,$l,$c,$l,"Reproducción del $l en función del
    alimento");
54
55 //decadencia(nombre,probabilidad,claseobjeto,descripcion)
56 $h1->reglacrear decadencia("$l.morir",4.5,$l,"El $l muere al año"); //por defecto prioridad = 0
57
58 echo "Etapa ".$id++." : Configuración del ambiente<br>\n";
59 $h1->numerofilas=25; //hasta 255
60 $h1->numerocolumnas=50; //hasta 255
61 $h1->tipolimites=1; //ciclico=1, finito=2, infinito=3
62
63 echo "Etapa ".$id++." : Crear Escenario<br>\n";
64 $numpasos=1000;
65 $h1->escenariocrear("Bosque",$numpasos);
66 $h1->escenarioseleccionar(0);
67 $h1->objetosinsertar("Conejo","Aleatorio",100,0,0);
68 $h1->objetosinsertar("Lobo","Aleatorio",10,0,0);
69 //color de las celdas de fondo, Gris claro
70 $h1->colorceldas=hexdec("CCCCC"); //

```

```

71
72 echo "Etapa ".$id++.".": Actualizar el ambiente<br>\n";
73 $h1->ambienteactualizar;
74 //hacer visible el ambiente y el proceso de simulación
75 $h1->visible=true;
76
77 echo "Etapa ".$id++.".": Ciclo de simulación<br>\n";
78 $pasofinal = $numpasos-1;
79 $h1->simulacioniniciar(false); //iniciar la simulacion
80 $ultimodato = $h1->objetosdarcantidad($pasofinal);
81 //continuar en el ciclo hasta que el valor del ultimo dato sea diferente de -1
82
83 while($h1->simulacionestado==1){
84 //procesar mientras se este ejecutando la simulacion, si cambia de estado se termina el ciclo. Pej:
    pausar la simulación manualmente
85 }
86
87 //debe decidirse en el momento de diseño del script si al terminar la simulacion el ambiente vuelve
    a su estado inicial o no
88 //las cantidades de objetos son siempre mayores que cero
89 $h1->simulacionterminar; //terminar la simulación
90 usleep(100000); //tiempo para operaciones finales del simulador
91
92 echo "Etapa ".$id++.".": Presentación de resultados<br>\n";
93 $resultados = $h1->objetosdarcantidad(0); //numero de clases objeto existentes en el modelo
94
95 //avance por el array de datos
96 echo"<table>";
97 for($paso=0;$paso<$numpasos;$paso++){
98     //retornar clases objeto de la fila actual
99     $resultados = $h1->objetosdarcantidad($paso);
100    //recorrer cada clase objeto en la fila actual
101    echo"<tr>\n";
102    echo "<td>$paso</td>\n";
103    foreach($resultados as $item)
104        echo "<td>$item</td>\n"; //tabla de resultados poblacionales por clase
105    echo"</tr>\n";
106 }
107 echo"</table>";
108
109 echo "Etapa ".$id++.".": Cierre del enlace y terminación del script<br>\n";
110 sleep(1);
111
112 $h1->proyectoguardar; // llamada explicita a la funcion de guardado del modelo
113 $h1->proyectocerrar(false); //cerrar y salir
114 $h1=null;
115 unset($h1);

```

E.4.5. Recreación del modelo Predador - Presa del Picture Simulator

Esta es una reimplementación del modelo descrito en el archivo Predador-Presa de A.S. Camara.prh[CFNF94] presente en la carpeta proyectos y que se incluye por defecto durante la instalación de Homos 1.0. Tomando los elementos característicos del modelo y expresándolos según la notación de transición propuesta por este proyecto, el resultado se muestra en la tabla E.19.

Estructuralmente, el modelo pertenece a un SPRT 3S2D que referencialmente pertenece a un 2RD. Todas las reglas tienen la misma prioridad de ejecución, todas son igualmente elegibles. Esto hace que durante el ciclo de cálculo del modelo se evite el descarte de reglas. La versión de seis reglas del modelo se hace uso de diferentes prioridades de ejecución y esto produce una forma de retardo en la actividad de los objetos de ese modelo. Esto se nota claramente al comparar las curvas de comportamiento reportadas en las figuras E.21 y E.24.

En general, la implementación de siete reglas es mucho más estable y dinámica que la versión de seis reglas de transición propuesta en el manual de Homos 1.0. El modelo presenta la dinámica de comportamiento esperado para el sistema predador-presa.

Se observa también que en comparación con el modelo anterior, r_5 hace que los lobos puedan reproducirse por si mismos sin requerir de los conejos. El aporte de r_5 es mínimo, la mayoría de la población de lobos está en función de r_6 , que tiene un mayor valor de probabilidad de ejecución.

La no asignación de diferentes prioridades de ejecución, los valores específicos de probabilidad de las reglas de transición y la presencia de r_5 configuran la diferencia en el comportamiento poblacional de los modelos de seis y siete reglas.

Tabla E.19: Modelo BOR de OLE_predador_presa_7rt.php

Símbolos	Regla	Notación	Metáfora	%	\$	Evaluación de celdas vecinas
c	Conejo	r_1	Movimiento del conejo Conejo.Mover	100	0	Toda dirección disponible distancia de paso = 1
1	Lobo	r_2	Reproducción por expansión Conejo.reproducir	30	0	Toda dirección disponible
-	Vacío	r_3	Decadencia del conejo Conejo.Morir	10	0	Ninguna
		r_4	Movimiento del lobo Lobo.Mover	100	0	Toda dirección disponible distancia de paso = 2
		r_5	Reproducción por expansión Lobo.Reproducir	5	0	Toda dirección disponible
		r_6	Reproducción por alteración Lobo.Comer	50	0	La del objeto relacionado c
		r_7	Decadencia del lobo Lobo.Morir	25	0	Ninguna
Metasímbolos				#		Distribución
?	Desconocido	s_1	Desconocido como espacio vacío en ambiente	∞		Uniforme
		s_2	Inserción de conejos en espacios vacíos del ambiente	100		Aleatoria
		s_3	Inserción de lobos en espacios vacíos del ambiente	10		Aleatoria

El modelo se implementa en la herramienta usando un ambiente de 25×50 celdas de borde cíclico, se insertan 100 conejos y 10 lobos.
Las reglas de transición han sido expresadas en notación de permutación. % y \$ son abreviaturas de Probabilidad y Prioridad de ejecución. # representa una función de conteo.
El movimiento es una transición de referencia, $[c, _ : _ , c]$ y $[1, _ : _ , 1]$ se interpretan como $[c, _ : \&2, \&1]$ y $[1, _ : \&2, \&1]$.
Este modelo está contenido dentro de los SPRT 3S2D y 2RD.

Fuente: Autor.

Tabla E.20: Caracterización en diferencias de OLE_predador_presa_7rt.php

Id	Regla	Acción bruta	Acción neta	Actividad	Aporte neto de primitivas	Población	Actividad poblacional
1	$c, -: -, c$	$-c \quad -_+ \quad +c$	0	n_1R_1	0	C	$n_2 - n_3 - n_6$
2	$c, -: c, c$	$-c \quad -_+ \quad +c \quad +c$	$-_+ \quad +c$	n_2R_2	$-n_{2-} + n_{2c}$	L	$n_5 - n_6 - n_7$
3	$c: -_+$	$-c \quad +_+$	$-c \quad +_+$	n_3R_3	$-n_{3c} + n_{3-}$	-	$n_3 + n_7 - n_2 - n_5$
4	$l, -: -, l$	$-l \quad -_+ \quad +_+ \quad +l$	0	n_4R_4	0		
5	$l, -: l, l$	$-l \quad -_+ \quad +l \quad +l$	$-_+ \quad +l$	n_5R_5	$-n_{5-} + n_{5l}$		
6	$l, c: l, l$	$-l \quad -c \quad +l \quad +l$	$-c \quad +l$	n_6R_6	$-n_{6c} + n_{6l}$		
7	$l: -_+$	$-l \quad +_+$	$-l \quad +_+$	n_7R_7	$-n_{7l} + n_{7-}$		

$Tr = -c + l$

Tabla E.21: Actividad de primitivas y patrones presentes en OLE_predador_presa_7rt.php

Id	Primitiva	Actividad	Tipo	Id	Patrón	Actividad
1	$-: c$	$n_1 + n_2$	c	1	A, B: B, A	$n_1 + n_4$
2	$-: l$	$n_4 + n_5$	c	2	A, B: A, A	$n_2 + n_5 + n_6$
3	$c: -_+$	$n_1 + n_3$	c	3	A: B	$n_3 + n_7$
4	$l: -_+$	$n_4 + n_7$	c			
5	$c: l$	n_6	c			
6	$c: c$	n_2	p			
7	$l: l$	$n_5 + n_6$	p			
Tp = $-c + l$		$\Sigma = \frac{5}{7}c + \frac{2}{7}p$				

El modelo presenta transición incompleta entre símbolos. Usa explícitamente 7 de 9 primitivas de transiciones posibles. Todos los flujos presentados se expresan en términos de los coeficientes de ejecución de las reglas de transición n_k . p y c son abreviaturas de Permanencia y Cambio. Tr y Tp son abreviaturas de tendencia de reglas y tendencia de primitivas del sistema

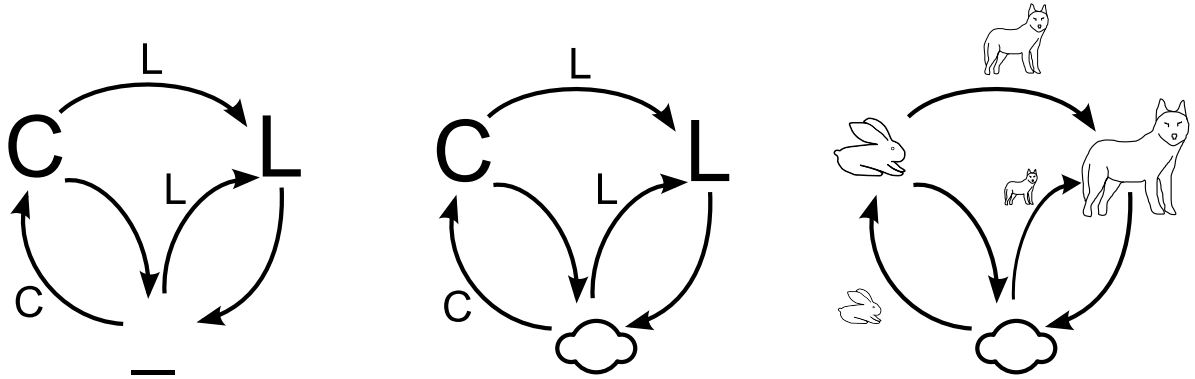
Fuente: Autor.

Representación transparente de 0LE_predador_presa_7rt.php.

En la ecuación (E.16) se agrupan las representaciones de transición por su similaridad estructural.

$$\begin{aligned}
 \text{Modelo} &= \sum n_i R_i \\
 &= n_1 R_1 + n_2 R_2 + n_3 R_3 + n_4 R_4 + n_5 R_5 + n_6 R_6 + n_7 R_7 \\
 &= n_1 \begin{bmatrix} c & \rightarrow & - \\ - & & c \end{bmatrix} + n_4 \begin{bmatrix} l & \rightarrow & - \\ - & & l \end{bmatrix} + \\
 &\quad + n_2 \left[- \xrightarrow{c} c \right] + n_5 \left[- \xrightarrow{l} l \right] + n_6 \left[c \xrightarrow{l} l \right] + \\
 &\quad + n_3 \left[c \rightarrow - \right] + n_7 \left[l \rightarrow - \right]
 \end{aligned} \tag{E.16}$$

Figura E.22: Representación de diagrama BOR del modelo predador presa de siete reglas



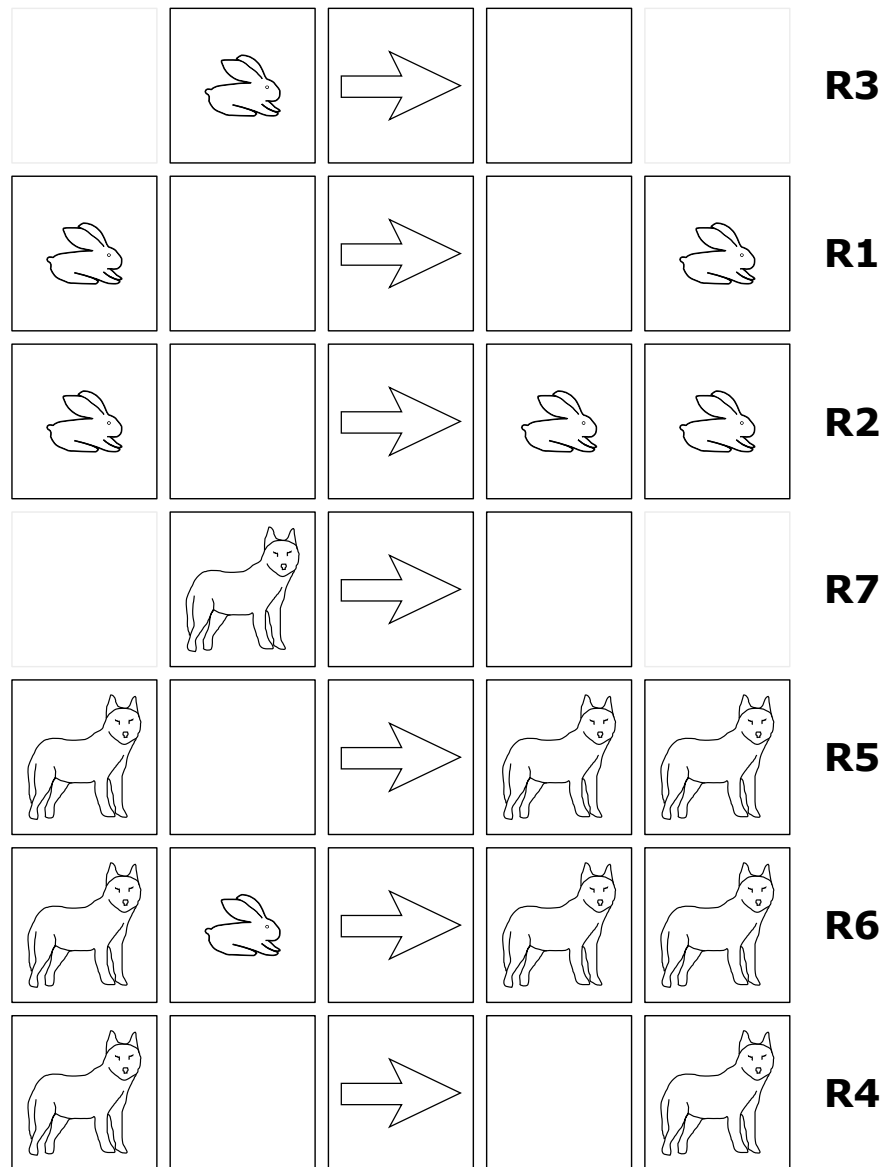
Se presentan de izquierda a derecha tres formas de representación gráfica del diagrama BOR

- Usando solo los símbolos o identificadores de clase.
- Mixto entre símbolos y nube que representa el espacio.
- Reemplazo de símbolos por imágenes o pictogramas.

Fuente: Autor.

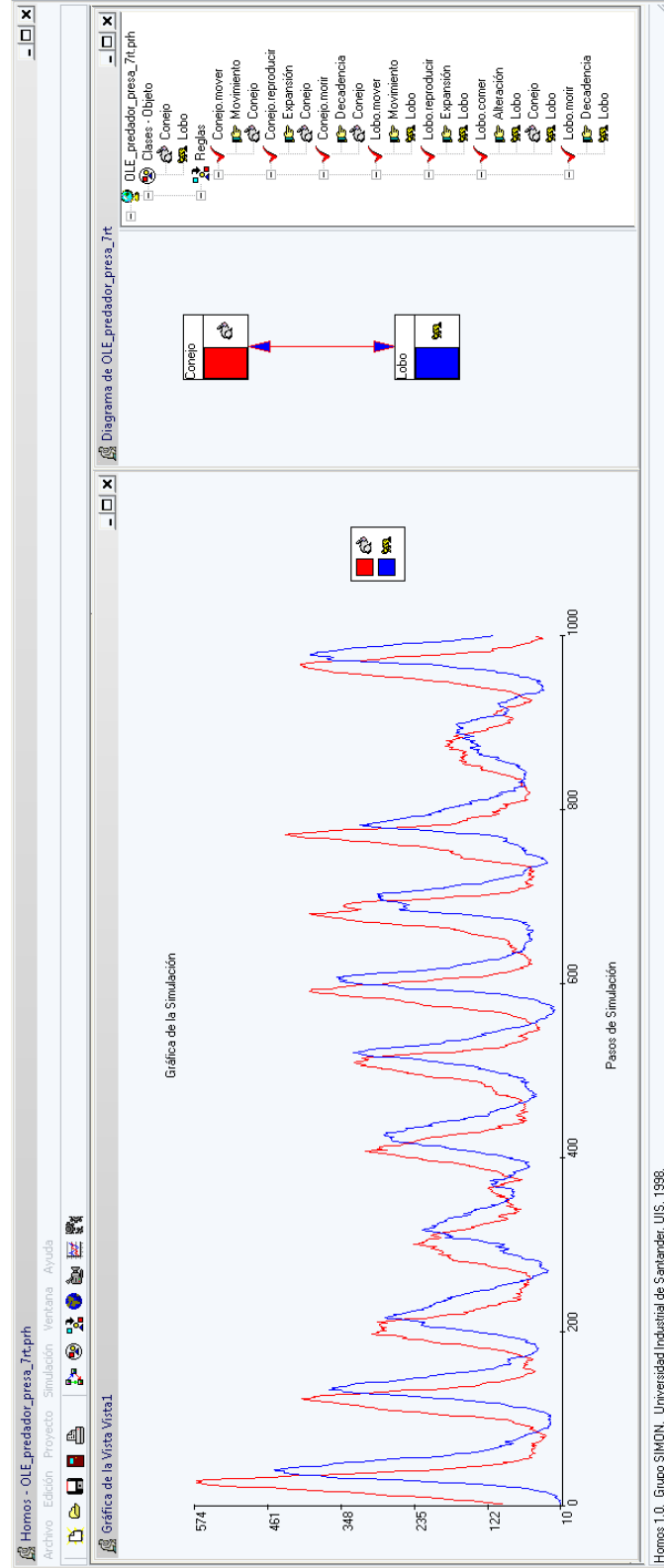
Figura E.23: Representación gráfica de transiciones de OLE_predador_presa_7rt.php

Transiciones Predador presa de 7 reglas



Fuente: Autor.

Figura E.24: Resultados de simulación de OLE_predador_presa_7rt.php



Nótese las diferencias entre la representación del diagrama BOR según Homos 1.0 versus la figura E.22 y la representación transparente (E.16).
Fuente: Autor.

Script E.20: Recreación del modelo Predador-Presa de siete reglas de transición

```
1 //descripción del script
2 set_time_limit(0); //elimina el límite de tiempo de ejecución del script
3 $phplet_dir_ = "c:/_phplets"; //directorio por defecto
4 $phplet_slf_ = "OLE_predador_presa_7rt.php"; //nombre archivo
5 //$phplet_lib_ = ""; //entrada de librería
6 $phplet_rows_ = 30; //filas
7
8 ///////////////////////////////////////////////////////////////////
9 //Reimplementación del modelo predador-presa de A.S. Camara
10 //con las funciones presentes en la interfaz OLE
11 ///////////////////////////////////////////////////////////////////
12
13 //enlace con la interfaz OLE de Homos 1.0
14 $id=0;
15 echo "Etapa inicial: Crear Enlace<br>\n";
16 $h1 = new COM("Homos.Sistema") or die("Imposible instanciar Homos.Sistema");
17
18 //enlace con la interfaz OLE de Homos 1.0
19 echo "Etapa ".$id++." : Crear elementos del modelo<br>\n";
20 $h1->visible=false;
21
22 //crear el archivo del proyecto
23 $h1->proyetonuevo('C:\Program Files\UIS\Homos\OLE_predador_presa_7rt.prh');
24
25 //cualquier error en la entrada de datos se traduce en una terminacion inmediata de la ejecucion de
    la interfaz
26
27 //nombre de clases-objeto
28 $c="Conejo";
29 $l="Lobo";
30
31 //creacion de las clases objeto del modelo
32 //hexdec(strrev("color_en_HEXadecimal")) rojo = FF0000 azul = 0000FF
33 $h1->claseobjetocrear($c,"C:\Program Files\UIS\Homos\Recursos\Conejo.bmp",hexdec(strrev("FF0000")),
    $c);
34 $h1->claseobjetocrear($l,"C:\Program Files\UIS\Homos\Recursos\Tigre.bmp",hexdec(strrev("0000FF")),
    $l);
35
36 //creacion de las reglas de la clase conejo y lobo
37 echo "Etapa ".$id++." : Crear clase $c<br>\n";
38
39 //movimiento(nombre,probabilidad,prioridad,direccion,numero_celdas_avance,claseobjeto1,descripcion)
40 $h1->reglacrearmovimiento("$c.mover",100,0,255,1,$c,"El conejo se mueve en cualquier dirección");
41
42 //expansion(nombre,probabilidad,prioridad,direccion,claseobjeto1,descripcion)
43 $h1->reglacrearexpansion("$c.reproducir",30,0,255,$c,"El $c se reproduce a una tasa estandar de
    crecimiento de población");
44
45 //decadencia(nombre,probabilidad,claseobjeto,descripcion)
46 $h1->reglacreardecadencia("$c.morir",10,$c,"El $c muere a los 4 meses");
47
48 echo "Etapa ".$id++." : Crear clase $l<br>\n";
49 //movimiento(nombre,probabilidad,prioridad,direccion,numero_celdas_avance,claseobjeto1,descripcion)
50 $h1->reglacrearmovimiento("$l.mover",100,0,255,2,$l,"El $l se mueve en cualquier dirección");
51
52 //expansion(nombre,probabilidad,prioridad,direccion,claseobjeto1,descripcion)
53 $h1->reglacrearexpansion("$l.reproducir",5,0,255,$l,"El $l se reproduce a una tasa estandar de
    crecimiento de población");
54
55 //alteracion(nombre,probabilidad,prioridad,direccion,claseobjeto1,descripcion)
56 $h1->reglacrearalteracion("$l.comer",50,0,$l,$c,$l,"Reproducción del $l en función del alimento");
57
58 //decadencia(nombre,probabilidad,claseobjeto,descripcion)
59 $h1->reglacreardecadencia("$l.morir",25,$l,"El $l muere al año");
60
61 echo "Etapa ".$id++." : Configuración del ambiente<br>\n";
62 $h1->numerofilas=25; //hasta 255
63 $h1->numerocolumnas=50; //hasta 255
64 $h1->tipolimit=1; //ciclico=1,finito=2,infinito=3
65
66 echo "Etapa ".$id++." : Crear Escenario<br>\n";
67 $numpasos=1000;
68 $h1->escenariocrear("Bosque",$numpasos);
69 $h1->escenarioseleccionar(0);
70 $h1->objetosinsertar("Conejo","Aleatorio",100,0,0);
71 $h1->objetosinsertar("Lobo","Aleatorio",10,0,0);
```

```

72 //color de las celdas de fondo, Gris claro
73 $h1->colorceldas=hexdec("CCCCC"); //
74
75 echo "Etapa ".$id++." : Actualizar el ambiente<br>\n";
76 $h1->ambienteactualizar;
77 //hacer visible el ambiente y el proceso de simulación
78 $h1->visible=true;
79
80 echo "Etapa ".$id++." : Ciclo de simulación<br>\n";
81 $pasofinal = $numpasos-1;
82
83 $h1->simulacioniniciar(false); //iniciar la simulacion
84
85 while($h1->simulacionestado==1){
86 //procesar mientras se este ejecutando la simulacion, si cambia de estado se termina el ciclo. Pej:
    pausar la simulación manualmente
87 }
88
89 //debe decidirse en el momento de diseño del script si al terminar la simulacion el ambiente vuelve
    a su estado inicial o no
90 //las cantidades de objetos son siempre mayores que cero
91 $h1->simulacionterminar; //terminar la simulación
92 usleep(100000); //tiempo para operaciones finales del simulador
93
94 echo "Etapa ".$id++." : Presentación de resultados<br>\n";
95 $resultados = $h1->objetosdarcantidad(0); //numero de clases objeto existentes en el modelo
96
97 //avance por el array de datos
98 echo"<table>";
99 for($paso=0;$paso<$numpasos;$paso++){
100     //retornar clases objeto de la fila actual
101     $resultados = $h1->objetosdarcantidad($paso);
102     //recorrer cada clase objeto en la fila actual
103     echo"<tr>\n";
104     echo "<td>$paso</td>\n";
105     foreach($resultados as $item)
106         echo "<td>$item</td>\n"; //tabla de resultados poblacionales por clase
107     echo"</tr>\n";
108 }
109 echo"</table>";
110
111 echo "Etapa ".$id++." : Cierre del enlace y terminación del script<br>\n";
112 sleep(1);
113
114 $h1->proyectoguardar; // llamada explicita a la funcion de guardado del modelo
115 $h1->proyectocerrar(false); //cerrar y salir
116 $h1=null;
117 unset($h1);

```

E.5. Algoritmo de teselado implementado en METAPOST

METAPOST³⁰ es un lenguaje de programación para crear gráficos en PostScript. Fue escrito por John D. Hobby basado en METAFONT³¹, un lenguaje de programación usado para definir fuentes vectoriales creado por Donald E. Knuth.[Knu86]

El algoritmo de relleno de superficies implementado en METAPOST fue desarrollado originalmente en Delphi durante el transcurso del proyecto. La visualización que hace el script `arsg.mp` de la forma de trabajo del algoritmo facilitó la depuración y simplificación de operaciones del código fuente en Delphi.

El script implementa la función `algoritmo_relleno` cuyos parámetros son:

- Ubicación de las coordenadas de la esquina superior e inferior del área a rellenar.
- Ubicación del punto de inicio de relleno, que debe estar contenido dentro del área a rellenar.
- Dos pares de ubicaciones centrados en el origen que establecen el sentido al avance en cada eje de movimiento X y Y dentro del área de relleno. Estos valores dependen de la configuración particular de la malla regular del grafómata a usar.
- A diferencia del algoritmo en Delphi, no se incluye dentro de los parámetros la información relacionada con el grafómata a reproducir.

Al final del script E.21 se incluye la función `beginfig(1)` donde se implementa el relleno de superficie en tres etapas, tal y como se usa el algoritmo en el prototipo software presentado en este trabajo de grado. Ejemplos de uso del script se presentan en las figuras C.60 en la página 416 y C.61 en la página 418. El diagrama de flujo del algoritmo de relleno se presenta en la figura C.59 en la página 413.

El script se encuentra en la carpeta `contenido/anexos/scripts/Metapost/arsg.mp`

Script E.21: Implementación del algoritmo de relleno de superficies en METAPOST

```
1 prologues:=3;
2
3 % establecer el nombre de los archivos como id-num.mps
4 if scantokens(mpversion) < 1.200:
5   filenameTEMPLATE
6 else:
7   outputtemplate :=
8   fi
9   "%j-%c.mps";
10
11 input automata; %version 0.3
12
13 verbatimtex
14 %\latex
15 % generic layout
16 \documentclass{article}
17 % additional packages
18 \usepackage[latin1]{inputenc}
19 \usepackage{amsmath}
20 \begin{document}
21 etex
```

³⁰<http://ect.bell-labs.com/who/hobby/MetaPost.html> y <http://www.tug.org/metapost.html>.

³¹<http://www-cs-faculty.stanford.edu/~uno/abcde.html>

```

22
23 looporientation := clockwiseorientation;
24 spacing := (30,30);
25
26 % algoritmo_relleno
27 % sentido de ejes de movimiento
28 % area a rellenar
29 % punto inicial de relleno
30
31 % codigo de algoritmo de relleno portable
32 vardef algoritmo_relleno(expr area_ulft,area_lrt,inicio,ejex,ejey) =
33 save ox,oy,en_zona,area_maxima_top,area_maxima_left,area_maxima_right,area_maxima_bottom,
34 pex,pey,em,antxy,regresando,estado,gx,gy,cslim,afuera,accion,hx,hy,dz,accion,pmx,pmy,
35 tmp,relleno;
36
37   e1x:=xpart(ejex);
38   e1y:=ypart(ejex);
39
40   e2x:=xpart(ejey);
41   e2y:=ypart(ejey);
42
43   path pex,pey,relleno;
44   pex:=origin--(e1x,e1y);
45   pey:=origin--(e2x,e2y);
46   pair em,antxy;
47
48   % maxx
49   tmp:=0;
50   if tmp<e1x: tmp:=e1x; fi;
51   if tmp<e2x: tmp:=e2x; fi;
52   maxx:=tmp;
53   % minx
54   tmp:=0;
55   if tmp>e1x: tmp:=e1x; fi;
56   if tmp>e2x: tmp:=e2x; fi;
57   minx:=tmp;
58   % maxy
59   tmp:=0;
60   if tmp<e1y: tmp:=e1y; fi;
61   if tmp<e2y: tmp:=e2y; fi;
62   maxy:=tmp;
63   % miny
64   tmp:=0;
65   if tmp>e1y: tmp:=e1y; fi;
66   if tmp>e2y: tmp:=e2y; fi;
67   miny:=tmp;
68
69   relleno:= (minx,miny)--(minx,maxy)--(maxx,maxy)--(maxx,miny)--cycle; %area de relleno real
70   draw relleno shifted inicio dashed evenly; %punto central
71
72 boolean ox,oy,en_zona,regresando;
73 string estado;
74
75 % avance de relleno
76 ox:=false;
77 oy:=ox;
78
79 % cuadrado a rellenar
80 area_maxima_top:=ypart(area_ulft);
81 area_maxima_left:=xpart(area_lrt);
82 area_maxima_right:=xpart(area_ulft);
83 area_maxima_bottom:=ypart(area_lrt);
84
85 % show area_maxima_top,area_maxima_left,area_maxima_right,area_maxima_bottom,"---";
86
87 draw (area_maxima_left,area_maxima_top)--(area_maxima_left,area_maxima_bottom)--(area_maxima_right,
88   area_maxima_bottom)--(area_maxima_right,area_maxima_top)--cycle;
89 drawdot origin shifted inicio withpen pencircle scaled 5pt;
90
91 % origen
92 gx:=xpart(inicio);
93 gy:=ypart(inicio);
94
95 em:=1/2[(e1x,e1y),(e2x,e2y)]; % punto medio
96
97 for i:= 0 upto 1: %realizar relleno de cada seccion [0,1] en el eje de avance
98   regresando:=false;

```

```

98  cslim:=0;
99  afuera:=0;
100 accion:=0;
101 dz:=0;
102 hx:=gx; %origen de relleno
103 hy:=gy;
104 antxy:=(hx,hy);
105 forever:
106     dz:=dz+1; % contador de longitud maxima
107     if (accion=2): % movimiento en eje de avance secundario
108         if oy: %movimiento en eje de avance secundario
109             hx:=hx+e2x; % moverse en el eje2
110             hy:=hy+e2y;
111         else:
112             hx:=hx-e2x; % moverse en el eje2
113             hy:=hy-e2y;
114         fi;
115         if ox: accion:=1; else: accion:=3; fi;
116         ox:=not ox; % cambio de sentido de movimiento en eje principal
117         dz:=0;
118     else: % accion = 0,1,3 = movimiento en eje de avance principal
119         if ox: % movimiento en eje de avance principal
120             hx:=hx-e1x; % moverse en eje el negativamente
121             hy:=hy-e1y;
122             accion:=3;
123         else:
124             hx:=hx+e1x; % moverse en eje el positivamente
125             hy:=hy+e1y;
126             accion:=1;
127         fi;
128 % actualizacion del limite de avance del eje principal para determinar el fin de ciclo de relleno
129     if (dz>cslim): cslim=dz; fi;
130     fi;
131
132 % presentacion del recorrido de dibujo
133 drawdot (hx,hy) withpen pencircle scaled 2pt;
134 drawarrow antxy--(hx,hy);
135 antxy:=(hx,hy);
136
137 draw relleno shifted (hx,hy) dashed evenly;
138
139     if ox: %movimiento en eje de avance principal
140         pmx:=hx+upart(em);
141         pmy:=hy+upart(em);
142     else:
143         pmx:=hx-upart(em);
144         pmy:=hy-upart(em);
145     fi;
146
147 % show pmx,pmy,"-*-";
148 % show (pmx>area_maxima_left);
149 % show (pmx<area_maxima_right);
150 % show (pmy>area_maxima_bottom);
151 % show (pmy<area_maxima_top);
152 % show "???" ;
153
154     en_zona:=(((pmx>area_maxima_left) and (pmx<area_maxima_right)) and ((pmy>area_maxima_bottom)
155         and (pmy<area_maxima_top))); %determinar en cada punto si esta dentro de cs o no.
156
157 % diferencia con el algoritmo original en pascal, cambio en el sentido de pmy
158     en_zona:=(((pmx>area_maxima_left) and (pmx<area_maxima_right)) and ((pmy>area_maxima_bottom)
159         and (pmy<area_maxima_top))); %determinar en cada punto si esta dentro de cs o no.
160
161     if en_zona:
162         regresando:=false;
163         afuera:=0;
164         estado:="dentro";
165     else:
166         if regresando:
167             afuera:=afuera+1;
168         else:
169             accion:=2;
170             regresando:=true;
171         fi;
172     fi;
173     estado:="fuera";
174     fi;
175 % show estado ,afuera ,cslim ;

```

```

173     exitif(afuera>cslim); %rellenar hasta donde es posible
174   endfor;
175   % inversión de dirección de avance en ejes , relleno de la siguiente sección
176   ox:=true;
177   oy:=ox;
178 endfor;
179 enddef;
180
181 % superposicion de rellenos
182 beginfig(1);
183 % etapa 1
184 algoritmo_relleno((1cm,1cm),(-1cm,-1cm),(0,0),(,5cm,.2cm),(0,.5cm));
185 % etapa 2
186 algoritmo_relleno((3cm,3cm),(-3cm,-3cm),(0,0),(2cm,.8cm),(0,2cm));
187 % etapa 3
188 algoritmo_relleno((9cm,9cm),(-9cm,-9cm),(0,0),(6cm,2.4cm),(0,6cm));
189 endfig;
190
191 end.

```


E.6. Hojas de cálculo y resultados de investigación

Se hace uso de hojas de cálculo para generar y organizar algunos de los resultados numéricos obtenidos durante el desarrollo de este proyecto. Específicamente se usó Excel® de Microsoft para ese propósito, dada su ubicuidad y por la experiencia manejo adquirida con esa herramienta a través de los años.

Se presenta en este apartado una prueba práctica de la validez de las relaciones que caracterizan numéricamente a los grafómatas, un reporte de las distintas distribuciones numéricas detalladas en este proyecto asociadas a los SPRT y una prueba mediante demostración conceptual aplicada de la caracterización en diferencias de los modelos BOR.

E.6.1. Prueba de relaciones numéricas entre características de Grafómatas

En la hoja de cálculo `grafomatas.xls` se valida numéricamente las funciones presentadas en la sección C.5 en la página 404. La tabla E.22 en la página siguiente contiene una descripción de las relaciones numéricas reportadas. Aunque la muestra sobre el cual se determinan las distintas relaciones pertenece a un conjunto limitado de datos, se considera que es suficiente para proponer que las relaciones descritas definen relaciones numéricas generales propias de los grafómatas.

El documento se encuentra en `contenido/anexos/scripts/Excel/grafomatas.xls`

E.6.2. Caracterización numérica de los SPRT

En la hoja de cálculo `sprt.xls` se presentan los valores numéricos que describen las distintas distribuciones de reglas de transición independientes, el total de patrones de transición presentes en los SPRT nSmD y su relación con secuencias enteras de OEIS. También se muestran valores relacionados con los límites de ejecución del script GSPRT. Listando en orden de aparición y contando desde la primera hoja, el contenido del documento es el siguiente:

1. Distribución del total de reglas de transición presente en cada SPRT nSkPmD. Se abstrae como k^m donde k es el total de primitivas usadas y m el total de dimensiones de evaluación.
2. Distribución del total de reglas de transición independientes existentes en los SPRT nSkPmD. Esta tabla presenta la relación entre la distribución de reglas independientes y el triángulo de Pascal.
3. Distribución de la cantidad de patrones de transición presentes en los nSmD completos. Se relacionan los valores estadísticos obtenidos por el script GSPRT para sistemas nSmD y algunas series de valores enteros existentes en OEIS. Se presenta la distribución de patrones de transición y dos tablas adicionales en las que se calculan valores de diferencias entre filas y columnas respectivamente. Se encontró que la diferencia entre filas fue más útil, se resaltan las secuencias OEIS implicadas en distintas las tablas.

Tabla E.22: Caracterización numérica de grafómatas

Relación	Descripción
$t_g = a_e + \frac{a_s}{2}$	El total de trazos de un grafómata es igual a la suma del total de aristas a vértices externos y la mitad de las aristas entre vértices internos o propios del grafómata.
$t_d = \frac{(a_e + a_s)}{2}$	El total de trazos en un dibúgrafo es igual al promedio de aristas necesarias para construir estructuralmente al grafómata que lo genera.
$a_i = 2t_d$	El total de aristas requeridos por un grafómata es igual al doble de trazos requeridos por su dibúgrafo.
$a_i = v_g + t_g + \{-1, 0, 1\}$	El total de aristas estructurales de un grafómata es igual a la suma del total de vértices, el total de trazos y un delta de ajuste en función de los vértices y trazos que componen la estructura.
$a_i(g_b) = a_i(g_d)$	El total de aristas en un grafómata es igual al total de aristas de su grafómata dual. g_b = grafómata base g_d = grafómata dual
$t_d(d_b) = t_d(d_d)$	El total de trazos de un dibúgrafo es igual al total de trazos de su dibúgrafo dual. d_b = dibúgrafo base d_d = dibúgrafo dual
Variable	Descripción
v_g	Total de vértices presentes en la estructura.
t_g	Total de trazos presentes en la estructura, líneas únicas requeridas por la estructura.
a_i	Total de aristas requeridas para definir una estructura de grafómata.
a_e	Total de aristas que van a vértices fuera de la forma principal del grafómata.
a_s	Total de aristas que van a vértices dentro de la forma principal del grafómata. Siempre es par.
v_d	Total de vértices requeridos para la construcción del dibúgrafo.
t_d	Total de trazos requeridos para la construcción gráfica del dibúgrafo.
$a_i(g)$	Total de aristas estructurales del grafómata g .
$t_d(d)$	Total de trazos en una estructura de dibúgrafo d .

Se propone que las relaciones encontradas tienen carácter general y se cumplen para todas las estructuras de grafómatas bidimensionales.
 Fuente: Autor.

4. Implementación de la secuencia OEIS A136565 en una hoja de cálculo. Esta serie predice la cantidad de patrones de transición presentes en los sistemas nSmD completos.
5. Cálculo del valor límite del ciclo primario o externo de construcción de sistemas SPRT nSmD. Este ciclo es de naturaleza exponencial, depende de la cantidad de símbolos y dimensiones usados en el sistema.
6. Distribución promedio del límite de diferencias entre símbolos en cada nSmD completo. Corresponde con la determinación de la cantidad de ciclos internos a realizar durante la búsqueda de patrones de transición, esta es una operación factorial que depende de la cantidad de diferencias entre símbolos presente en las expresiones de transición contenidas en cada SPRT.
7. Una tabla que reúne a las secuencias OEIS relacionadas en este trabajo.

En las hojas 1 y 2, Las franjas en amarillo indican los valores para los sistemas completos, aquellos donde se usan todas las primitivas que es posible construir con los símbolos y dimensiones de evaluación establecidos para el SPRT. En estas hojas se listan por columnas de izquierda a derecha: La cantidad de símbolos y el total de primitivas de transición usados en el sistema seguida del número total de reglas de transición producidas en cada dimensión de evaluación.

Los valores resaltados en las celdas de color naranja de la hoja 3 corresponden con valores sugeridos por este proyecto. Estos datos han sido deducidos mediante la relación entre el valor máximo en cada columna en la tabla principal (Secuencia A052171) y los valores de sus diferencias máximas entre filas (Secuencia A144047). Los valores resaltados esperan por ser corroborados analíticamente.

El documento se encuentra en `contenido/anexos/scripts/Excel/sprt.xls`

E.6.3. Prueba práctica de caracterización en diferencias de modelos BOR

En la hoja de cálculo `mbor.xls` se presentan pruebas de uso y resultados de una implementación práctica de la caracterización en diferencias de los modelos `OLE_EJ_cristales_pulso.php` y `OLE_predador_presa_7rt.php` planteados en la sección E.3 en la página 584. Compárense los resultados gráficos presentados en las figuras E.7 en la página 589 y E.24 en la página 628 con E.25 en la página 642 y E.35 en la página 647. La diferencia fundamental entre los sistemas de cálculo usados para la generación de las gráficas comparadas es la complejidad que representa el autómata celular. En la versión de la hoja de cálculo no hay tal autómata, se lo simuló mediante el uso de valores aleatorios uniformes y la siguiente función que realiza la acción poblacional de las reglas de transición sobre la clases a la que esta asociada:

$$n_k R_k = \psi(R_k) \cdot R_k \quad (E.17)$$

$$n_k = f(R_k) \cdot [\%R_k \cdot \%R_{ka} \cdot Pob(c)] \quad (E.18)$$

- R_k : Regla de transición k del modelo BOR.
- n_k : Coeficiente de actividad o frecuencia de ejecuciones de la regla k por paso de simulación.
- $f(R_k)$: Función compleja de aplicación de la regla en el ambiente. Esta función determina globalmente si se realizan transiciones con la regla asociada, reemplaza en las hojas de cálculo al autómata celular correspondiente.
- $\%R_k$: Probabilidad de ejecución de la regla de transición, la propuesta por el modelador.
- $\%R_{ka}$: Probabilidad de ejecución *real* de la regla k en el ambiente.
- $Pob(c)$: Total de población actual de la clase / símbolo c sobre la que aplica la regla de transición.

n_k es un valor entero ya que las reglas no se ejecutan parcialmente. Los corchetes en la expresión representan la función parte entera³², $int(x) = [x]$.

La función (E.18) simplifica en extremo la representación de las complejidades relacionadas con la implementación de autómatas en la hoja de cálculo, ya que asume totalidades mediante uniformidad. Se asemeja esto a los supuestos con los cuales se hace uso de las ecuaciones diferenciales en las ecuaciones de flujo-nivel en dinámica de sistemas.

La estructura de la función (E.18) es preliminar y sirve solo al propósito de prueba de concepto, ya que aplica solo a reglas simples, que afectan solo a una clase. Se espera implementar una expresión de esta función que permita representar reglas que relacionan más de una clase en su definición, que tenga en cuenta elementos como la probabilidad de encuentro en el espacio entre objetos, que incorpore condiciones de ejecución específicas de la regla de transición, etc.

³² $[x] = \begin{cases} [x] & \text{si } x \geq 0, \\ \lceil x \rceil & \text{si } x < 0 \end{cases}$

E.6.3.1. Generación de series de datos mediante caracterización en diferencias

Haciendo uso de $A(tal)$ como el total de actividad de tal , se listan en orden de aparición, las primeras curvas obtenidas del proceso de caracterización en diferencias de los modelos BOR.

1. Curvas poblacionales de las clases previstas en el modelo:

$$Pob_i(c) = Pob_{i-1}(c) + \sum n_k p_k, \text{ con } p_k = f(c) \quad (E.19)$$

El total de población de la clase c actual es igual a la suma del valor de población c anterior más la sumatoria de aportes de las reglas de transición del paso i que están en función de la clase c .

2. Curva de probabilidad *real* de ejecución de reglas de transición en el ambiente:

$$\%R_{ka} = 0 \leq x \leq 1$$

Hace uso de valores aleatorios uniformes, representa en forma simplificada la acción del autómata celular que interviene dentro del proceso de simulación.

3. Actividad de reglas de transición:

$$A(R_k) = n_k = f(R_k) \cdot [\%R_k \cdot \%R_{ka} \cdot Pob(c)] \quad (E.20)$$

La ejecución de una regla k esta en función de la población de las clases requeridas para su ejecución. En los ejemplos presentados, cada regla k está asociada solo a una población c específica. La función (E.20) es la fuente principal de los valores con los cuales se calculan las demás series de datos. Ver ecuación (E.18).

4. Actividad poblacional de la clase c presente en el modelo:

$$A(c) = \sum n_k p_k, \text{ con } p_k = f(c) \quad (E.21)$$

Es la sumatoria de toda la actividad de primitivas que están función de la clase c . Cabe destacar que en cada paso de simulación $\sum_{i=1}^k A(c_i) = 0$, el neto de la sumatoria de aportes poblacionales realizados es cero.

5. Actividad de cada primitiva presente en el modelo:

$$A(p_i) = \sum n_k R_k, \text{ con } p_i \in R_k \quad (E.22)$$

Sumatoria de toda actividad de reglas que contenga la primitiva i .

6. Actividad de primitivas de cambio y permanencia del modelo:

$$A(p_f) = \sum n_k p_k, \text{ con } p_k \in p_q \mid q \in \{Cambio, Permanencia\} \quad (E.23)$$

Es la sumatoria de toda la actividad de primitivas agrupadas según su clasificación como primitivas de Cambio o Permanencia.

7. Actividad total de primitivas del modelo:

$$A(\sum p) = \sum n_k p_k = \sum A(p_i) = \sum A(p_f) \quad (E.24)$$

Es la sumatoria de la actividad total de primitivas en cada paso de simulación.

8. Actividad de las transiciones fundamentales presentes en el modelo:

$$A(p_f) = \sum n_k p_k, \text{ con } p_k \in p_q \mid q \in \{Idem, Inst, Trans, Term\} \quad (E.25)$$

Es la sumatoria de toda la actividad de primitivas según el tipo de transición fundamental a la que pertenecen. Existen cuatro formas fundamentales: Idempotencia, Instanciación, Transformación y Terminación.

9. Actividad de cada patrón de transición presente en el modelo:

$$A(P_i) = \sum n_k R_k, \text{ con } R_k \in P_i \quad (E.26)$$

Es la sumatoria de toda la actividad de las reglas que pertenecen al patrón i .

10. Actividad total de patrones de transición presentes en el modelo:

$$A(\sum P) = \sum A(P_i) = \sum n_k R_k \approx \frac{\sum A(p_i)}{2}. \quad (E.27)$$

La actividad total de patrones es igual a la actividad total de reglas de transición o aproximadamente la mitad de toda la actividad de primitivas presente en el modelo en cada paso de simulación. La aproximación se debe a que la cantidad de primitivas que se ejecutan en las reglas no es uniforme, cada primitiva puede repetirse varias veces en una o más reglas de transición simultáneamente.

Cabe recordar que en el proceso de caracterización:

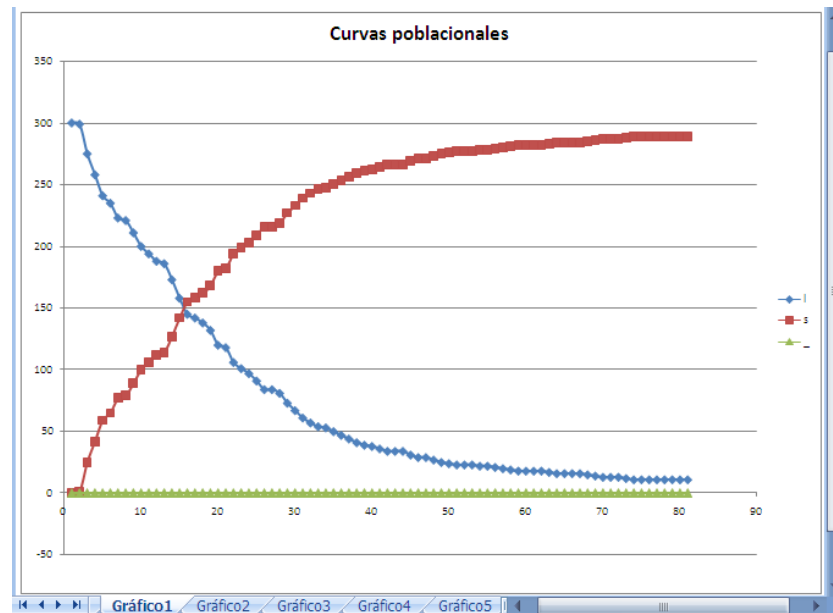
- Las primitivas hacen parte de una o más reglas de transición existentes en el modelo.
- Los patrones son excluyentes, una regla no pertenece simultáneamente a varios patrones.
- El método solo aplica a las reglas que generan cambio poblacionales netos. Esto descarta a reglas de referencia como el movimiento y el intercambio o swap.
- Las distintas sumatorias mostradas se realizan en cada paso de simulación.

En [Ake04, pág. 6] se habla de actividad en el mismo sentido que en el propuesto en este trabajo de grado; el documento hace referencia a la frecuencia de actividad en celdas, nosotros hablamos de frecuencia de ejecuciones de reglas, primitivas, patrones y demas elementos establecidos en este

apartado. Los tipos de curvas obtenidos y los demas formalismos matematicos presentados pueden ser aplicados con ajustes a lo tratado en este documento. Ambos trabajos están relacionados con sistemas de eventos discretos o DEVS. Para profundizar en estos sistemas ver [ZPK00].

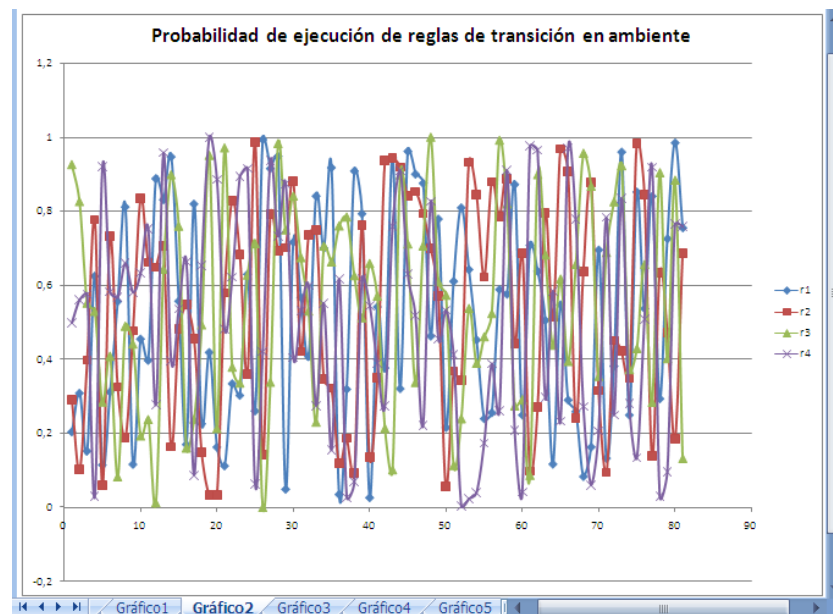
La forma de las curvas obtenidas en el proceso está en función directa de las transiciones establecidas pero también de la aleatoriedad empleada para su generación. Una aleatoriedad pobre produce resultados de igual calidad.

Figura E.25: Comportamiento poblacional de OLE_EJ_cristales_pulso



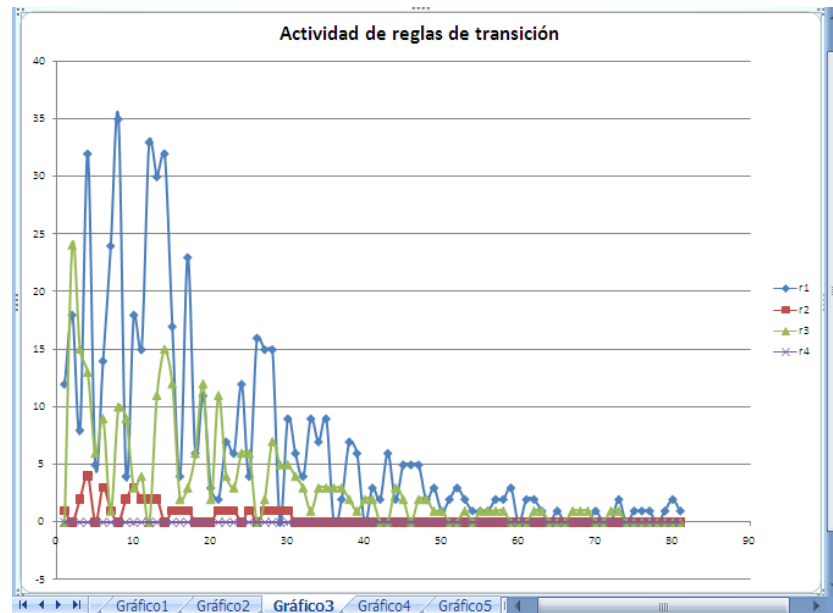
Fuente: Autor.

Figura E.26: Valores de aleatoriedad en OLE_EJ_cristales_pulso



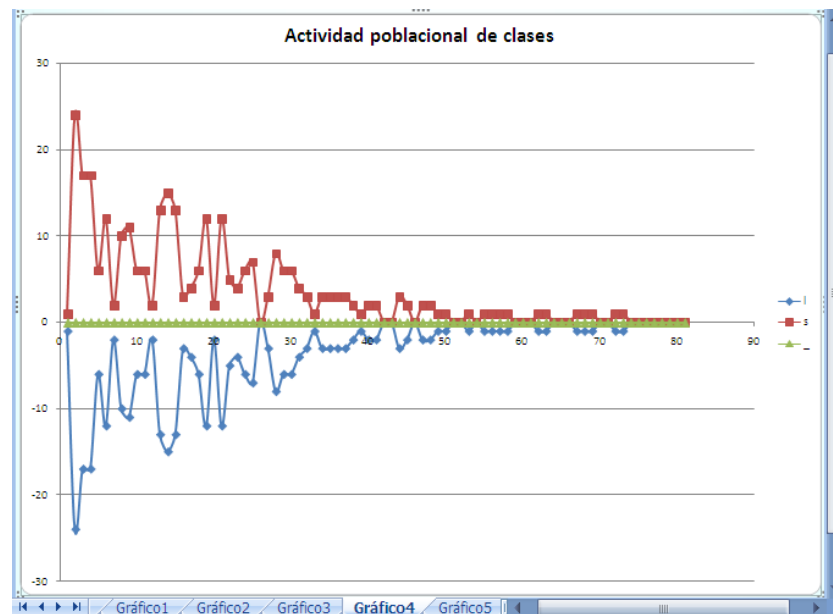
Fuente: Autor.

Figura E.27: Actividad de reglas de transición en OLE_EJ_cristales_pulso



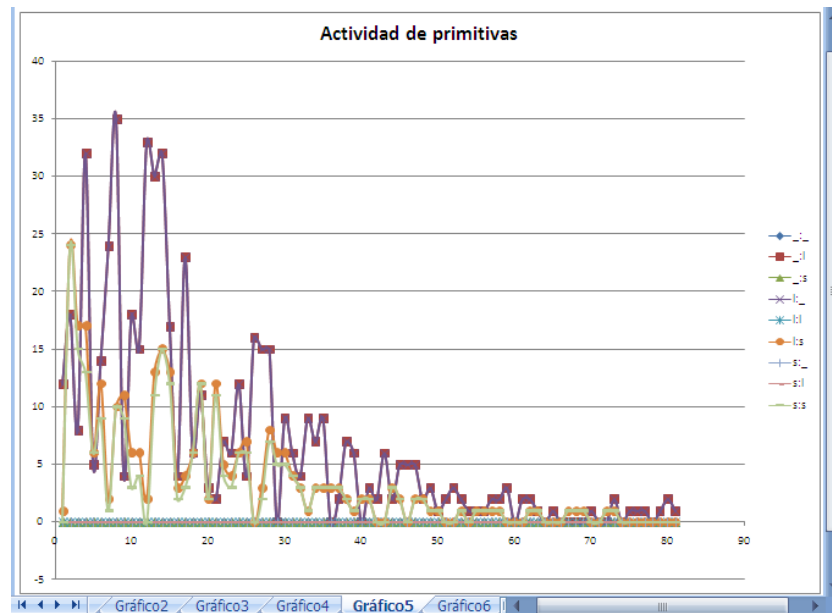
Fuente: Autor.

Figura E.28: Actividad de clases en OLE_EJ_cristales_pulso



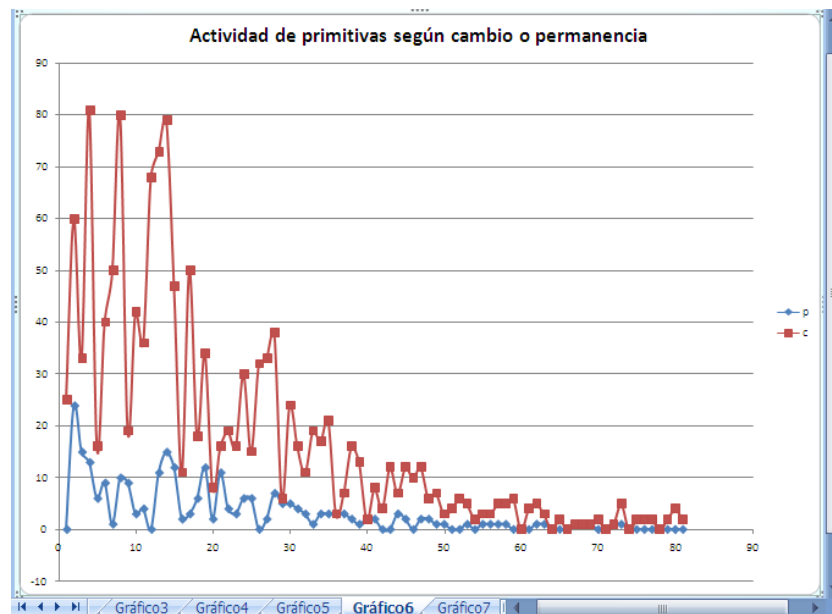
Fuente: Autor.

Figura E.29: Actividad de primitivas en OLE_EJ_cristales_pulso



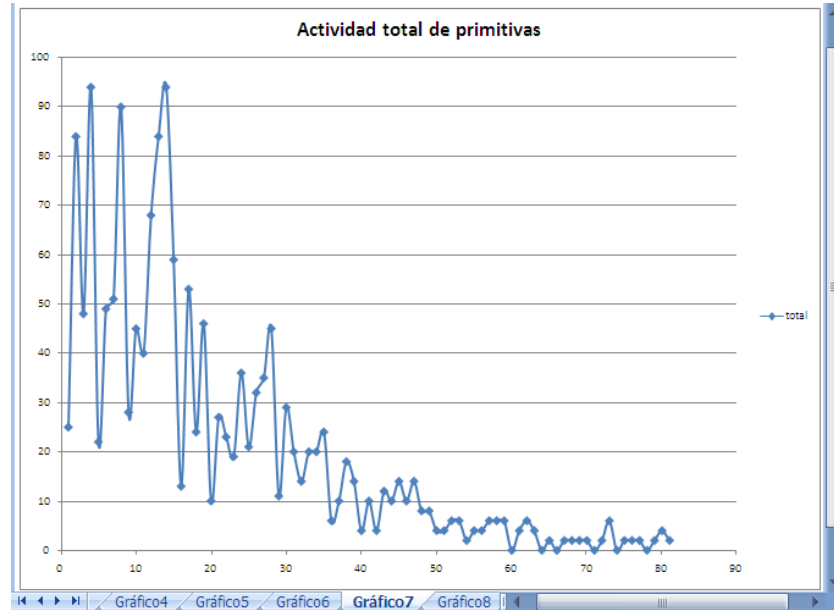
Fuente: Autor.

Figura E.30: Permanencia y cambio de primitivas en OLE_EJ_cristales_pulso



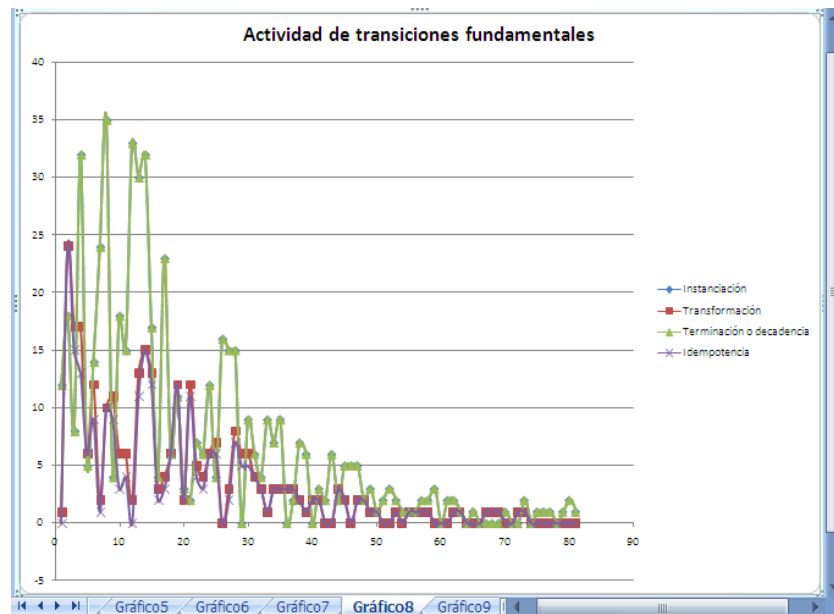
Fuente: Autor.

Figura E.31: Actividad total de primitivas en OLE_EJ_cristales_pulso



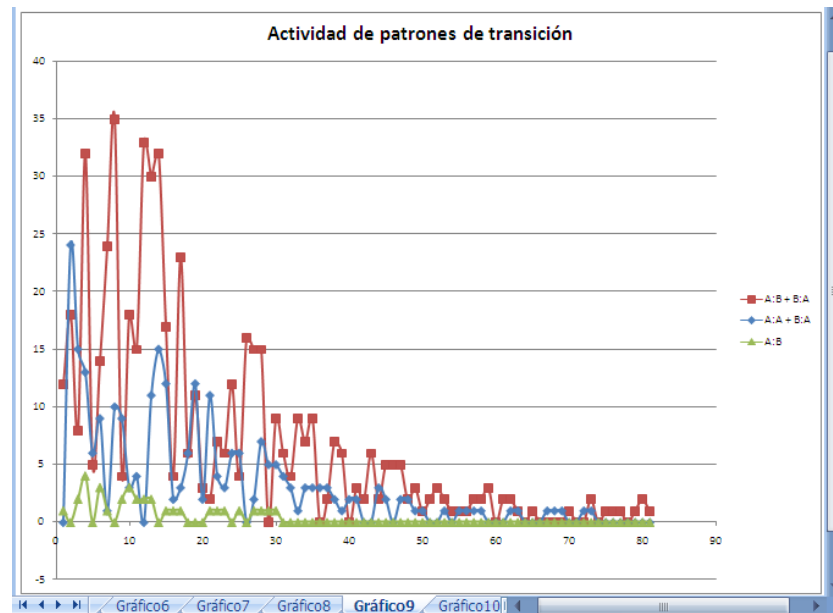
Fuente: Autor.

Figura E.32: Actividad de primitivas como transiciones fundamentales en OLE_EJ_cristales_pulso



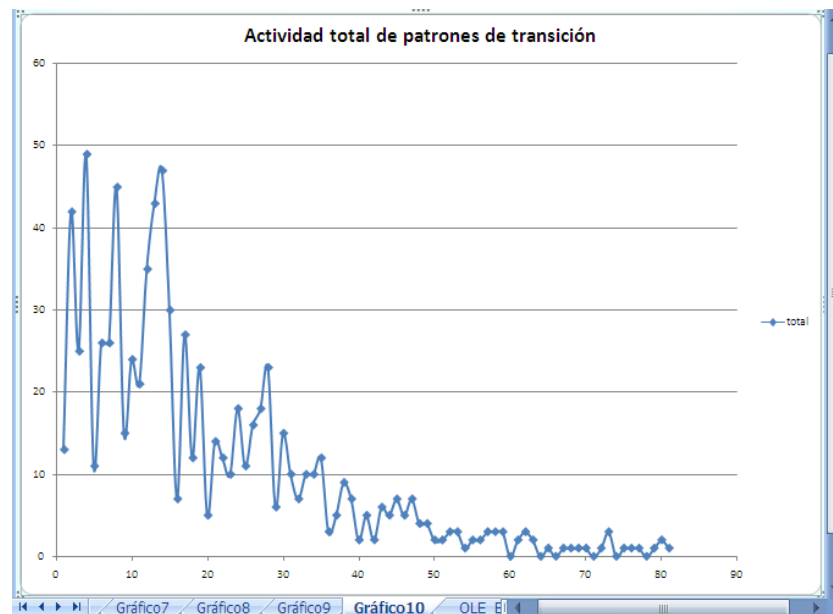
Fuente: Autor.

Figura E.33: Actividad de patrones de transición en OLE_EJ_cristales_pulso



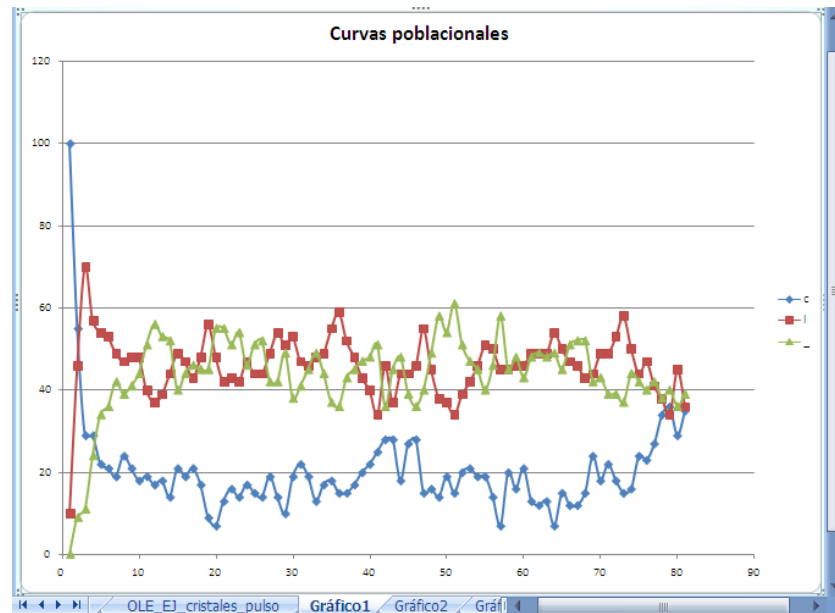
Fuente: Autor.

Figura E.34: Actividad total de patrones en OLE_EJ_cristales_pulso



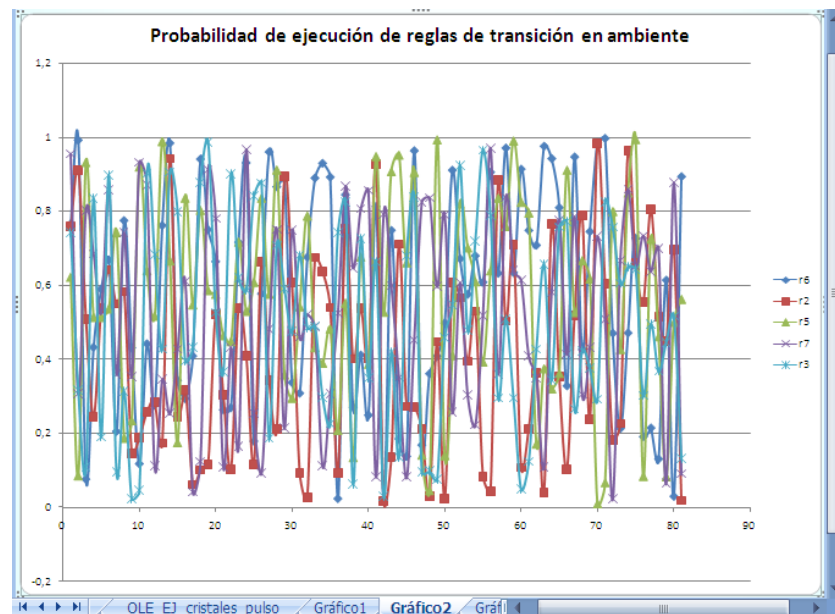
Fuente: Autor.

Figura E.35: Comportamiento poblacional de OLE_predador_presa_7rt



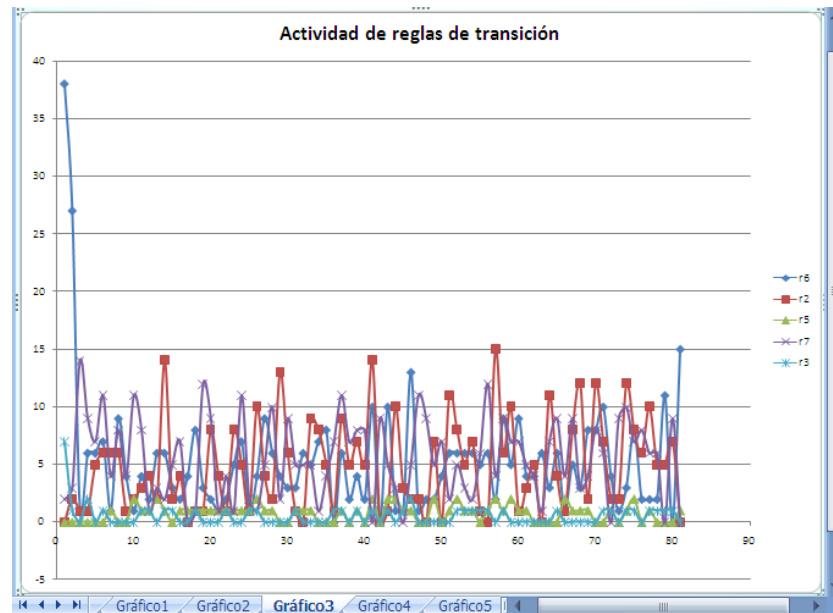
Fuente: Autor.

Figura E.36: Valores de aleatoriedad en OLE_predador_presa_7rt



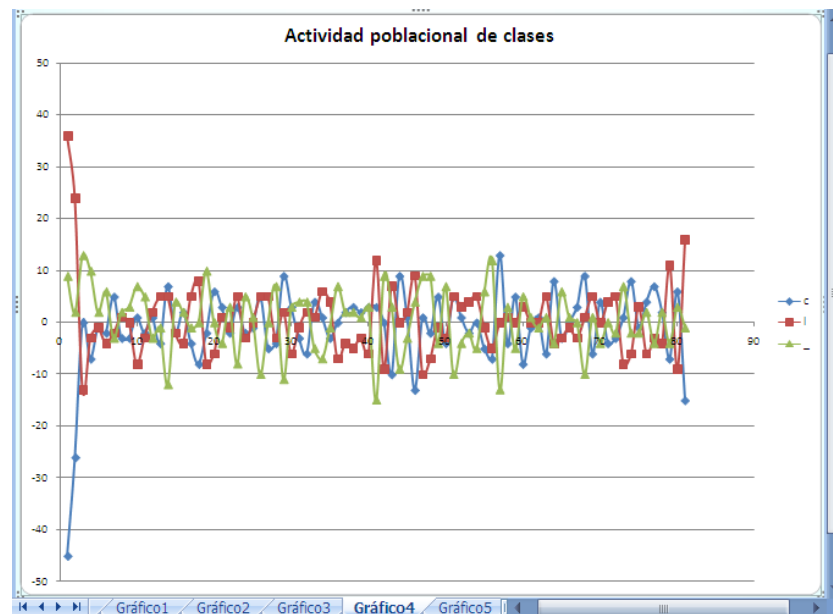
Fuente: Autor.

Figura E.37: Actividad de reglas de transición en 0LE_predador_presa_7rt



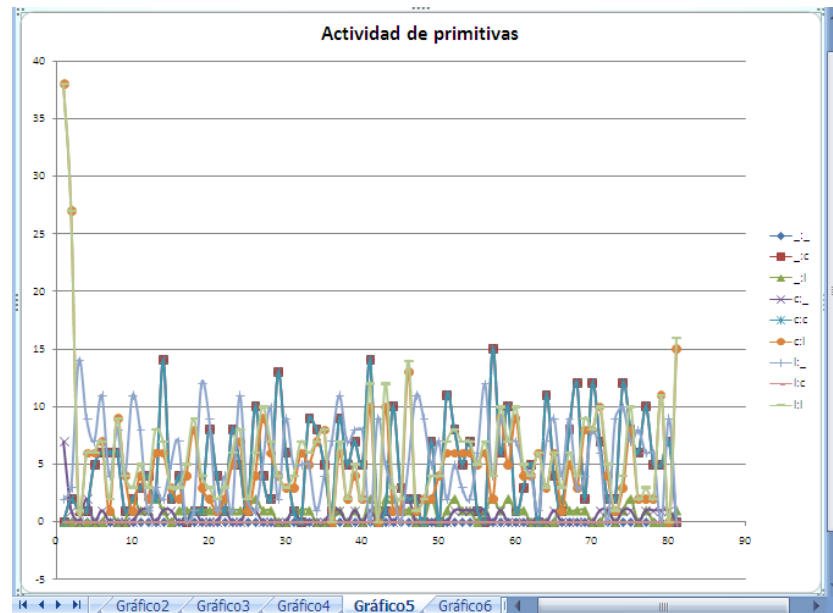
Fuente: Autor.

Figura E.38: Actividad de clases en 0LE_predador_presa_7rt



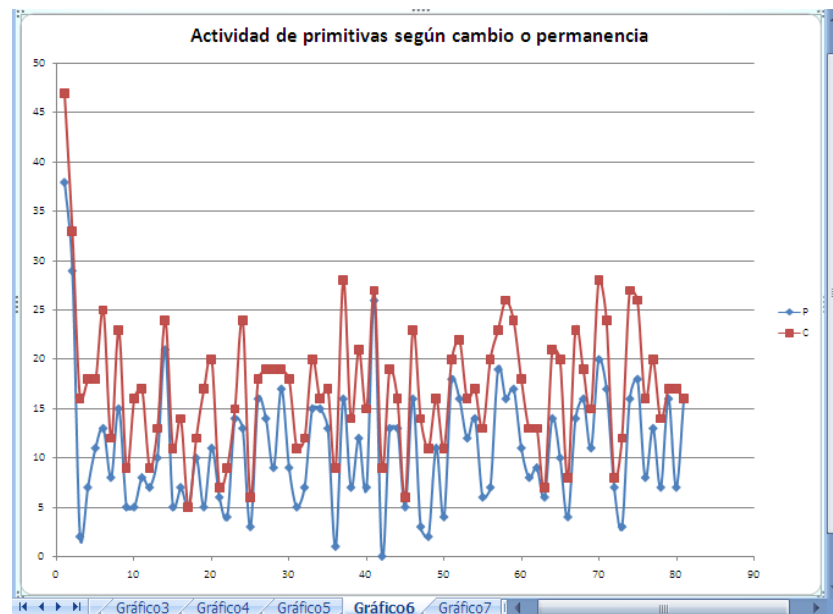
Fuente: Autor.

Figura E.39: Actividad de primitivas en OLE_predador_presa_7rt



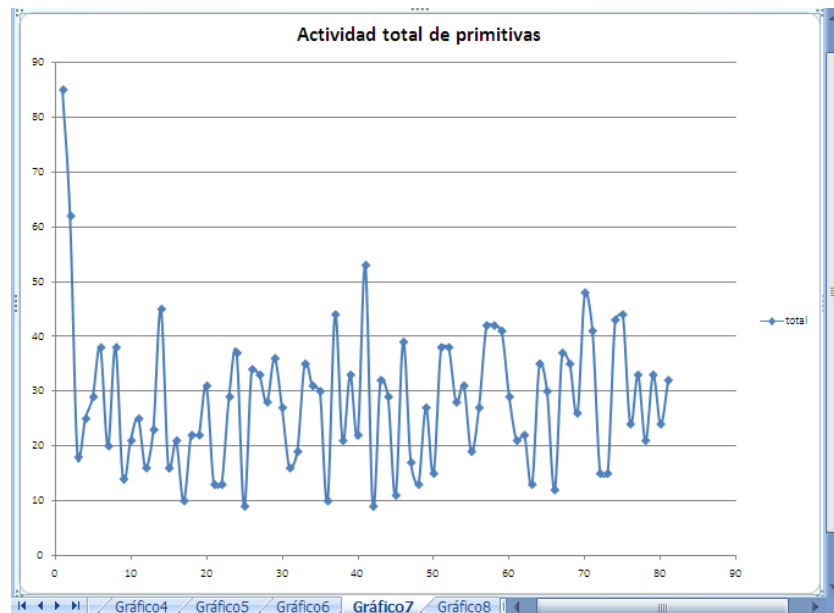
Fuente: Autor.

Figura E.40: Permanencia y cambio de primitivas en OLE_predador_presa_7rt



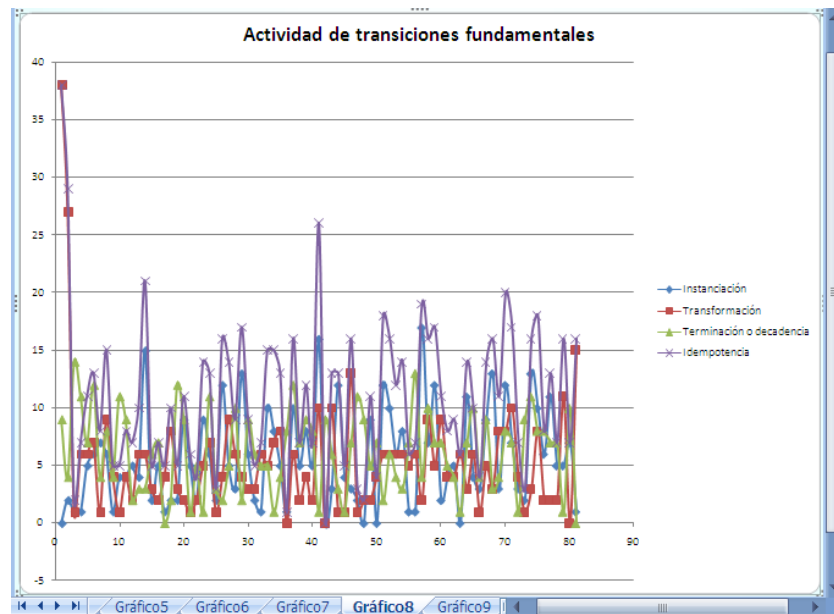
Fuente: Autor.

Figura E.41: Actividad total de primitivas en OLE_predador_presa_7rt



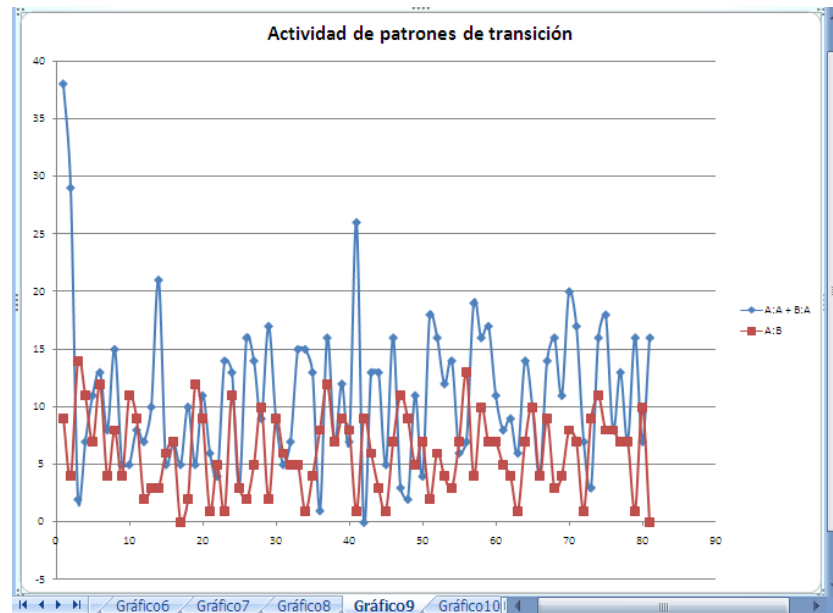
Fuente: Autor.

Figura E.42: Actividad de primitivas como transiciones fundamentales en OLE_predador_presa_7rt



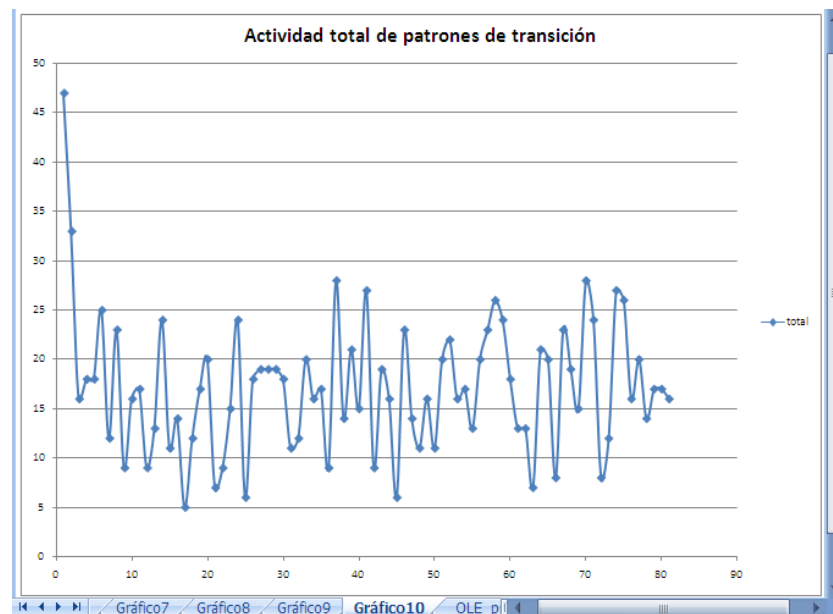
Fuente: Autor.

Figura E.43: Actividad de patrones de transición en OLE_predador_presa_7rt



Fuente: Autor.

Figura E.44: Actividad total de patrones en OLE_predador_presa_7rt



Fuente: Autor.

E.6.3.2. Implementación de caracterizaciones en hoja de cálculo

En forma general, el método de creación y uso de modelos BOR en Excel es el siguiente:

1. Caracterizar en diferencias el modelo BOR a representar.
2. Implementar en las celdas los valores de las diferencias netas encontradas para cada una de las reglas presentes en el modelo.
3. Establecer la función de caracterización de cada regla, se propone inicialmente usar la ecuación (E.18).
4. Organizar los elementos de cálculo y la generación de series de datos como se presentan en `mbor.xls`. La estructura propuesta facilita la extensión de las series.
5. Construir las distintas curvas a partir de los valores calculados, arrastrar y pegar las series de datos hasta el valor de paso de simulación deseado. Ver E.6.3.1 en la página 639.
6. Actualizar la hoja, dar [F9] para recalcular valores y observar los cambios producidos.

E.6.3.3. Notas de implementación

Se observa en las pruebas realizadas que hay similitud entre las curvas poblacionales producidas por la hoja de cálculo y las reportadas por Homos 1.0. Conceptualmente, la caracterización en diferencias sirve al propósito esperado de recreación de la dinámica de los modelos. Adicionalmente se afirma que:

- Los modelos implementados en Excel están en primeras etapas de desarrollo. Solo sirven al propósito de demostración de capacidades de la caracterización en diferencias de los modelos BOR.
- Se calcularon en cada modelo solo 81 pasos de simulación. Por la organización misma del contenido de los datos en las hojas, es posible aumentar la cantidad de pasos hasta el valor deseado y hacerlo simultáneamente para todos los tipos de curvas implementadas.
- Se encontró que es realmente molesto el constante recálculo de valores en la hoja de cálculo ante el más mínimo cambio de datos. Un paliativo a esto es la incorporación de macros que controlen este comportamiento. Esto último se consideró excesivo para el actual propósito de demostración y por eso no se implementó dentro de los ejemplos presentados. Se pospone esta actividad para futuros trabajos relacionados.
- Se esperan mejoras y cambios expresión interna del útil usado para la presentación de los ejemplos incluidos en este apartado, especialmente en la construcción de la función característica de cada regla de transición. Esta es una actividad a realizar en futuras iteraciones de investigación.

- Los tipos de curvas introducidos en este apartado son prueba suficiente de funcionalidad de para los propósitos expositivos de este proyecto; la lista mostrada no es exhaustiva. Se presume que la generación de otros tipos de curvas son posibles mediante el mecanismo de caracterización.
- Dada la naturaleza de implementación en la hoja de cálculo, cabe preguntarse si posible generalizar lo conceptualizado en una forma intermedia de construcción de modelos BOR que obvie completamente la necesidad de los autómatas celulares. Responder a esta pregunta requiere más exploración y trabajo sobre el tema, fundamenta una nueva línea de investigación sobre la MBOR.

El documento que contiene los ejemplos de caracterización en diferencias se encuentra en `contenido/anexos/scripts/Excel/mbor.xls`

E.7. Diagramas de flujo y guías de código fuente

Con el propósito de ilustrar gráficamente algunos de los algoritmos presentes en el código fuente Pascal/Delphi analizado y producido durante este proyecto, se estableció una versión particular de diagramas de flujo. La representación gráfica de los diagramas de flujo implementados tienen las siguientes características:

- Hace uso de un conjunto mínimo de recursos gráficos: líneas como conectores, rectángulos como bloques de código, rombos como elementos de decisión y elipses como marcadores de inicio y fin de diagrama. La selección concuerda con los elementos estándar³³ de los diagramas de flujo.
- Presenta de arriba a abajo el orden de ejecución de los distintos bloques que componen el código, resalta el flujo principal y los flujos dependientes existentes en el código fuente.
- Presenta de izquierda a derecha la profundidad de anidamiento de bloques de código, notada en la figura E.45 como niveles: N1, N2...
- Resalta el punto de retorno de ciclos y los distintos bloques de control disponibles:
 - Los bloques if/si y demás condicionales se implementan de esta forma: el conjunto de bloques a la derecha del símbolo de decisión o rombo es el resultado de la condición verdadera. El siguiente conjunto de bloques configura la condición falsa. Se da un ejemplo de la estructura en la figura E.46b en la página 656.
 - Se usan las líneas de segmentos largos para indicar el flujo normal del código y las de segmentos cortos para indicar el retorno de código desde un estructura de control de ciclos. Los bloques for/while están implementados de forma convencional. Ver figura E.45 en la página siguiente.
- Define una asociación entre colores y función de cada bloque. La figura E.45 en la página siguiente establece la interpretación de los colores de las distintas figuras usadas en el diagrama. Se usan colores para establecer contraste entre los elementos funcionales componentes.

La forma del diagrama facilita la comprensión de la complejidad algorítmica del código. El diagrama se hace más extenso según la cantidad de operaciones presentes, se hace más profundo según el anidamiento de las estructuras de control implementadas. La asociación organizada entre diagramas y bloques funcionales en cada unidad disponible conforma una guía de código. Las guías sirven al propósito de análisis ya que permiten observar la complejidad global del software. Cada guía contiene un índice general de unidades y cada unidad contiene los diagramas del código fuente asociado. Las figuras de los diagramas están implementadas en la sección de definición de cada *.svg, se nombra a los diagramas con el siguiente formato: [nombre bloque]_[orden en unidad].svg.

En la figura E.46a y E.46b en la página 656 se muestra un ejemplo de las guías de código que acompañan este proyecto, en la sección F en la página 693 se listan todas las guías de código disponibles.

³³Se implementan algunas de las formas geométricas en diagramas de flujo según las normas FIPS 24, ANSI X3.5 and ISO 1028. Una versión comercial en: http://www.timelytemplates.com/f_computer/Tcomputer.htm

Figura E.45: Características de los diagramas de flujo implementados en este proyecto

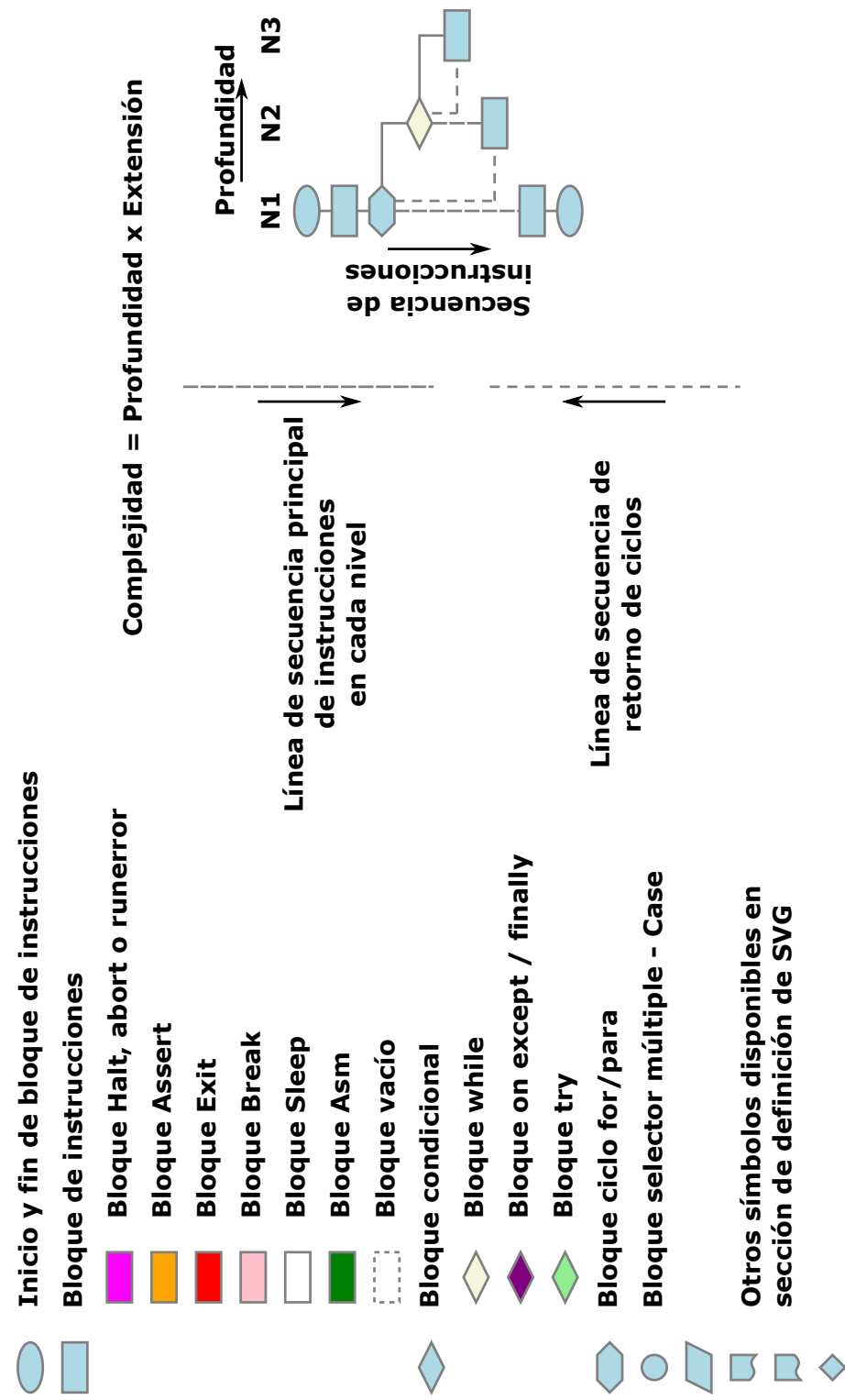
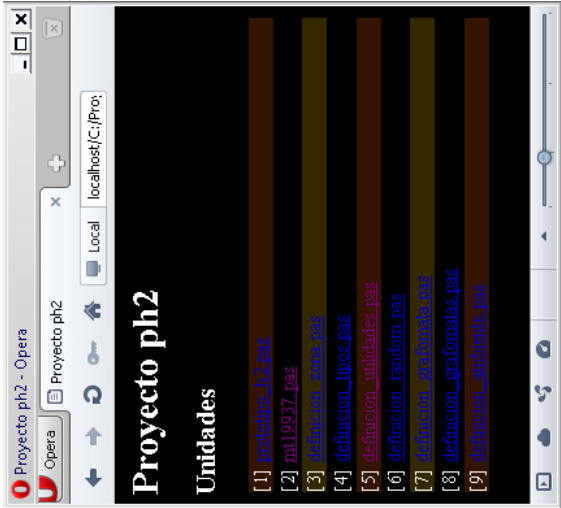
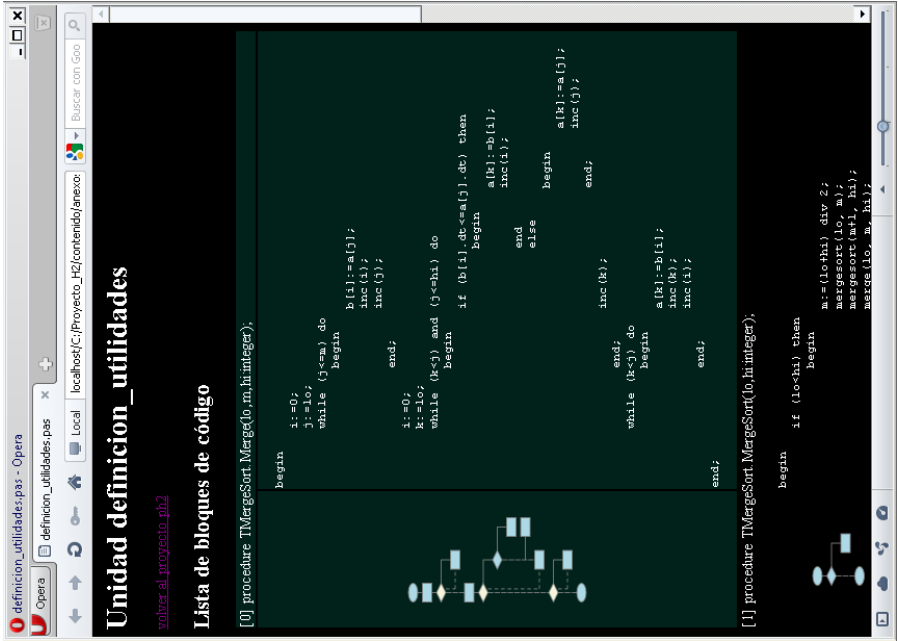


Figura E.46: Ejemplo de Guia de código

(a) Ejemplo de índice guía de código fuente implementado en este proyecto



(b) Ejemplo de diagrama de flujo y código fuente asociado



Índice de guía de código fuente del proyecto PH2 .exe.

Diseño de contenido de guía: Letras blancas, fondo negro con bloques de colores cíclicos verde, negro, azul, negro. Se usa esta combinación de colores por su contraste.

Fuente: Autor:

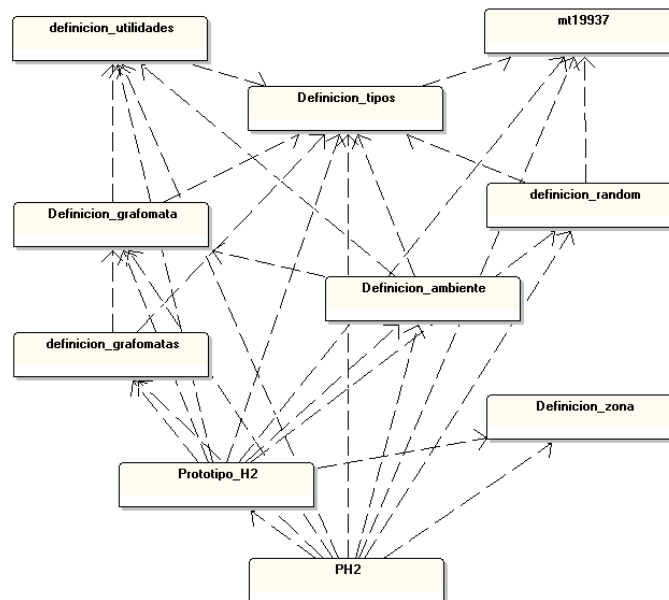
E.8. Primer prototipo para Homos 2.0

El prototipo software Homos 2.0 [PH2] presentado en este proyecto es uno de los primeros resultados obtenidos del proceso de construcción iterada con el cual se desarrolla la investigación acerca de las características y facilidades que deben ser parte de una aplicación que de soporte a la siguiente versión de la MBOR. Su propósito es permitir la exploración de ideas y facilitar la selección de elementos que puedan incorporarse en futuras herramientas software que implementen la MBOR.

PH2 es una prueba práctica de la viabilidad de los cambios propuestos por la investigación realizada, se espera que las funciones aquí planteadas hagan parte de los útiles incorporados en las siguientes versiones de la herramienta software.

E.8.1. Productos de investigación integrados en el prototipo software PH2

Figura E.47: Productos de investigación y primer prototipo de Homos 2.0



Las flechas indican las dependencias inmediatas entre módulos.

Fuente: Autor.

E.8.2. Desarrollo del prototipo

El lenguaje de programación usado para la construcción del prototipo fue Delphi 7.0. El desarrollo de la aplicación se realizó mediante prototipado evolutivo. Este proceso se inició apartir de varios núcleos temáticos (Grafómatas, Zonas, Interfaz gráfica y Simulador) determinados durante la etapa

de análisis, que luego fueron integrándose en el orden en que aquí se listan hasta conformar la unidad del conjunto de funciones implementados en PH2.

A continuación se presenta cada núcleo temático, indicando las principales necesidades encontradas:

1. Desarrollo e implementación práctica del concepto de grafómata:³⁴

- Generación de estructuras de representación interna de grafómatas. Se dio una primera aproximación de representación mediante la implementación de grafómatas contruidos con distancias enteras entre vértices.
- Desarrollo de funciones que permitan realizar las operaciones requeridas para el manejo y construcción de grafómatas.
- Encapsulación, construcción de un objeto que centraliza las funciones requeridas para la manipulación en código de los grafómatas.
- Determinación de un medio de almacenamiento para las estructuras creadas, una forma de organización para su posterior utilización.

2. Desarrollo del concepto de zonas:

- Determinación del algoritmo a usar para implementar el concepto de zonas.
- Generación de un objeto que encapsula las funciones requeridas para su construcción y manejo.
- Prueba de concepto de la utilidad de zonas en ambiente.

3. Desarrollo de la interfaz gráfica:

- Visualización de los grafómatas, mediante la construcción en el ambiente de la superficie de teselaciones que cada uno representa.
- Implementación de opciones gráficas tales como rotación y vista panorámica del ambiente.
- Implementación de distintas formas de presentación de los objetos en el ambiente.

4. Desarrollo del simulador:

- Implementación de un algoritmo de simulación que haga uso de los grafómatas como estructura que determine las capacidades de movimiento de los objetos en el ambiente.
- Desarrollo de controles de simulación mediante la implementación del control básico y avanzado de simulación.
- Actualización síncrona entre el estado del simulador y el ambiente.
- Incorporación de funcionalidad de conteo de las zonas en el simulador.

Luego de alcanzado el punto de integración entre los distintos núcleos temáticos, se da por terminada la evolución del prototipo. PH2 es ahora la primera iteración en el ciclo de desarrollo de la versión completa de la herramienta software Homos 2.0 y posteriores.

³⁴Ver anexo C en la página 309

E.8.3. Alcances y limitaciones del prototipo

PH2 implementa una primitiva interfaz gráfica de ambiente en la que se muestran:

- Distintas opciones de configuración de división del espacio.
- Varias formas de representación gráfica de objetos en ambiente.
- Un menú simple sin iconos que implementa las distintas opciones disponibles en el prototipo.

Las características generales del ambiente son:

- No tiene límites³⁵, los objetos pueden estar ubicados mas allá del espacio representable por la ventana gráfica y aún pertenecer al espacio de simulación.
- Puede representar hasta 37 tipos³⁶ distintos de geometrías / teselaciones de 41 formas disponibles³⁷ con las cuales caracterizar el espacio.
- No hay límites en la cantidad de objetos que pueden coexistir en una misma celda del espacio

Las características gráficas iniciales del ambiente son:

- Color blanco para el fondo del ambiente, colores azul y gris para representar las líneas que dibujan a las teselaciones base y dual que configuran los vecindarios y las áreas de influencia de los objetos en el espacio.
- La geometría base por defecto es la malla cuadrada que implementa un vecindario de Moore de 9 celdas. Una imagen de esta configuración inicial se presenta en la figura E.50 en la página 669.
- El centro del ambiente se indica mediante un círculo relleno del color de la teselación base.
- Es posible rotar y mover lateralmente el ambiente. Ver la figura E.56 en la página 672.

Las características de representación de los objetos en el ambiente son:

- Los objetos están centrados sobre los vértices presentes en la malla base o dual elegida como sistema de interacción en el menú ambiente.
- En esta etapa de desarrollo del prototipo no hay un límite fijo al número de objetos a manejar en el ambiente.
- Se incluyen seis formas distintas de visualización de objetos. Ver las figuras E.56 a E.67.

³⁵... Inmediatos. La ubicación de coordenadas en el espacio va hasta el alcance máximo y mínimo del tipo de datos Integer o Longint comprendido entre -2147483648..2147483647. Este es un tipo de datos con signo de 32 bits.

³⁶La malla cuadrada es su propio dual, la malla triangular y hexagonal son mutuamente duales, las teselaciones regulares y no regulares presentadas tienen duales distintos, las mallas de variaciones muestran alteraciones en la representación gráfica pero repiten la esencia de las estructuras base y dual de las cuales han sido tomadas. Los vecindarios de Moore tienen el mismo tipo de dual, la malla cuadrada. Ver la tabla

³⁷Al número de tipos se suma la cantidad de variaciones implementadas.

- Las imágenes de los objetos usados en este prototipo son archivos *.bmp de 16×16 bits, semejantes a los usados en Homos 1.0.

Las características del simulador son:

- El proceso de simulación implementado en el prototipo se restringe inicialmente el uso aplicado de los grafómatas como estructuras de movimiento de los objetos en el espacio.
- El modelo simulado en el prototipo es el movimiento aleatorio de los objetos según una de las configuraciones geométricas planteadas en PH2 para representar el espacio.
- Al iniciar la simulación todos los objetos en el prototipo se ubican inicialmente en el origen del ambiente.
- Todos los objetos instanciados tienen 100 % de probabilidad de ejecución de movimiento.

E.8.4. Descripción de elementos implementados en la interfaz gráfica de PH2

A continuación se hará una descripción de cada uno de los elementos funcionales presentes en el menú principal del prototipo. Algunas de las opciones listadas tienen atajos de teclado asociados, los cuales se mostrarán inmediatamente después del identificador de elemento. La presentación de los componentes del menú se organiza por bloques funcionales.

El menú del prototipo Homos 2.0 implementa las siguientes opciones:

- Menú principal \ Ambiente \ Geometría \ Grafómatas prediseñados: Se incorporaron en la herramienta varias formas de división del espacio, cada una está construida sobre un grafómata diferente y específico. La elección de un grafómata particular implica el uso de su grafómata dual y sobre cualquiera de las teselaciones presentadas, base o dual, es posible ubicar objetos en el ambiente. El prototipo actual implementa 27 grafómatas y sus duales, la selección de uno de los títulos presente en cada entrada de este submenú identifica el grafómata a usar en el ambiente.

Las teselaciones regulares usadas como formas de división del espacio implementadas en este prototipo son:

- Cuadrada: Es dual de si misma. Corresponde al vecindario de Von Newman. Ver figura C.64 en la página 423.
- Triangular y Hexagonal: Estas mallas son mutuamente duales. Ver figuras C.63 y C.65 en la página 423.
- 3.3.3.4.4: Arreglo de triángulos y cuadrados. Ver figura C.66.
- El cairo | 3.3.4.3.4: Sistema de pentágonos cuyo dual es un arreglo de cuadrados y triángulos. Ver figura C.75 en la página 427.
- 4.8.8: Arreglo de cuadrados y octágonos. Ver figura C.72 en la página 426.
- Estrella de David o 3.6.3.6: Arreglo de hexágonos y triángulos cuyo dual configura una malla de paralelogramos. Ver figura C.67 en la página 424.
- 3.3.3.3.6: Arreglo de triángulos que rodean por completo a hexágonos, cuyo dual es un arreglo de pentágonos que arman *flores* de 6 pétalos. Ver figura C.69 en la página 425.
- 3.4.6.4: Arreglo de cuadrados, hexágonos y triángulos cuyo dual es un arreglo de cuadriláteros con forma de punta que ensamblan a su vez hexágonos y triángulos. Ver figura C.70 en la página 425.
- 4.6.12: Arreglo de cuadrados, hexágonos y dodecágonos cuyo dual presenta un arreglo de triángulos escalenos. Ver figura C.71 en la página 425.
- 3.12.12: Arreglo de triángulos y dodecágonos, su dual es un arreglo de triángulos isósceles. Ver figura C.68 en la página 424.

Los elementos del menú denominados como variantes son realizaciones de alteraciones en la dirección de algunas aristas de los grafómatas que los originan. Reflejan lo que pueden producir pequeños cambios en la dirección de una arista en la estructura de un grafómata. Se adicionan:

- Variante de triángulos: Desarrollo similar a 3.3.3.4.4 pero no hay cuadrados, solo triángulos. Ver figura C.78 en la página 428.
- Variante 4.8.8: Arreglo de estrellas de cuatro puntas y cuadrados. Ver figura C.79 en la página 428.
- Variante 4.6.12: Arreglo de estrellas de 6 puntas y triángulos equiláteros. Ver figura C.80 en la página 428.
- Variante el Cairo: Arreglo de pentágonos de base cuadrada. Ver figura C.76 en la página 427.

Se incluyen además 5 ejemplos de grafómatas construidos con polígonos que no tienen lados iguales y que no hacen parte del conjunto de estructuras presentes en las teselaciones regulares.

- Pentágonos y octágonos: Su dual es una malla de triángulos. Ver figura C.83 en la página 429
- Pentágonos y rombos: Su dual configura un arreglo de cuadrados y triángulos. Ver figura C.82 en la página 429.
- Cuadrados y hexágonos: Configura una teselación conocida como ojo de buey. Su dual presenta una malla cuadrada que contienen triángulos isósceles rellenando algunos cuadrados. Ver figura C.73 en la página 426.
- Waffles: Similar a cuadrados y hexágonos, su dual configura un arreglo de octágonos y triángulos isósceles. Ver figura C.74 en la página 426.
- Molinetes: Similar a la teselación el cairo, su dual genera una malla de octágonos, cuadrados y triángulos isósceles. Ver figura C.77 en la página 427.

Se incorporan 6 variaciones del vecindario de Moore, una geometría en la que se resalta el tamaño del vecindario y la relación entre los puntos el espacio:

- Vecindarios de Moore de 9, 25, 49, 81, 121 y 169 celdas: El dual de cada sistema es una malla cuadrada. La diferencia entre sistemas consiste en el alcance de acción - avance - influencia del objeto sobre el ambiente, los números indican la cantidad de celdas requeridas por la estructura. Al restar uno al número de celdas se obtiene la cantidad de vecinos a los que tiene acceso el objeto central dentro de su vecindad inmediata.
- Menú principal \ Ambiente \ Geometría \ Sistema de interacción: Permite elegir sobre cual teselación o estructura de división del espacio se van a mover los los objetos en el ambiente.
- Grafómata base: Elige una definición particular de grafómata implementado, aquella identificada por el nombre que aparece en el listado de teselaciones disponibles. Cada grafómata constituye el patrón de construcción de las formas de teselación presentadas en el software.
 - Grafómata dual: Elige el dual del grafómata como la estructura de movimiento a usar por los objetos en el ambiente. El dual se construye internamente según el grafómata base elegido en el submenú Grafómatas prediseñados.

- Menú principal \ Ambiente \ Grafómata \ Visualizar: Establece las opciones de visualización de las mallas de división del espacio en el ambiente. La teselación elegida como base se presenta en azul y la teselación dual en gris. El color de la teselación indica cual fue la selección realizada en el submenú sistema de interacción.
 - Espacio libre: No se dibujan teselaciones en el ambiente. Ver figura E.55 en la página 671.
 - Grafómata Base: Dibuja en el ambiente la teselación base. Ver figura E.52 en la página 670.
 - Grafómata dual: Dibuja en el ambiente la teselación dual. Con esta opción activada se hace posible rellenar el área de influencia con el color de clase a la que pertenece el objeto. Ver figuras E.53 en la página 670 y E.65 en la página 676.
 - Visualizar sistemas base y dual: Presentación simultánea de los sistemas base y dual. Ver figura E.54 en la página 671.
 - Norte del ambiente: [Des]activa en pantalla una flecha de color rojo que parte del origen del ambiente y tiene al extremo una letra N que permite determinar la orientación del ambiente. Es de utilidad cuando se realiza la rotación del ambiente. Ver figura E.55 en la página 671.
- Menú principal \ Ambiente \ Objetos \ Representar...: Permite establecer seis opciones de visualización para los objetos en el ambiente:
 - En uniformidad, como círculos rellenos de rojo. Ver figura E.62 en la página 675.
 - En diferencias mediante círculos rellenos del color de clase a la que pertenece el objeto. Ver figura E.63 en la página 675.
 - En diferencias presentando el identificador numérico de cada objeto. Ver figura E.64 en la página 676.
 - Presentando el área de influencia del objeto rellena del color de clase del objeto. Requiere que se visualice en el ambiente solo la teselación dual. Ver Figura E.65 en la página 676.
 - Presentando la imagen de la clase a la que pertenece el objeto, igual que en Homos 1.0. Ver figura E.66 en la página 677.
 - Visualiza los objetos como hormigas, Ver figura E.67 en la página 677.
- Menú principal \ Ambiente \ Objetos \ Cantidad máxima: Permite instanciar en el ambiente 100, 1.000, 10.000, 100.000 y hasta 1.000.000 de objetos simultáneos. Todos los objetos se mueven sobre los vértices que estructuran a la teselación base. Ver figura E.61 en la página 674.
- Menú principal \ Ambiente \ Objetos \ Opciones de visualización \ ver última ubicación: Presenta un segmento de línea rojo que indica gráficamente en el ambiente la dirección de movimiento del objeto al que esta asociado. El destino es la ubicación donde está dibujado el objeto. Ver figura E.60 en la página 674.
- Menú principal \ Ambiente \ Zonas \ Prediseñadas: Introduce el concepto de zonas en el ambiente de simulación. Delimita áreas específicas dentro del ambiente con el propósito de

realizar operaciones tales como realizar conteos, filtrar, contener o limitar el movimiento de ciertos objetos etc..

- Espacio libre: No existen zonas definidas en el ambiente. Ver figura E.55 en la página 671.
- Zonas 1 a 5: Se delimita un área con una forma establecida, para realizar con ella alguna actividad relacionada con los objetos presentes en la simulación. Con cada una de las zonas disponibles el simulador realiza el conteo de objetos contenidos. Se visualiza el resultado del conteo en una de las esquinas del borde que describe la zona.
Las forma de las zonas en orden de aparición el en menú es: Cuadrado centrado en el origen, Figura en forma de Y, Reloj de arena, Anillo cuadrado y zona compuesta de áreas cuadradas separadas. El reloj y el anillo se muestran en las figuras E.58 y E.59 en la página 673.

■ Menú principal \ Ambiente \ Rotación:

- 15° Izquierda: [Ctrl+Left] Decrementa el ángulo de rotación del ambiente en 15 grados.
- 15° Derecha: [Ctrl+Right] Incrementa el ángulo de rotación del ambiente en 15 grados.
- Sin rotación: [Ctrl+3] Hace que el ángulo de rotación del ambiente sea igual a cero.

Se muestra una rotación del ambiente de 45° en las figuras E.52 a E.59.

■ Menú principal \ Ambiente \ Panorámica:

- Ampliar: [Ctrl+Down] Aumenta 1x de panorámica, permite ver más espacio en el ambiente. Nótese que al ir aumentando la panorámica, la representación gráfica de la estructura geométrica con la que se divide el espacio se hace tan densa que finalmente se pierde, el fondo se hace uniforme.
- Reducir: [Ctrl+Up] Reduce 1x la panorámica.
- Valor por defecto: [Ctrl+2] Establece la panorámica en 6x por defecto.

Cuando está disponible, la rueda del ratón permite ampliar o reducir la panorámica del ambiente. Las Figuras E.56 E.57 en la página 672 muestran ejemplos cambio de panorámica. En los casos presentados no se hace visible la teselación con la que se organiza el espacio.

- Menú principal \ Ambiente \ Centrar ambiente: [Ctrl+C] Reubica el ambiente al origen de coordenadas.
- Menú principal \ Simulación: Aquí se implementan los distintos controles del proceso de ejecución del modelo en curso.
 - Iniciar: [F9] Comienza el proceso de simulación del modelo.
 - Detener/Continuar: [Space] Interrumpe o reinicia el proceso de simulación
 - Finalizar: [Ctrl+F9] Detiene el simulador y termina el ciclo de simulación actual.

- Menú principal \ Simulación \ Espera entre pasos: Este submenú permite configurar el tiempo de pausa entre pasos de simulación.
 - Pausa máxima: [Alt+F5] Establece el tiempo máximo de espera entre pasos de simulación.
 - Incrementar: [Alt+F6] Aumenta el tiempo de espera entre pasos de simulación.
 - Reducir: [Alt+F7] Disminuye el tiempo de espera entre pasos de simulación.
 - Sin pausa: [Alt+F8] Hace que la realización de cada paso de simulación sea inmediato, no hay tiempo de espera entre pasos.
 - Avance manual: [Ctrl+Enter] Permite [des]activar el control manual del proceso de simulación. Se calcula el siguiente paso de simulación con cada pulsación de la barra espaciadora.

Durante la simulación se deshabilitan los menús Selección de vecindario y Sistema de interacción, no se tienen en cuenta los cambios realizados a la cantidad de objetos a simular. Se permite cambiar la función fuente de los valores de aleatoriedad.

- Menú principal \ Simulación \ Proceso: Implementa en el prototipo la acción de ejecución en reversa o vuelta atrás de los pasos de una simulación previamente calculada. Internamente consiste en el registro y almacenamiento de las condiciones de aleatoriedad y el estado del ambiente en ciertos pasos o momentos de simulación. Los momentos almacenados se visualizan en el ambiente según los controles de manejo histórico implementados. Cada avance o retroceso presenta solo los momentos de simulación registrados, no los pasos intermedios. Las opciones de control del registro histórico de simulación son:
 - Activar registro de momentos: [Shift+Ctrl+Alt+R] Permite el proceso grabación de momentos. Puede ser [des]activado solo si no se ha iniciado el proceso de simulación. El almacenamiento de momentos se hace efectivo durante la ejecución de la simulación posterior al registro histórico.
 - Grabar ahora!: [Shift+Ctrl+Alt+G] Permite determinar a partir de que momento de simulación iniciar el registro de momentos. En esta versión se almacenan hasta 100 momentos de simulación, con una separación entre momentos de 5 pasos. El delta de pasos y la cantidad de momentos a registrar están fijos en este prototipo.
 - Momento inicial: [Ctrl+F5] Presenta en el ambiente el momento inicial de simulación, el paso desde el cual se inicia el registro de la actividad de simulación.
 - Retroceder: [Ctrl+F6] Presenta la ejecución del modelo en dirección al inicio del registro de momentos de simulación actual.
 - Avanzar: [Ctrl+F7] Presenta la ejecución del modelo en dirección al final del registro de momentos de simulación actual.
 - Momento final: [Ctrl+F8] Presenta en el ambiente el momento final de simulación calculado en la ejecución actual del modelo.

- Devolucionar: [Ctrl+Alt+F5] Esta opción inicia el proceso de presentación en sentido inverso del orden de la simulación realizada. A partir del último momento de simulación almacenado se presentan uno a uno los momentos almacenados hasta llegar al paso de simulación desde el cual se dio comienzo al registro histórico. En el prototipo actual, devolucionar es posible solo si existen momentos históricos recientemente calculados. Desactivar el registro de momentos, anula los momentos históricos vigentes de la simulación actual.
- Detener: [Ctrl+Espacio] Hacer una pausa en el proceso de [De]evolución.
- Evolucionar: [Ctrl+Alt+F8] Esta opción presenta los momentos de simulación almacenados en el orden de ejecución normal del modelo.

En general, las opciones de pausa alteran la velocidad de ejecución de la simulación y el funcionamiento de las opciones evolucionar y devolucionar del submenú Proceso. Las opciones de pausa hacen posible observar con detenimiento los detalles del proceso de simulación. Ver Figura E.51 en la página 669.

- Menú principal \ Opciones \ Aleatoriedad \ Origen de Secuencias: Permite elegir la fuente de los valores aleatorios a usar en la simulación:
 - LCG³⁸ Random: Usa la rutina de valores aleatorios provista por Delphi 7.0.
 - Mersenne Twister: Se usa una implementación en Delphi del generador de números aleatorios Mersenne Twister creado por Makoto Matsumoto y Takuji Nishimura ³⁹[MN98]. Es la opción de generación de valores aleatorios establecida por defecto en el prototipo. Este generador es ideal para simulaciones de tipo Monte Carlo.
- Menú principal \ Opciones \ Preferencias.: Muestra un cuadro de diálogo desde el cual es posible cambiar el color del fondo del ambiente. Por defecto el fondo es blanco.
- Menú principal \ Archivo \ Salir: [Ctrl+X] Cierre de la aplicación.

Los demás elementos incluidos en el menú y no referidos en la descripción anterior se presentan solo como guía de opciones | funciones a implementar en una futura versión de la herramienta software, se presenta un contexto que justifica su inclusión en la sección E.8.5 en la página 678. La estructura completa del menú de la aplicación se muestra en las figuras E.48 y E.49 en la página 668. En los menús del prototipo se marcan con [•] las opciones establecidas por defecto.

³⁸Linear Congruential Generator

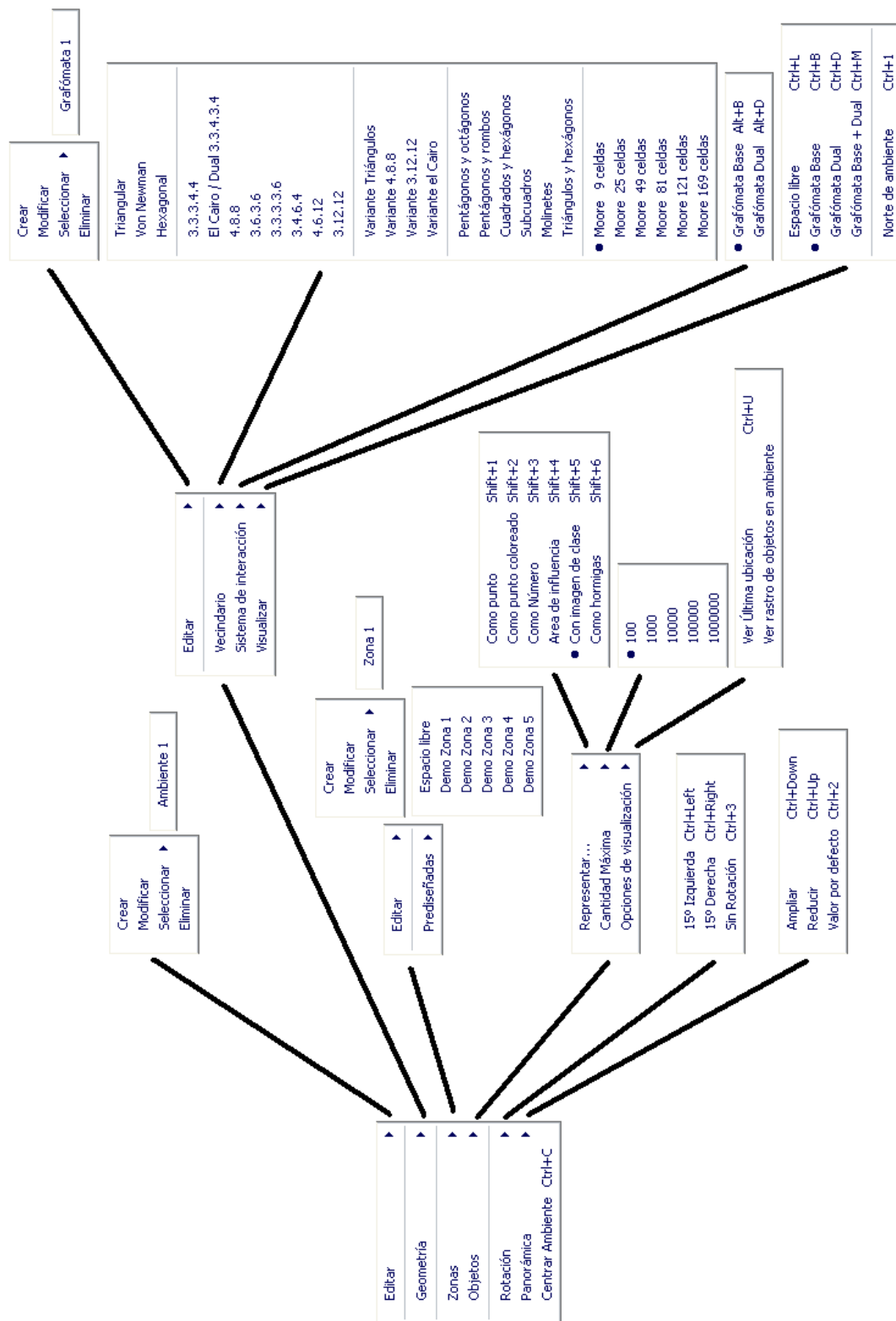
³⁹<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>

Figura E.48: Estructura del menú principal del prototipo Homos 2.0



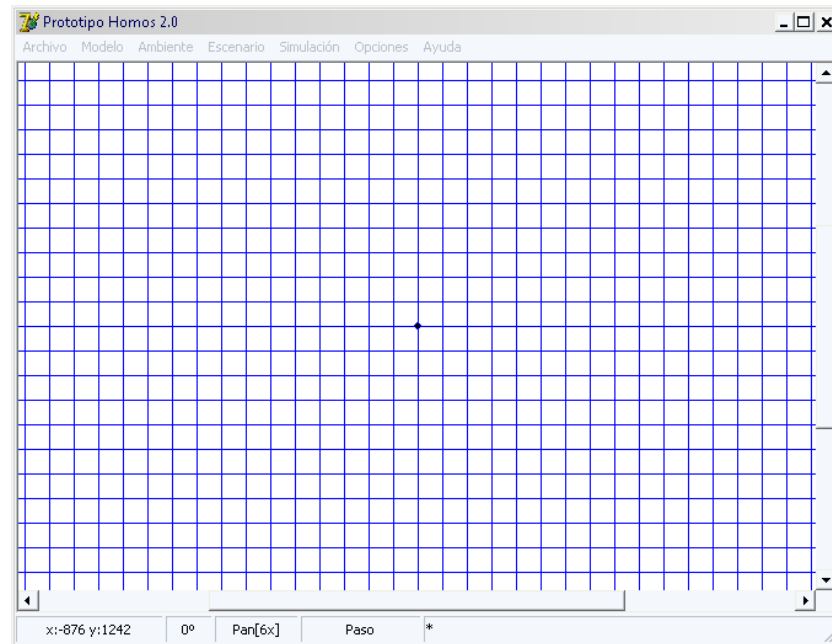
Fuente: Autor.

Figura E.49: Estructura del submenú Ambiente en el prototipo Homos 2.0



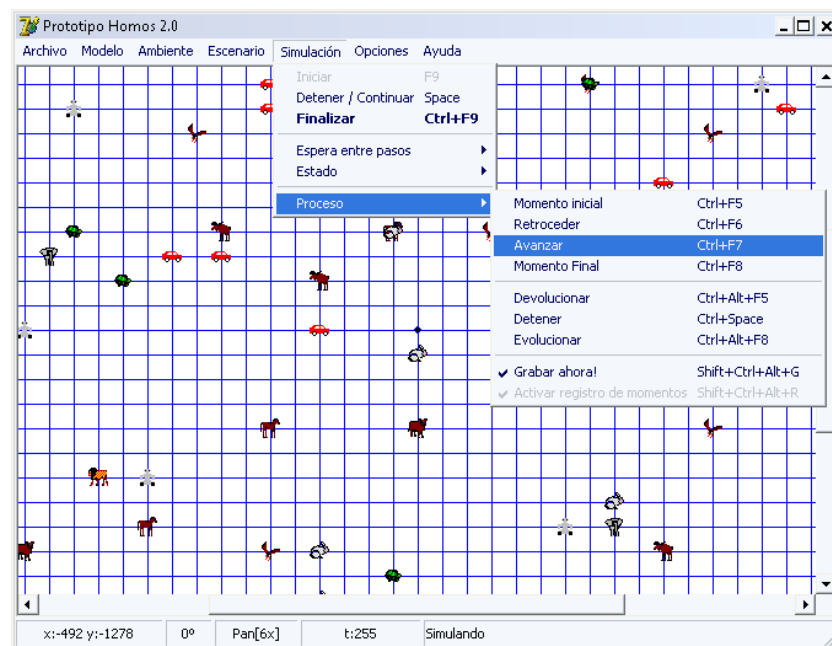
Fuente: Autor.

Figura E.50: Estado inicial del prototipo de Homos 2.0



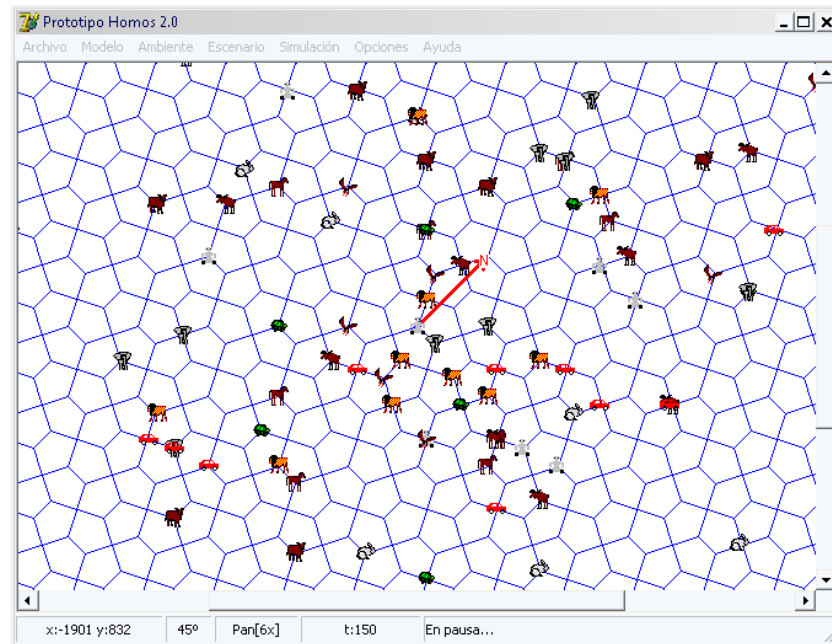
Fuente: Autor.

Figura E.51: Registro de momentos y controles de desarrollo de la simulación



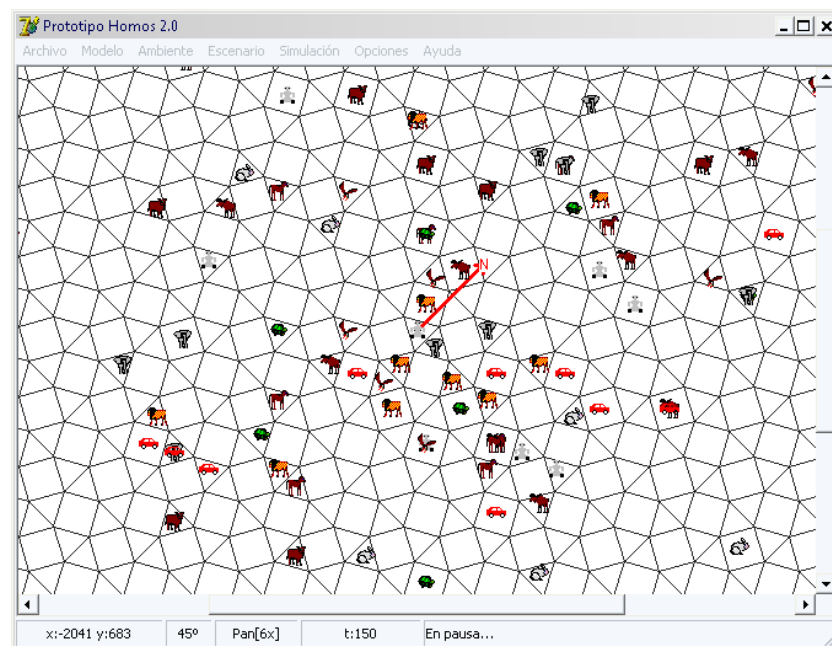
Fuente: Autor.

Figura E.52: Visualizar opciones de movimiento de objetos en espacio según geometría [Ctrl+B]



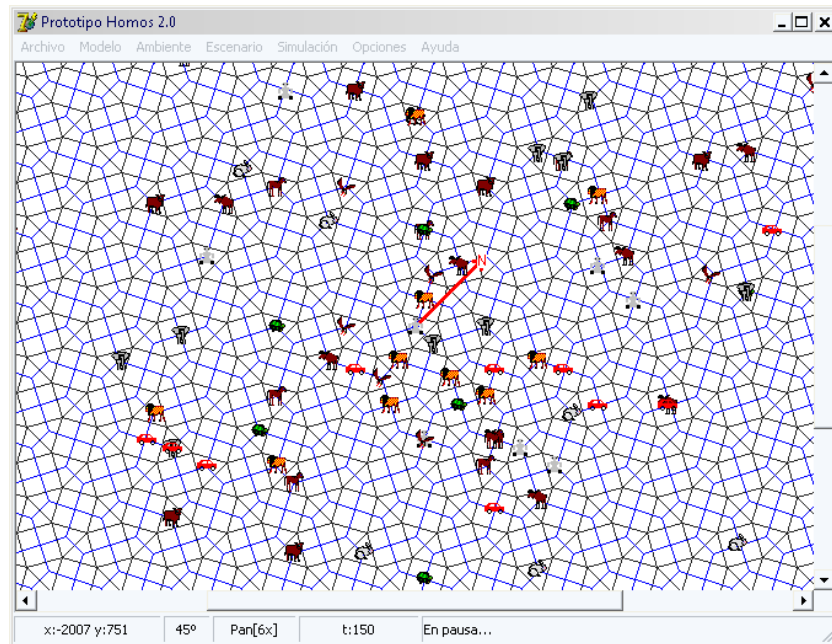
Fuente: Autor.

Figura E.53: Visualizar áreas de influencia espacial de objetos según geometría [Ctrl+D]



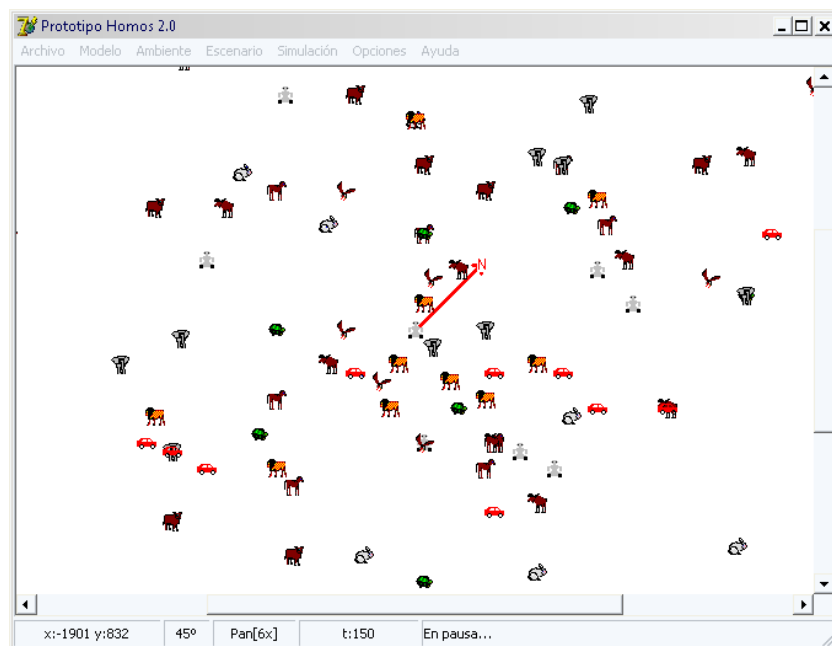
Fuente: Autor.

Figura E.54: Visualizar áreas de influencia y opciones de movimiento según geometría [Ctrl+M]



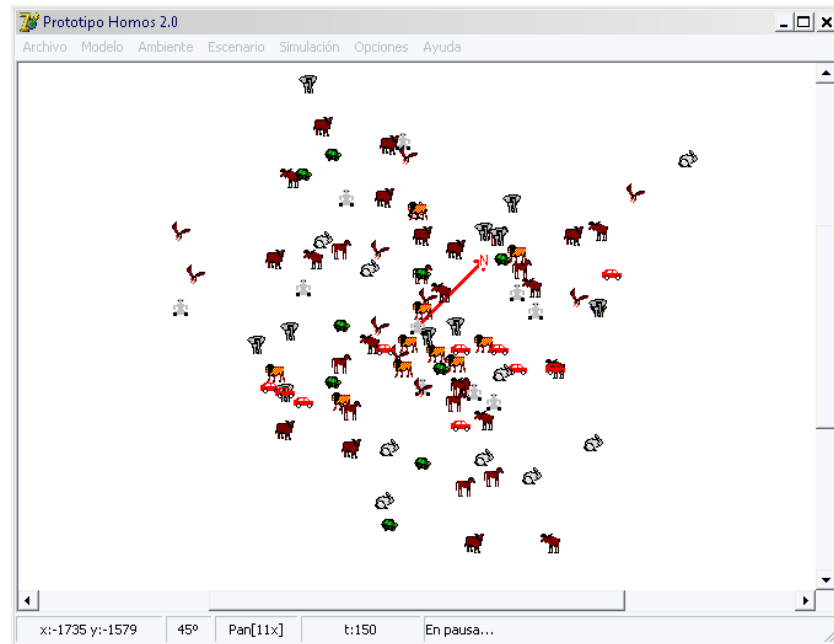
Fuente: Autor.

Figura E.55: Visualizar objetos en espacio libre [Ctrl+L]



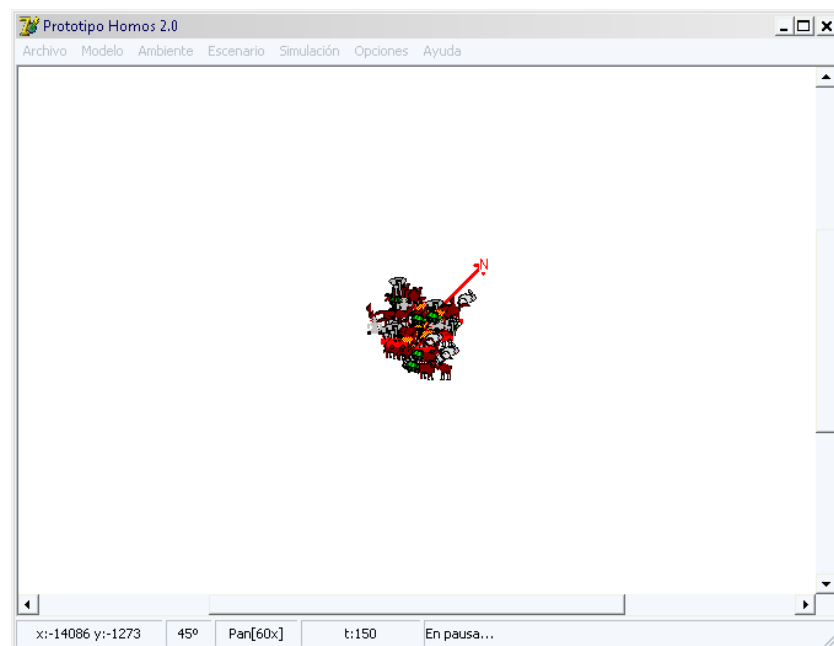
Fuente: Autor.

Figura E.56: Ambiente en panorámica cercana, 11x



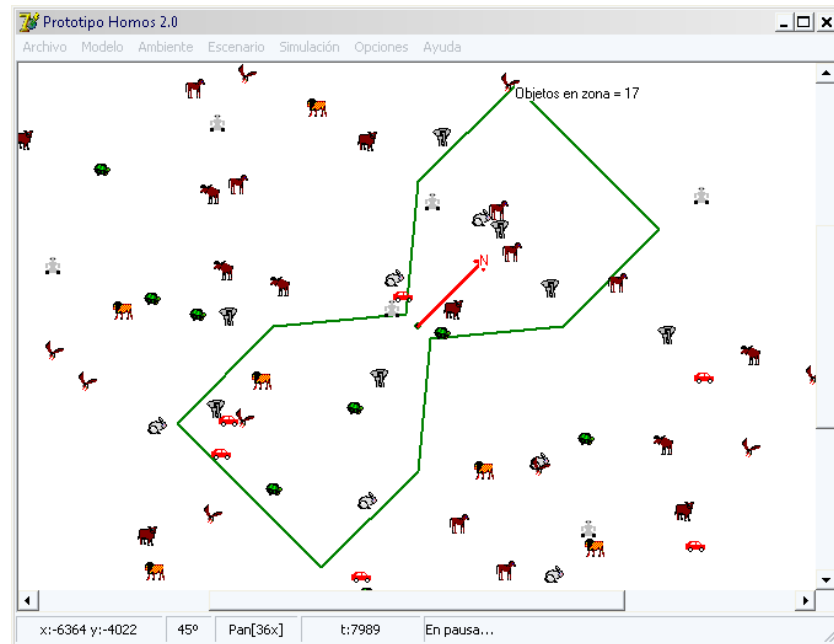
Fuente: Autor.

Figura E.57: Ambiente en panorámica máxima, 60x



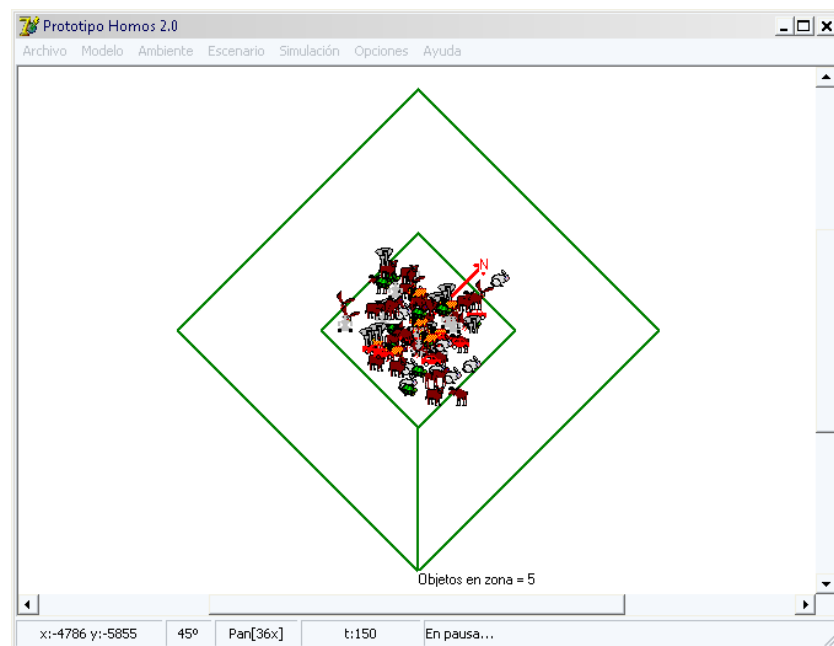
Fuente: Autor.

Figura E.58: Visualización de Zona 3: Reloj de arena



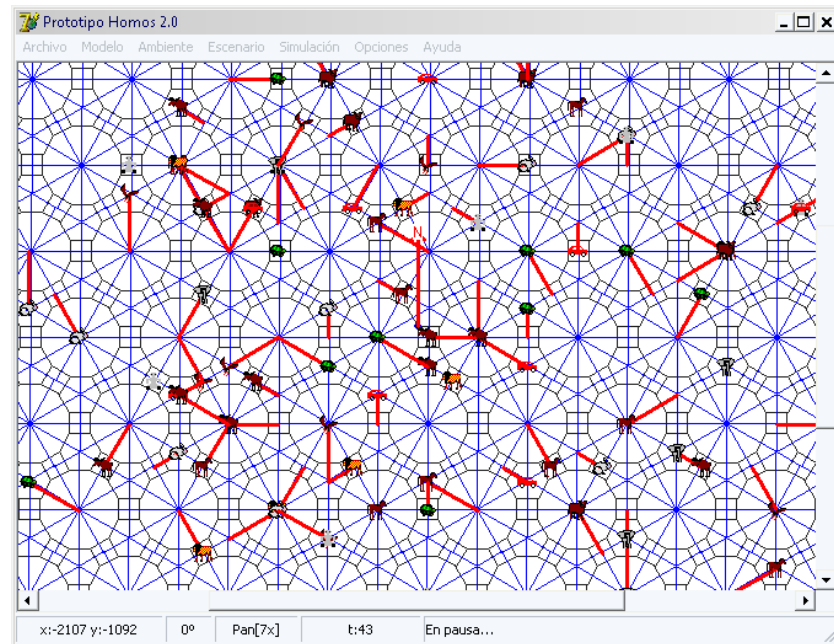
Fuente: Autor.

Figura E.59: Visualización de Zona 4: Anillo cuadrado



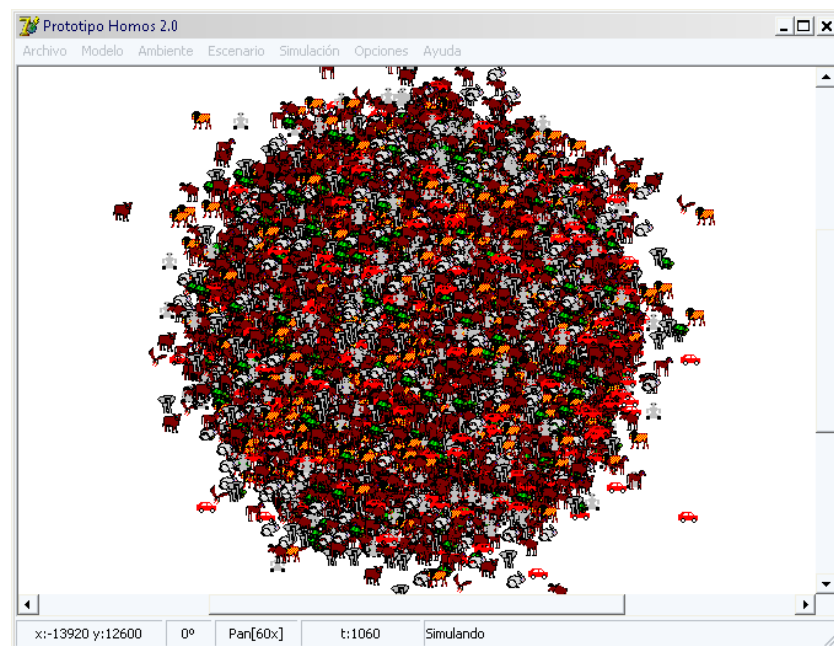
Fuente: Autor.

Figura E.60: Presentar la última ubicación de objeto [Ctrl+U]



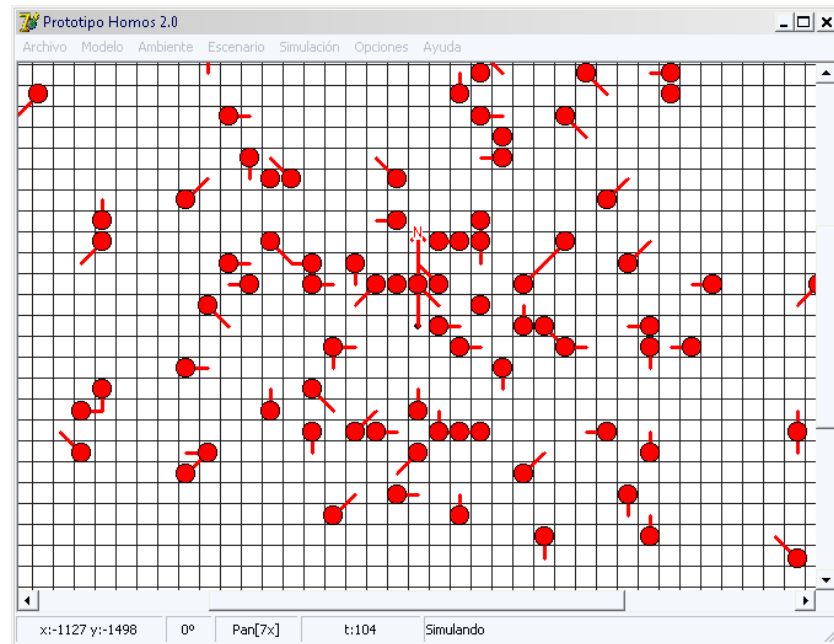
Fuente: Autor.

Figura E.61: 100.000 objetos en ambiente



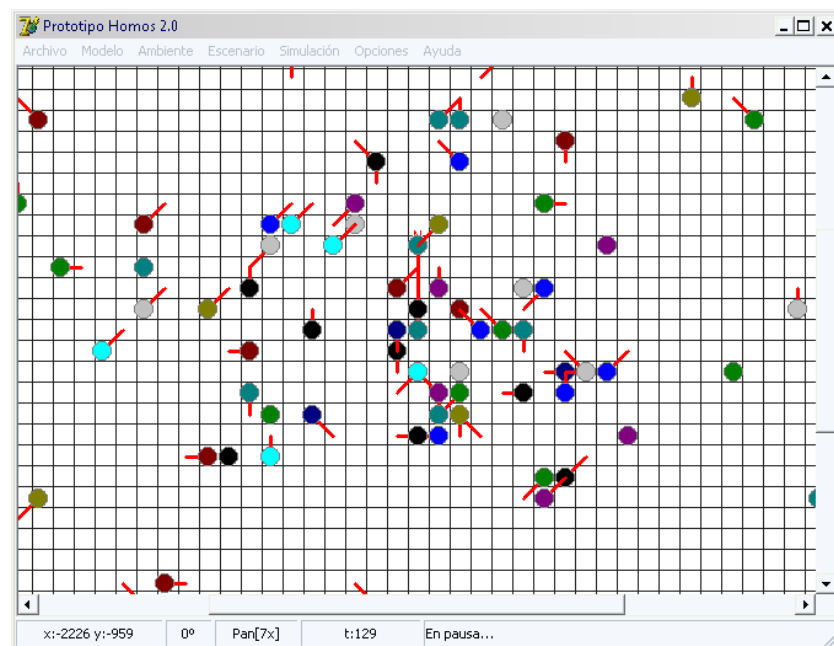
Fuente: Autor.

Figura E.62: Objetos en ambiente como puntos / círculos uniformes [Shift+1]



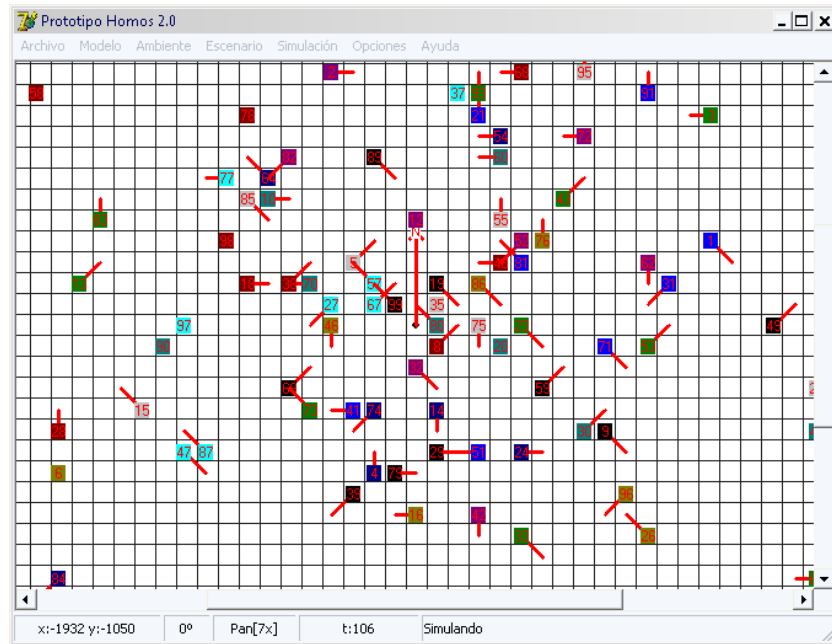
Fuente: Autor.

Figura E.63: Objetos en ambiente como círculos coloreados según su clase [Shift+2]



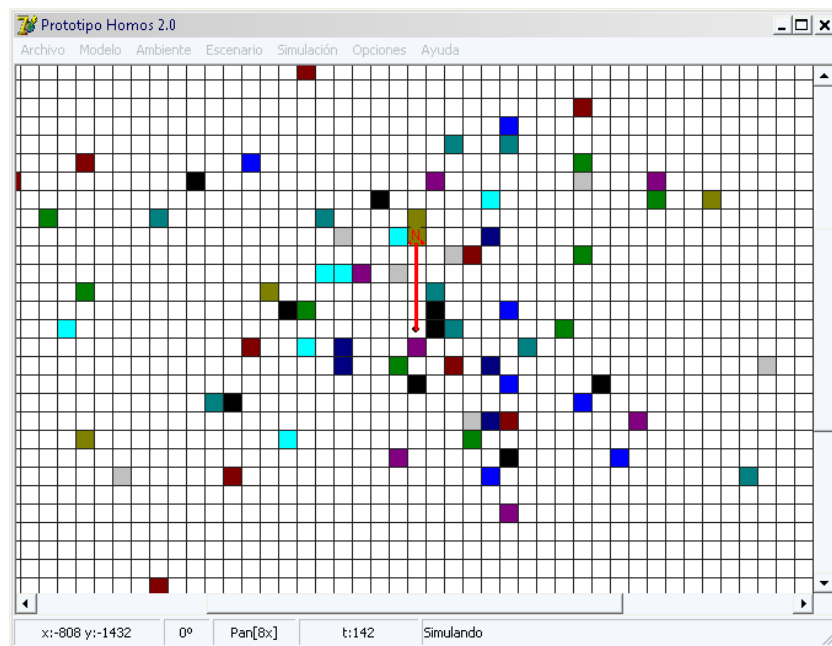
Fuente: Autor.

Figura E.64: Objetos en ambiente identificados serialmente [Shift+3]



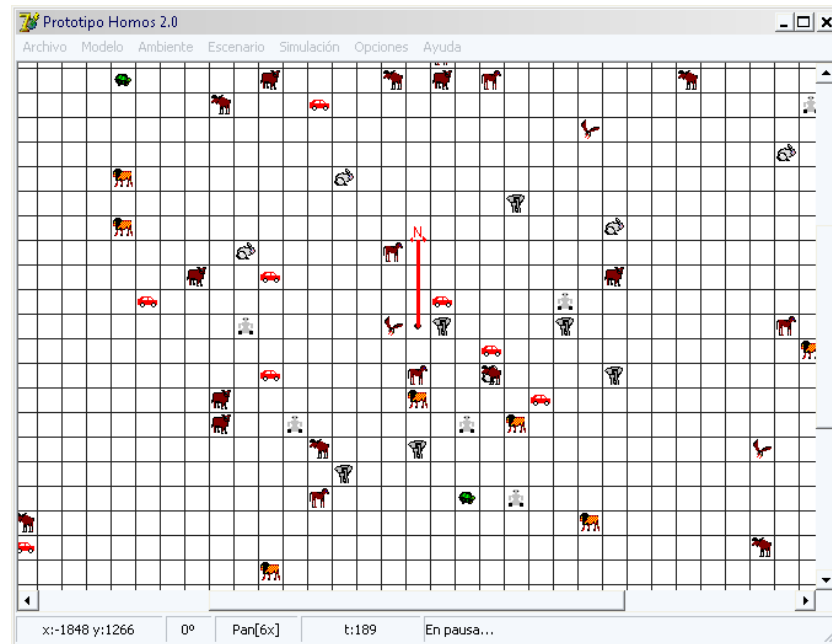
Fuente: Autor.

Figura E.65: Área de influencia inmediata de objetos coloreada según su clase [Shift+4]



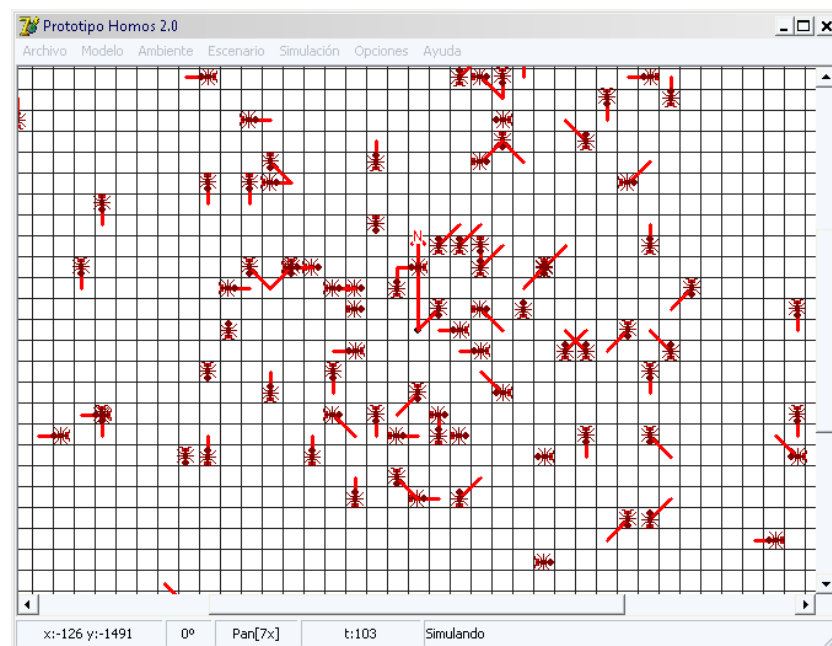
Fuente: Autor.

Figura E.66: Objetos en ambiente según imagen de clase [Shift+5]



Fuente: Autor.

Figura E.67: Objetos en ambiente como hormigas [Shift+6]



Fuente: Autor.

E.8.5. Descripción de elementos de menú no implementados en PH2

Con el propósito de dar completitud al PH2, se integraron al menú principal un conjunto de submenús de funciones no implementadas. El objetivo de esta acción es:

- Observar la sinergia de integración de los módulos propuestos⁴⁰ y distintos elementos que los caracterizan.
- Orientar el desarrollo de los módulos, componentes y elementos faltantes en la dirección de integración de funciones propuesta en PH2.
- Servir de plantilla y registro para la continuación del desarrollo del prototipo, completando los elementos faltantes hasta alcanzar la versión estable o final de la herramienta software.

El orden en que se listan los menús corresponde al orden presentado en el prototipo. Este orden se justifica ya que corresponde con la forma más común de desarrollo de modelos en la herramienta software, conclusión se obtuvo gracias a los distintos procesos de análisis realizados a Homos 1.0. Se dan a conocer a continuación el contexto sobre el cual se soporta la aparición de cada elemento no implementado en el menú principal de PH2.

E.8.5.1. Menú archivo

Este menú integra las funciones relacionadas con la recuperación y almacenamiento de los modelos creados en la herramienta.

- Almacenamiento y recuperación de archivos de modelo tanto en formato texto y binario. Se espera la posibilidad de la carga simultánea de múltiples archivos de proyecto. La carga de archivos y la terminación de ejecución de la simulación debe ser tan inmediata como sea posible.
- Opciones para compactar mediante algoritmo de compresión los modelos resultantes. No se hace énfasis específico en cuanto a la seguridad de los archivos, puesto que existen aplicaciones externas que realizan compresión y seguridad con gran eficiencia, P.ej.: Winrar de de RARLAB⁴¹.
- En cuanto a opciones de impresión, se espera reportar en papel cada elemento que constituya una forma de representación intermedia y final tanto del modelo como de la información obtenida del proceso de simulación. Se supone una variedad de resultados mayor a la existente en Homos 1.0. Ejemplos de nuevos elementos a reportar se presentan en la sección E.3 en la página 584.

⁴⁰Cada elemento raíz en el menú principal.

⁴¹<http://www.winrar.es>

E.8.5.2. Menú Modelo

A largo plazo este es el menú más complejo a implementar, ya que busca integrar funcionalmente la administración de los componentes de representación de transiciones de los modelos:

- Se propone la administración y manejo de múltiples modelos simultáneos, brindando soporte a operaciones de edición a nivel de modelo entendidas como crear, seleccionar y eliminar modelos. Al contar con la carga simultánea de varios modelos, puede suponerse por ejemplo una opción en la cual puedan integrarse varios modelos en un mismo ambiente de simulación. Sobre integración de múltiples modelos ver [Sar06].
- Opciones de edición de clases y reglas: crear, modificar, eliminar clases y reglas del modelo integración con SPRT: En las reglas, se propone adicionalmente presentar, generar, manipular y hacer uso de elementos característicos de los SPRT que sirvan de utilidad para la descripción de transiciones en los modelos construidos con la herramienta software.
- Creación manual y automática de diagramas relacionados con el modelo activo. En el menú se hace mención del modelo de clases y el modelo BOR, pero como se ha visto en apartados anteriores de este documento, otros tipos de modelos y representaciones son posibles. Se espera contar con una amplia variedad de representaciones que faciliten la comprensión de la dinámica interna y la estructura de las reglas de transición planteadas en los modelos.
- En este menú se da espacio para la determinación de nuevas propiedades específicas del modelo, no contempladas en esta etapa de desarrollo de PH2.

No se presenta en este menú el repositorio o depósito de reglas⁴², debido a su actual impracticidad. Con la representación escrita de reglas de transición propuesta en este trabajo, es posible la implementación de una forma efectiva de esta función que pueda ser integrada naturalmente dentro del proceso de construcción de modelos en la herramienta software. Se preve una reaparición de esta funcionalidad en futuros prototipos.

E.8.5.3. Menú Ambiente

Este menú se encarga de la configuración y caracterización del espacio de simulación. Debido a la naturaleza propia de sus operaciones, se evalúa que le corresponde un segundo lugar en cuanto al nivel de complejidad de implementación.

- Soporte a múltiples ambientes simultáneos, se plantean opciones para editar, modificar, seleccionar y eliminar ambientes. Esta opción permite imaginar nuevas formas de uso de la aplicación tales como: ejecución de un modelo simultáneamente en varios ambientes, ejecución de varios modelos en uno o más ambientes simultáneamente, etc. Otras opciones de uso son igualmente posibles.

⁴²Función existente en la anterior versión del software, muy poco usada.

- A largo plazo se espera contar con librerías de grafómatas para la representación del espacio. A futuro es posible pensar también en la generación automática de grafómatas desde el propio software o desde una herramienta especializada. La integración de grafómatas externos al funcionamiento de la herramienta software se realizará como archivos especializados o como conjuntos de librerías listas para usar. La presentación actual de un menú con varias opciones de vecindarios será obsoleta a corto plazo, puesto que se requerirá una interfaz especializada en la administración y manejo de grafómatas para condensar el volumen de estructuras de este tipo que es posible usar. También se requiere de esta interfaz especializada si por ejemplo se desea mezclar dos o más de estas estructuras en un mismo ambiente.
- No se esperan mayores cambios en cuanto a opciones de visualización y de selección de la estructura que caracterizará la movilidad de los objetos en el ambiente, se suponen más y distintas opciones de representación de objetos en el ambiente distintas a las presentadas en este prototipo.
- Se plantean mejoras y cambios en cuanto a las zonas. Este es un elemento multipropósito y flexible que puede ser caracterizado de muchas formas útiles. Se esperan opciones de edición completa de zonas incluyendo también el aspecto gráfico, p.ej: en la inclusión de trazos de líneas curvas en sus formas básicas. La asociación de zonas con otras funcionalidades distintas del conteo de objetos, tales como la determinación de contención o pertenencia de un objeto a un espacio específico, la creación de zonas de exclusión⁴³, entre otros usos posibles. Contar con zonas prediseñadas facilitará el uso de esta característica en futuras representaciones del ambiente de simulación.
- En general el número de objetos en el ambiente constituyen una limitante para el funcionamiento del simulador⁴⁴. A largo plazo se espera contar un mecanismo tal que permita usar racionalmente, según sea el caso, todos los recursos computacionalmente disponibles para la representación y ejecución de modelos en la herramienta software.
- La ubicación de los objetos en el espacio es importante, por esta razón se esperan nuevas formas de visualización relacionada con la posición del objeto en el ambiente, como ya se ha mostrado con las opciones ver última ubicación [Ctrl+U] y rastro de objetos en el ambiente.
- Como se hace notar al observar el prototipo, las opciones gráficas de presentación del ambiente están en primerísimas etapas de desarrollo. Estas son 2D al igual que en la versión anterior software, posteriores desarrollos pueden extenderse a las tres dimensiones y otros tipos de espacios, p.ej. el hiperbólico.

Adicionalmente a la caracterización de funciones presentada, se promueve el uso de imágenes vectoriales⁴⁵ para la representación gráfica de objetos, clases y elementos en el ambiente, junto con el soporte de tipos de imágenes tradicionales *.bmp, *.jpg, *.ico y otros tipos *Raster*⁴⁶ con o sin compresión. El uso de imágenes vectoriales se justifica debido a la calidad del resultado gráfico.

⁴³Como remplazo al recurso de objetos / clases inertes como bordes o limitadores de movimiento usados en Homos 1.0

⁴⁴A mayor cantidad de objetos, más cálculos.

⁴⁵Por ejemplo de tipo *.svg. Para conocer mas ver <http://www.w3.org/Graphics/SVG/>

⁴⁶Mapa de bits, imagen matricial o bitmap.

E.8.5.4. Menú Escenario

La etapa presimulación es compleja debido a la cantidad de condiciones y valores adicionales con los cuales controlar la simulación y caracterizar aun más el modelo y el ambiente de simulación. El escenario se encarga de estos detalles.

- En PH2 se propone centralizar en este submenú la edición gráficas, reportes, hitos, la narración de la simulación y creación de micromundos como parte de la configuración natural de los escenarios.
- Múltiples escenarios hacen posible que se pueda evaluar un modelo en un mismo tipo de ambiente en condiciones completamente diferentes, propias de cada escenario. Esta es una valiosa cualidad heredada de Homos 1.0.
- Se plantea la edición completa y centralizada de vistas, gráficas comparación entre curvas poblacionales, tablas de datos y demás funciones relacionadas con la presentación de datos en forma gráfica o tabulada. La edición de reportes comprende el conjunto de actividades que permiten determinar específicamente cual es la información a ser retornada por el simulador al terminar la ejecución del modelo.
- Los hitos son condiciones a ser evaluadas dentro del proceso de simulación del modelo. Permiten caracterizar y controlar el desarrollo de la simulación. Los hitos pueden ser automáticos, manuales o predefinidos por el modelador antes de iniciar el proceso de simulación. Una vez que se reconoce el cumplimiento de un hito, este se registra. La colección de estos registros permite determinar las condiciones de ejecución naturales del modelo o las particulares buscadas por el modelador.
- La narración de simulaciones es una actividad propuesta por este proyecto para la descripción en prosa u otro medio de los eventos acontecidos dentro de la simulación del modelo. Su propósito es revelar la secuencia de acciones producidas así como destacar determinados eventos dentro del proceso de construcción de los resultados de simulación. Inicialmente se plantea la producción de narraciones como resultados escritos, a largo plazo es posible pensar en la narración hablada de la simulación en curso así como el uso de otras formas de expresión posibles.
- Los gráficos y sus datos, los hitos y la narración aunque surgen como entidades independientes, se seleccionan e integran según necesidades al escenario deseado.
- Incluir la construcción de micromundos en la herramienta software refuerza la integridad de la aplicación en cuanto al proceso de construcción de materiales educativos. Ver más sobre esto en el capítulo 4.5 de [ADE⁺01]. La construcción de un micromundo implica el uso de todos los recursos y funciones implementados en la herramienta software junto con la capa gráfica intermedia necesaria para la construcción del micromundo, la capa de control y de administración de usuarios. Se profundiza esto en [IMS00, RC08].
El prototipo plantea la integración del ciclo completo de edición de micromundos en la

herramienta software: Crear, modificar, seleccionar y eliminar micromundos. Incorporar esta funcionalidad cierra el círculo de utilidades para la creación de materiales educativos con la MBOR y Homos. Los micromundos pueden hacer uso en su configuración de uno o más escenarios.

E.8.5.5. Menú Simulación

Adicionalmente a los controles de simulación implementados en PH2, se espera contar con funciones que permitan la recuperación y almacenamiento de uno o más estados (pasos) de simulación. Estas funciones permitirán evitar partir desde cero en el estudio particular de un modelo, también servirán como punto de partida desde el cual realizar contrastes entre la configuración actual y la producida por cambios en la configuración del modelo, del escenario u otros elementos con el objetivo de determinar los efectos producidos por dichos cambios.

E.8.5.6. Menú Opciones

Cada elemento presentado en los distintos módulos que componen la herramienta software son caracterizables, múltiples opciones se derivan de cada una de ellas. Se hace necesaria la creación de un medio dentro de la aplicación en el cual se reúnan las opciones de configuración de los distintos elementos previstos en el software.

- Una vez alcanzada una representación flexible de las teselaciones en el ambiente, es posible pensar en la implementación de deformaciones y cambios de densidad de las estructuras gráficas, estas son funciones que permiten la representación de formas especializadas de teselaciones en el ambiente.
- En las opciones de aleatoriedad, se espera incluir nuevos tipos de generadores de números aleatorios. Esto a largo plazo también implica la adición de funciones estadísticas para la validación y estudio de los resultados producidos por los generadores usados.
- También se incluye dentro de las opciones la internacionalización de la interfaz, el soporte a múltiples idiomas. Esta es una función necesaria en la actual realidad globalizada. Su impacto y acción esta dirigido a la emancipación y el dar a conocer de la MBOR y de herramienta software en otras latitudes. La valía de esta función se destaca aun más cuando se acompaña de acciones de carácter administrativo como promoción de contenidos, creación de materiales, etc.. actividades que generalmente están fuera del alcance del proceso básico de desarrollo del software.

E.8.5.7. Menú Ayuda

Al igual que en los procesos de internacionalización, el módulo de ayuda requiere para su evolución integral de acciones fuera de alcance del proceso de desarrollo de software. Sincronizar lo existente

en el mundo real (las ayudas con las que se puede contar) con las indicaciones de ayuda planteadas en el producto, requieren de voluntad de permanencia del productor del software y de planeación para hacer de la ayuda ofrecida algo estable.

- El programa pueda ofrecer por si mismo auxilio de primera mano mediante la implementación de ayudas contextuales.
- La creación de manuales y materiales de ayuda es una actividad que requiere tiempo, dedicación y experiencia de uso con la herramienta. Igualmente, no es posible reducir la importancia del material de referencia para el sano desarrollo de la actividad de modelado. En futuras versiones de la herramienta software se espera contar con los siguientes materiales:
 - Manual de usuario de la aplicación.
 - Guía de desarrollo de modelos con la herramienta software.
 - Guía de desarrollo de modelos con la herramienta software con énfasis en cada interfaz disponible implementada.

Es natural esperar que esta información se encuentre disponible en el sitio web particular de la herramienta software.

- El dar a conocer y la integración con la comunidad es parte fundamental de las actividades de crecimiento y desarrollo de la influencia de una metodología como la MBOR. Se espera la implementación de un sitio web específico para la herramienta software y su temática: esta constituiría un punto de encuentro necesario para usuarios, modeladores, desarrolladores y programadores de scripts relacionados con el software, un lugar de depósito de modelos desarrollados y una fuente de información sobre novedades acerca de la herramienta software y su metodología.

E.8.6. Instrucciones de uso y prueba de manejo del prototipo

El prototipo software desarrollado durante esta investigación se llama `ph2.exe` y se encuentra en la carpeta `contenido\anexos\Prototipo Homos 2.0` dentro del CD-ROM adjunto al documento de proyecto de grado. Con el propósito de facilitar el primer contacto con el prototipo, se presenta la secuencia esperada de operaciones de uso y un conjunto básico de pruebas de las funciones implementadas. Al final de este apartado se presenta una guía gráfica de acciones disponibles, en la figura E.68 en la página 686. La guía indica el orden de la secuencia esperada de operaciones y el momento de realización de las distintas pruebas sugeridas.

E.8.6.1. Secuencia de operaciones

Se han establecido 7 etapas de uso dentro del flujo de acciones desarrollables con el prototipo software:

1. Inicio de operaciones: el prototipo muestra un ambiente vacío.
2. Configuración del simulador: se hace uso del menú principal para configurar:
 - La cantidad de objetos a simular: Se ofrecen opciones para usar 100, 1.000, 10.000, 100.000 y 1.000.000 de objetos. Claramente la última opción es la más lenta de todas, inicialmente se ofrecen 100 objetos.
 - El tipo de aleatoriedad a usar en los cálculos: Actualmente se usan LCG Random y Merssene Twister. Este último es la opción por defecto.
 - El tipo de vecindario a usar: Se activa solo una geometría de vecindario de entre 27 opciones disponibles. Se activa por defecto el vecindario de Moore de 9 celdas.
 - Determinar el grafómata a usar como vecindario: Los objetos en el ambiente hacen uso del grafómata base para determinar el camino de movimiento en el ambiente, el dual se usa para establecer el área de influencia inmediata de cada objeto presente en el ambiente. Al elegir uno como base el otro necesariamente será el dual. Ver la prueba (E) en la página siguiente.
3. Iniciar la simulación.
4. Terminar la simulación.
5. Finalizar la ejecución del programa.
6. Volver a usar el simulador: Este paso representa la simulación después de terminar una simulación. Esta nueva ejecución del simulador hace uso de los últimos valores de configuración establecidos. En esta versión del prototipo no se preserva el estado del simulador, siempre se inicia con el ambiente vacío.
7. Este paso solo es posible en la prueba (D), en la cual se activa y hace uso del control de evolución de la simulación.

E.8.6.2. Prueba de funciones incorporadas

Se proponen cinco actividades de prueba de las funciones implementadas:

- (A) Prueba de manejo de opciones de manejo del ambiente:
Consiste en hacer uso de los atajos de teclado correspondientes al manejo del ambiente para la realización explícita de las funciones asociadas.
- (B) Prueba de visualización de divisiones del espacio:
Esta operación es válida para todos los tipos de vecindarios previstos, la prueba consiste en hacer uso de los atajos de teclado agrupados en (B). En los vecindarios de Moore el dual siempre es la malla cuadrada.
- (C) Prueba de zonas en ambiente:
Existen 5 ejemplos + el ambiente sin zonas que es la opción por defecto. Al activar una zona, esta aparece en el ambiente como una figura geométrica de color verde. El simulador realiza el conteo de objetos que están dentro de la zona. La prueba consiste en hacer uso del menú Ambiente/Zonas/Prediseñadas y elegir una de las opciones disponibles. Solo se activa una zona a la vez.
- (D) Prueba de mecanismo de avance y retroceso de simulación:
Esta prueba es posible solo si antes de iniciar la simulación se activa el submenú registro de simulación con [Shift+Ctrl+Alt+R]. La grabación se inicia al activar el submenú grabar ahora (desde este momento) con [Shift+Ctrl+Alt+G].

La grabación se puede activar:

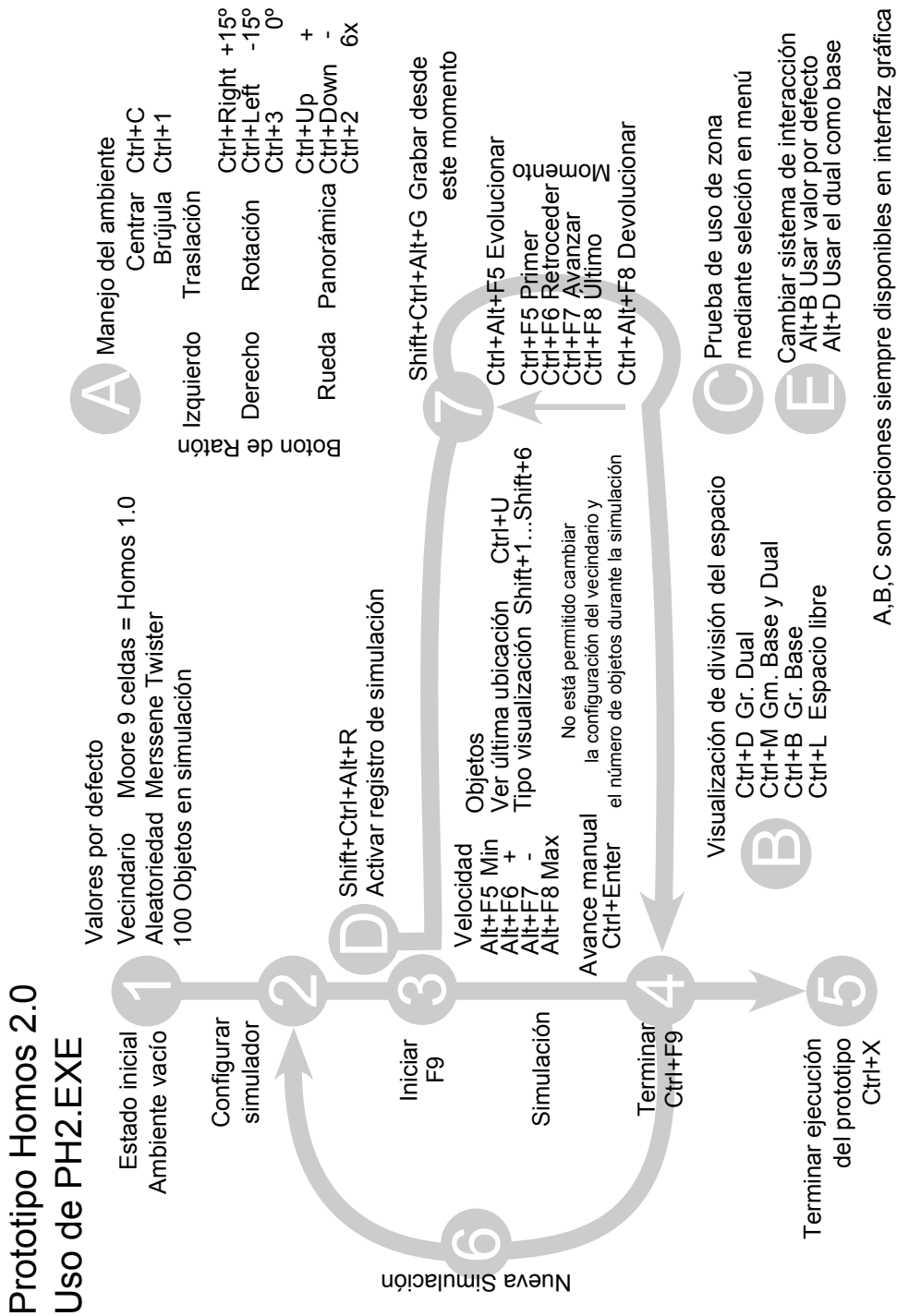
- Antes de iniciar la simulación, si se desea volver al primer paso de la simulación.
- Durante la ejecución del simulador, activar en cualquier momento a partir del punto de interés.

En este prototipo se almacenan hasta 100 momentos, separados entre sí cada 5 pasos de simulación. La simulación se detiene una vez que se ha alcanzado el máximo número de pasos de grabación. Esta es la oportunidad de hacer uso de los controles de proceso / evolución de la simulación. La rapidez en la reproducción de la evolución está sujeta a la última configuración realizada sobre los controles de velocidad de la simulación.

- (E) Prueba de cambio de sistema de interacción
Consiste en cambiar el tipo de grafómata sobre el cual se mueven los objetos.
 - [Alt+D] Hace que el grafómata dual sea quien determine el movimiento de los objetos en el ambiente.
 - [Alt+B] Restablece el valor por defecto al hacer uso del grafómata base.

Para realizar esta prueba se requiere realizar los cambios antes de iniciar la simulación.

Figura E.68: Guía de acciones disponibles en el prototipo de Homos 2.0

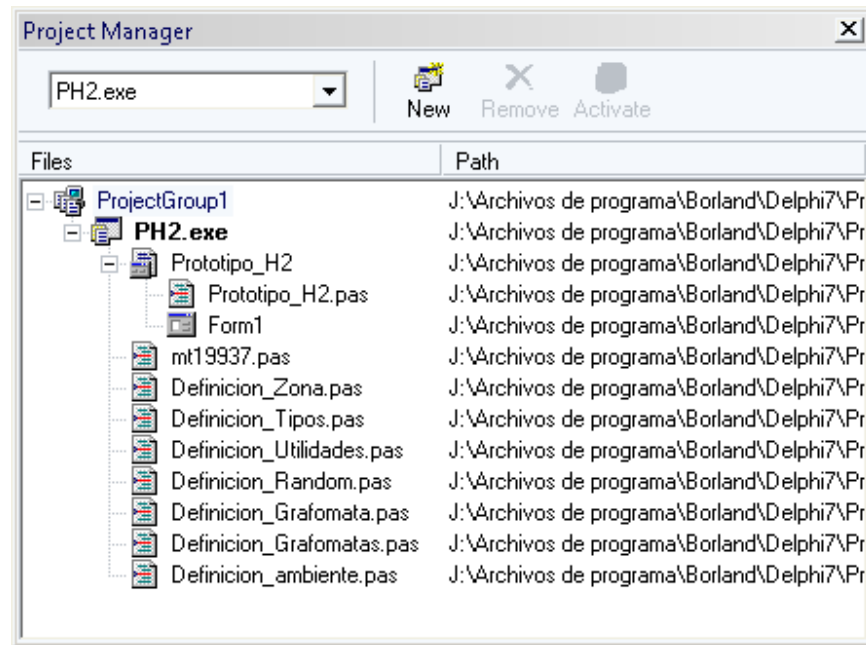


Fuente: Autor.

E.8.7. Componentes del prototipo

La figura E.69 presenta la estructura del proyecto y las unidades que componen al prototipo PH2.exe:

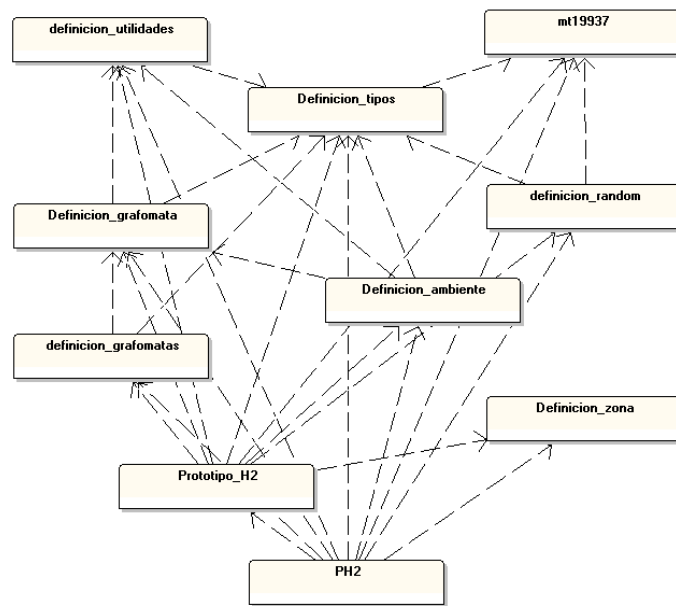
Figura E.69: Estructura de unidades componentes del prototipo PH2.exe



El uso de cada una de las unidades componentes es la siguiente:

- El prototipo usa solo un formulario que sirve de interfaz gráfica a la aplicación. Prototipo_H2 contiene el código de las funciones implementadas en la interfaz gráfica.
- mt19937.pas contiene el algoritmo del generador de números aleatorios Mersenne Twister [MT].
- Definicion_Grafomata.pas implementa el objeto Tgrafomata y varias funciones auxiliares con las cuales se construyen los grafómatas.
- Definicion_Grafomatas.pas implementa cada grafómata y una librería interna que organiza estas estructuras.
- Definicion_tipos.pas contiene la declaración de los tipos de datos principales usados en el proyecto, especialmente la estructura interna de los grafómatas.
- Definicion_ambiente.pas contiene dos versiones del algoritmo de relleno de superficies, uno para el trazado del dibúgrafo sobre el área de dibujo y otro para la etapa de relleno final mediante copia de porciones de memoria en la pantalla.
- Definicion_Utilidades.pas contiene funciones de control de tiempo y el TAD Mergesort.

Figura E.71: Diagrama de dependencias



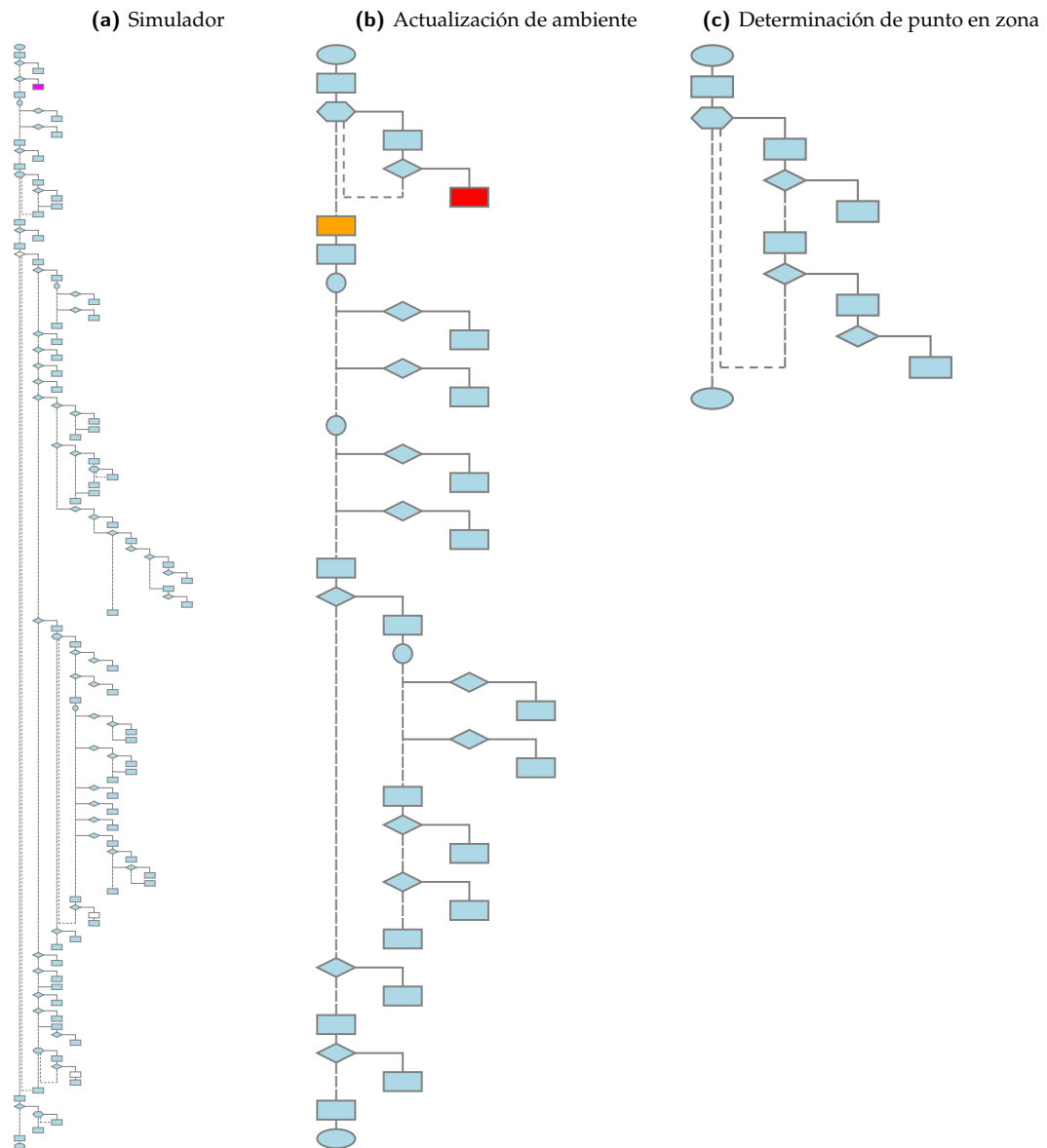
Las flechas indican las dependencias inmediatas entre módulos.

Fuente: Autor.

E.8.9. Recursos relacionados

En los anexos presentados en el CD-ROM que acompaña esta tesis se incluye: El código fuente del prototipo PH2, una guía de diagramas de flujo, el modelo UML y el archivo binario producido. Se muestran a continuación ejemplos de diagramas de flujo que componen la guía:

Figura E.72: Muestra de diagramas de flujo del código fuente de PH2



Detalles acerca de los diagramas de flujo se presentan en la sección E.7 en la página 654.

Fuente: Autor.

E.8.10. Conclusiones

A partir del estado de implementación de los distintos elementos incluidos en PH2, se afirma que el prototipo se encuentra en una etapa inicial de desarrollo. Por ahora es solo un útil de investigación y como tal, permite establecer pautas y elementos de cambio a incorporar o desechar en una futura versión completa de la herramienta software.

Sobre los elementos incorporados en el prototipo podemos concretar que:

- La adición del avance y retroceso junto con las opciones de pausa en ejecución del proceso de simulación ayudan a mejorar la comprensión del desarrollo del modelo. Incrementa la interactividad con la aplicación, constituye una adición relevante para futuras versiones de la herramienta software.
- El mecanismo implementado de avance y retroceso de la simulación no está optimizado. Se requieren mejoras en el algoritmo que permitan la realización de las funcionalidades descritas con un menor consumo de recursos.
- La representación de gran cantidad de objetos en el ambiente implica tiempo de proceso y recursos de memoria para su representación. Para manejar grandes poblaciones se necesita de una estructura interna de administración y acceso a ubicaciones en memoria que reduzca al máximo la estructura de almacenamiento requerida y mejore el tiempo de acceso a los datos. Constituye un elemento principal dentro de la arquitectura de una futura herramienta.
- El rastreo de objetos en el ambiente sirve al propósito de explicación del desarrollo espacial de la simulación del modelo en curso. Por este medio se reflejan tanto la influencia / alcance de un objeto como la tendencia general de movimiento de los objetos en el ambiente.
- El concepto de zonas es un útil de simulación con el cual se pueden obtener resultados en una dimensión distinta al simple conteo poblacional de objetos en el ambiente. Facilitar la selección y configuración de formas de las zonas permitirá la construcción de modelos relacionados con cambios poblacionales en sectores específicos del ambiente, permitirá una forma de control de simulación centrada en la configuración misma del espacio, entre otras posibles aplicaciones prácticas.
- El prototipo implementa un conjunto limitado de teselaciones regulares y sus duales. Los grafómatas base que estructuran las teselaciones incorporadas están hechos a mano, solo sus duales son calculados, todo el conjunto se almacena internamente en forma vectorial en una librería. Aumentar la variedad de estructuras redundará en una mejora sustancial en la representación del espacio.
- Gracias a la experiencia adquirida en el desarrollo, implementación y uso de los grafómatas, puede afirmarse que hacerlos manualmente no es algo práctico. A largo plazo la mejor opción es la construcción automática de grafómatas. La automatización de este proceso facilitará el estudio y aplicación práctica de este tipo de estructuras.

- La representación gráfica de teselaciones en el ambiente se encuentra en una etapa inicial: los grafómatas implementados usan ubicaciones enteras. Esto da estabilidad, velocidad y reducción del tamaño de datos requerido para su almacenamiento, pero trae consigo deformidad gráfica de la estructura debido al uso de aproximaciones y redondeos para calcular las distintas ubicaciones. Se requieren mejoras en la representación interna de los grafómatas, en la velocidad de dibujo y la calidad gráfica de las imágenes producidas.
- El uso de la vista panorámica en el ambiente permite concluir que para la representación de divisiones del espacio:
 - La forma del vecindario es importante para la interrelación local: Los objetos responden distinto a cada configuración del espacio, ver de cerca los objetos destaca este efecto.
 - La estructura del vecindario a nivel general se hace uniforme: Cuando se usan valores de panorámica mayores, las distancias entre las divisiones del espacio se hacen tan pequeñas que producen un efecto óptico de similitud entre tipos de geometría diferentes.
- Al combinar el movimiento aleatorio de los objetos con cada una de las distintas teselaciones incorporadas y sus duales, se observa que cada forma de división trae consigo la realización de dinámicas de desplazamiento por el espacio completamente diferentes entre sí. Analizar el impacto, las implicaciones, ventajas y desventajas de cada sistema para un modelo dado es fuente natural de interrogantes por resolver y propias de este estudio.
- El color y forma de los distintos elementos presentados en el prototipo es optativo, constituyen solo sugerencias. Se eligieron porque presentaron ventajas visuales inmediatas frente a otras combinaciones posibles, por ejemplo un mejor contraste. Se espera que cambien en próximas iteraciones de desarrollo de la herramienta.
- El uso de distintos tipos de generadores de valores aleatorios en el prototipo permite establecer lo que es mejor para la herramienta software a largo plazo: conviene integrar en la aplicación más de un sistema de generación de números aleatorios y que estos sean los mejores en su clase, para que en términos de los números generados se mejore la calidad de la simulación y sus resultados.
- Proponer entradas inoperantes en el menú principal del prototipo parece contraproducente, ¿Porqué incluir estos elementos?. En términos del proceso desarrollado, la acción se justifica ya que permite ubicar en perspectiva el conjunto de funciones que se espera implementar a futuro. Este ejercicio obliga al diseñador de la interfaz a pensar en la sinergia total entre funciones, la modularidad y la no redundancia de las características ofrecidas en el software.
- Desarrollar una guía de uso del prototipo y establecer métodos para la prueba de funciones implementadas constituye una muestra de seguridad y confianza en el producto ofrecido.
- Anexar el código fuente, los diagramas de flujo y demás recursos asociados a PH2 mejora la documentación, provee a largo plazo datos e información útil para posteriores desarrollos.

Apéndice F

Anexos en CD-ROM

F.1. Recursos relacionados con el desarrollo del proyecto de investigación

En los anexos presentados en el CD que acompaña esta tesis se incluye:

el código fuente del prototipo PH2 y una guía de diagramas de flujo del código fuente que compone a PH2.

Esta acción se realiza como continuación de una actividad ya establecida en Homos 1.0 que consiste en la incorporación de detalles sobre la documentación, tan completa como sea posible de los productos obtenidos durante la investigación.

La utilidad principal de las distintas guías de código fuente presentando diagramas y código es la ilustración rápida de la complejidad implicada en cada una de las funciones que componen la aplicación. Sirven como método de documentación técnica, son de utilidad para la comprensión del código del cual han sido generados.

Todos los diagramas incluidos en este documento han sido generados con una herramienta interna de desarrollo mantenida por el autor de este documento. Esta se encuentra en fase de pruebas pero los resultados obtenidos son suficientes para los propósitos de ilustración de este documento.

Con el propósito de presentar en forma clara la investigación

cada elemento tiene que ver con los distintos procesos llevados a cabo durante la investigación

Estos son los elementos incluidos como los anexos:

\latex\contenido\anexos\Homos 1.0

binario principal version 1.0.0.73 = Homos.exe

Texto de la licencia de Homos 1.0 en español e inglés = licencia_esp.txt + licencia_ing.txt

ejemplo de “Resultados_impresos_en_Homos_1.0.pdf”

Diagrama de clases de Homos 1.0

¿¿y la prueba de realizacion de analisis funcionales??

se realizaron guías de código de

homos 1.0.0.23

homos 1.0.0.50

homos 1.0.0.73

prototipo homos 2.0 PH2.exe

- SPRT
 - Sistemas nSmD
 - Sistemas nRD
 - Sistemas 2mSmD
 - Caracterización numérica de los Sistemas de Producción de Reglas de Transición [SPRT]
 - Código fuente de scripts en PHP
 - AegesPHP.php
 - Gspirt.php
- Código fuente y resultados de Modelos-script
- Código fuente en Delphi
 - Documentación de elementos existentes
 - Modelo UML de Homos 1.0 original
 - Modelo UML de Homos completo
 - Guía Homos [1.0.0.23]
 - Guía visor de Homos [1.0.0.23]
 - Guía Homos versión completa [1.0.0.50]
 - Documentos particulares de este proyecto
 - Código fuente de Homos 1.0 [1.0.0.73]
 - Código fuente de PH2
 - Scripts en Metapost
- Gráficos realizados
- Bibliografía:
Archivos *.bib

etimologías proporcionadas por

<http://etimologias.dechile.net/griego/?Raices>

http://www.etymonline.com/index.php?l=a&p=53&allowed_in_frame=0

<http://contraestoyaquello.blogspot.com/2011/01/automatas.html>

Herramientas usadas en el desarrollo de este proyecto

edición de texto

latex

lyx + ...

Diapositivas de sustentación

inkscape 0.48. + jessyink + Texttext 0.4.4 + pstoeedit + ...

programación

Organización de los contenidos de la carpeta Proyecto_H2

- Proyecto_H2
 - contenido
 - anexos
 - ◇ sprt.xls
 - ◇ grafomatas.xls
 - ◇ mbor.xls
 - imágenes