COBOR 2.0
SOFTWARE PARA EL CONTROL BORROSO

SOFTWARE PARA EL CONTROL BORROSO BASADO EN ALGORITMOS MODERNOS "COBOR 2.0"

ELKIN JAVIER MENDEZ

OSCAR MAURICIO ORDÓÑEZ CAMARGO

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2005

COBOR 2.0
SOFTWARE PARA EL CONTROL BORROSO

SOFTWARE PARA EL CONTROL BORROSO BASADO EN ALGORITMOS MODERNOS "COBOR 2.0"

ELKIN JAVIER MENDEZ

OSCAR MAURICIO ORDOÑEZ

Trabajo de grado presentado como requisito parcial para optar a los títulos de:

INGENIEROS DE SISTEMAS

DIRECTOR

JUAN CARLOS REYES FIGUEROA

INGENIERO DE SISTEMAS

CODIRECTOR

FERNANDO RUIZ DIAZ

INGENIERO DE SISTEMAS, ME

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2005

COBOR 2.0
SOFTWARE PARA EL CONTROL BORROSO

TABLA DE CONTENIDO.

LISTADO DE FIGURAS

COBOR 2.0
SOFTWARE PARA EL CONTROL BORROSO

RESUMEN

TITULO:

"SOFTWARE PARA EL CONTROL BORROSO BASADO EN ALGORITMOS MODERNOS -COBOR 2.0 -"

AUTORES:

ELKIN JAVIER MENDEZ
OSCAR MAURICIO ORDÓÑEZ CAMARGO.

PALABRAS CLAVES:

Control borroso, Controlador borroso basado en Algoritmos modernos, Diseño de controladores borrosos, apoyo para el diseño del control borroso.

CONTENIDO:

En los sistemas de control, se presentan procesos en los que controladores diseñados según otros patrones (clásicos, adaptables.) no dan los resultados deseados. Se trata de procesos de elevada complejidad que en la mayoría de los casos, operadores humanos diestros y experimentados alcanzan resultados satisfactorios. Cuando se investiga el proceder de esos expertos, se concluye de inmediato que, tanto la información acerca del estado del proceso como sus acciones de control, las procesan y manejan de un modo básicamente cualitativo, usando su intuición, habilidad heurística y experiencia.

La importancia de la lógica borrosa desde el punto de vista de la teoría del control de procesos, es la de proveer soporte cuándo se quiere traducir el conocimiento heurístico experimentado, expresado en frases lingüísticas imprecisas a algoritmos numéricos. Todo basado en la teoría de conjuntos que posibilita imitar el comportamiento de la lógica humana. El término "borroso" procede de la palabra inglesa "fuzzy" que significa: confuso, difuso, indefinido.

COBOR 2.0 es un nuevo software para los sistemas de control, que utiliza la lógica borrosa como una tecnología adaptativa. Mediante el monitoreo de las variables de control, se determina la acción a tomar por el controlador sobre la inferencia de una base de reglas de conocimiento definida a partir de la experiencia dada por un experto en el control de procesos, tomando un soporte formal de los fundamentos teóricos de los conjuntos.

COBOR 2.0
SOFTWARE PARA EL CONTROL BORROSO

ABSTRACT

TITLE:

SOFTWARE TO THE FUZZY CONTROL BASED ON MODERN ALGORITHMS
- COBOR 2.0 –

AUTHORS:

ELKIN JAVIER MENDEZ
OSCAR MAURICIO ORDOÑEZ CAMARGO.

KEY WORDS:

Fuzzy Control, Fuzzy Controller based on modern Algorithms, Design of Fuzzy Controllers, Support For the design of the Fuzzy Control.

CONTENT:

In the systems of control, they present processes in which controllers designed according to other bosses (classic, adaptable.) they do not give the wished results. It is a question of processes of high complexity that in the majority of the cases, human skillful and experienced operators reach satisfactory results. When it is investigated to come from these experts, one concludes at once that, so much the information brings over of the state of the process as their control actions, are processed and managed in a qualitative way, using their intuition, heuristic ability and experience.

The importance of the Fuzzy logic from the point of view of the theory of the process control, is it of providing support when there wants to be translated the experienced heuristic knowledge, expressed in imprecise linguistic sentences to numerical algorithms. Quite based on the theory of sets that facilitates to imitate the behavior of the human logic. The "Fuzzy" term comes from the English word "fuzzy" that it means: confused, diffuse, indefinite.

COBOR 2.0 is a new software for the systems of control, which uses the fuzzy logic as an adaptative technology. By means of the monitoring of the variables of control, the action decides to take for the controller on the inference of a base of rules of knowledge defined from the experience given by an expert in the process control, taking a formal support of the theoretical foundations of the sets.

INTRODUCCION

En la vida práctica como estudiantes y profesionales es importante conocer y rendir informes sobre diversos tópicos que nos ayudan a enriquecer nuestra capacidad intelectual. Este documento contiene el informe final del proyecto titulado “Software para el control borroso basado en Algoritmos Modernos COBOR 2.0”. Proyecto que ha surgido en el grupo GEMA de la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander, y se inscribe dentro del área de investigación, descubrimiento de conocimientos y tecnologías adaptativas que se encuentran en pleno desarrollo a nivel mundial.

El problema del control en los procesos de ingeniería ha sido uno de los factores fundamentales en el desarrollo de nuevas tecnologías de razonamiento aproximado, esto con el fin de reemplazar o en su defecto dar apoyo a los responsables del control de procesos. Dado que el diagnóstico de los expertos no siempre será el más acertado en procesos de este tipo, el objetivo es crear una herramienta capaz de generar una aproximación a la solución en los casos donde se requiera una vasta experiencia y sapiencia de las personas encargadas de esta clase de procesos.

La lógica borrosa surge hoy como una opción para realizar control en los procesos de alta complejidad que se manejan en las diferentes áreas de las ingenierías, luego se pretende crear un software interdisciplinario, que se utilice en los diferentes campos de la ingeniería y que además muestre el proceso que se sigue para obtener la matriz de acciones que es el corazón del controlador Lógico Borroso.

1. PRESENTACIÓN DEL PROYECTO.

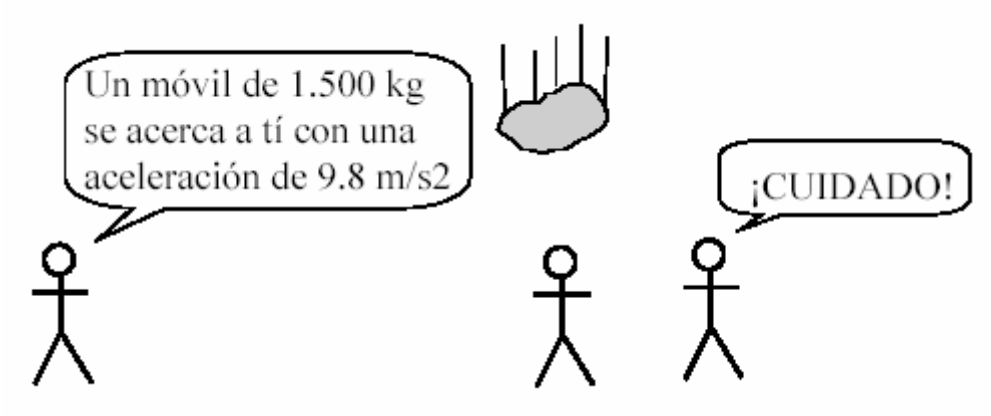
1.1 .JUSTIFICACIÓN.

En los últimos años se han incorporado sistemas de automatización que permiten rebajar costos y hacer sencillas las tareas para el operador de procesos complejos. En un principio surgieron los controladores convencionales, usados aún en la industria, presentando el problema que necesitan un modelo cuantitativo del proceso a automatizar, lo cual no siempre es posible, especialmente en problemas complejos, en los que el experto juega un papel muy importante usando su intuición, su habilidad heurística y su experiencia. Una de las técnicas efectivas para resolver este problema, es la *teoría de los conjuntos borrosos* que ha hecho posible establecer una modalidad del llamado "Control Inteligente".

La importancia de la lógica borrosa desde el punto de vista de la teoría del control de procesos, es la de proveer un soporte cuando se quiere traducir el conocimiento heurístico experimentado y expresado en frases lingüísticas imprecisas, a algoritmos numéricos.

La lógica borrosa es un área de la investigación, ya que realiza un trabajo conjugando la significancia y la precisión. En la figura 1 vemos un claro ejemplo de lo que significan estas dos palabras.

Figura 1 Significancia y precisión en el mundo real.



De las muchas formas de hacer funcionar la caja negra, frecuentemente la mejor opción es la lógica borrosa. Como alguna vez dijo Lotfi Zadeh, considerado el padre de la lógica borrosa: “En casi todos los casos se puede construir el mismo producto sin lógica borrosa, pero ésta es más rápida y barata”.

1.2 OBJETIVO

1.2.1 OBJETIVO GENERAL

Diseñar e implementar un software para el control de procesos, desarrollando la segunda versión del “Software de apoyo para el control inteligente basado en lógica borrosa COBOR 2.0”, para sistemas de múltiple entrada / salida basados en reglas borrosas tipo MANDAMI y tipo TSK (Takagi, Sugeno y Kang).

1.2.2 OBJETIVOS ESPECIFICOS

El prototipo Software COBOR 2.0 tendrá las siguientes características:

1. Brindar al usuario un ambiente de trabajo adecuado para el diseño, simulación e implementación de sistemas de control basado en la lógica borrosa, que permita trabajar cinco variables de control.

2. Permitir la definición de los conjuntos borrosos y sus particiones correspondientes a variables de estado y acción.

3. Generar un controlador representado por una matriz de control n – dimensional, donde n es el número de variables de estado y de acción a monitorear.

4. Ilustrar de manera didáctica y sistemática las operaciones borrosas que soportan la construcción de un sistema de inferencia borroso.

5. Facilitar la construcción de una base de conocimiento definida por reglas de la forma IF [Condición] THEN [Acción], donde la condición se forme con expresiones lógicas del tipo NOT, AND, OR, para todas las variables.

6. Implementar los sistemas de inferencia borrosa: Tipo MANDAMI Y Tipo TSK, considerando cinco funciones de pertenencia.

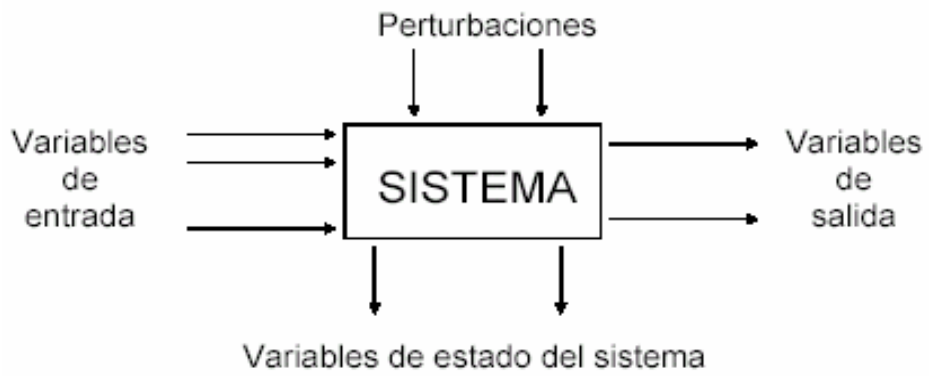
2. MARCO TEÓRICO.

2.1. MARCO TEÓRICO GENERAL.

2.1.1 SISTEMA DE CONTROL.

Un sistema de Control es un Conjunto o combinación de componentes que actúa conjuntamente y que cumple un determinado objetivo.

Figura 2 Sistema de Control.



Las ventajas de utilizar un sistema de control son las siguientes:

- Las técnicas de control automático tienen un campo prácticamente ilimitado de aplicación.

- Es útil contar con sistemas capaces de mantener todos los parámetros “controlados” sin la intervención humana.

- En ocasiones se consigue optimizar la evolución del proceso.
    - ✓ Situaciones de elevada complejidad
    - ✓ Situaciones en las que se debe operar en tiempo de respuesta corto
- Pueden eliminar fallas (distracciones, cansancio, tensión...).

2.1.2 CLASIFICACIÓN DE LAS TÉCNICAS DE CONTROL.

Entre las principales técnicas de control tenemos:

Sistemas de control continuo.

- Sistemas que operan con señales continuas.
- Normalmente la función de control se implementa con circuitos electrónicos.

Sistemas de control digital.

- Utilizan tecnología digital (Mayor flexibilidad en diseño).
- Un controlador digital para plantas continuas necesita conversión analógica-digital (y viceversa).
- “Problemas” de retardos y longitud de palabra inapreciables con las mejoras en la tecnología digital.
- Mayor capacidad para almacenar y manipular datos.
- Permite la inclusión de procesos de aprendizaje, control adaptativo, conocimiento experto y otros conceptos avanzados.

Sistemas de eventos discretos.

(Control secuencial, control lógico programable, control dinámico de eventos discretos).
Acciones de control determinadas como respuesta a las características secuénciales y combinaciones observadas de un conjunto de órdenes y condiciones sensoriales.

- Entradas y realimentación suelen ser binarias.
- Salida también suele ser binaria.
- Se diseñan mediante el desarrollo de una tabla de transición de estados.

2.1.2. SISTEMAS INTELIGENTES DE CONTROL.

Desarrollo de métodos de control para emular características importantes de la inteligencia humana como:

- Adaptación,
- Aprendizaje,
- Tratamiento de grandes cantidades de datos, y
- Tratamiento de incertidumbre.

Lo que es control inteligente hoy será simplemente control mañana; esto quiere decir que son áreas de límites cambiantes.

El informe Task Force on Intelligent Control define el control inteligente a través de varias propiedades propias de los sistemas inteligentes:

- Adaptación y aprendizaje: Capacidad para adaptarse a condiciones cambiantes.
- Autonomía e inteligencia: Habilidad para actuar adecuadamente en un entorno con incertidumbre.
- Estructuras y jerarquías: Arquitectura funcional apropiada para afrontar problemas complejos.

El objetivo es el de diseñar sistemas de control automático (controladores) que sean Robustos, Adaptables, con capacidad de aprendizaje de la experiencia y de la intervención humana y autónomos.

Concretamente se estudiarán los controladores borrosos y los modelos de aprendizaje neuronal y evolutivo.

Definiciones más utilizadas:

- Sistema: Combinación de componentes que actúan juntos y realizan un objetivo determinado.
- Variable controlada: Condición que se mide y controla.
- Variable manipulada: Condición que el controlador modifica para afectar el valor de la variable controlada.
- Controlar: Medir la variable controlada y alterar la variable manipulada para corregir o limitar la variable controlada.
- Planta: Parte del sistema que se controla.
- Proceso: “Operación continua, marcada por cambios graduales que se suceden uno a otro de una forma relativamente fija y que conduce a un resultado determinado”.
- Perturbación: Señal que tiende a afectar negativamente el valor de la salida de un sistema.

2.1.3 CONTROL INTELIGENTE.

Entre los sistemas de control inteligente que existen hoy en día encontramos tres clases de control:

- Sistemas Neuro-Borrosos.
- Sistemas Genético-Borrosos.
- Sistemas basados en reglas borrosas.

2.1.3.1 SISTEMAS DE CONTROL NEURO-BORROSOS.

Las redes neuronales artificiales y la lógica borrosa son dos de los avances científicos recientes más importantes del conocimiento. Mediante estas herramientas se ha intentado simular dos de las características más importantes con las que cuenta el cerebro humano: la capacidad de aprendizaje y el poder procesar información incompleta o que no es

precisa. Estas técnicas se han podido utilizar para la solución de problemas tanto científicos como de la vida diaria. Con una buena combinación de estas dos técnicas es posible un desarrollo tecnológico más amplio en diversos campos, desde la medicina hasta el mejoramiento de electrodomésticos.

2.1.3.1.1 REDES NEURONALES.

Introducción.

Mediante esta técnica se intenta imitar el proceso de aprendizaje del cerebro humano. El cerebro esta formado por miles de millones de neuronas conectadas entre sí. Utiliza una información que es percibida, transmitida hasta las neuronas y allí es procesada por ellas para dar una respuesta a cada uno de los diferentes estímulos. Cada neurona tiene tres partes: un cuerpo celular, una estructura de entrada (Dendrita), y una de salida (Axon). La mayoría de los terminales de los axones se conectan con las dendritas de otras neuronas (Sinapsis). El comportamiento de una neurona es el siguiente: recibe una señal de entrada con una fuerza determinada, dependiendo de ellas la neurona emite una señal de respuesta, las sinapsis pueden variar en fuerza, algunas pueden enviar una señal fuerte y otras débiles. A una neurona pueden llegar miles de señales de entrada, cada una con una fuerza o peso diferente. Matemáticamente el comportamiento de una neurona puede representarse por una lista de sus señales de entrada que son multiplicadas por sus pesos respectivos y posteriormente sumados, el resultado es llamado nivel de activación de la neurona del que depende la señal de salida que es enviada a cada una de las neuronas que están conectadas a ella.

Una red neuronal artificial (RNA) es un sistema compuesto de muchos sistemas procesadores simples conectados en paralelo, cuya función es determinada por la estructura de la red, la fuerza en las conexiones y el procesamiento realizado por los elementos en los nodos (Jang 1997). Las RNA, igual que las personas, aprenden de la experiencia.

Referencia Histórica.

Mcculloch y Pitts (1943)

- Propusieron un modelo de una neurona.
- Modelo de McCulloch-Pitts, Hebb.
- Se presenta el funcionamiento de la regla de aprendizaje fisiológico mediante modificación sináptica.

Rosenblatt (1958)

- Se introduce un nuevo enfoque al problema de reconocimiento de patrones; perceptron.

Widrow y Hoff (1960)

- Se introduce el algoritmo de mínimos cuadrados (LMS).
- Se formula el Adaline (elemento lineal adaptativo).

Hopfield (1982)

- Se usa la idea de la función de energía para formular el cálculo realizado por redes recurrentes.
- Redes estables dinámicas.

Barto, Sutton y Anderson (1983)

- Se propone el Aprendizaje por Refuerzo.

Rumelhart, Hinton, Williams (1986)

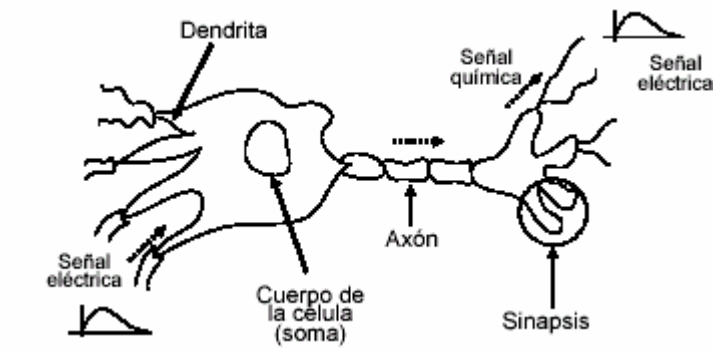
- Se propone el algoritmo de retro-propagación (backpropagation)

Broomhead y Lowe (1988).

- Se propone la Red de Funciones de Base Radial (radial basis function, RBF).

Neurona Biológica.

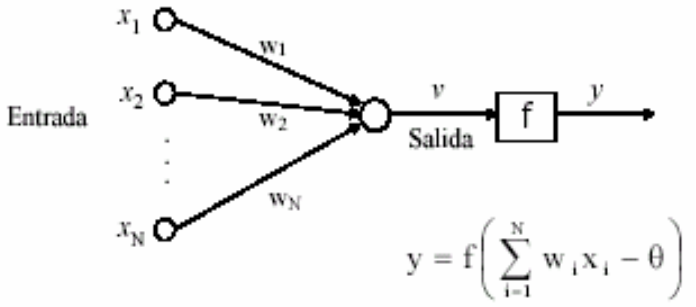
Figura 3 Neurona Biológica.



El cerebro humano contiene aproximadamente 1.5×10^{10} neuronas y cada neurona recibe señales de 104 sinapsis. Esta es la metáfora que se quiere implementar en los modelos de redes neuronales artificiales.

Modelo de una Neurona Artificial.

Figura 4 Modelo de una Neurona artificial.



Los siguientes son los componentes de una neurona artificial:

- Un conjunto de enlaces de entrada desde otros nodos
- Un conjunto de pesos que ponderan cada entrada
- Una salida.
- Una función de activación, generalmente no lineal.

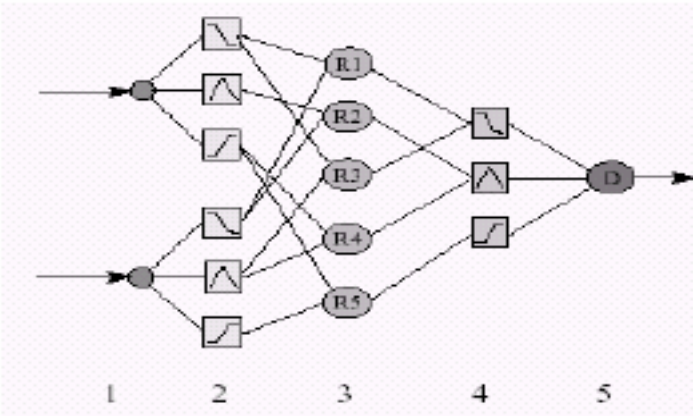
2.1.3.1.2 SISTEMAS NEURO - BORROSOS.

Son sistemas híbridos que combinan técnicas de redes neuronales y sistemas de inferencia borrosa. Asociando de este modo la capacidad de aprendizaje de las RNA con la tolerancia a fallos, interpretabilidad y robustez de los sistemas borrosos. Además permiten la integración de conocimiento (métodos previos, expertos, etc.) siendo posible extraer el conocimiento incluido en la RNA en formato de reglas borrosas (por eso son considerados modelos de “caja gris”) [6].

La arquitectura más común consta de 5 capas:

1. Entradas
2. Emborronado.
3. Reglas.
4. Consecuentes.
5. Desemborronado.

Figura 5 Sistema Neuro – Borroso.



2.1.3.1.2.1 LIMITACIONES DE LOS SISTEMAS NEURO – BORROSOS.

Entre las principales limitaciones de los sistemas Neuro -Borrosos tenemos:

- Número pequeño de entradas (curso de la dimensionalidad: crecimiento geométrico de la complejidad según el número de entradas).
- Dificultad para aprender la estructura de las reglas. Generalmente, sólo aprenden la forma de las funciones de pertenencia y los coeficientes del consecuente (en TSK).
- Dificultad para tratar funciones no diferenciables.
- Problemas de convergencia: caída en óptimos locales.
- Problemas de sobre aprendizaje: Error de aproximación (conjunto de entrenamiento) mucho menor que el de generalización (conjunto de validación).

Algunos ejemplos de Sistemas Neuro -Borrosos:

NEFCLASS (D. Nauck, 1994)

- Particionamiento de rejilla.
- Reglas de clasificación. FSOM (P. Vuorimaa, 1996)
- Particionamiento de grafos borrosos (SBRD aproximativo). NFH (F.J. de Soutza, 1997)
- Particionamiento jerárquico.

ANFIS.

Adaptive Network based Fuzzy Inference System (Jyh-Shing Roger Jang, 1993).

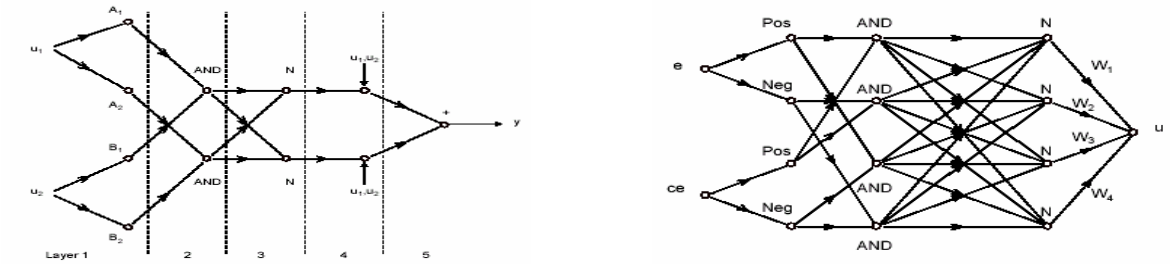
- Utiliza variables lingüísticas (particionamiento de rejilla).
- Únicamente ajusta las funciones de pertenencia.
- Apto sólo para reglas TSK de orden 0 (el consecuente es un número real) ó 1 (el consecuente es un polinomio de primer grado).

Entrenamiento en dos pasos:

- Fijar el consecuente y ajustar los parámetros del antecedente (funciones de pertenencia) mediante Gradiente Descendente.
- Fijar el antecedente y ajustar los parámetros del consecuente (coeficientes de los polinomios) mediante Optimización por Mínimos Cuadrados.

Estructura de Anfis.

Figura 6 Estructura de Anfis.



2.1.4 SISTEMAS DE CONTROL GENÉTICO- BORROSOS.

Hoy en día las aplicaciones de sistemas de control borroso apoyados con algoritmos genéticos están alcanzando niveles muy altos de aceptación debido a su gran potencialidad y bajo costo computacional. De ahí que se esté presentando un gran fenómeno de personas interesadas en investigar acerca de las grandes ventajas y virtudes que estos sistemas pueden brindar en un futuro.

2.1.4.1 INTRODUCCIÓN SOBRE ALGORITMOS GENÉTICOS.

Los Algoritmos Genéticos (GA) fueron introducidos por John Holland en 1970, inspirándose en el proceso observado en la evolución natural de los seres vivos. Los biólogos han estudiado con detenimiento los mecanismos de la evolución y aunque quedan términos por entender, muchos aspectos están bastante explicados. De manera muy general podemos decir que en la evolución de los seres vivos el problema al que cada individuo se enfrenta cada día es la supervivencia. Para ello cuenta con las habilidades innatas provistas en su material genético. En el ámbito de los genes, el problema es buscar aquellas adaptaciones beneficiosas en un medio hostil y cambiante. Debido en parte a la selección natural, cada especie gana una cierta cantidad de "conocimiento", que es incorporado a la información de sus cromosomas. Así pues, la

evolución tiene lugar en los cromosomas, en donde está codificada la información del ser vivo. La información almacenada en el cromosoma varía de unas generaciones a otras. En el proceso de formación de un nuevo individuo, se combina la información cromosómica de los progenitores aunque la forma exacta en que se realiza es aún desconocida.

Aunque muchos aspectos están todavía por discernir, existen unos principios generales ampliamente aceptados por la comunidad científica. Algunos de estos son:

- La evolución opera en los cromosomas en lugar de los individuos a los que representan.
- La selección natural es el proceso por el que los cromosomas con "buenas estructuras" se reproducen más a menudo que los demás.
- En el proceso de reproducción tiene lugar la evolución mediante la combinación de los cromosomas de los progenitores. Se denomina Recombinación a este proceso en el que se forma el cromosoma del descendiente. También son de tener en cuenta las mutaciones que pueden alterar dichos códigos.
- La evolución biológica no tiene memoria en el sentido que en la formación de los cromosomas únicamente se considera la información del período anterior.

Los algoritmos genéticos establecen una analogía entre el conjunto de soluciones de un problema y el conjunto de individuos de una población natural, codificando la información de cada solución en un string (vector binario) a modo de cromosoma. En palabras del propio Holland:

"Se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de "evolución simulada".

A tal efecto se introduce una función de evaluación de los cromosomas, que se llama calidad ("fitness") y que está basada en la función objetivo del problema. Igualmente se

introduce un mecanismo de selección de manera que los cromosomas con mejor evaluación sean escogidos para "reproducirse" más a menudo que los que la tienen peor.

Los algoritmos desarrollados por Holland inicialmente eran sencillos pero dieron buenos resultados en problemas considerados difíciles [10].

Computación Evolutiva

Está compuesta por modelos de evolución basados en poblaciones cuyos elementos representan soluciones a problemas. La simulación de este proceso en un ordenador resulta ser una técnica de optimización probabilística, que con frecuencia mejora a otros métodos clásicos en problemas difíciles.

Existen cuatro paradigmas básicos:

Algoritmos Genéticos: que utilizan operadores genéticos sobre cromosomas.
Estrategias de Evolución: que enfatizan los cambios de comportamiento al nivel de los individuos.
Programación Evolutiva: que enfatizan los cambios de comportamiento al nivel de las especies.
Programación Genética: que evoluciona expresiones representadas como árboles.

Dominios De Aplicación.

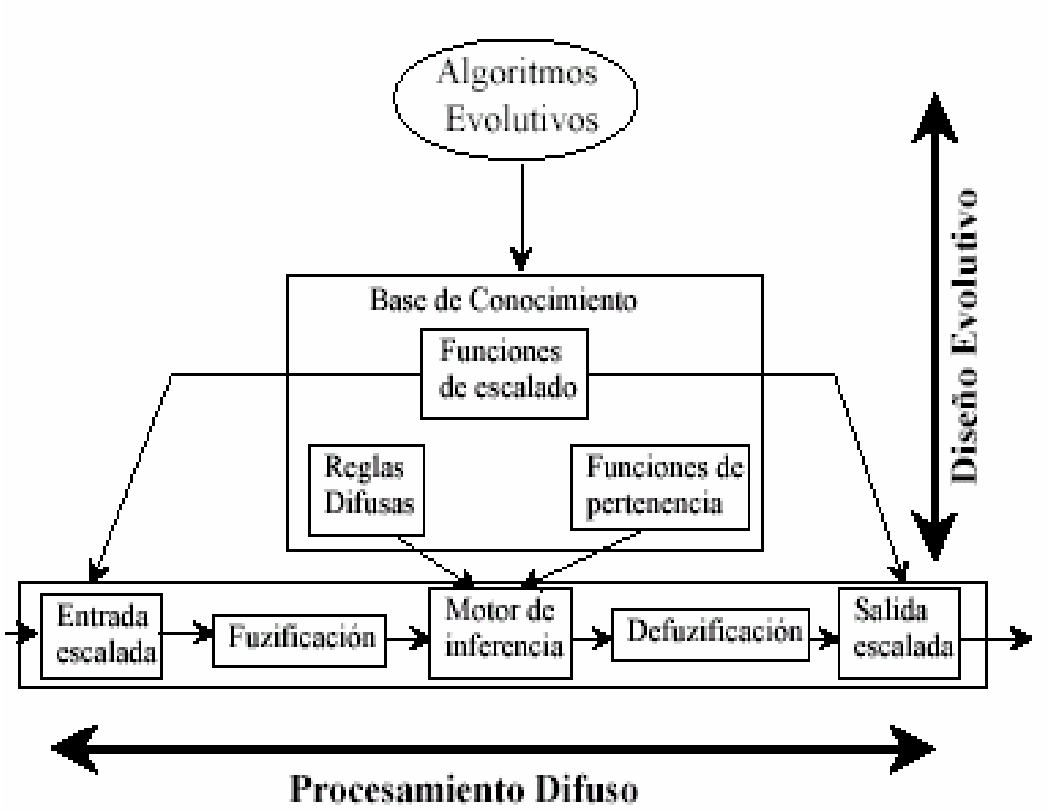
- Optimización combinatoria y en dominios reales.
- Modelado e identificación de sistemas.
- Planificación y control.
- Ingeniería.
- Vida artificial.
- Aprendizaje y minería de datos.
- Internet y Sistemas de Recuperación de Información.

2.1.4.2 SISTEMAS GENÉTICO-BORROSOS.

Diseño.

En la actualidad se están desarrollando aplicaciones de sistemas que integran los algoritmos genéticos con sistemas basados en reglas borrosas, dando excelentes resultados en las aplicaciones ejecutadas a través de ellos.

Figura 7 Diseño de un Sistema Genético Borroso.



Tipos de Sistemas basados en reglas Genéticos-borrosas:

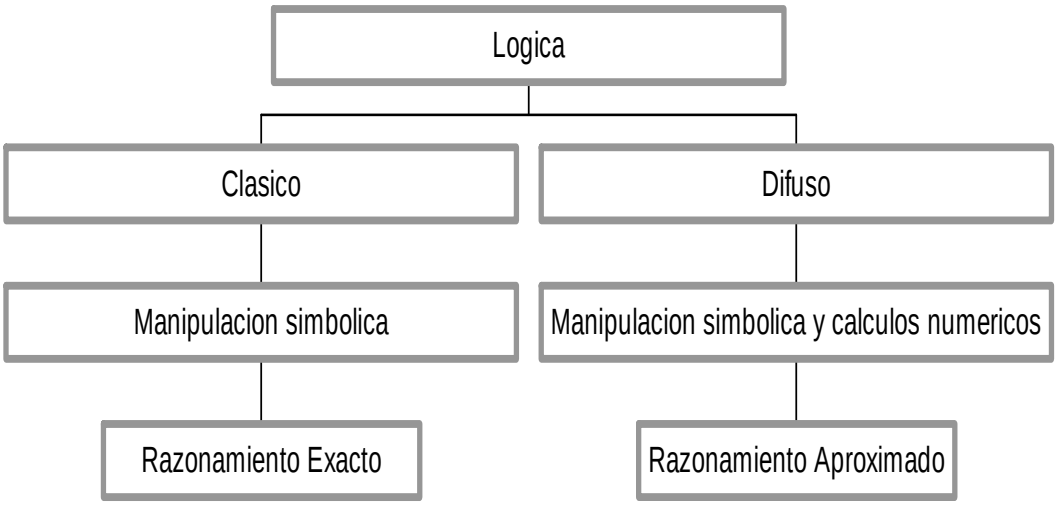
- Sistemas con ajuste genético de la Base de Datos
- Sistemas con aprendizaje genético de la Base de Reglas
- Sistemas con aprendizaje genético de la Base de Conocimiento
- Sistemas con aprendizaje genético del Mecanismo de Inferencia (Poco usuales)

2.2 MARCO TEÓRICO ESPECÍFICO

2.2.1 LOGICA CLÁSICA vs. LOGICA BORROSA

Aunque existen mucha similitud entre la lógica tradicional y la lógica borrosa en la parte operacional (Figura 8), también existen grandes diferencias que han dado pie para nuevos aportes que han significado grandes avances en todos los campos de la ingeniería.

Figura 8 Lógica Clásica vs. Lógica Borrosa.



La Lógica Borrosa es una técnica de la inteligencia computacional que permite trabajar información con alto grado de imprecisión. En esto se diferencia de la lógica convencional que trabaja con información bien definida y precisa.

El concepto de Lógica Borrosa fue concebido por Lofti Zadeh, profesor de la Universidad de California en Berkeley, quien inconforme con los conjuntos clásicos (“crisp-sets”) que sólo permiten dos opciones, la pertenencia o no de un elemento a dicho conjunto, la presentó como una forma de procesar información permitiendo pertenencias parciales a unos conjuntos, que en contraposición a los clásicos los denominó Conjuntos Borrosos

(“fuzzy-sets”). El concepto de conjunto borroso fue expuesto por Lofti Zadeh en un artículo hoy clásico en la literatura de la lógica borrosa, en el año de 1965 titulado "Fuzzy Sets" publicado en la revista “Information and Control”. El mismo Zadeh publica en 1971 el artículo, "Quantitative Fuzzy Semantics", en donde introduce los elementos formales que acabarían componiendo el cuerpo de la doctrina de la lógica borrosa y sus aplicaciones tal como se conocen en la actualidad.

Pocos años después (1974), el británico Ebrahim Mandami demuestra la aplicabilidad de la lógica borrosa en el campo del control, desarrollando el primer sistema de control borroso (Fuzzy Control) práctico, la regulación de un motor de vapor. Las aplicaciones de la lógica borrosa en el control no se pudieron implementar con anterioridad a estos años debido a la poca capacidad de cómputo de los procesadores de la época.

2.2.1.1 LOGICA CLÁSICA DE CONJUNTOS.

Historia.

La lógica de conjuntos fue creada por George Cantor, aunque George Boole dio los primeros pasos en su libro “Investigations of the laws of thought”. El concepto de infinito fue tratado por Zenón de Elea y sus célebres paradojas. Cantor publicó varios artículos entre 1867 y 1871 sobre teoría de números de gran calidad, pero nada indicaba que su autor cambiaría el curso de la matemática. Tiempo después empezó a trabajar en series trigonométricas y aquí aparecen las primeras ideas sobre teoría de conjuntos. Cantor demostró que los números reales algebraicos se podían poner en correspondencia uno a uno con los números naturales pero que esto no se podía hacer con los números reales (que incluyen, además de los reales algebraicos, los transcendentes). El primer intento de axiomatizar la Teoría de Conjuntos la hizo Zermelo en 1908. Después lo intentaron Fraenkel, Von Neumann, Bernays y Godel. Godel mostró las limitaciones de cualquier teoría axiomática.

La mejor prueba de que la teoría de conjuntos no ha logrado unificar a las matemáticas es que éstas se han ramificado en áreas muy diferenciadas, como la aritmética, el álgebra, la

trigonometría y geometría. También se han separados distintos campos como el cálculo, la topología, la teoría de conjuntos, la teoría de los números y la estadística.

2.2.1.1.1 CONJUNTOS CLÁSICOS.

Notación.

Ordinariamente se usan letras mayúsculas para representar los conjuntos que incluiremos sus elementos dentro de llaves separados por comas, {}. El símbolo $\in$ significa (es elemento de). Análogamente, $\notin$ significa (no es elemento de).

Conjuntos Iguales.

Se dice que dos conjuntos S y T son iguales si cada elemento de S es elemento de T y viceversa. Se escribe S = T. Se usa el signo de igualdad para indicar que dos símbolos representan al mismo conjunto.

Conjuntos Vacíos.

Un conjunto sin elementos recibe el nombre de conjunto vacío o conjunto nulo y se representa por [ ] o por $\varnothing$. Es útil tener el concepto de un conjunto sin elemento.

Subconjuntos.

Se dice que un conjunto S es subconjunto T, si todos los elementos de S los son T. El símbolo $\subseteq$ se lee (es subconjunto de).

Conjuntos Equivalentes.

Cuando los elementos de un conjunto se corresponden con los de un segundo conjunto de modo que cada elemento de cada conjunto tenga uno, y solo uno, asociado en el otro conjunto, decimos que hay una correspondencia uno a uno entre ambos conjuntos.
Dos conjuntos que se pueden poner en correspondencia uno a uno entre sí, se dice que son equivalentes. Si A es equivalente a B, se escribe A ~ B.

Cardinalidad de un Conjunto.

Contar es el proceso por el cual ponemos en correspondencia los elementos de un conjunto con algún subconjunto propio de N, comenzando con 1 y usando los elementos de N en orden y sin saltar ninguno. Un subconjunto así se llama subconjunto estándar de N. Por ejemplo, el subconjunto estándar de N = {1,2,3,4} es equivalente a = {a,b,c,d} Decimos entonces que S tiene cuatro elementos. Esto lleva a la siguiente definición: "Cuando un conjunto S se equipara con un subconjunto estándar de N, el ultimo elemento de N usado se llama cardinalidad del conjunto S y se denota por n (S)".

Dos números enteros no negativos m y n, son iguales si ambos son la cardinalidad del mismo conjunto o de conjuntos equivalentes. En tal caso, escribimos m = n.

Conjuntos Finitos e Infinitos.

Si es posible encontrar un subconjunto estándar de N que se puede hacer corresponder uno a uno con un conjunto dado S, o si S es el conjunto vacío, se dice que S es finito. En caso contrario, se dice que es infinito. Nótese que, sin embargo, si hay un equiparamiento de N con uno de sus subconjuntos, que no es un subconjunto estándar como el conjunto de los números pares, vemos que N se puede poner en correspondencia con un subconjunto propio de sí mismo. Esto solo se puede hacer en un conjunto infinito.

CONCEPTOS BÁSICOS SOBRE LOS CONJUNTOS CLÁSICOS.

Un Conjunto es cualquier colección de objetos el cual puede ser tratado como una entidad y un objeto de la colección se dice que es un elemento o miembro del conjunto. Dado un objeto x, y un conjunto S, si x es un elemento del conjunto S, lo podemos escribir como "x $\in$ S"; si x no es un elemento del conjunto S, podemos escribirlo como "$\neg$(x $\in$ S)" o también "x $\notin$ S". Los términos conjunto, colección y clase son usados como sinónimos, así como también los términos elemento o miembro.

Casi cualquier cosa puede ser tratada como un conjunto, viéndolo desde un punto de vista muy matemático, lo que se trata de ilustrar con los siguientes ejemplos.

- ✓ El conjunto de enteros no negativos menores que 4. Éste es un conjunto finito con cuatro miembros: 0, 1, 2 y 3.
- ✓ El conjunto de enteros mayores que 3. Como es de suponerse, éste se trata de un conjunto infinito y no hay ninguna dificultad para definir cualquiera de los miembros de éste conjunto.

Dado que un conjunto es caracterizado por sus miembros, un conjunto puede ser especificado por declaración cuando un objeto está en el conjunto. Un conjunto finito puede ser especificado explícitamente por una lista de sus elementos. Los elementos de la lista deben ser separados por comas y la lista encerrada en llaves { }, como lo muestran los siguientes ejemplos:

- ✓ El conjunto que contiene los elementos A, B y C está denotado por {A, B, C}.

Los elementos de un conjunto infinito no pueden ser listados explícitamente; en consecuencia, necesitamos una forma para describirlos implícitamente. La especificación implícita frecuentemente es hecha por el significado de predicados con una variable libre. El conjunto es definido de manera que los elementos del universo establecido por el conjunto hagan el predicado verdadero. De aquí, si P (x) es un predicado con una variable

libre, el conjunto {x | P(x)} denota el conjunto S tal que c $\in$ S, si y sólo si, P(c) es verdadero.

Si un conjunto es finito pero muy largo como para listarse fácilmente o si es un conjunto infinito, las elipses suelen ser usadas para especificar implícitamente un conjunto. Las siguientes especificaciones usan elipses para caracterizar una lista de los elementos de un conjunto.

- ✓ El conjunto de enteros del 1 al 50 es especificado por {1, 2, 3,…, 50}
- ✓ El conjunto de enteros pares no negativos es especificado por {0, 2, 4, 6,…}

En un desarrollo más formal de la teoría de conjuntos, el siguiente axioma es usado para establecer que los conjuntos son completamente especificados por sus elementos. El axioma nos sirve como una definición de igualdad de conjuntos.

Axioma de Extensión: Dos conjuntos A y B son iguales si y sólo si tienen los mismos elementos.
El axioma de extensión puede ser expresado en notación lógica de dos maneras:

$$A = B \Leftrightarrow \forall x\, [x \in A \Leftrightarrow x \in B\ ]$$

$$A = B \Leftrightarrow \{\ \forall x\, [\ x \in A \Rightarrow x \in B\ ]\ \wedge \forall x\, [\ x \in B \Rightarrow x \in A\ ]\}$$

El axioma de extensión declara que si dos conjuntos tienen los mismos elementos, aún sin considerar como están especificados, son iguales. Es decir, si un conjunto es especificado explícitamente con una lista, el orden en el que esté listado es irrelevante. Por ejemplo: el conjunto denotado por {A, B, C} es el mismo que (igual a) el conjunto denotado por {C, B, A} y {B, C, A}. Además, no importa el número de veces que aparezca un elemento en el conjunto: {A, B, A}, {A, B} y {A, A, A, B, B}

Son diferentes especificaciones de un mismo conjunto. Un conjunto finito puede ser caracterizado implícita o explícitamente, como lo demuestran los conjuntos:

$$\{1, 2, 3, 4, 5\} \text{ y } \{x \mid x \in \mathbf{I} \wedge 1 \leq x \leq 5\}$$

Que son el mismo conjunto.

2.2.2 LOGICA BORROSA.

Es una lógica basada en la teoría de conjuntos que posibilita imitar el comportamiento de la lógica humana. El término "borroso" procede de la palabra inglesa "fuzzy" que significa "confuso, difuso, indefinido o desenfocado".

La lógica borrosa es una rama de la inteligencia artificial que se fundamenta en el concepto "Todo es cuestión de cuanto se cumple", lo cual permite manejar información vaga o de difícil especificación. Es entonces posible con la lógica borrosa gobernar un sistema por medio de reglas de 'sentido común', las cuales se refieren a cantidades indefinidas.

Las reglas involucradas en un sistema borroso pueden ser desarrolladas con sistemas adaptativos, que aprenden al ' observar ' como las personas operan los dispositivos, o simplemente ser formuladas por un experto humano. En general, la lógica borrosa se aplica tanto a sistemas de control como para modelar cualquier sistema continuo de ingeniería, física, biología o economía, definiendo así un sistema matemático que modela funciones no lineales, convirtiendo unas entradas en salidas acordes con los planteamientos lógicos que usan el razonamiento aproximado.

Se fundamenta en los denominados conjuntos borrosos y un sistema de inferencia borroso basado en reglas de la forma " SI..... ENTONCES.....", donde los valores

lingüísticos de la premisa y el consecuente están definidos por conjuntos borrosos, es así como las reglas siempre convierten un conjunto borroso en otro.

La lógica borrosa es más flexible que la lógica clásica o bivaluada; define la realidad en diferentes grados de verdad siguiendo patrones de razonamiento similares a los del pensamiento humano. Permite también cierta "imprecisión" en la representación de un problema y aún así llegar a una muy buena solución, maneja la incertidumbre y la imprecisión, también se reconocen más que simples valores verdaderos y falsos y las proposiciones pueden ser representadas con grados de veracidad o falsedad. Por ejemplo, la sentencia "hoy es un día soleado" puede ser 100% verdad si no hay nubes, 80% verdad si hay pocas nubes, 50% verdad si existe neblina y 0% si llueve todo el día.

La Lógica Borrosa ha sido probada para ser particularmente útil en sistemas expertos y otras aplicaciones de inteligencia artificial. Es también utilizada en algunos correctores de voz para sugerir una lista de probables palabras a reemplazar un término erróneo. La Lógica Borrosa, que hoy en día se encuentra en constante evolución, nació en los años 60 como la lógica del razonamiento aproximado y en ese sentido podía considerarse una extensión de la Lógica Multivaluada.

Psicológicamente la lógica borrosa puede definirse como la resolución de problemas que implica cierto grado de inferencia e intuición para lograr la conclusión propia; vista como una distinción crucial entre la inteligencia humana y la mecánica. Desde el punto de vista de la inteligencia artificial, es un método de razonamiento de maquina similar al pensamiento humano que puede procesar información incompleta o incierta, característico de muchos sistemas expertos. Un modelo de lógica borrosa consiste básicamente en que cada variable se asocia en cierto grado a alguna categoría particular y en cierto grado a otras. Por ejemplo, si el número 1 es considerado "pequeño" y el número 10 "grande", ¿Entonces como se considera el número 6? Podría ser considerado como “mediano", o en realidad también como "grande" si lo miramos con relación al número 1. Si nos fijamos bien, el número 6 tiene algo de grande, pero también algo de mediano. Es decir, su definición es "borrosa".

Los sistemas de control borrosos tienen una gran variedad de aplicaciones que van desde la estimación de parámetros, toma de decisiones, sistemas mecánicos de control tales como el aire acondicionado o lavadoras automáticas, hasta el control de automóviles o casas "inteligentes". Las nociones como "más bien caliente" o "poco frío" pueden formularse matemáticamente y ser procesados por computadoras. De esta manera, se ha realizado un intento de aplicar una forma más humana de pensar en la programación de computadoras [2].

2.2.2.1 CONJUNTOS BORROSOS.

Historia

Los conjuntos borrosos fueron introducidos por primera vez en 1965. En cierto nivel, la disciplina de la lógica borrosa puede ser vista como un lenguaje que permite trasladar sentencias sofisticadas en lenguaje natural, a un lenguaje matemático formal. Mientras la motivación original fue ayudar a manejar aspectos imprecisos del mundo real, la práctica temprana de la lógica borrosa permitió el desarrollo de aplicaciones prácticas. Aparecieron numerosas publicaciones que presentaban los fundamentos básicos con aplicaciones potenciales. Esta fase marcó una fuerte necesidad de distinguir la lógica borrosa de la teoría de probabilidad. Tal como la entendemos ahora, la teoría de conjuntos borrosos y la teoría de probabilidad tienen diferentes tipos de incertidumbre.

En 1994, la teoría de la lógica borrosa se encontraba en la cumbre, pero esta idea no es nueva. Para muchos, estuvo bajo el nombre de lógica borrosa durante 25 años, pero sus orígenes se remontan hasta 2,500 años. Aún Aristóteles consideraba que existían ciertos grados de veracidad y falsedad. Platón había considerado ya grados de pertenencia.

En el siglo XVIII, el filósofo y obispo anglicano Irlandés George Berkeley y David Hume describieron que el núcleo de un concepto atrae conceptos similares. Hume en particular,

creía en la lógica del sentido común, el razonamiento basado en el conocimiento que la gente adquiere en forma ordinaria mediante vivencias en el mundo. En Alemania, Emmanuel Kant, consideraba que sólo los matemáticos podían proveer definiciones claras y muchos principios contradictorios no tenían solución. Particularmente la escuela americana de la filosofía llamada pragmatismo, fundada a principios de siglo por Charles Sanders Pierce, cuyas ideas se fundamentaron en estos conceptos, fue la primera en considerar "vaguedades", más que falso o verdadero, como forma de acercamiento al mundo y a la forma en que las personas funcionan.

La idea de que la lógica produce contradicciones fue popularizada por el filósofo y matemático británico Bertrand Russell a principios del siglo XX. Estudió las vaguedades del lenguaje, concluyendo con precisión que la vaguedad es un grado. El filósofo austriaco Ludwing Wittgenstein estudió las formas en las que una palabra puede ser empleada para muchas cosas que tienen algo en común. La primera lógica de vaguedades fue desarrollada en 1920 por el filósofo Jan Lukasiewicz. Visualizó los conjuntos con un posible grado de pertenencia con valores de 0 y 1, después los extendió a un número infinito de valores entre 0 y 1. En los años sesentas, Lotfi Zadeh inventó la lógica borrosa, que combina los conceptos de la lógica y de los conjuntos de Lukasiewicz mediante la definición de grados de pertenencia [3].

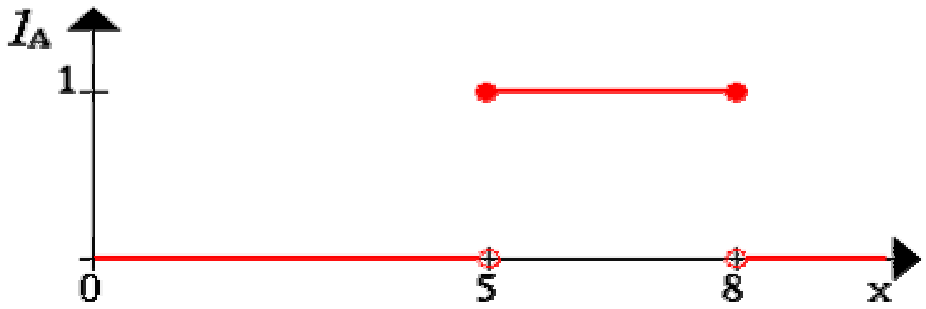
Notación.

En primer lugar, se considera un conjunto X con todos los números reales entre 0 y 10 llamado el universo de discurso. Ahora, se define un subconjunto A de X con todos números reales en el rango entre 5 y 8.

$$A = [5,8]$$

Ahora se muestra el conjunto A por su función característica, es decir esta función asigna un número 1 ó 0 al elemento en X, dependiendo de si el elemento está en el subconjunto A ó no. Esto conlleva a la figura siguiente:

Figura 9 Función Característica de un conjunto clásico.



Se podrán interpretar los elementos que han asignado el número 1 como los elementos que están en el conjunto A y los elementos que han asignado el número 0 como los elementos que no están en el conjunto A.

Este concepto es suficiente para muchas áreas de aplicación. Pero se podrán encontrar fácilmente situaciones donde carece de flexibilidad, como por ejemplo:

Para describir el conjunto de gente joven, formalmente se denota como:

B = {conjunto de gente joven}

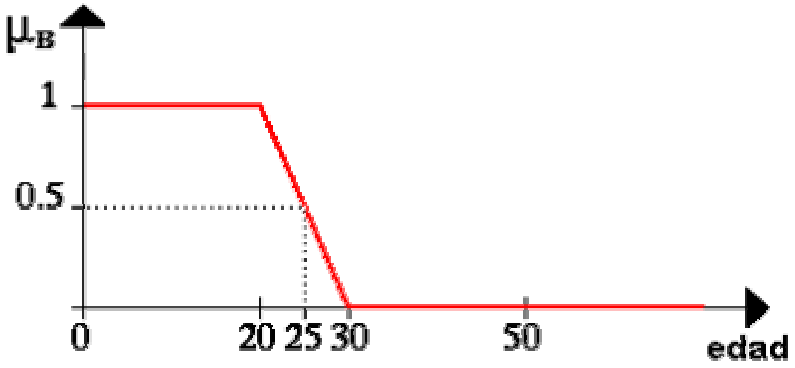
Como en general la edad comienza en 0, el rango más inferior de este conjunto está claro. El rango superior, por otra parte, es más bien complicado de definir. Como un primer intento se establece el rango superior en 20 años. Por lo tanto, B se define como un intervalo denominado:

B = [0,20]

Ahora la pregunta es: ¿por qué alguien es en su 20 cumpleaños joven y al día siguiente no? Obviamente, este es un problema estructural, porque si se mueve el límite superior del rango desde 20 a un punto arbitrario se podrá plantear la misma pregunta.

Una manera más natural de construir el conjunto B estaría en suavizar la separación estricta entre el joven y el no joven. Esto se hará para permitir no solamente la “crispada” decisión: "él / ella SI está en el conjunto de gente joven" o "él / ella NO está en el conjunto de gente joven", sino también frases más flexibles como: "él / ella SI pertenece un poquito más al conjunto de gente joven" o "él / ella NO pertenece aproximadamente al conjunto de gente joven".

Figura 10 Conjunto Gente joven.



De lo anterior se llega a un concepto a priori: “Se pueden usar conjuntos borrosos para hacer computadoras más sabias”. Resta codificar la idea más formalmente. En el primer ejemplo se codifican todos los elementos del Universo de Discurso con 0 o 1. Una manera de generalizar este concepto está en permitir más valores entre 0 y 1. De hecho, permitiendo infinitas alternativas entre 0 y 1, denominando el intervalo de unidad Y_o = [0,1].

La interpretación de los números ahora asignados a todos los elementos del Universo de Discurso es algo más difícil. Por supuesto, el número 1 asignado a un elemento significa que el elemento está en el conjunto B y 0 significa que el elemento no está definitivamente en el conjunto B. El resto de valores significan una pertenencia gradual al conjunto B.

De esta forma, 25 años de edad todavía sería joven al grado de 50 por ciento.

2.2.2.2 OPERACIONES CON CONJUNTOS BORROSOS.

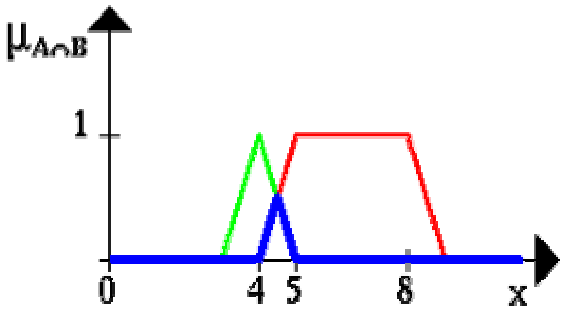
Para operar con conjuntos borrosos se extienden las operaciones con conjuntos clásicos:

Igualdad $A = B \Leftrightarrow \forall x \in X : \mu_A(x) = \mu_B(x)$

Inclusión $A \subseteq B \Leftrightarrow \forall x \in X : \mu_A(x) \leq \mu_B(x)$

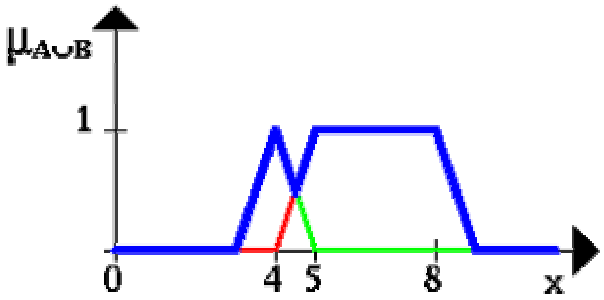
Intersección $A_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$.

Figura 11 Intersección de conjuntos Borrosos.



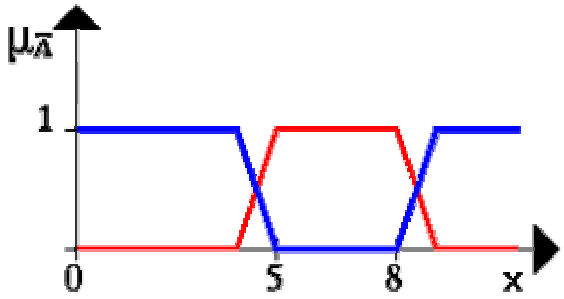
Unión $A_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$

Figura 12 Unión de Conjuntos Borrosos.



Complemento $A_{\bar{A}}(x) = 1 - \mu_A(x)$

Figura 13 Complemento de un Conjunto Borroso.



2.2.3 SISTEMAS DE CONTROL BORROSO.

Los diseñadores de sistemas de control han buscado incesantemente nuevos caminos para la solución del problema del control de procesos complejos. Esto se da en los casos en que no existe un modelo preciso del proceso y la información apriorística acerca de este sea esencialmente cualitativa.

Por otro lado, el problema de control constituye uno típico de toma de decisiones y para este tipo de problemas la creación de la Lógica borrosa significó la aparición de una nueva y poderosa herramienta para enfrentarlo en situaciones de imprecisión esencial en modelos, información, objetivos, restricciones, y acciones de control.

En el campo específico de los sistemas de control, es frecuente el caso de procesos o plantas en las que controladores diseñados según otros patrones (clásicos, adaptables, etc.) no dan los resultados deseados o simplemente fallan del todo. Se trata de procesos de elevada complejidad y a pesar de ello, en la inmensa mayoría de los casos operadores humanos diestros y experimentados alcanzan resultados satisfactorios en su dirección (quizá “óptimos” en esas circunstancias). Cuando se investiga el proceder de esos expertos, se concluye de inmediato que, tanto la información acerca del estado del

proceso como sus acciones de control, las procesan y manejan de un modo básicamente cualitativo. Esto es típico para el proceder humano aún en casos de la dirección de procesos relativamente simples. Observando la actuación de un conductor de automóvil se aprecia que nunca accionará sobre el timón en términos de radianes o de grados, ni sobre el pedal del freno en términos de milímetros o newton. El, ante una esquina cerrada gira el timón mucho, al estar cerca de la señal de pare frena un poco, etc. Sin embargo, nadie pone en duda las excelentes virtudes del ser humano para conducir autos, las que frecuentemente llegan al virtuosismo (¿lo óptimo?).

Con sus principales antecedentes en la lógica multinivel de Lukasiewicz de los años 30, el apelativo “Fuzzy”, que se traduce como borroso es introducido por Zadeh en 1962 en un trabajo que vincula la teoría de los circuitos eléctricos con la de sistemas. Tres años más tarde (Zadeh 1965) da a luz la llamada teoría de conjuntos borrosos que echa los cimientos de la llamada síntesis lingüística, mostrando como pueden usarse planteos lógicos vagos para derivar inferencias (también vagas) a partir de datos vagos. Aunque este último trabajo sugiere aplicaciones a sistemas humanísticos en realidad la extensión al control de procesos industriales se hizo evidente. Al menos así lo fue para Ebrahim Mandami y su grupo del Queen Mary College de Londres, así como para su contemporáneo Richard M. Tong de la cercana universidad de Cambridge. Fue un numeroso colectivo de varias nacionalidades europeas como pioneros de la aplicación de la lógica borrosa al control, a partir de la segunda mitad de los 70.

La temática de la lógica borrosa ha sido abordada en la literatura por un número geométricamente creciente de trabajos, desde 2 en 1965, 69 en 1970, alrededor de 600 en 1975, más de 1500 en 1979 y, aunque no hay referencias exactas, puede estimarse en varios miles la cifra acumulada hasta ahora. Inclusive, desde 1978 existe la publicación Fuzzy set and Systems. Artículos como el de Maiers y Sherif (1985) y Lee (1990), por sólo citar algunos, incluyen referencias a centenares de trabajos, a pesar de ser seleccionados y con preferencia para las aplicaciones en control. Llegado este punto es bueno subrayar que las aplicaciones de Lógica Borrosa abarcan, además del control automático, infinidad

de esferas de la ciencia y la técnica, desde la economía y la biología hasta las ciencias sociales.

Vale la pena comentar brevemente el hecho de que, aunque hoy la teoría de los conjuntos borrosos goza de la aceptación generalizada, el reconocimiento a su validez, rigor y aplicabilidad no ha estado exento a controversias. Algunos científicos de renombre han llegado a lanzarle fuertes invectivas imputándole ingenuidad filosófica, carencia de aplicabilidad e interés, excepto el puramente matemático. Sin embargo, las aplicaciones introducidas y el desarrollo teórico paralelo, de los últimos 20 años se han encargado de darle a esta teoría su justo valor, probablemente menos "milagrosa" que como nos la describiera Zadeh al principio, pero indudablemente muy útil y poderosa.

Los cuestionamientos sobre la lógica borrosa están centrados básicamente en su similitud con la teoría de probabilidades, no obstante como fruto de la teoría borrosa surgió la llamada teoría de posibilidades que guarda diferencias esenciales con la primera.

La teoría borrosa es una teoría matemática que trabaja con borrosidad como tipo de incertidumbre, la borrosidad es la ambigüedad que puede encontrarse en la definición de un concepto o en el significado de una palabra, es decir en el término lingüístico que describe la situación.

La teoría matemática solo ha trabajado con la incertidumbre involucrada en la probabilidad esta incertidumbre generalmente esta relacionada con la ocurrencia de un fenómeno y simbolizada con el concepto de aleatoriedad.

La aleatoriedad y la borrosidad son dos tipos diferentes de incertidumbre. La aleatoriedad es la casualidad natural y se maneja a través de probabilidades. Y la borrosidad es la ambigüedad causada por la imprecisión, expresa mucho más de la incertidumbre diaria y hace parte del significado de las palabras inherentes al pensamiento humano.

Este tipo de incertidumbre tiene que ver con todo el mundo, pues todos piensan y transmiten su información por medio de palabras. Así esa es la clase de incertidumbre que cualquiera puede entender. Por ejemplo la incertidumbre de "lloverá mañana" viene

de una predicción que se hace antes de que llegue el día de mañana, lo que se comprobara al pasar el tiempo llegado el otro día. Sin embargo la incertidumbre encontrada en “persona vieja” o “alta temperatura”, no se comprobara con el paso del tiempo o mediante mediciones. La ambigüedad radica en el significado de las palabras.

Describiremos mejor la diferencia entre la posibilidad y la probabilidad a través de un ejemplo sencillo, utilizamos el planteamiento “Jacinto y Ana tengan X hijos en su matrimonio” donde podemos asociar la distribución de posibilidades con X, interpretando a F(a) como el grado de facilidad con la que Jacinto y Ana tuvieran hijos en su matrimonio. También podemos asociar una distribución de probabilidades con X, interpretando a P(a) como la probabilidad de que Jacinto y Ana tuvieran hijos en su matrimonio, los valores de F(a) y P(a) se muestran en la siguiente tabla:

X	1	2	3	4	5	6	7	8
F(X)	1	1	1	1	0.8	0.6	0.4	0.2
P(X)	0.1	0.8	0.1	0	0	0	0	0

Se observa que mientras la posibilidad de tener 3 hijos es de 1, la probabilidad de que lo hiciera puede ser muy pequeña, 0.1. De aquí que un alto grado de posibilidad no implique un alto grado de probabilidad ni un grado bajo de probabilidad implica un grado bajo de posibilidad. Por supuesto, si un evento es imposible queda circunscrito a ser improbable.

Esta referencia sirve para subrayar la aplicación de la teoría de conjuntos borrosos al control, objetivo primordial de este estudio. Viene de interpretar planteos imprecisos de los operadores como la imposición de restricciones posibilísticas a la clase de eventos que los satisfacen. Tales restricciones se hacen corresponder con pertenencias graduadas, de modo que cada evento tiene su grado de pertenencia dentro del conjunto que delimita la medida en que es consistente con la restricción. Para un planteo como “Si la temperatura es alta...” existirá una gradación posibilística, entre 0 y 1. Entonces, una temperatura de 300 grados puede considerarse como perteneciente al conjunto de las

“altas” (claro esta, en dependencia del proceso en cuestión), con un cierto grado de pertenencia, por ejemplo de 0.9, claro que los planteos no sólo implicarán la simple correspondencia de eventos con conjuntos (unión, intersección, etc.) que tuvieron sus primeras definiciones formales con los trabajos iniciales de Zadeh [4].

2.2.3.1 SISTEMAS BASADOS EN REGLAS BORROSAS.

Sistemas que emplean reglas borrosas del tipo SI - ENTONCES para representar el conocimiento.

Aplicaciones:
Modelado de sistema:

- Obtención de modelos que representan realidades complejas.

Control:

- Plantas industriales complejas

Control en línea

- Sistemas de navegación con perturbaciones

Clasificación:

- Detección de patrones, diagnóstico médico.

Sistemas expertos:

- Ayuda a la decisión, recuperación de información, planificadores financieros.

Minería de datos y descubrimiento de información:

- Extracción del conocimiento intrínseco contenido en grandes bases de datos con reglas de asociación borrosas.

Alternativas para Controles Digitales.

PID (proporcional-integral-derivativo).

- Problemas en entornos de control cambiantes o sistemas no lineales.

MRAC (control adaptativo de modelo de referencia).

- Resuelve el problema anterior ajustando los parámetros del controlador comparando la salida con un modelo de referencia.
- Necesita un modelo matemático.

Control Borroso

- Las entradas, salidas y respuesta de control se especifican con términos similares a los utilizados por un experto en control.
- No se requiere un modelo del sistema.
- Aprendizaje y ajuste automático fácil de realizar.

2.2.3.1.1 TIPOS DE SISTEMAS BASADOS EN REGLAS BORROSAS.

Reglas

En función del tipo de regla borrosa que utilicen se pueden distinguir principalmente dos tipos.

Sistemas basados en reglas borrosas tipo Mamdani.

- SI X1 es Alto y X2 es Bajo ENTONCES Y es Alto.

Sistemas basados en reglas borrosas tipo TSK (Takagi, Sugeno y Kang).

- SI X1 es Alto y X2 es Bajo ENTONCES Y = f (X1, X2).

A continuación se darán a conocer las principales características que identifican a cada uno de estos sistemas borrosos:

Motor de inferencia.

Tipo Mandami.

- Utiliza reglas borrosas para obtener la respuesta del sistema borroso ante una determinada entrada.
- Hay dos formas de realizar este proceso:

- ✓ Inferencia basada en reglas individuales: Aplicar la entrada a la primera regla, a la segunda y así sucesivamente. Posteriormente las salidas de las reglas se unen para obtener una única salida.
- ✓ Inferencia basada en la composición: Calcular la relación borrosa que representa el significado de toda la base de reglas para aplicar la entrada a esa relación borrosa global.

Esquema simplificado de un motor de inferencia basado en reglas individuales:
Disparo de reglas:
Una regla se dispara si el grado de emparejamiento (o aplicabilidad) del antecedente de la regla con la entrada es mayor que cero.

Cálculo del grado de emparejamiento.

- Antecedente con una variable.
- Antecedente con más de una variable.

Escalado o corte de la salida borrosa.
Agregación de las salidas (si es necesario).

Parámetros de diseño para el motor de inferencia:
Elección del tipo de motor de inferencia.

- Basado en reglas individuales.
- Basado en la composición de reglas.

Elección de la representación del significado de las reglas borrosas.

- Operadores de conjunción, disyunción, complemento, modificadores lingüísticos, según el caso.
- Operador de implicación.
- Operador de agregación de reglas.

Tipo TSK (Takagi, Sugeno, Kang).

Normalmente $f(x_1 ,..., xn) = a_0 + a_1 x_1 +... + a_n x_n$

- El antecedente se procesa igual que el de las reglas tipo Mamdani.
- Para una entrada específica, el resultado de disparar una regla es un valor nítido.
- Finalmente los valores nítidos obtenidos al dispararse distintas reglas se combinan para obtener una única salida (máximo, media aritmética ponderada, etc.).

Ventajas y desventajas de los sistemas basados en Reglas Borrosas.

Ventajas.

Tipo Mandami.

- Facilidad para la derivación de reglas.
- Interpretabilidad de las reglas borrosas.
- Fueron propuestos antes y se han utilizado con más frecuencia.

Tipo TSK.

- Incrementan la precisión.
- Mayor eficiencia computacional.
- Facilidad para el análisis del sistema.
- No necesitan interfaz de Desemborronado.
- Garantizan la continuidad de la superficie de salida.

Desventajas.

Tipo Mandami.

- No garantizan la continuidad de la superficie de salida.
- Menor eficiencia computacional.

Tipo TSK.

- El consecuente es una fórmula matemática y no proporciona un marco natural para representar conocimiento humano.
- Limitan la representación de los principios de la lógica borrosa [1].

2.2.3.2 CONTROLADORES BORROSOS.

Principales definiciones:

- ✓ Son Sistemas expertos en tiempo real que implementan una parte de la experiencia del operador humano o ingeniero de procesos que no se presta a ser fácilmente expresado con parámetros PID o ecuaciones diferenciales pero sí mediante reglas de situación o acción.
- ✓ Matriz multidimensional donde los subíndices son variables físicas de entrada, y el valor del elemento matricial representa la acción para obtener el estado deseado del sistema a controlar.
- ✓ Forma heurística y modular de definir sistemas de control basados en tablas no lineales.

Objetivos de un controlador borroso.

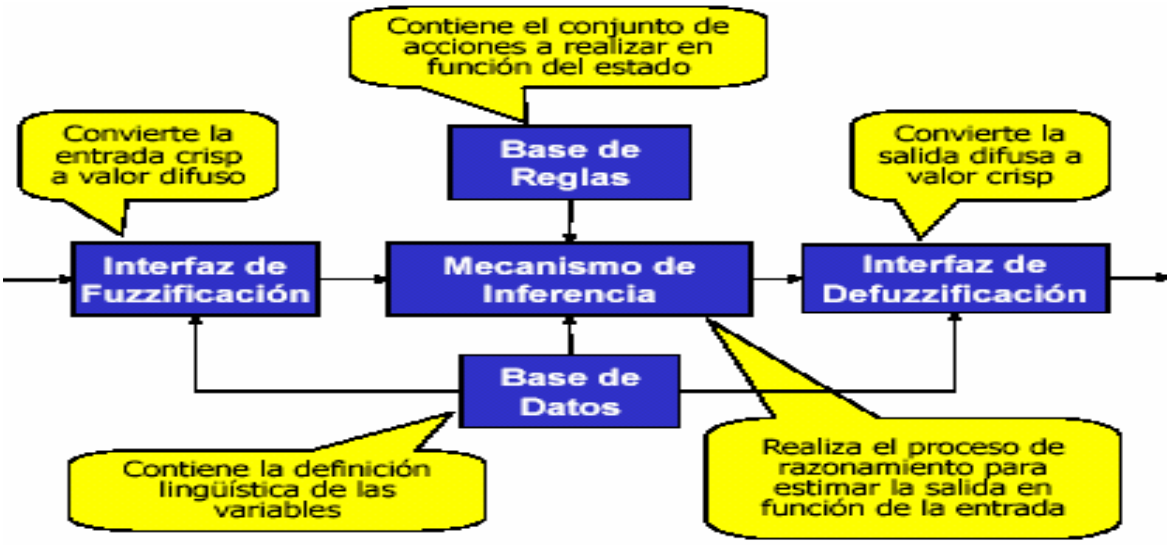
Entre los principales objetivos de los controladores borrosos tenemos:

- Clasificar los sistemas basados en reglas borrosas con base a su estructura y a la estructura de regla borrosa utilizada.
- Conocer ventajas e inconvenientes de cada uno de los tipos de sistemas basados en reglas borrosas.
- Comprender la función de las distintas componentes de un controlador borroso y las posibilidades de diseño disponibles para cada caso.
- Comprender el funcionamiento global de un sistema borroso para control.

2.2.3.2.1 ESTRUCTURA DE UN CONTROLADOR BORROSO.

Esta es la estructura básica que posee un controlador borroso tradicional:

Figura 14 Estructura de un Controlador Borroso.



Según los enfoques tradicionales, el diseño de un regulador se basa en un análisis cabal del proceso y una vez que se posee un modelo cuantitativo de este, todas las decisiones se calculan usando algoritmos estrictamente numéricos. Es obvio que solo a procesos "Dóciles" al análisis cuantitativo le son aplicables técnicas de control de entidad numérica. Por otra parte, existe un número de procesos difíciles de controlar automáticamente en virtud de los obstáculos que se precisan para su modelado, sin embargo los operadores humanos exhiben magníficos resultados en su dirección. Esto, junto a la aparición de la teoría de los conjuntos borrosos y la asequibilidad de medios de cómputo poderosos, estimuló la investigación acerca de las estrategias de control de esos operadores, expresadas por medio de reglas heurísticas, para convertirlas en estrategias de control automático. El controlador lógico borroso resultante se basaría entonces en el modelo lingüístico de la estrategia del operador humano, es decir, en un modelo decisional del experto. La esencia de tal modelo es un programa basado en reglas, por lo que clasifica entre los llamados sistemas expertos [5].

2.2.3.3 FASES DEL CONTROL BORROSO

2.2.3.3.1 EMBORRONADO.

Es la transformación de la información determinista, enviada por el proceso al controlador lógico borroso, en información cualitativa que toma como referencia a conjuntos borrosos. También se evidencia de la explicación que esta transformación depende de parámetros fijados a priori al controlador lógico borroso asociados a la caracterización de las variables medidas del proceso, como variables lingüísticas, numero de conjuntos y forma de las funciones de pertenencia.

2.2.3.3.1.1 INTERFAZ DE EMBORRONADO.

Para cada una de las entradas del sistema:

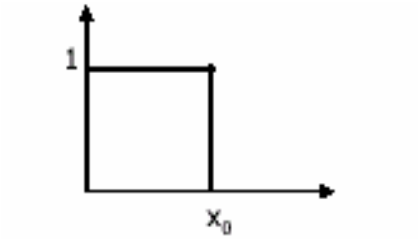
1. Adquirir los valores nítidos de las variables de entrada.
2. Trasladar los valores de las variables a los universos de discurso correspondientes.
3. En función del tipo de sistema borroso, se puede optar por:

- Convertir cada valor crisp en un conjunto borroso con grado de pertenencia igual a 1 para ese valor y 0 para el resto (fuzzy singleton)
- Hacer corresponder a cada valor crisp el término lingüístico más adecuado
- Calcular el grado de pertenencia a cada uno de los conjuntos borrosos utilizados para dicha variable lingüística.

Algunas posibilidades:

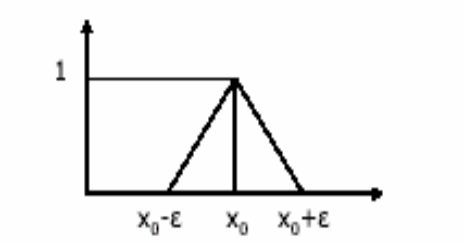
- El valor crisp se convierte en un conjunto borroso tipo singleton Ver figura 15
- Es la opción más sencilla y la más utilizada.
- Adecuado cuando la medición de las variables de estado.

Figura 15 Conjunto singleton.



- Se genera un conjunto borroso con centro en el valor crisp y un soporte acorde con la incertidumbre de la medición Ver figura 16.

Figura 16 Conjunto Triangular.



Funciones de pertenencia.

Las funciones de pertenencia representan la manera grafica de representar un conjunto borroso. Estas funciones son expresadas por medio de formulas matemáticas, son definidas para el espacio unidimensional, bidimensional y multidimensional.
Las mas utilizadas son las triangulares, trapezoidales, tipo campana de Gauss, tipo campana de Bell, sigmoidal.

FUNCIONES DE PERTENENCIA UNIDIMENSIONALES.

Función de Pertenencia Triangular.

Una función de pertenencia triangular se determina a través de tres parámetros (a, b, c) de la siguiente manera:

$$Triangular\ (x,a,b,c) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & c \le x \end{cases}$$

Su forma grafica quedaría así:

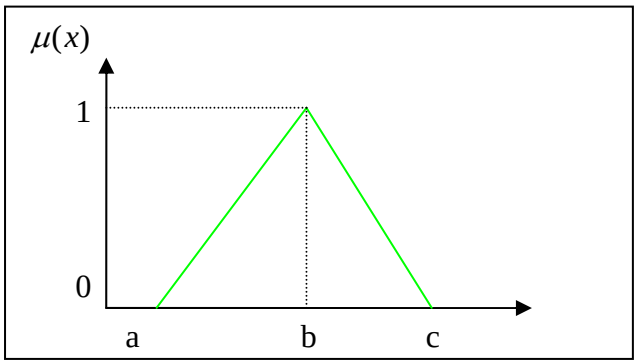


Figura 17 Función de Pertenencia Triangular

Los parámetros a, b, c determinan los tres ángulos de la función de pertenencia triangular, con $(a,b,c) > 0 \quad y \quad a < b < c$.

Función de Pertenencia Trapezoidal.

La función de pertenencia trapezoidal se determina por cuatro parámetros (a, b, c, d) de la siguiente manera:

$$Trapezoidal\ (x,a,b,c,d) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 0 & d \le x \end{cases}$$

Su forma gráfica quedaría así:

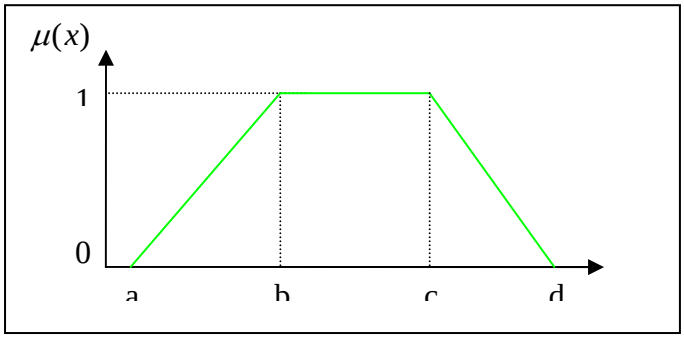


Figura 18 Función de Pertenencia Trapezoidal.

Los parámetros (a, b, c, d) determinan las coordenadas de los cuatro ángulos de la función de pertenencia trapezoidal, con $(a,b,c,d) > 0 \quad y \quad a < b < c < d$

Las funciones de pertenencia triangular y trapezoidal son las más utilizadas en las aplicaciones industriales reales pero presentan la desventaja de no ser muy suaves en los puntos donde se forman los ángulos.

Función de Pertenencia Tipo Campana de Gauss.

La función de pertenencia tipo campana de gauss se determina por dos parámetros. (c, σ).

$$\text{Gaussiana } (x, c, \sigma) = e^{-\frac{1}{2}\left[\frac{x-c}{\sigma}\right]^2}$$

Su forma gráfica quedaría así:

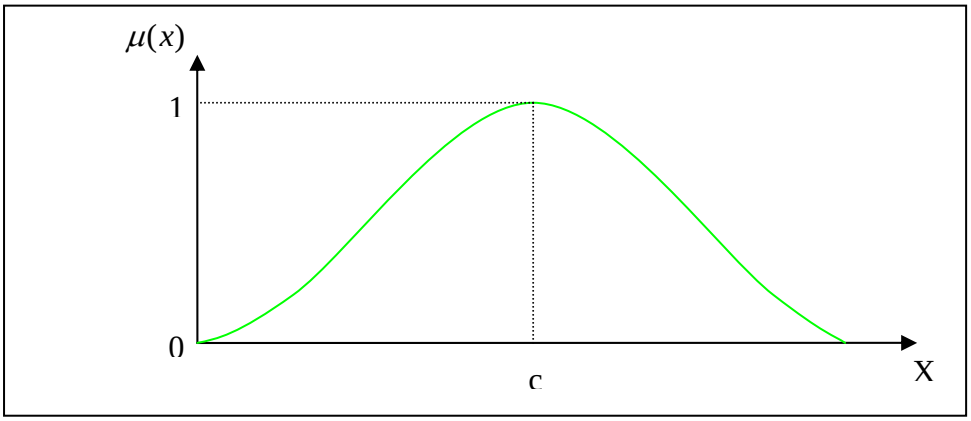


Figura 19 Función de pertenencia Gaussiana.

Las funciones de pertenencia tipo campana de gauss se determinan completamente por los parámetros **c** y σ, donde **c** representa el centro y σ representa el ancho de la función de pertenencia.

Función de Pertenencia Tipo Campana de Bell.

La función de pertenencia generalizada de Bell se determina por tres parámetros (a, b, c) de acuerdo a la expresión:

Bell (x, a, b, c) = $\frac{1}{1+\left|\frac{x-c}{\sigma}\right|^{2*b}}$ con b > 0.

Su forma grafica quedaría así:

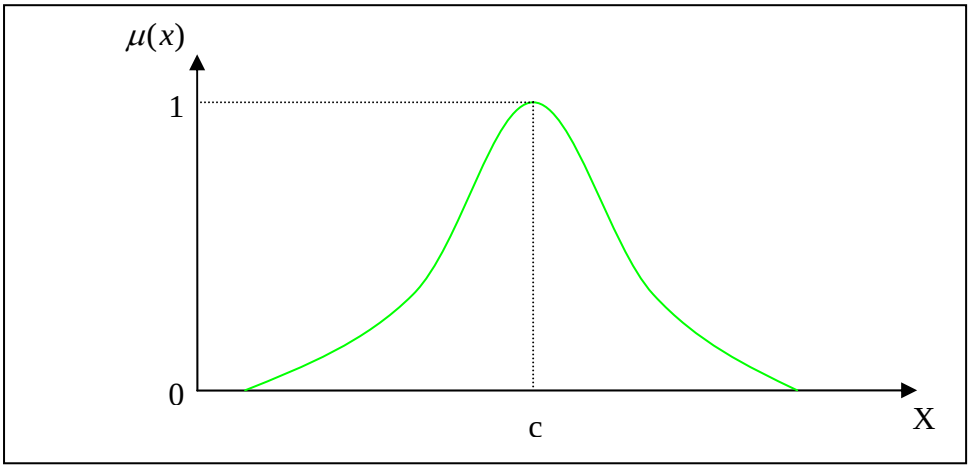


Figura 20 Función de Pertenencia Campana de Bell.

Las funciones de pertenencia es una generalización directa de la distribución de Cauchy utilizada en la teoría de las probabilidades por eso también se le conoce como función de pertenencia de Cauchy, una función de pertenencia particular de Bell se puede obtener a través de una selección correcta del conjunto de parámetros (a ,b, c). Específicamente podemos ajustar **c** y **a** para variar el centro y el ancho de la función de pertenencia y utilizamos **b** para controlar la inclinación de los puntos extremos.

Función de Pertenencia Sigmoidal.

La función de pertenencia tipo Sigmoidal se determina por la siguiente formula:

$$\text{Sig}(x, a, c) = \frac{1}{1 + e^{[-a(x-c)]}}$$

Su forma gráfica quedaría así:

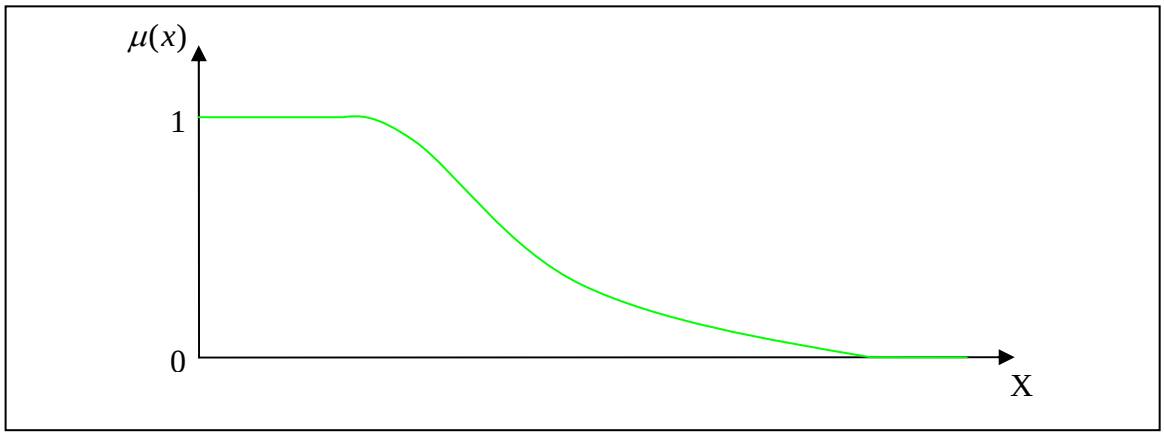


Figura 21 Función de Pertenencia Sigmoidal.

El parámetro **a** controla la inclinación del punto de transición x = **c**, esta función de pertenencia se utiliza en las redes neuronales como función de activación; y para simular el comportamiento del sistema de inferencia borroso.

Aparte de las funciones de pertenencia anteriormente mencionadas, también se pueden crear otro tipo de funciones especializadas y de cualquier tipo según la necesidad practica y especifica de cada aplicación.

FUNCIONES DE PERTENENCIA BIDIMENSIONALES.

En algunas ocasiones es ventajoso emplear funciones de pertenencia con dos entradas cada una en un universo diferente, un método natural para extender las funciones de pertenencia es mediante la extensión cilíndrica de las funciones de pertenencia unidimensionales.

2.2.3.3.2 DESEMBORRONADO.

Es la transformación de un valor borroso dado después del proceso de inferencia, en un valor cuantitativo o crisp.

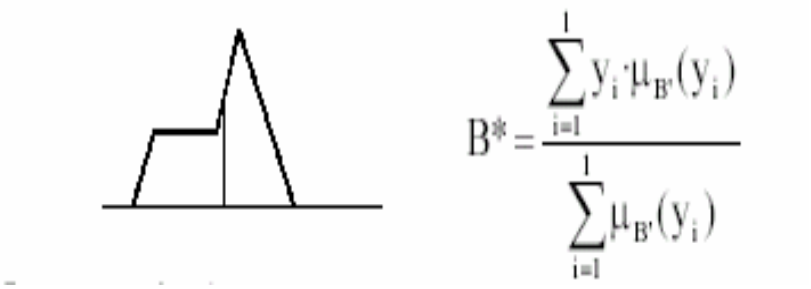
Interfaz de Desemborronado.

El Desemborronado transforma el conjunto borroso de salida en un valor crisp mediante métodos desarrollados por investigadores de la lógica borrosa.

Entre los principales métodos de desemborronado tenemos:

1. **Centro de área o centro de gravedad.**

Figura 22 Método del Centro de Área.

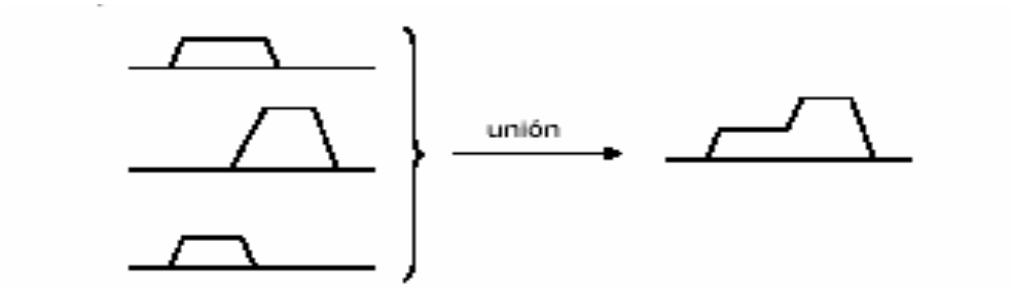


$$B^* = \frac{\sum_{i=1}^{l} y_i \cdot \mu_{B'}(y_i)}{\sum_{i=1}^{l} \mu_{B'}(y_i)}$$

Inconvenientes:

- El cálculo del conjunto borroso agregado es costoso.
- No tiene en cuenta el hecho de que dos áreas se solapen.

Figura 23 Agregación de Particiones activas.



2. **Centro de sumas.**

$$B = \frac{\sum_{i=1}^{l} y_i \cdot \sum_{k=1}^{m} \mu_{B^{(k)}}(y_i)}{\sum_{i=1}^{l} \cdot \sum_{k=1}^{m} \mu_{b^{(k)}}(y_i)}$$

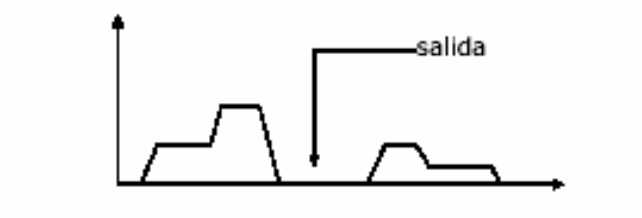
- Considera la contribución de cada área de forma independiente. El método del centro de área toma la unión de los B'(k), mientras que este método toma la suma de los conjuntos. De esta forma, si un área se repite, se considera de nuevo, evitando el problema de solapamiento visto anteriormente.

- No requiere el cálculo del conjunto borroso de salida.

3. **Centro de mayor área.**

Problema: si B' no es convexo, el centro de área y de sumas da una salida en la zona intermedia, donde el conjunto borroso tiene baja importancia.

Figura 24 Método del centro de Mayor Área.



- Solución: se determina el conjunto borroso con mayor área y se calcula su centro de gravedad.
- Es un método muy costoso computacionalmente hablando.

4**. Método de la altura.**

- No requiere el cálculo del conjunto borroso de salida.
- Rápido.
- Requiere la definición del punto umbral (primer punto de un conjunto borroso con grado de pertenencia máximo).

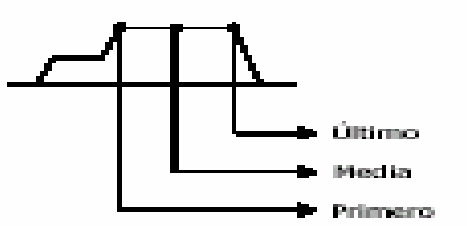
$$B = \frac{\sum_{k=1}^{m} c^{(k)} \cdot \mu_{B^{(k)}}(c^{(k)})}{\sum_{k=1}^{m} \mu_{B^{(k)}}(c^{(k)})}$$

Siendo c (k) el valor umbral del conjunto borroso B’ (k).

5. **primero del máximo, ultimo del máximo, y media de los máximos.**

Toma el valor más pequeño, más grande o medio del núcleo del conjunto borroso resultante.

Figura 25 Primer máximo, Ultimo máximo y Media de Máximos.



Ventaja:

- costo computacional muy bajo.

Inconvenientes:

- Valor de salida menos representativo.
- Puede producir discontinuidades, es decir, genera una salida no continua para pequeños cambios en la entrada [7].

2.2.3.3.3 BASE DE CONOCIMIENTO.

Está formada por la Base de Reglas y la Base de Datos.

Parámetros de diseño implicados:

- Elección de las variables de estado del proceso y de control del mismo.
- Elección del conjunto de términos lingüísticos para las variables de estado y de control.
- Elección de la estructura del antecedente y consecuente de las reglas.
- Conjunto de reglas borrosas.

Formas de obtención de la base de conocimiento:

A través de experiencia, conocimiento de ingeniería de control o acciones de un operador de control experimentado.

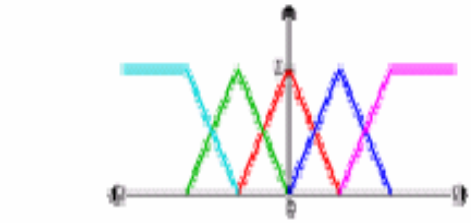
- Experto capaz de describir de forma lingüística sus reglas de decisión (factores de escala, semántica de los conjuntos borrosos, operadores implicados, etc.).
- Información extraída a partir de un cuestionario realizado al experto.
- Información extraída a partir de la observación de las acciones de control de un operador.
- Obtención basada en un modelo borroso.
- Obtención basada en aprendizaje automático (métodos ad hoc, computación evolutiva, redes neuronales, clustering, etc.).

2.2.3.3.4 BASE DE DATOS.

Proporciona la información necesaria para el funcionamiento del módulo de Emborronado, de desemborronado y del motor de inferencia.

- Se define la semántica de cada variable lingüística.

Figura 26 Base de datos.



- También se pueden definir factores de escalado para extender o reducir el universo de discurso, así como cambiar la sensibilidad.

2.2.3.3.5 BASE DE REGLAS.

Representa de forma estructurada la política de control experto.

Se deben determinar los siguientes aspectos:

- ✓ Qué variables de estado y de control se considerarán.
- ✓ Qué estructura tendrá la regla borrosa.
- ✓ Qué conjunto de reglas (en su representación simbólica) se utilizará.

Ejemplo: Base de reglas para el controlador borroso de una aspiradora

Objetivo: Regular la fuerza de aspiración

¿Variables de entrada?

Cantidad de suciedad:

{muy sucio, sucio, algo sucio, casi limpio, limpio}.

¿Variable de control?

Fuerza:

{muy fuerte, fuerte, normal, débil, muy débil}.

Propuesta 1 para la base de reglas:

R1: SI la superficie está sucia ENTONCES la fuerza es fuerte.

R2: SI la superficie está algo sucia ENTONCES la fuerza es normal.

R3: SI la superficie está casi limpia ENTONCES la fuerza es débil.

R4: SI la superficie está limpia ENTONCES la fuerza es muy débil.

Se puede mejorar el rendimiento incluyendo más información:

¿Variables de entrada?

Cantidad de suciedad:

{muy sucio, sucio, algo sucio, casi limpio, limpio}.

Tipo de superficie:

{madera, caucho, alfombra}.

¿Variable de control?

Fuerza:

{muy fuerte, fuerte, normal, débil, muy débil}.

Propuesta 2 de base de reglas:

	Limpio	Casi limpio	Algo sucio	sucio	Muy sucio
Madera	Muy débil	Muy débil	Débil	Normal	Fuerte
Caucho	Muy débil	Débil	Normal	Fuerte	Muy fuerte
Alfombra	Débil	Normal	Normal	Fuerte	Muy fuerte

2.2.3.4 ANÁLISIS DE CONTROLADORES BORROSOS.

Objetivos principales:

- Entender la importancia de la etapa de análisis de un controlador borroso.
- Conocer las propiedades estáticas y dinámicas en un controlador borroso y su influencia en el comportamiento del sistema.
- Conocer algunos métodos de análisis de estabilidad para control borroso.
- Conocer formas de valorar la interpretabilidad de un controlador borroso.

Introducción.

- El análisis de la fiabilidad de un controlador borroso: se estudia la coherencia del conocimiento usado y su comportamiento dinámico.

    - ✓ Propiedades estáticas: relacionadas con la base de conocimiento usada.
    - ✓ Propiedades dinámicas: relacionadas con el comportamiento del controlador.
- Análisis de la interpretabilidad de un controlador borroso: se estudia en qué grado es fácil comprender el conocimiento empleado en un controlador borroso.

Análisis de fiabilidad.

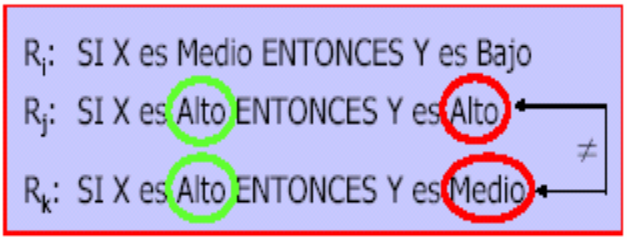
- Un controlador borroso no requiere un modelo matemático de la dependencia entre las salidas y entradas de control.
- La base de conocimiento se obtiene generalmente de conocimiento experto y es tan buena como el conocimiento humano que representa.
- Es fácil obtener un sistema borroso para control, pero es necesario un análisis y simulación para determinar la confianza en su fiabilidad.

Consistencia

Un conjunto de reglas es consistente si no contiene contradicciones.
Existen varias posibilidades para definir la contradicción:

- Un conjunto de reglas es inconsistente si existen dos reglas con igual antecedente pero distinto consecuente.

Figura 27 Consistencia de Reglas.

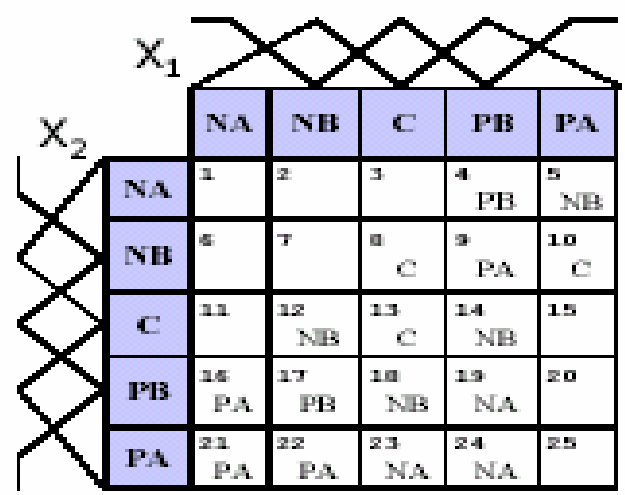


- Algo más flexible sería decir que son inconsistentes cuando, teniendo el mismo antecedente, los consecuentes son de intersección vacía.

Completitud.

- Un sistema borroso de control debe ser capaz de inferir una acción de control para cualquier estado del proceso.
- Un conjunto de reglas es completo si cualquier combinación de entrada produce como resultado un conjunto borroso de salida.
- Esto no significa que deban existir todas las reglas posibles, ya que debido al solapamiento de los conjuntos borrosos, algunas entradas pueden cubrirse indirectamente por reglas adyacentes.

Figura 28 Completitud de Reglas.



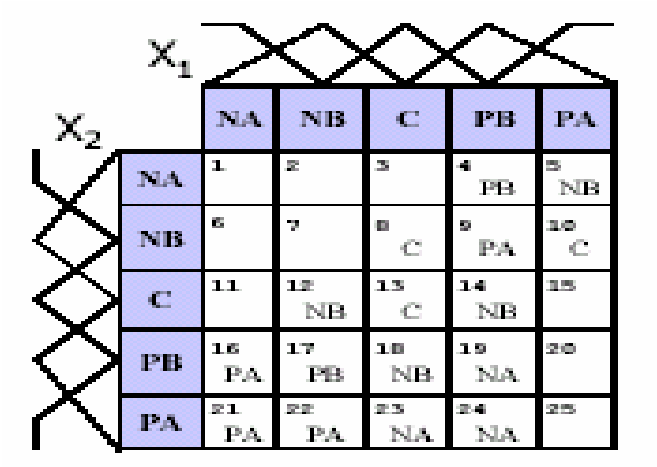
- El subespacio de entrada correspondiente a R15 está cubierto indirectamente por R9, R10, R14 y R19.
- Sin embargo, los subespacios de entrada correspondientes a R1, R2 y R6 no los cubre ninguna otra regla y, por tanto, el conjunto de reglas del controlador no es completo.

Continuidad.

Un conjunto de reglas borrosas es continuo si los consecuentes asociados a cada par de reglas vecinas tienen una intersección no vacía.

-Dos reglas borrosas se consideran vecinas si sólo difieren en el término lingüístico de una única variable de entrada, y dichos términos son consecutivos.

Figura 29 Continuidad de Reglas.



- Vecinas de R18: R13, R17, R19 y R23.
- Dada la siguiente semántica para el consecuente: desemborronado.

- R18 y R17 provocan un conjunto de reglas no continuo porque NB y PB tienen intersección vacía.

Interacción.

$$R_k : \text{SI X es } A_k \text{ ENTONCES Y es } B_k$$

- Si la entrada X toma el valor A_k se espera que la salida sea exactamente B_k.
- Existe interacción cuando no ocurre esto (por el efecto de otras reglas).
- Factores que contribuyen a la interacción:

  Uso de un conjunto de reglas inconsistente.

  Definición inapropiada de los conjuntos borrosos y sus funciones de pertenencia en el universo de discurso.

  Elección inapropiada del método de inferencia y el operador de Desemborronado.

Robustez.

La robustez refleja la habilidad del sistema borroso de control para resistir a cambios abruptos en las entradas.

A esta propiedad afecta:

- La resolución de control.
- La partición del espacio de entrada.
- El número de reglas de control.

Las funciones de pertenencia de los conjuntos borrosos

- Se ha demostrado experimentalmente que el uso de funciones de pertenencia trapezoidales colabora a que el sistema sea más robusto ante cambios drásticos.

Estabilidad en Control Borroso.

Habitualmente, en un controlador borroso, se desconoce el modelo matemático de la planta, esto dificulta (o hace imposible, según el caso) la aplicación de las técnicas de análisis de estabilidad existentes en control clásico.

- El problema del análisis de estabilidad en sistemas borrosos para control no está resuelto completamente.
- En general, se puede decir que un controlador es estable si produce una respuesta acotada ante una entrada o distorsión acotada.

Técnicas de análisis:

1. Si se conoce el modelo de la planta:
    - ✓ Planta lineal: técnica de descripción de la función, criterio de estabilidad L2.
    - ✓ Planta no lineal: principio de invarianza / alfa-estabilidad, función de Lyapunov.
2. Sin modelo de la planta:

    Análisis de indicadores de estabilidad.

    Análisis de la trayectoria de disparo de reglas.

- Estas técnicas de análisis sirven para comprobar la estabilidad del controlador borroso. Si se detecta que es inestable, habrá que ajustar las funciones de pertenencia, las reglas borrosas y los operadores de inferencia para reducir la inestabilidad.

Indicadores de estabilidad.

- Un controlador es estable cuando se observan las siguientes características en la curva de respuesta del sistema.
    - ✓ Radio de exceso (overshoot) bajo.
    - ✓ Tiempo de reposo (settle time) corto.

Análisis de interpretabilidad.

- Generalmente, en los sistemas borrosos para control no es tan importante la interpretabilidad.

- No obstante, si se desea comprender el funcionamiento de la planta a través del controlador diseñado, la interpretabilidad sí es importante.
- La interpretabilidad también cobra importancia en control borroso cuando se desea integrar el conocimiento extraído por varios expertos.
- En muchas ocasiones, forzar un controlador borroso a ser interpretable lo hace más rígido y, generalmente, menos preciso.
- Esta restricción de interpretabilidad puede favorecer o perjudicar, según el caso, la fiabilidad del controlador.

Trayectoria de disparo de reglas.

Figura 30

Orden de disparo de reglas típico en un sistema inestable

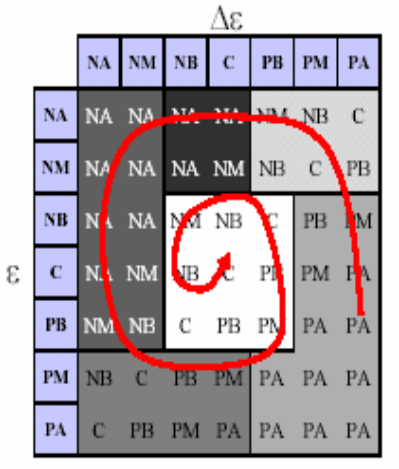
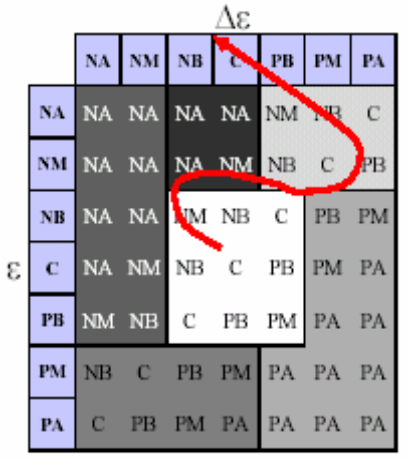


Figura 31

Orden de disparo de reglas típico en un sistema estable



Sencillez de la estructura.

- El Sistema basado en reglas borrosas de tipo Mandami es más fácilmente interpretable que el TSK debido a la estructura del consecuente.
- El uso de reglas con variables lingüísticas (donde existe un conjunto de términos común) en lugar de reglas con variables borrosas (donde cada variable en cada

regla usa un conjunto borroso distinto) hace que el controlador sea más interpretable.

- Si se añaden modificadores lingüísticos, factores de escala, pesos para indicar el factor de importancia de cada regla, etc., la estructura del controlador se hace más flexible pero se pierde interpretabilidad.

Simplicidad de la base de conocimiento.

- Un controlador borroso será más fácilmente interpretable cuanto más simple sea.
- La simplicidad viene determinada por el tamaño del conocimiento empleado.
- Número de términos lingüísticos reducido.
- Número de reglas borrosas reducido.

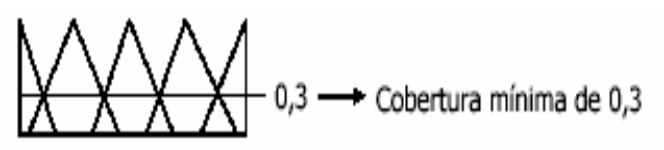
2.2.3.5 ERRORES EN LA SEMÁNTICA.

- Cada valor del universo de discurso debería pertenecer al menos a un término lingüístico.

Figura 32 errores en la Semántica. (Intervalos)

- Alternativamente, se puede definir un criterio más estricto estableciendo un nivel mínimo de grado de pertenencia a algún término lingüístico para cada valor del universo de discurso.

Figura 33 Errores en la Semántica (cobertura).

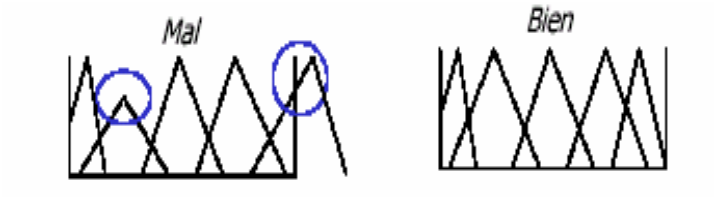


Normalidad de los Conjuntos Borrosos.

- Cada conjunto borroso debería mostrar un grado de pertenencia máximo (generalmente 1) con al menos un valor del universo de discurso.

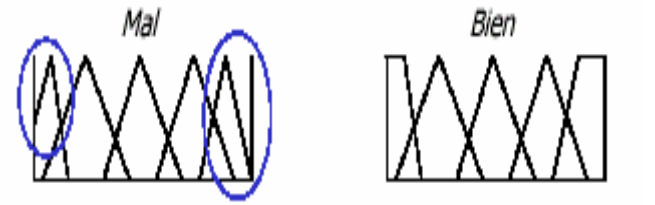
- Para ello, basta con que los conjuntos borrosos sean normales.

Figura 34 Normalidad de los Conjuntos Borrosos.



- Además, algunas veces es deseable que los valores de los extremos del universo de discurso tengan un grado de pertenencia máximo a algún conjunto borroso.

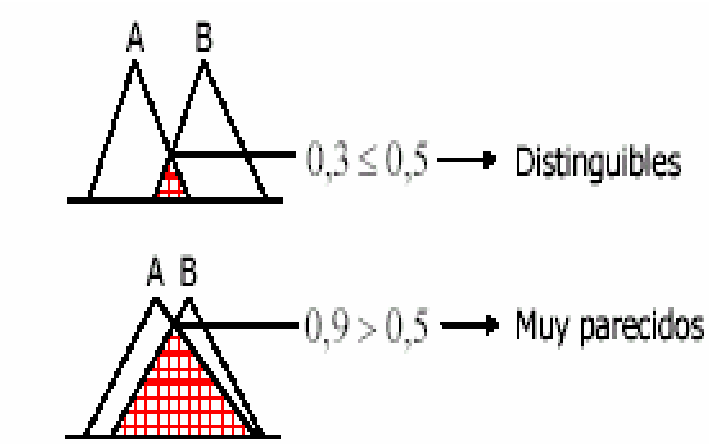
Figura 35 Grados de Pertenencia De un Conjunto borroso.



Distinguibilidad entre los conjuntos borrosos.

- Cada término lingüístico debe tener un significado claro y el conjunto borroso asociado debe definir claramente un rango del universo de discurso.
- Las funciones de pertenencia deben ser suficientemente distintas unas de otras.
- Se pueden usar distintas medidas de similitud entre conjuntos borrosos. Por ejemplo, que la altura de la intersección sea inferior a un valor.

Figura 36 Distinguibilidad de un Conjunto Borroso.



3. EJEMPLOS DE LOS MODELOS BORROSOS MANDAMI Y TSK

3.1 MODELO BORROSO DE MANDAMI PARA UNA ENTRADA Y UNA SALIDA

3.1.1 REGLAS DEL MODELO

- IF X es pequeño THEN Y es pequeño.
- IF X es medio THEN Y es medio.
- IF X es grande THEN Y es grande.

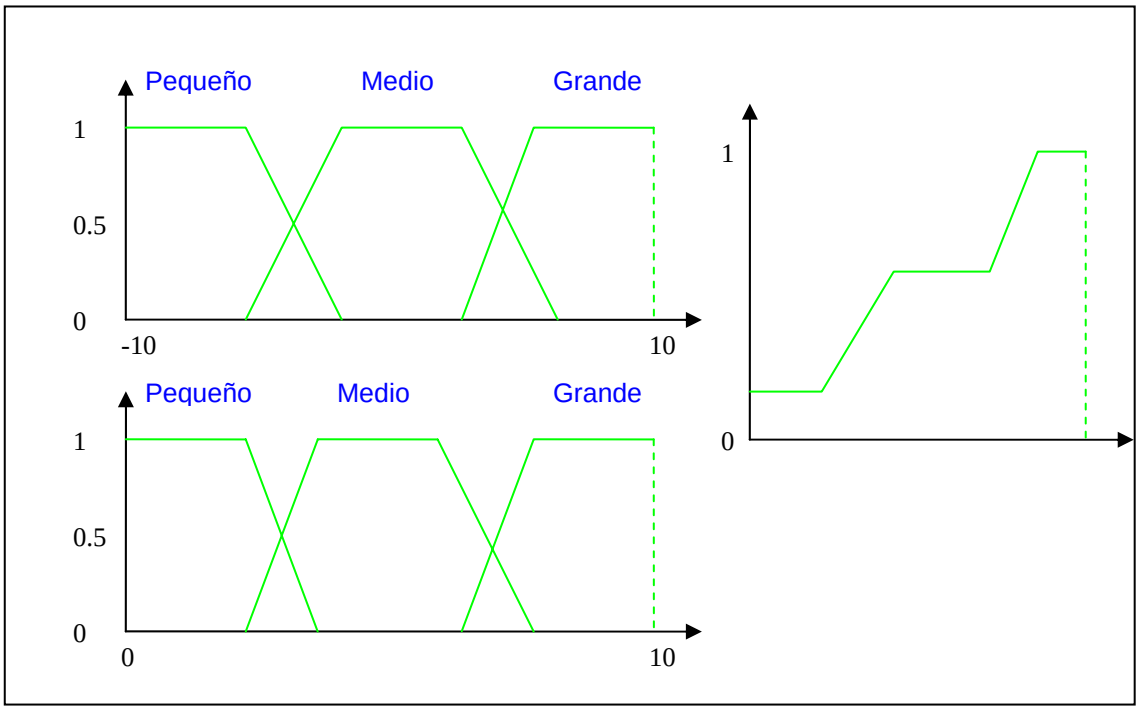


Figura 37 Modelo Borroso de Mandami.

Si consideramos la eficiencia de cómputo o la facilidad de manejo matemático, los sistemas de inferencia borrosos de Mandami no siguen estrictamente la definición de la regla de composición Máx. – min. De inferencia.

- Operador AND (generalmente T – norm) para calcular la fuerza de disparo de una regla con antecedentes con AND.
- Operador OR (generalmente T – conorm) para calcular la fuerza de disparo de la regla con antecedentes con OR.
- Operador de implicación (generalmente T- norm) para calcular las funciones de pertenencia calificando al consecuente basado en la fuerza del disparo de la regla.

- Operador de agregación (generalmente T – conorm) para la agregación de las funciones de pertenencia calificando el consecuente par generar una salida general de las funciones de pertenecía.
- Operador de desemborronado para transformar una salida de la función de pertenencia en un valor único binario de salida.

3.2 MODELO BORROSO DE MANDAMI PARA DOS ENTRADAS Y UNA SALIDA

Se considera un proceso cuyo estado esta caracterizado por las variables lingüísticas presión y temperatura, y por una variable de acción representada por la apertura de una válvula de control (figura 38).

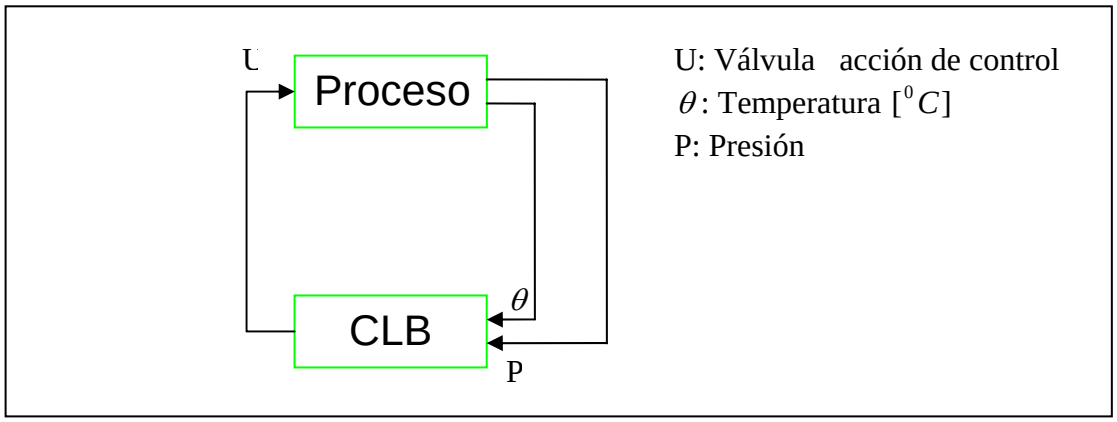


Figura 38 Lazo de control con C LB.

Variables.

Las variables propias del proceso se denotan de la siguiente manera. La variable presión tiene un universo de discurso de [0 10] HPa (Hectopascales) y 3 particiones borrosas, figura 39. De modo similar la variable temperatura puede moverse en un rango de [50-200] ^{o}C , con tres particiones borrosas, figura 40. Para la válvula de control, la apertura y cierre en porcentaje [-100 0 100]% respecto a la posición dada (valor incremental) se representa como variable lingüística según figura 41.

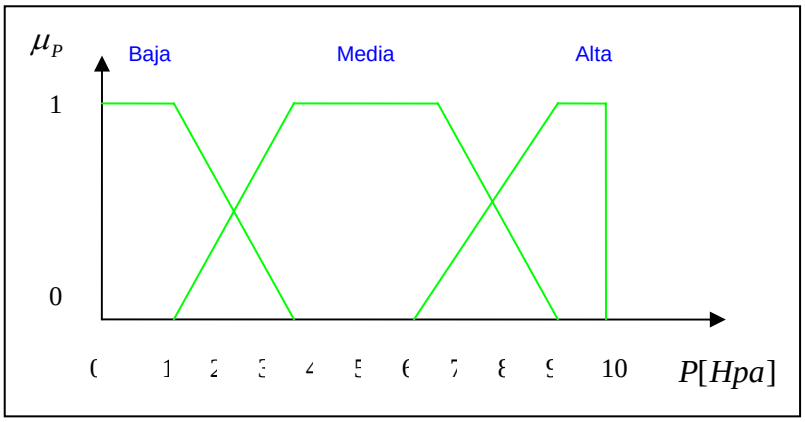


Figura 39 Variable Presión

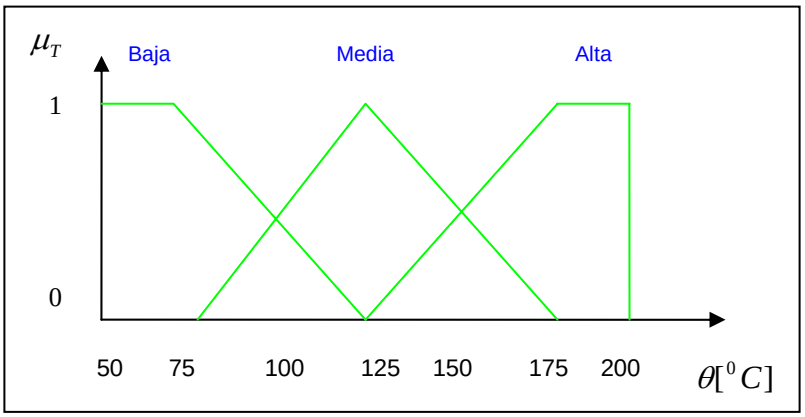


Figura 40 Variable Temperatura.

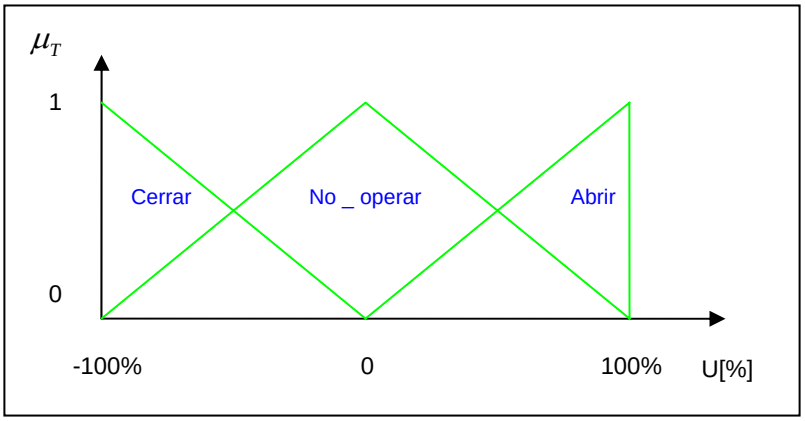


Figura 41 Variable de acción (válvula)

3.2.1 BASE DE REGLAS

La base de reglas consta de hasta 9 reglas. Se suponen todas las posibles combinaciones dadas por las particiones borrosas de las variables que componen el sistema.

$R_1 : Si\ p\ es\ baja\ Y\ \theta\ es\ baja\ ENTONCES\ U\ es\ abrir$

$R_2 : Si\ p\ es\ media\ Y\ \theta\ es\ baja\ ENTONCES\ U\ es\ abrir$

$R_3 : Si\ p\ es\ alta\ Y\ \theta\ es\ baja\ ENTONCES\ U\ No\_operar$

$R_4 : Si\ p\ es\ baja\ Y\ \theta\ es\ media\ ENTONCES\ U\ es\ abrir$

$R_5 : Si\ p\ es\ media\ Y\ \theta\ es\ media\ ENTONCES\ U\ No\_operar$

$R_6 : Si\ p\ es\ alta\ Y\ \theta\ es\ media\ ENTONCES\ U\ es\ cerrar$

$R_7 : Si\ p\ es\ baja\ Y\ \theta\ es\ alta\ ENTONCES\ U\ es\ No\_operar$

$R_8 : Si\ p\ es\ media\ Y\ \theta\ es\ alta\ ENTONCES\ U\ es\ cerrar$

$R_9 : Si\ p\ es\ alta\ Y\ \theta\ es\ alta\ ENTONCES\ U\ es\ cerrar$

Después de especificar el sistema borroso se quiere encontrar la acción a tomar para el siguiente caso particular.

$$p = 7Hpa \qquad \theta = 150^0 C$$

Aplicando el proceso de emborronado tenemos:

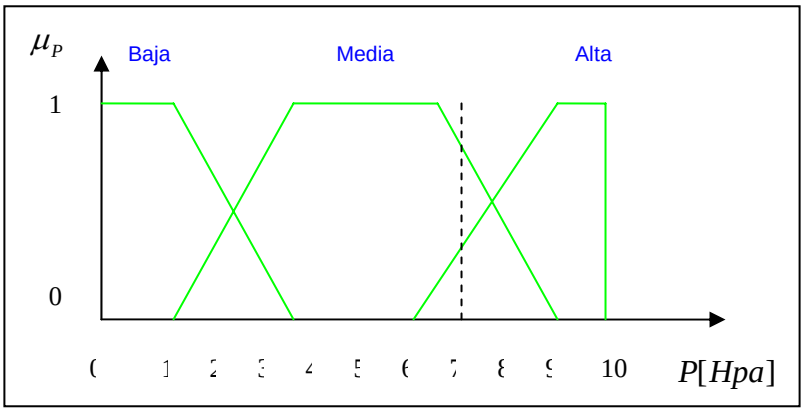


Figura 42 Emborronado variable presión

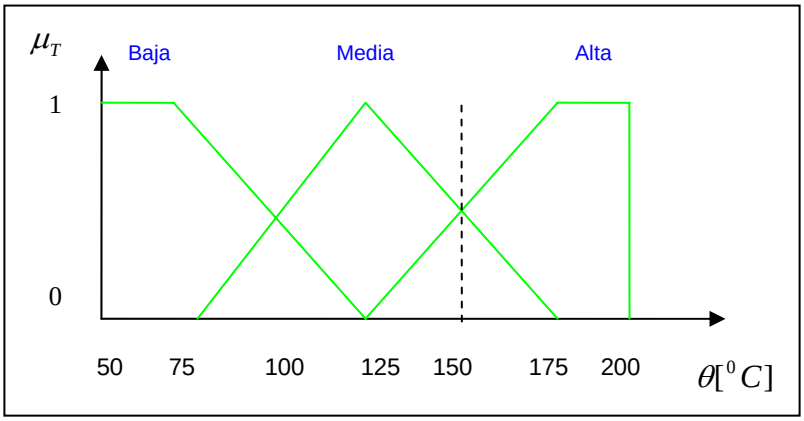


Figura 43 Emborronado variable temperatura

De las anteriores graficas podemos concluir:

$$p : 0.33\ alta\ y\ 0.67\ media$$
$$\theta : 0.5\ alta\ y\ 0.5\ media$$

Procedemos a la activación y desactivación de reglas. Por la presión se desactivan $R_1, R_2\ y\ R_3$. Por la temperatura se desactivan además $R_4\ y\ R_7$, Quedan activas solamente las siguientes reglas:

R_5 : *Si p es media Y θ es media ENTONCES U No_operar*

R_6 : *Si p es alta Y θ es media ENTONCES U es cerrar*

R_8 : *Si p es media Y θ es alta ENTONCES U es cerrar*

R_9 : *Si p es alta Y θ es alta ENTONCES U es cerrar*

Para este ejercicio se utilizara la intersección (mínimo) para los antecedentes de las reglas:

$$R_5 : \min\{0.67, 0.5\} = 0.5$$
$$R_6 : \min\{0.33, 0.5\} = 0.33$$
$$R_8 : \min\{0.67, 0.5\} = 0.5$$
$$R_9 : \min\{0.33, 0.5\} = 0.33$$

Si no se hubieran desactivado las demás reglas, es fácil ver que de todas maneras su resultado serio nulo ya que $(\min\{0, \alpha\} = 0 \text{ para todo } \alpha \geq 0)$.

Agregación de Reglas

Graficando las respectivas reglas activas para este proceso:

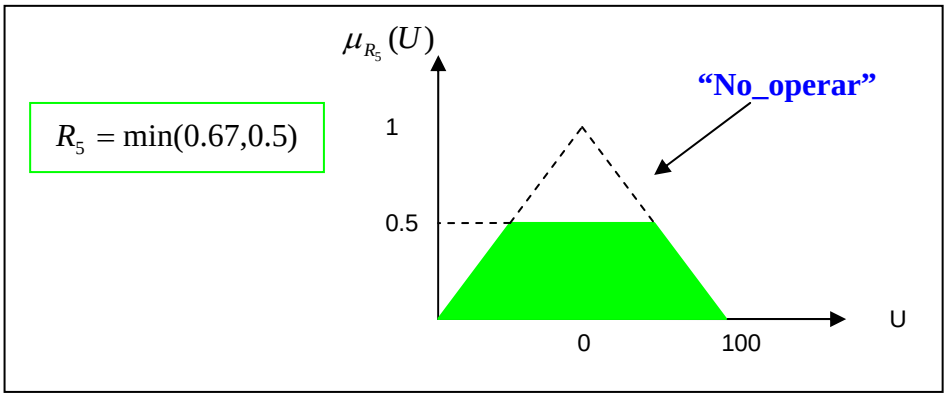


Figura 44 Regla activa #5

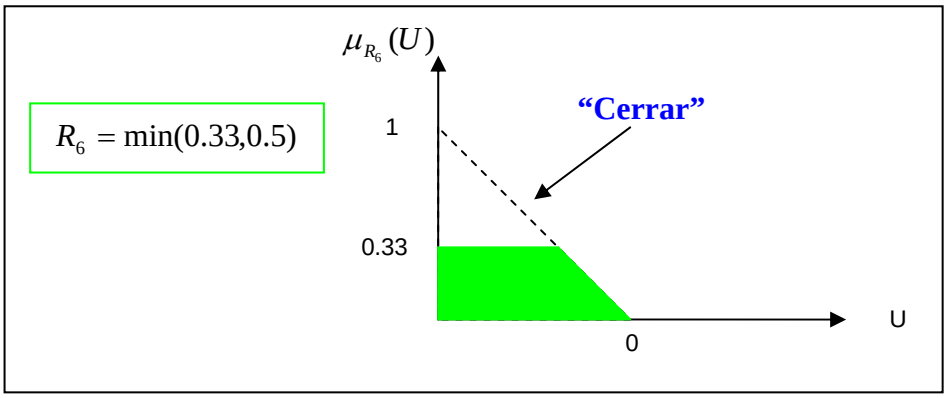


Figura 45 Regla activa #6

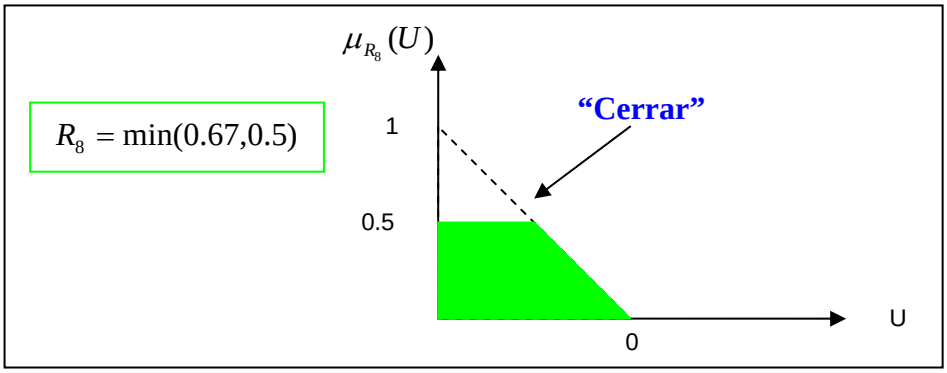


Figura 46 Regla activa #8

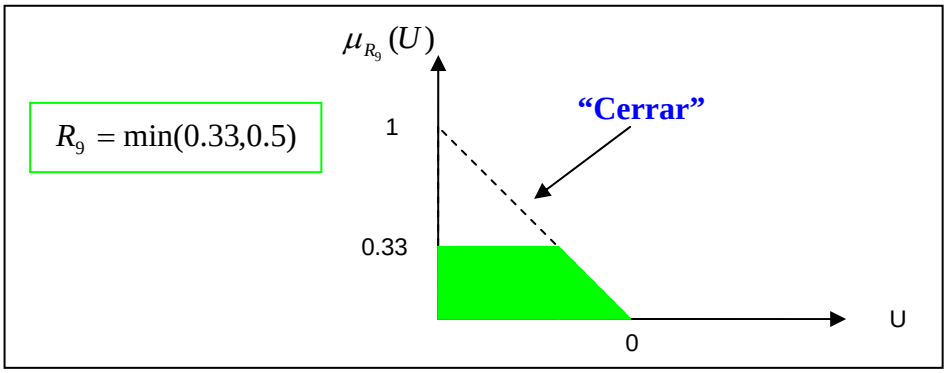


Figura 47 Regla activa #9

Procedemos a la agregación de reglas utilizando el máximo, obteniendo la siguiente grafica:

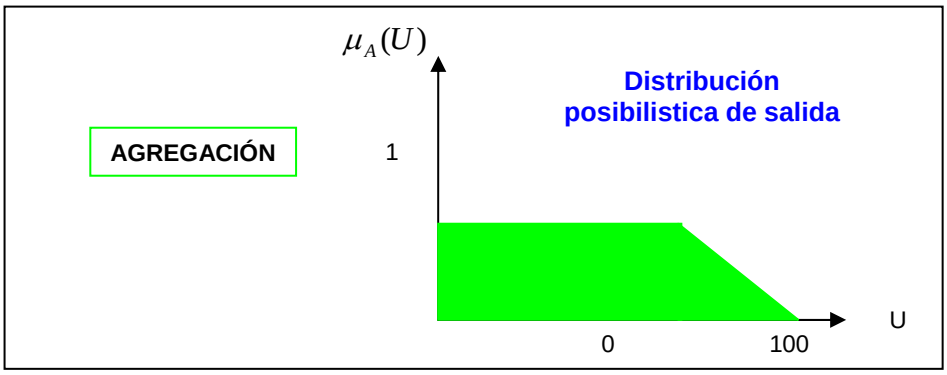


Figura 48 Agregación de Reglas activas.

En la agregación obtenemos el conjunto borroso de salida al que debemos aplicarle un método de desemborronado para encontrar una respuesta cuantitativa, el más conveniente para este caso en particular es el centro de área discreto.

$$X = \frac{\sum_x x * \mu_A(x)}{\sum_x \mu_A(x)}$$

Reemplazando obtenemos:

$$X = -12.5$$

Quiere decir que la válvula debe cerrarse un 12.5 % de estado que tiene actualmente.

3.3 MODELO BORROSO DE MANDAMI PARA TRES ENTRADAS Y DOS SALIDAS

Se trata de modelar la situación que se presenta comúnmente en las universidades y entidades educativas, la cuestión de cómo infieren algunos aspectos relacionados como la calidad del profesor, el interés del alumno y otros factores sobre el rendimiento de los alumnos y su asistencia a clase.

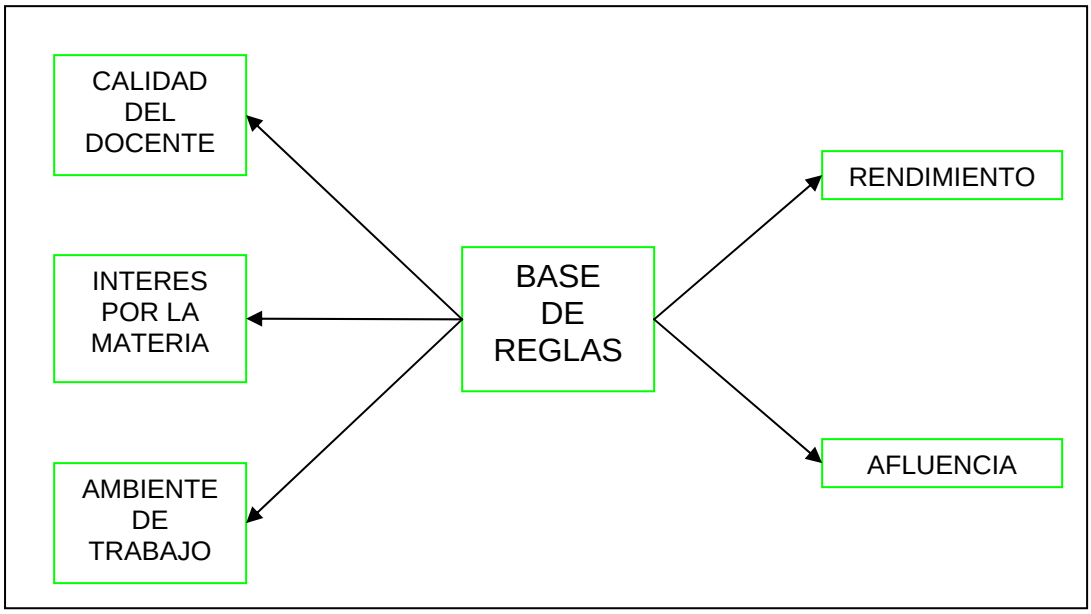


Figura 49 Esquema del ejemplo

Variables.

Se tendrán en cuenta tres variables de entrada y dos variables de salida. Como entradas tenemos Calidad del docente (C_doc) que tendrá tres particiones borrosas y un rango de [0 100] % ver figura 50, Interés por la materia (I_mat) con tres particiones borrosas y un rango de [0 100] % ver figura 51, y por ultimo el ambiente de trabajo (A_trab) con tres particiones borrosas y un rango de [0 100] % ver figura 52. Las salidas están conformadas por el rendimiento de los alumnos (Rend) con tres particiones borrosas y un rango de [0 100] % ver figura 53, y la afluencia de alumnos (Aflue) con tres particiones borrosas y un rango de [0 100] % ver figura 54.

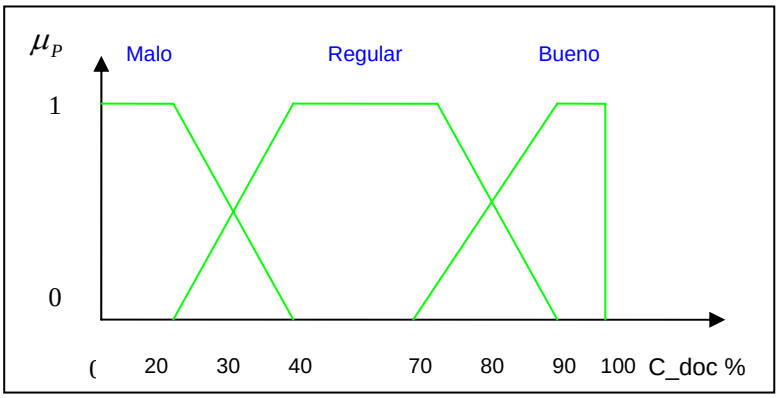


Figura 50 Variable Calidad del Docente.

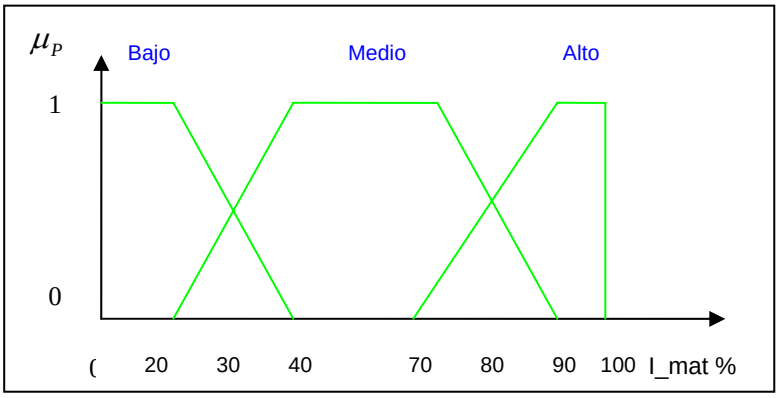


Figura 51 Variable Interés por la Materia.

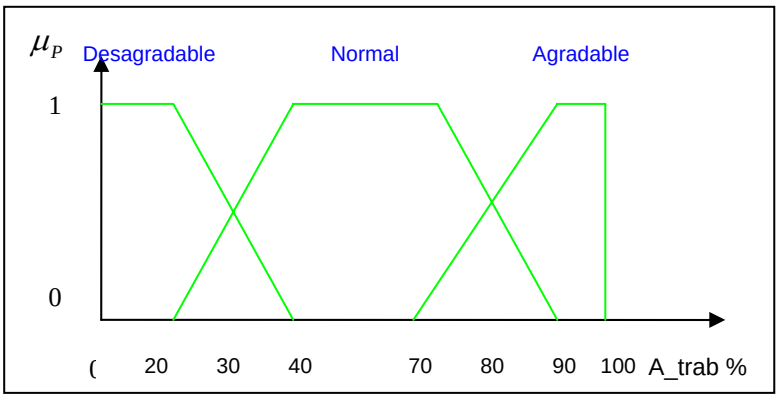


Figura 52 Variable Ambiente de Trabajo.

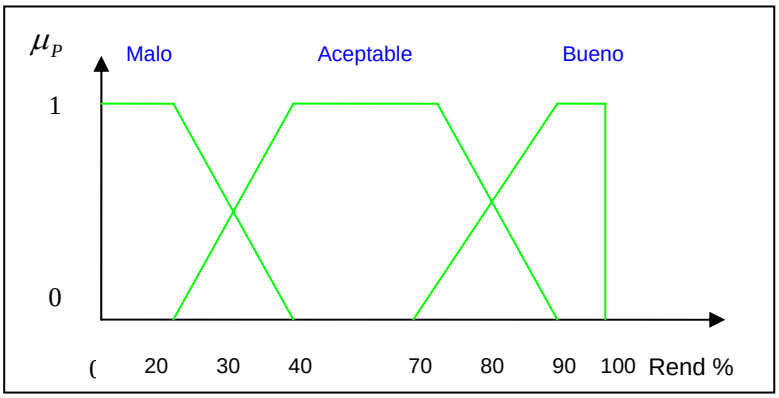


Figura 53 Variable Rendimiento.

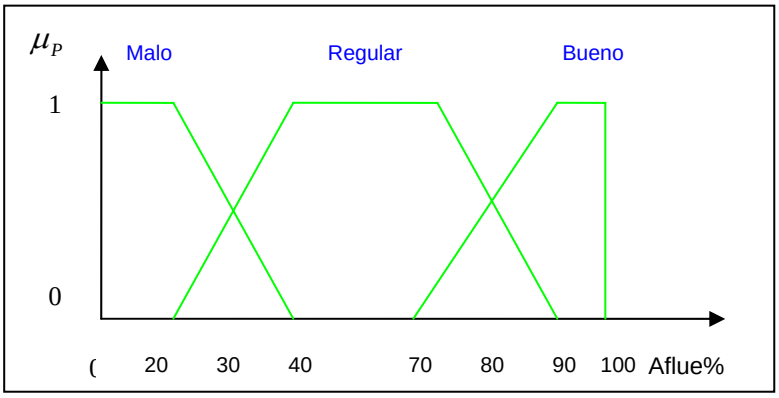


Figura 54 Variable Afluencia.

3.3.1 BASE DE REGLAS

Se tomaron todas las posibles combinaciones de las particiones borrosas a partir de las entradas, dando como resultado 27 reglas borrosas que se aprecian a continuación.

R_1 *si* C_prof *es malo* Y I_mat *es bajo* Y A_trab *es desagradable entonces* Rend *es malo* Y *Aflue es baja*

R_2 *si* C_prof *es regular* Y I_mat *es bajo* Y A_trab *es desagradable entonces* Rend *es malo* Y *Aflue es baja*

R_3 *si C_prof es bueno Y I_mat es bajo Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es media*

R_4 *si C_prof es malo Y I_mat es medio Y A_trab es desagradable entonces* Re*nd es malo Y Aflue es baja*

R_5 *si C_prof es regular Y I_mat es medio Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es baja*

R_6 *si C_prof es buena Y I_mat es medio Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es media*

R_7 *si C_prof es malo Y I_mat es alto Y A_trab es desagradable entonces* Re*nd es malo Y Aflue es baja*

R_8 *si C_prof es regular Y I_mat es alto Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es baja*

R_9 *si C_prof es bueno Y I_mat es alto Y A_trab es desagradable entonces* Re*nd es bueno Y Aflue es media*

R_{10} *si C_prof es malo Y I_mat es bajo Y A_trab es normal entonces* Re*nd es malo Y Aflue es baja*

R_{11} *si C_prof es regular Y I_mat es bajo Y A_trab es normal entonces* Re*nd es malo Y Aflue es baja*

R_{12} *si C_prof es bueno Y I_mat es bajo Y A_trab es normal entonces* Re*nd es aceptable Y Aflue es media*

R_{13} *si C_prof es malo Y I_mat es medio Y A_trab es normal entonces* Re*nd es malo Y Aflue es baja*

R_{14} *si C_prof es regular Y I_mat es medio Y A_trab es normal entonces* Re*nd es aceptable Y Aflue es media*

R_{15} *si C_prof es buena Y I_mat es medio Y A_trab es normal entonces* Re*nd es bueno Y Aflue es media*

R_{16} *si C _ prof es malo Y I _ mat es alto Y A _ trab es normal entonces* Re *nd es malo Y Aflue es baja*

R_{17} *si C _ prof es regular Y I _ mat es alto Y A _ trab es normal entonces* Re *nd es aceptable Y Aflue es media*

R_{18} *si C _ prof es bueno Y I _ mat es alto Y A _ trab es normal entonces* Re *nd es bueno Y Aflue es alta*

R_{19} *si C _ prof es malo Y I _ mat es bajo Y A _ trab es agradable entonces* Re *nd es malo Y Aflue es media*

R_{20} *si C _ prof es regular Y I _ mat es bajo Y A _ trab es agradable entonces* Re *nd es malo Y Aflue es media*

R_{21} *si C _ prof es bueno Y I _ mat es bajo Y A _ trab es agradable entonces* Re *nd es aceptable Y Aflue es media*

R_{22} *si C _ prof es malo Y I _ mat es medio Y A _ trab es agradable entonces* Re *nd es malo Y Aflue es media*

R_{23} *si C _ prof es regular Y I _ mat es medio Y A _ trab es agradable entonces* Re *nd es aceptable Y Aflue es media*

R_{24} *si C _ prof es buena Y I _ mat es medio Y A _ trab es agradable entonces* Re *nd es bueno Y Aflue es alta*

R_{25} *si C _ prof es malo Y I _ mat es alto Y A _ trab es agradable entonces* Re *nd es aceptable Y Aflue es media*

R_{26} *si C _ prof es regular Y I _ mat es alto Y A _ trab es agradable entonces* Re *nd es aceptable Y Aflue es alta*

R_{27} *si C _ prof es bueno Y I _ mat es alto Y A _ trab es agradable entonces* Re *nd es bueno Y Aflue es alta*

Después de especificar el sistema borroso se quiere encontrar las acciones a tomar para el siguiente caso particular de entrada.

Calidad del profesor 30% Interés por la materia 40% Ambiente de trabajo 85%.

$$C\_prof\ 30\%\ \ I\_mat\ 40\%\ \ A\_trab\ 85\%$$

Aplicando el proceso de emborronado obtenemos los siguiente valores.

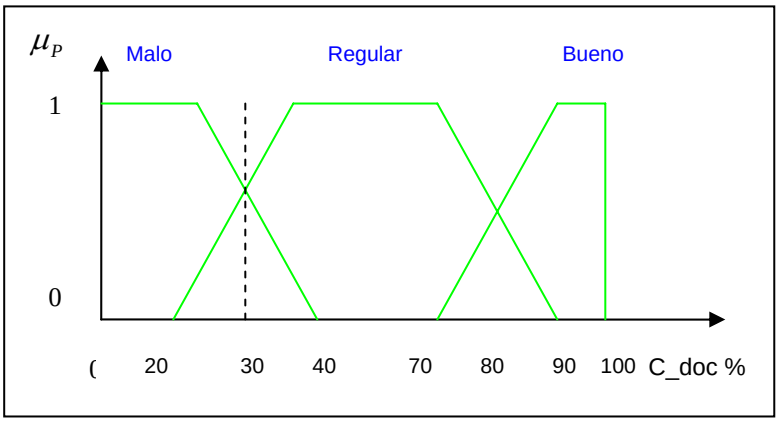


Figura 55 Emborronado variable C_doc

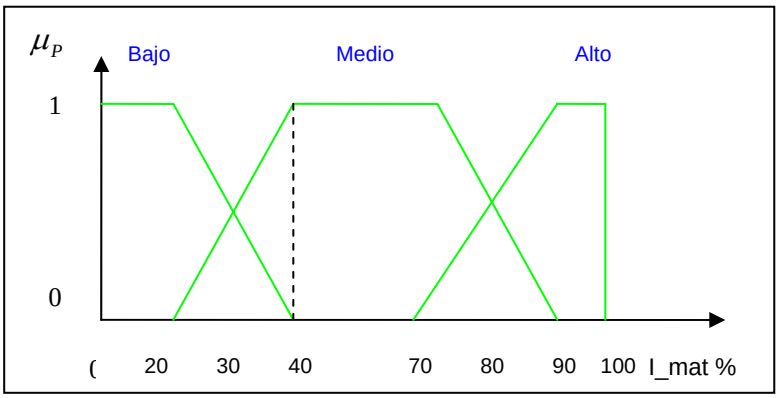


Figura 56 Emborronado Variable I_mat.

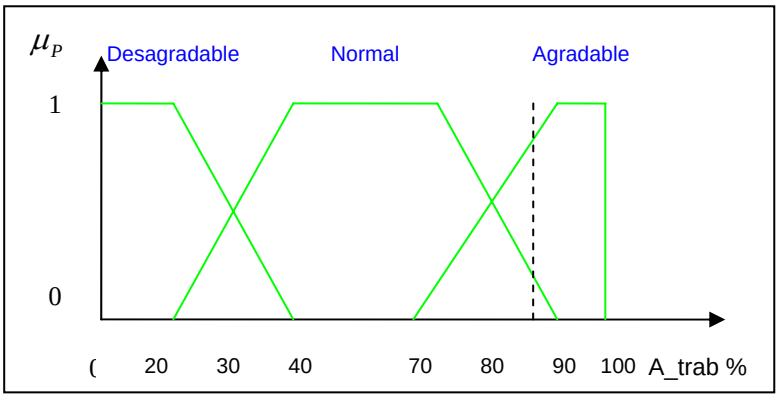


Figura 57 Emborronado Variable A_trab.

Del proceso anterior podemos concluir:

$$C\_doc \quad 0.5\ regular \quad 0.5\ malo$$
$$I\_mat \quad 0\ bajo \quad 1\ medio$$
$$A\_trab \quad 0.75\ agradable \quad 0.25\ normal$$

Procedemos a la activación y desactivación de reglas. Después del proceso de emborronado, encontramos que las reglas que se activan para este caso en particular son:

R_{10} *si* C_prof *es malo* Y I_mat *es bajo* Y A_trab *es normal entonces*
Re *nd es malo* Y *Aflue es baja*

R_{11} *si* C_prof *es regular* Y I_mat *es bajo* Y A_trab *es normal entonces*
Re *nd es malo* Y *Aflue es baja*

R_{13} *si* C_prof *es malo* Y I_mat *es medio* Y A_trab *es normal entonces*
Re *nd es malo* Y *Aflue es baja*

R_{14} *si* C_prof *es regular* Y I_mat *es medio* Y A_trab *es normal entonces*
Re *nd es aceptable* Y *Aflue es media*

R_{19} *si C_prof es malo Y I_mat es bajo Y A_trab es agradable entonces* Rend *es malo Y Aflue es media*

R_{20} *si C_prof es regular Y I_mat es bajo Y A_trab es agradable entonces* Rend *es malo Y Aflue es media*

R_{22} *si C_prof es malo Y I_mat es medio Y A_trab es agradable entonces* Rend *es malo Y Aflue es media*

R_{23} *si C_prof es regular Y I_mat es medio Y A_trab es agradable entonces* Rend *es aceptable Y Aflue es media*

Para este ejercicio se utilizara la intersección (mínimo) para los antecedentes de las reglas:

$$R_{10} : Min\{0.5 , 0 , 0.25\} = 0$$
$$R_{11} : Min\{0.5 , 0 , 0.25\} = 0$$
$$R_{13} : Min\{0.5 , 1 , 0.25\} = 0.25$$
$$R_{14} : Min\{0.5 , 1 , 0.25\} = 0.25$$
$$R_{19} : Min\{0.5 , 0 , 0.75\} = 0$$
$$R_{20} : Min\{0.5 , 0 , 0.75\} = 0$$
$$R_{22} : Min\{0.5 , 1 , 0.75\} = 0.5$$
$$R_{23} : Min\{0.5 , 1 , 0.75\} = 0.5$$

Agregación de Reglas.

Graficando las respectivas reglas activas para este proceso con respecto primero a la variable Rendimiento (Rend), y luego para la variable Afluencia (Aflue), tenemos:

Variable rendimiento.

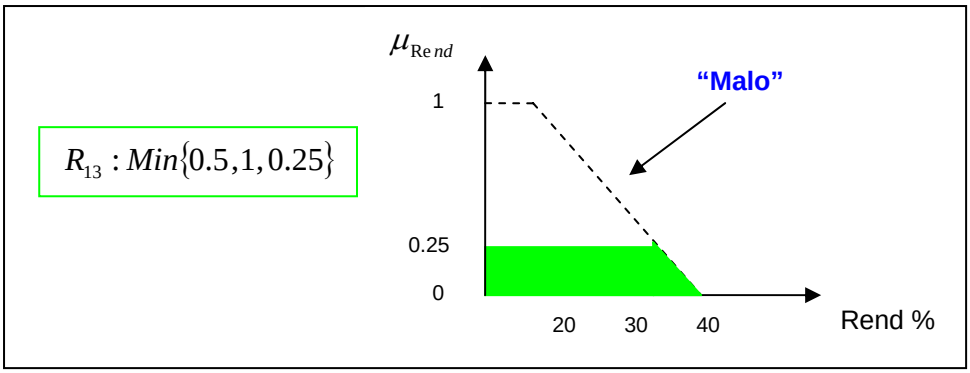


Figura 58 Regla #10 Variable Rendimiento.

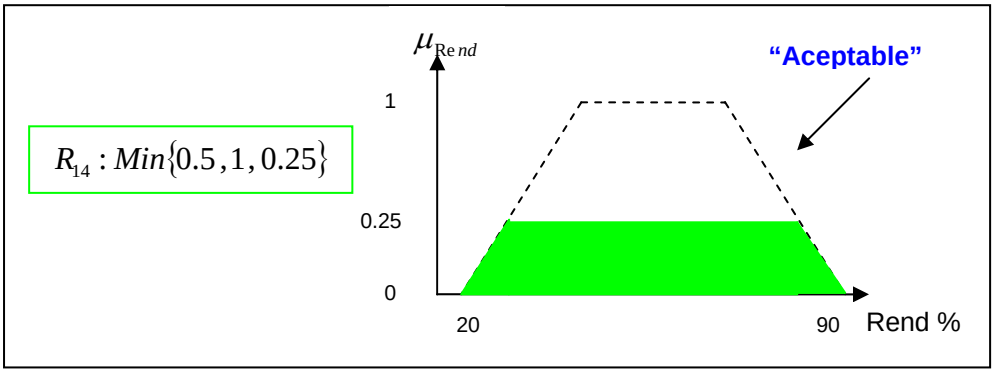


Figura 59 Regla # 14 Variable Rendimiento.

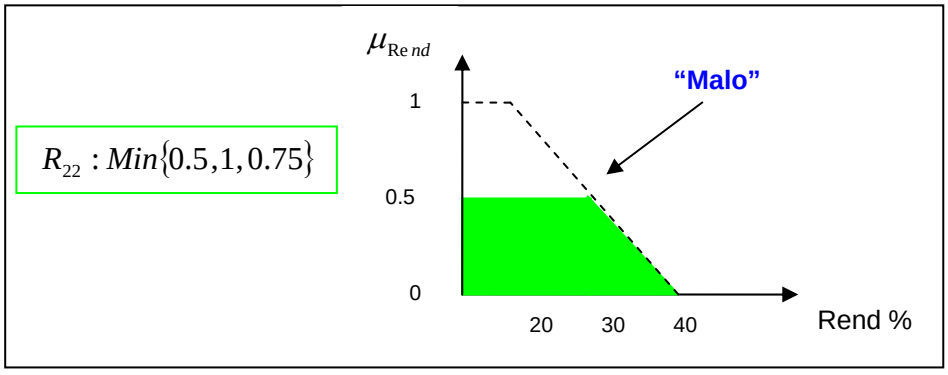


Figura 60 Regla # 22 Variable Rendimiento.

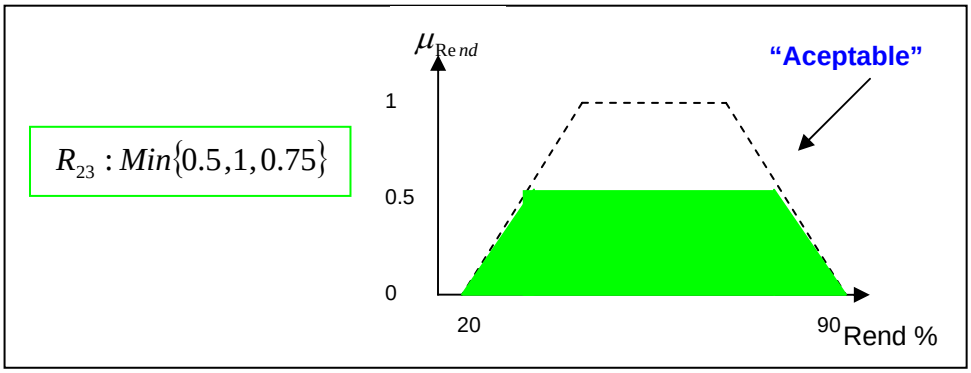


Figura 61 Regla # 23 Variable Rendimiento.

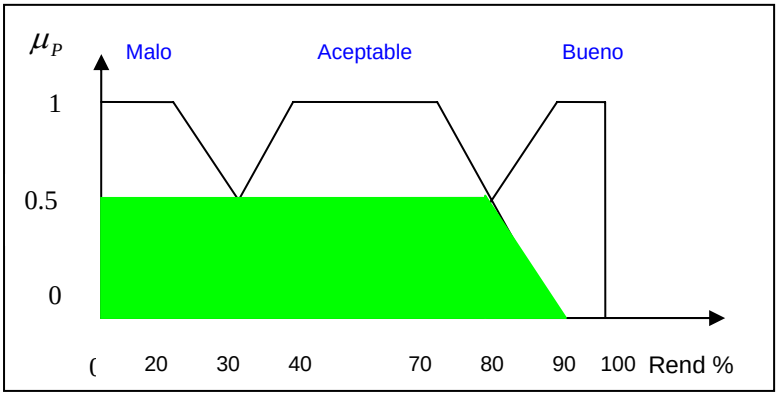


Figura 62 Agregación de Reglas Variable Rendimiento.

En la agregación obtenemos el conjunto borroso de salida al que debemos aplicarle un método de desemborronado para encontrar una respuesta cuantitativa, el más conveniente para este caso en particular es el centro de área discreto.

$$X = \frac{\sum_x x * \mu_A(x)}{\sum_x \mu_A(x)}$$

Reemplazando obtenemos:

$X = 43.8$

Variable Afluencia.

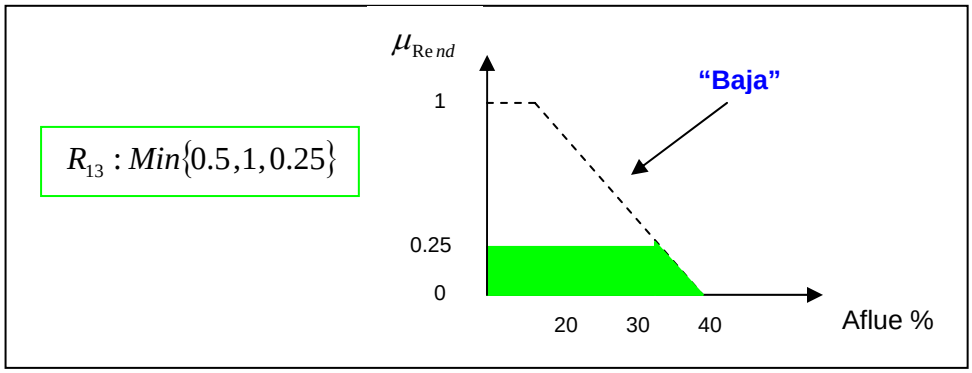


Figura 63 Regla # 13 Variable Afluencia.

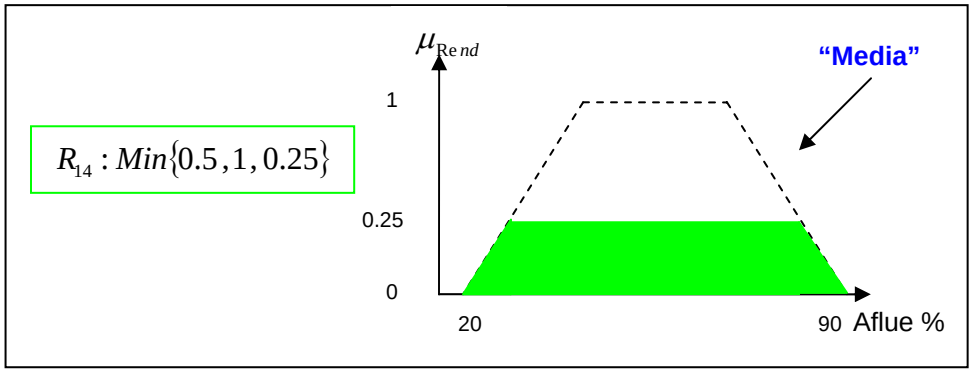


Figura 64 Regla # 14 Variable Afluencia.

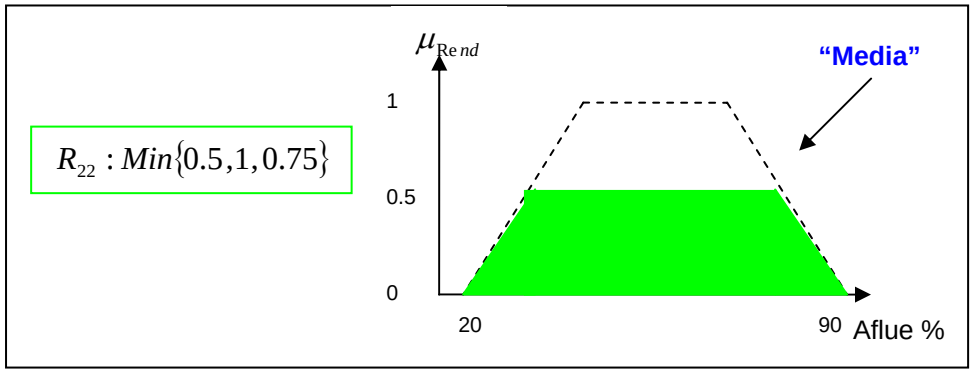


Figura 65 Regla # 22 Variable Afluencia.

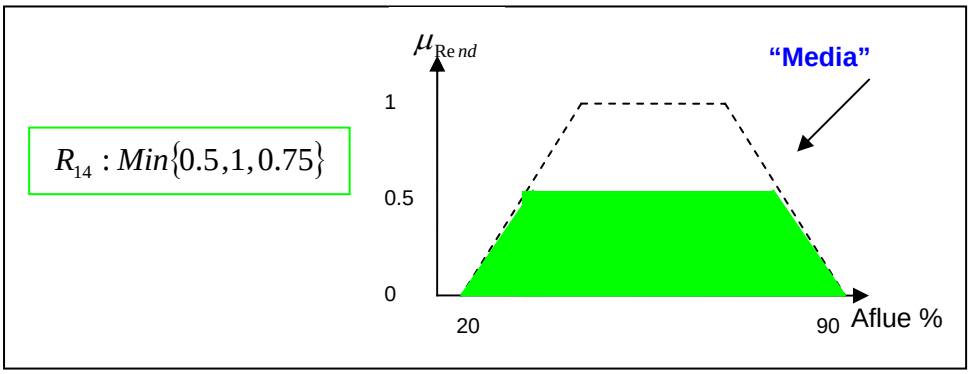


Figura 66 Regla # 23 Variable Afluencia.

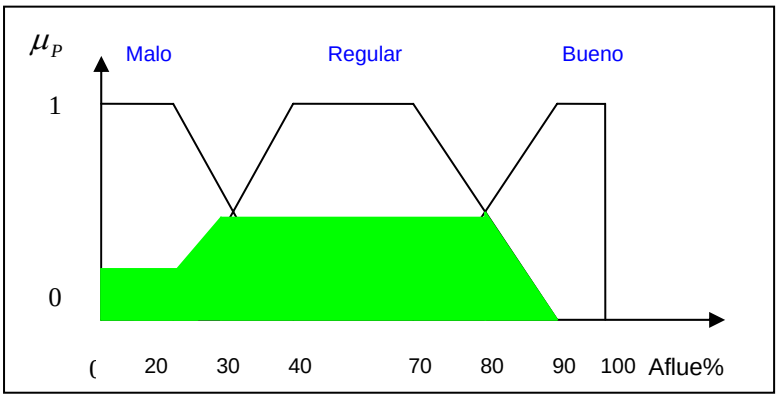


Figura 67 Agregación de Reglas Variable Afluencia.

En la agregación obtenemos el conjunto borroso de salida al que debemos aplicarle un método de desemborronado para encontrar una respuesta cuantitativa, el más conveniente para este caso en particular es el centro de área discreto.

$$X = \frac{\sum_x x * \mu_A(x)}{\sum_x \mu_A(x)}$$

Reemplazando obtenemos:

$$X = 48.6$$

3.4 MODELO BORROSO TSK

También conocido como el modelo de Sugeno, fue propuesto por Takagi, Sugeno y Kant como un esfuerzo para desarrollar un enfoque sistemático para generar las reglas borrosas en un conjunto de datos entrada – salida dado.

Una regla borrosa típica en un modelo de Sugeno tiene la forma:

$$IF\ "x"\ is\ A\ AND\ "y"\ is\ B\ THEN\ z = f(x, y)$$

Donde A y B son conjuntos borrosos en el antecedente, mientras que $z = f(x, y)$ es una función "crisp" en el Consecuente. Generalmente $f(x, y)$ es un polinomio.

3.4.1 MODELO BORROSO DE SUGENO CON DOS ENTRADAS Y UNA SALIDA

3.4.1.1 MODELADO BORROSO TIPO TSK DEL PROBLEMA DEL PÉNDULO INVERTIDO.

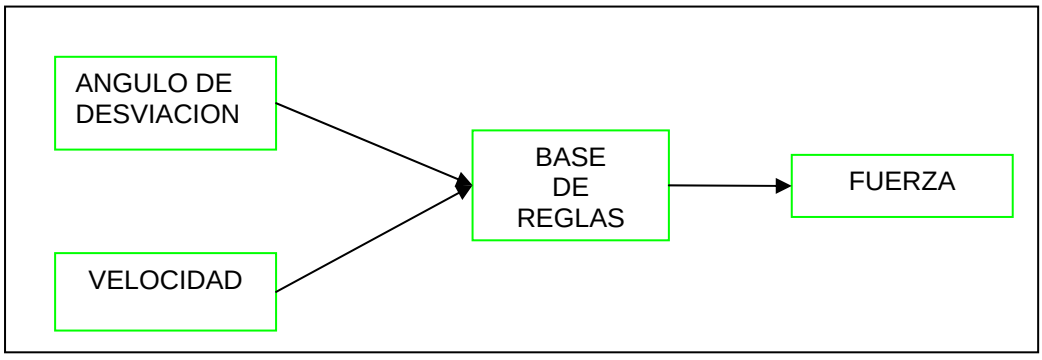


Figura 68 Esquema del ejemplo controlador TSK

Variables.

Las variables del proceso están determinadas de la siguiente manera, dos variables de entrada, la variable ángulo de desviación θ, conformada por dos funciones de pertenencia tipo triangular, y con rango de [0 1] (figura 69), la variable velocidad v, conformada por dos funciones de pertenencia tipo triangular, y con un rango de [0 1] (figura 70), la variable de salida denominada fuerza f, conformada por cuatro polinomios lineales que relacionan por medio de un muestreo y técnicas de mínimos cuadrados las entradas con las salidas deseadas (figura 71), un polinomio por cada regla del sistema.

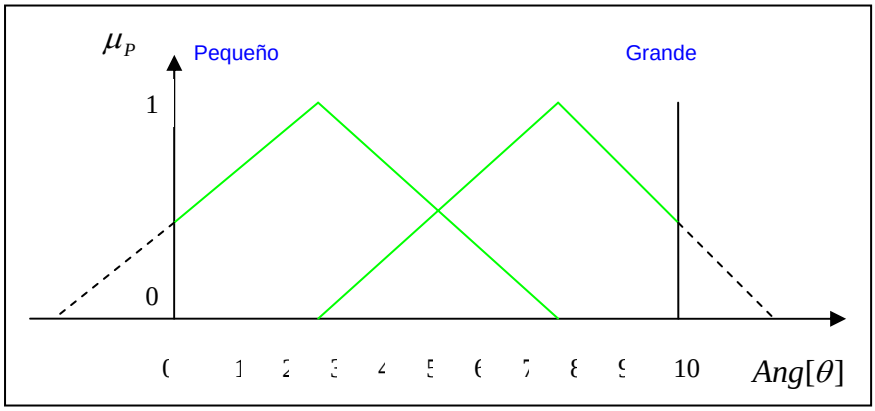


Figura 69 Variable Angulo de Desviación

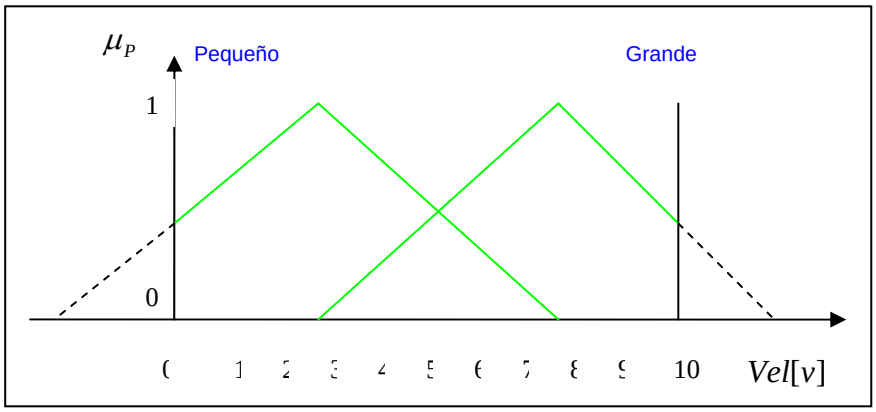


Figura 70 Variable Velocidad

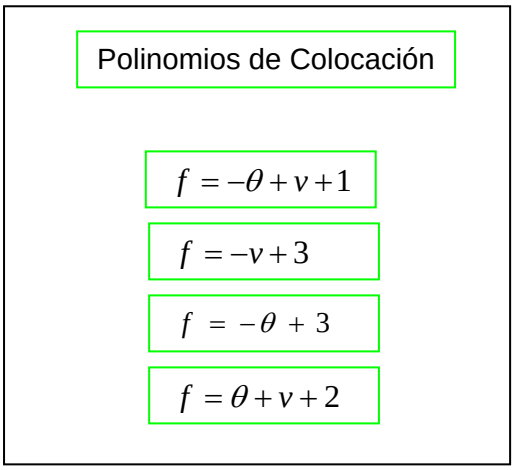


Figura 71 Polinomios de Aproximación.

3.4.1.2 BASE DE REGLAS

La base de reglas esta conformada de la siguiente manera:

R_1 *si* θ *es pèqueño AND v es pequeño THEN* $f = -\theta + v + 1$

R_1 *si* θ *es pèqueño AND v es grande THEN* $f = -v + 3$

R_1 *si* θ *es grande AND v es pequeño THEN* $f = -\theta + 3$

R_1 *si* θ *es grande AND v es grande THEN* $f = \theta + v + 2$

Después de especificar el sistema borroso se quiere encontrar la accione a tomar para el siguiente caso particular de entrada.

Angulo de desviación $\theta = 3$ Velocidad $v = 6$

Aplicando el proceso de emborronado:

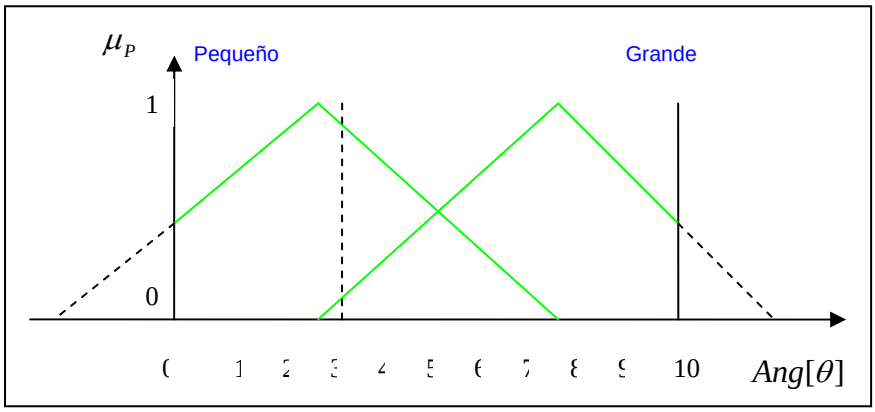


Figura 72 Emborronado Variable Angulo de colocación

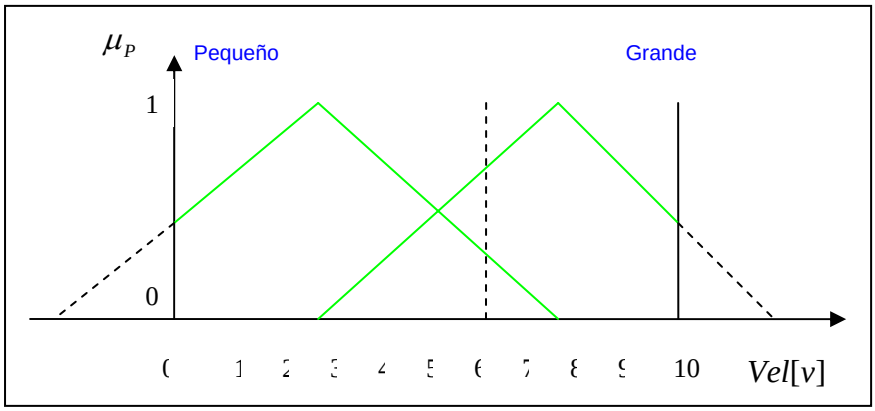


Figura 73 Emborronado Variable Velocidad.

Aplicando la formula para la función triangular tenemos los siguientes valores:

$$0.9\ pequeño \qquad 0.1\ grande$$
$$0.3\ pequeño \qquad 0.9\ grande$$

Para este ejemplo todas las reglas han quedado activas, utilizando el minimo para el operador AND, tenemos:

$$R_1 : Min\,[0.9 \quad 0.3] = 0.3$$
$$R_2 : Min\,[0.9 \quad 0.7] = 0.7$$
$$R_3 : Min\,[0.1 \quad 0.3] = 0.1$$
$$R_4 : Min\,[0.1 \quad 0.7] = 0.1$$

Obtenemos el vector $W_i = [0.3 \ 0.7 \ 0.1 \ 0.1]$

El siguiente paso es obtener el vector Z_i, del conjunto de los polinomios que se activaron con las reglas para el caso específico $\theta = 3 \quad y \quad v = 6$.

$$f = -\theta + v + 1 = -3 + 6 + 1 = 4$$

$$f = -v + 3 = -6 + 3 = -3$$

$$f = -\theta + 3 = -3 + 3 = 0$$

$$f = \theta + v + 2 = 3 + 6 + 2 = 11$$

Obteniendo el vector $Z_i = [4 \quad -3 \quad 0 \quad 11]$

Aplicando el proceso de desemborronado:

$$X = \frac{\sum_{i=1}^{N} W_i * Z_i}{W_i}$$

Reemplazando:

$$X = \frac{[(0.3*4) + (0.7*-3) + (0.1*0) + (0.1*11)]}{[0.3 + 0.7 + 0.1 + 0.1]} = \frac{0.2}{1.2} = 0.167$$

Para el caso de $\theta = 3 \quad y \quad v = 6$ la acción a tomar es $f = 0.167$

El modelo de Sugeno es el candidato más popular para el modelado borroso basado en datos de muestreo.

4 DISEÑO Y SISTEMA COBOR 2.0

4.1 DISEÑO

El diseño del software se realizó utilizando el lenguaje Unificado de Modelado (Unified Modeling Language, UML), donde UML es un lenguaje o forma estandarizada de desplegar y escribir la estructura de un software o sistema por medio de conceptos orientados a objetos. Puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

A continuación se presentan los diferentes elementos de diseño del proyecto COBOR 2.0.

4.1.1 DIAGRAMA DE CLASES

Un diagrama es una presentación gráfica de un conjunto de elementos, que la mayoría de las veces se dibuja como un grafo conexo de nodos (elementos) y arcos (relaciones). Las Clases son la base para la construcción de un sistema orientado a objetos. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Un Diagrama de Clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Una relación es una conexión entre elementos, en el modelado orientado a objetos las tres relaciones más importantes son las *dependencias* (son relaciones de uso, un cambio en un elemento puede afectar a otro elemento), *generalizaciones* (conectan clases generales con otras mas especializadas, por ejemplo subclase/superclase o hijo/padre) y las *asociaciones* son

relaciones estructurales entre instancias, cuando existen elementos compuestos por otros elementos).

COBOR 2.0 (Software para el Control Borroso), se realizó bajo el paradigma de la Ingeniería del software llamado ***Construcción de prototipos***, que nos ofrece el mejor enfoque.

El paradigma de construcción de prototipos, metodología usada en este proyecto, es un proceso de desarrollo de software que comienza con la recolección de requisitos. Se definen unos objetivos globales para el software, identificando los requisitos conocidos generando un diseño rápido que se centra en esos aspectos del software que serán visibles. (Por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo es evaluado y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para cumplir los objetivos, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

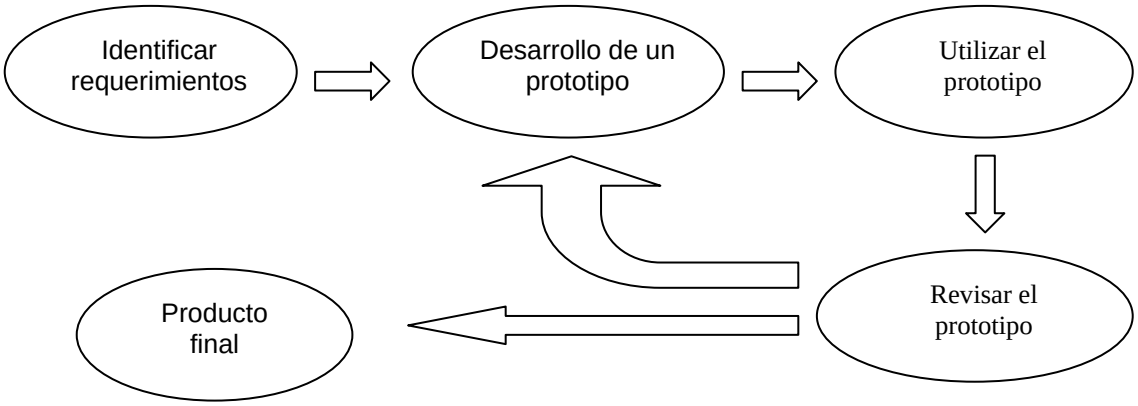


Figura 74 Proceso de Evaluación de Prototipos.

El paradigma de creación de prototipos es abierto. El enfoque abierto, denominado prototipo evolutivo, emplea el prototipo como primera parte de una actividad de análisis a la que seguirá el diseño y construcción. El prototipo del software es la primera evolución del sistema terminado.

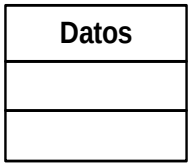
4.1.1.1 DIAGRAMA DE CLASES DEL PROYECTO COBOR 2.0.

Para la realización del proyecto COBOR 2.0 se elaboraron cuatro prototipos en los cuales se fueron agregando progresivamente las clases correspondientes para añadir las funcionalidades o componentes requeridos.

4.1.1.1.1 PROTOTIPO NO 1.

El diagrama de clases prototipo No 1 se muestra a continuación:

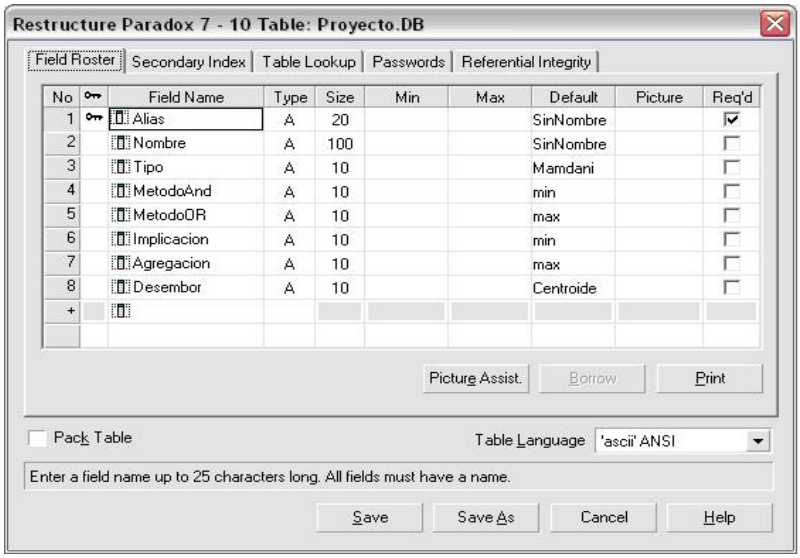
Figura 75 Diagrama de Clases del Prototipo No 1



En este prototipo se implementaron los métodos de:

- Entrada de datos.
- Manipulación de datos permitiendo Agregar, Eliminar y Modificar conjuntos y reglas.
- Manipular subconjuntos de datos, con las funcionalidades dadas por el sistema.

El sistema COBOR 2.0 se desarrolló con una Estructura de bases de datos que se realizó en PARADOX, como un manejo de archivos Indexados, a continuación se presenta la estructura de las tablas que se diseñaron:



Se presenta un pequeño ejemplo del almacenamiento de datos, para ello crea los campos que requiera el proyecto.

El código fuente que se desarrolló en el lenguaje DELPHI para este prototipo es el siguiente:

```
unit UndVariables;

interface
uses
 Forms,SysUtils,BDE, UndDataModulo,Graphics, DBTables, Math, Windows,
 StdCtrls, chart,ComCtrls, StrUtils, Series, TeEngine, db;

Const
 Colores :array[1..30] of Tcolor=
 (ClRed, ClYellow, ClBlack, ClLime, ClAqua,Clmaroon, ClNavy, ClFuchsia, Clteal,
Clpurple,$000080FF,$00FF80FF,$00408000,ClRed, ClYellow, ClBlack, ClLime,
ClAqua,Clmaroon, ClNavy, ClFuchsia, Clteal,
Clpurple,$000080FF,$00FF80FF,$00408000,ClRed, ClYellow, ClBlack, ClLime );
Var
  DirEjecutable: string; // path del ejecutable
  DirProyecto: string; // path del proyecto
  DirDatos: string; //path de la carpeta de datos
  DirEstructura: string; //path de la estructura de datos
  Mensaje0 :String ='El Codigo ya existe';
  TitMensajeError : String ='COBOR 2.0 ERROR';
```

```
TitMensajeExc: String ='COBOR 2.0 MENSAJE';
alpha: double; //intervalo del mouse

//******* 0..49 significa hasta 50 reglas
//******* 0..50  significa que se tiene en cuenta la linea adicional para resultado

rangos: Array[0..14, 0..2] of Tlabel; //aca quedan consignados min, max, paso
graficas: Array[0..14,0..10000] of TChart;  //la grafica como tal
titulos: Array[0..14] of Tlabel;         //Donde está el nombre de la variable de ENTRADA
alturas: Array[0..14,0..9999] of TEdit;    //valor de la altura y
titvalor: Array[0..14] of TEdit;         //el valor de las x
FMNombres: Array[0..14,0..10000] of TEdit;  //FuncionMiembro Nombre
FMNot:Array[0..14,0..9999] of TCheckBox;   //FuncionMiembro Not
LineasMarca: Array[0..14] of TTrackBar;  //slide
ReglasConector: Array[0..2,0..9999] of TLabel; // en el 1o va el numero de la regla, en el
segundo la conjuncion
VarEntradas: integer;
NoVariables:integer;


Procedure CopiarTablaYa1(NombreTbOrg,NombreTbDest,Origen,Destino:String);
Procedure EjecutarConsulta(QrConsulta:TQuery;Cadena:String;Tipo:byte;Valor:byte);
procedure AddRI(Master, Detail: TTable; RIName: string);         //; ModOp, DelOp:
RINTQual


//******* FUNCIONES MATHS****************
function gauss(x,c,d:double):double;
function sigmoidal(x,a,c:double):double;
function bell(x,b,c,d: double):double;
function FuncNot(y:double;Fnot: boolean):double;

//******* FUNCIONES ALTURA ************************
function AltTriangular(x,a,b,c:double):double;
function AltTrapezoidal(x,a,b,c,d:double):double;
function CalcularAltura(x:double; VarFMNombre: string): double;

//******* FUNCIONES DE IMPLICACION ****************
Function ImpPro (Noregla: Integer): double;
Function ImpMax (Noregla: Integer): double;
Function ImpMin (Noregla: Integer): double;
Function Implicacion(): boolean;

//******* FUNCIONES DE DESEMBORRONADO ************
Function DesCentroide(Serie: TChartSeries):double;
```

```
  Function DesCentroSumas(x,y:integer): double;
  Function DesPMaximo(Serie:TChartSeries):double;
  Function DesUMaximo(Serie:TChartSeries):double;
  Function DesMMaximo(Serie:TChartSeries):double;

  //******** FUNCIONES DE PRESENTACION ***************
  Function NotNatural(FM:string; esNot: Boolean; conj: string):string;

implementation
uses UndFrmResultados;

Procedure CopiarTablaYa1(NombreTbOrg,NombreTbDest,Origen,Destino:String);
Begin
 if not FileExists(Destino+NombreTbDest) then

DbiCopyTable(DataModulo.DBCobor.Handle,False,PChar(Origen+NombreTbOrg),nil,PCh
ar(Destino+NombreTbDest));
    End;

Procedure EjecutarConsulta(QrConsulta:TQuery;Cadena:String;Tipo:byte;Valor:byte);
Begin
 With QrConsulta do
  begin
   DisableControls;
   try
    Close;
    Sql.Clear;
    Case Valor of
     0:Sql.Add(Cadena);
     1:Sql.LoadFromFile(Cadena);
    end;
    Prepare;
    Case Tipo of
     0:ExecSQL;
     1:Open;
     2: begin
        ExecSQL;
        Open;
        end;
    end;
    finally
    EnableControls;
   end;
  end;
End;
```

```
procedure AddRI(Master, Detail: TTable; RIName: string);
var
  MasterProps, DetailProps: CURProps;
  hDb: hDBIDb;
  TableDesc: CRTblDesc;
  Op: CROpType;
  RInt: RINTDesc;
  MIndex, DIndex: IDXDesc;
  MNo, DNo: Word;
  ModOp, DelOp: RINTQual;

begin

ModOp:= rintCASCADE;
DelOp:= rintRESTRICt;
  if Master.Active = False then
    raise EDatabaseError.Create('Master table: ' + Master.TableName +
                          ' is not opened');
  if Detail.Active = False then
    raise EDatabaseError.Create('Detail table: ' + Detail.TableName +
                          ' is not opened');

  if Master.Exclusive = False then
    raise EDatabaseError.Create('Table: ' + Master.TableName +
                          ' must be opened exclusively');
  if Detail.Exclusive = False then
    raise EDatabaseError.Create('Table: ' + Detail.TableName +
                          ' must be opened exclusively');

  // Make sure the tables are opened with an index and get their descriptors...
  FillChar(DIndex, sizeof(DIndex), 0);
  FillChar(MIndex, sizeof(MIndex), 0);
  Check(DbiGetIndexDesc(Detail.Handle, 0, DIndex));
  Check(DbiGetIndexDesc(Master.Handle, 0, MIndex));

  // Get the table properties to determine table type...
  Check(DbiGetCursorProps(Master.Handle, MasterProps));
  Check(DbiGetCursorProps(Detail.Handle, DetailProps));

  // If the table is not a Paradox table, raise an error...
  if (MasterProps.szTableType <> szPARADOX) and
     (MasterProps.szTableType <> szDBASE) then
     raise EDatabaseError.Create('Master table: ' + Master.TableName +
```

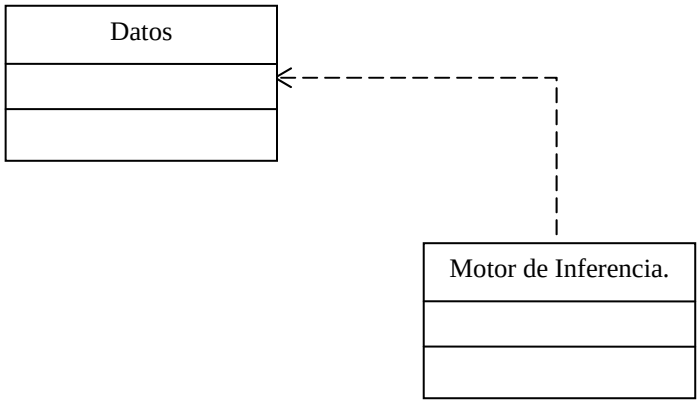
```
                        ' must be a Paradox or dBASE table type');
if (DetailProps.szTableType <> szPARADOX) and
   (DetailProps.szTableType <> szDBASE) then
   raise EDatabaseError.Create('Detail table: ' + Detail.TableName +
                        ' must be a Paradox or dBASE table type');
if MasterProps.szTableType <> DetailProps.szTableType then
   raise EDatabaseError.Create('Master and Detail tables must be of same type');

// Blank out the structures...
FillChar(TableDesc, sizeof(TableDesc), 0);
FillChar(RInt, sizeof(RInt), 0);
//  Get the database handle from the table's cursor handle...
Check(DbiGetObjFromObj(hDBIObj(Master.Handle), objDATABASE, hDBIObj(hDb)));
// Put the table name in the table descriptor...
StrPCopy(TableDesc.szTblName, Detail.TableName);
// Put the table type in the table descriptor...
TableDesc.szTblType := MasterProps.szTableType;
// Set the operation type...
Op := crADD;
TableDesc.pecrRintOp := @Op;
// Set the amount of new RI descriptors...
TableDesc.iRintCount := 1;
// Connect the table descriptor to the RI descriptor...
TableDesc.printDesc := @RInt;
// Setup the RI descriptor...
// Put in the name of the RI...
StrPCopy(RInt.szRintName, RIName);
// Do the restructure on the dependent (detail) table...
RInt.eType := rintDEPENDENT;
// Add the master table name...
StrPCopy(RInt.szTblName, Master.TableName);
// Modify operations will be restricted (this can be changed to rintCASCADE)...
RInt.eModOp := ModOp;
// Delete operations will be restricted (NOTE: rintCASCADE will not work)...
RInt.eDelOp := DelOp;
// Only one field in link...
RInt.iFldCount := 1;
// If the tables are Paradox, then put the associated field numbers in the descriptor...
if (MasterProps.szTableType = szPARADOX) then
begin
  //
  if RInt.eDelOp = rintCASCADE then
    raise EDatabaseError.Create('Cannot use cascading delete RI with Paradox tables');
  // Put the detail field index in the array...
  RInt.aiThisTabFld := DIndex.aiKeyFld;
```

```
    // Put the master field index in the array...
    RInt.aiOthTabFld := MIndex.aiKeyFld;
  end;

  // If the tables are dBASE, then put the sequence number in the descriptor...
  if MasterProps.szTableType = szDBASE then
  begin
    Check(DbiGetIndexSeqNo(Master.Handle,  MIndex.szName,  MIndex.szTagName,  0,
MNo));
    Check(DbiGetIndexSeqNo(Detail.Handle,  DIndex.szName,  DIndex.szTagName,  0,
DNo));
    // Put the detail field index in the array...
    RInt.aiThisTabFld[0] := DNo;
    // Put the master field index in the array...
    RInt.aiOthTabFld[0] := MNo;
  end;

  try
    Master.Close;
    Detail.Close;
    Check(DbiDoRestructure(hDb, 1, @TableDesc, nil, nil, nil, FALSE));
  finally
    Master.Open;
    Detail.Open;
  end;
end;
```

4.1.1.1.2 PROTOTIPO NO 2.

El diagrama de clases de este prototipo se aprecia en la siguiente figura:

Figura 76 Diagrama de Clases del Prototipo No 2



En este prototipo se agrego la clase Motor de inferencia la cual permite la fácil creación de la base de reglas borrosas con todas las características que se tienen en cuenta para el normal funcionamiento del controlador lógico borroso. Teniendo la posibilidad de crear las reglas de dos formas según la necesidad del experto que maneje el proceso.

El código fuente que se generó para este prototipo es el siguiente:

```
//****************** F U N C I O N E S  M A T H S ******************
function gauss(x,c,d:double):double;
begin
//e^(-0.5)[(X-c)/d]^2
gauss := exp(-0.5*power((x-c)/d,2));
end;

function sigmoidal(x,a,c:double):double;
begin
//1/(1+e^(-a*(x-c)))
sigmoidal :=  1/(1+exp((a)*(-1)*(x-c)));
end;

function bell(x,b,c,d: double):double;
begin
if (b<=0) then
  begin
//    application.MessageBox('El valor de la variable a debe ser positiva','Error');
    bell:=0;
  end
else
begin
bell:= 1/(1+ power(abs((x-c)/d),2*b)) ;
end;
// 1/(1+  power(abs((x-c)/d),2*a)) a>0
end;


//****************** F U N C I O N E S   A L T U R A *******************

function AltTriangular(x,a,b,c:double):double;
begin
AltTriangular:=0;
if x<=a then AltTriangular:=0
else if ((x>=a) and (x<=b)) then AltTriangular:=((x-a)/(b-a))
else if ((x>=b) and (x<=c)) then AltTriangular:=((c-x)/(c-b))
```

```
else if  (x>=c) then AltTriangular:=0;
end;


function AltTrapezoidal(x,a,b,c,d:double):double;
begin
AltTrapezoidal:=0;
if x<=a then AltTrapezoidal:=0
else if ((x>=a) and (x<=b)) then AltTrapezoidal:=((x-a)/(b-a))
else if ((x>=b) and (x<=c)) then AltTrapezoidal:= 1
else if ((x>=c) and (x<=d)) then AltTrapezoidal:=((d-x)/(d-c))
else if  (x>=d) then AltTrapezoidal:=0;
end;

function FuncNot(y:double;Fnot: boolean):double;
begin
 If Fnot then
   FuncNot:=(-y)+1
  else FuncNot:=y;
end;

function CalcularAltura(x:double; VarFMNombre: string): double;
var tipofuncion: string;
a,b,c,d: double;
begin
CalcularAltura:=0;
  if DataModulo.QFMiembros.Locate('FNombre',VarFMNombre,[]) then
    begin
    TipoFuncion:=DataModulo.QFMiembrosFTipo.AsString;

  if (TipoFuncion ='triangular') then
     begin
      a:=DataModulo.QFMiembrosFParam1.AsFloat;
      b:=DataModulo.QFMiembrosFParam2.AsFloat;
      c:=DataModulo.QFMiembrosFParam3.AsFloat;
      CalcularAltura:= AltTriangular(x,a,b,c);
     end
  //trapezoidal
     else if (tipoFuncion='trapezoidal') then
      begin
      a:=DataModulo.QFMiembrosFParam1.AsFloat;
      b:=DataModulo.QFMiembrosFParam2.AsFloat;
      c:=DataModulo.QFMiembrosFParam3.AsFloat;
      d:=DataModulo.QFMiembrosFParam4.AsFloat;
      CalcularAltura:= AltTrapezoidal(x,a,b,c,d);
```

```
    end
  // gbell  1/(1+  power(abs((x-c)/d),2*a)) a>0
  else if (tipoFuncion ='gbell') then
    begin
     c:=DataModulo.QFMiembrosFParam1.Value;
     d:=DataModulo.QFMiembrosFParam2.Value;
     b:=DataModulo.QFMiembrosFParam3.Value;
     CalcularAltura:= bell(x,b,c,d);
    end
  //gaussiana  e^(-0.5)[(X-c)/d]^2
  else if (tipoFuncion='gaussiana') then
    begin
     c:=DataModulo.QFMiembrosFParam1.Value;
     d:=DataModulo.QFMiembrosFParam2.Value;
     CalcularAltura:= gauss(x,c,d);
    end
  //sigmoidal 1/(1+e^(-a*(x-c)))
  else if (tipoFuncion='sigmoidal') then
    begin
     c:=DataModulo.QFMiembrosFParam1.Value;
     a:=DataModulo.QFMiembrosFParam2.Value;
     CalcularAltura:= sigmoidal(x,a,c);
    end;
end
else
CalcularAltura:=0; //en los no aplica

end;
```

Si a futuro se desea incluir una nueva función de implicación, ésta debe hacerse mediante el código que se presenta a continuación:

```
//********** F U N C I O N E S   D E   I M P L I C A C I O N ************************

Function ImpMin (Noregla: Integer ): double;
var i: integer;
 auxmin: double;
begin
 auxmin:= StrtoFloat(Alturas[0, Noregla].Text);
 for i:= 1 to NoVariables do
 if     ((Alturas[i,     Noregla].Text<>'')     and     (leftstr(titulos[i].Caption,1)     ='E')     and
(FmNombres[i,Noregla].Text <> 'NoAplica' ))  then
 if (StrtoFloat(Alturas[i, Noregla].Text)< auxmin) then
    auxMin:= StrtoFloat(Alturas[i, Noregla].Text);
 ImpMin:= auxmin;
```

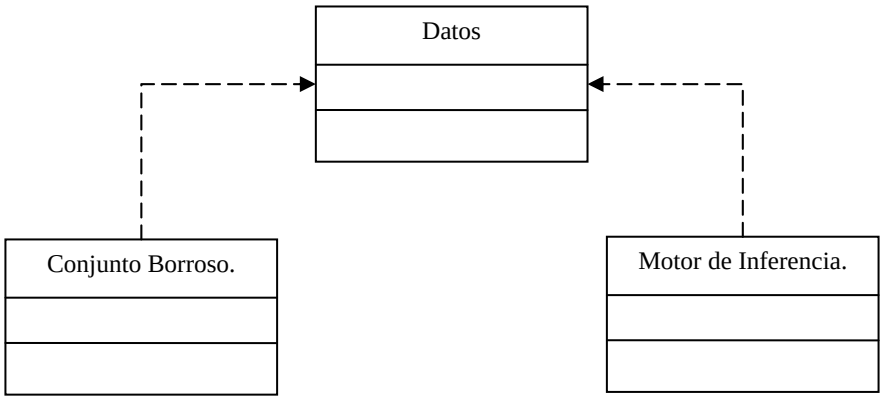
```
end;

Function ImpMax (Noregla: Integer): double;
var i: integer;
 auxmax: double;
begin
 auxmax:= StrtoFloat(Alturas[0, Noregla].Text);
 for i:= 1 to NoVariables do
 if ((Alturas[i, Noregla].Text<>'') and (leftstr(titulos[i].Caption,1)='E') and
(FmNombres[i,Noregla].Text <> 'NoAplica' )) then
 if (StrtoFloat(Alturas[i, Noregla].Text)> auxmax) then
    auxMax:= StrtoFloat(Alturas[i, Noregla].Text);
 ImpMax:= auxmax;
end;

Function ImpPro (Noregla: Integer): double;
var i: integer;
 AuxPro: double;
begin
 AuxPro:= StrtoFloat(Alturas[0, Noregla].Text);
 for i:= 1 to NoVariables do
 if ((Alturas[i, Noregla].Text<>'') and (leftstr(titulos[i].Caption,1)='E') and
(FmNombres[i,Noregla].Text <> 'NoAplica' )) then
     auxPro:= auxPro* StrtoFloat(Alturas[i, Noregla].Text);
 ImpPro:= auxPro;
end;

Function Implicacion:boolean;
var i,j:integer;
AlturaImp: double;
begin
  for j:=0 to DataModulo.TReglas.RecordCount-1 do    // esto se hace por cada regla
     begin
       if ReglasConector[1,j].Caption = 'Y' then  //si el conector es Y
             begin
               if DataModulo.TProyectoMetodoAnd.AsString = 'min' then
                  AlturaImp:=ImpMin(j)
                else //prod
                 AlturaImp:=ImpPro(j)
             end
           else //si el conector es O
             begin
                if DataModulo.TProyectoMetodoOR.AsString = 'max' then
                 AlturaImp:=ImpMax(j)
                else //probor
```

```
            AlturaImp:=ImpPro(j)
          end;
    for i:=0 to 14 do
    begin
    if (leftstr(titulos[i].Caption,1)='S') then
      begin
      Alturas[i,j].Text:= FormatFloat('##0.##0',AlturaImp);
      FrmResultados.RepintarSalida(i,j,AlturaImp);
      end;
    end;
   end;
end;
```

4.1.1.1.3 PROTOTIPO NO 3.

El diagrama de clases de este prototipo se aprecia en la siguiente figura:

Figura 77 Diagrama de Clases del Prototipo No 3



-En este prototipo se ha agregado la clase Conjunto Borroso en la que se maneja toda la información de los controladores borrosos, como lo son conjuntos borrosos, funciones de membresía.

En este prototipo también se implementaron los siguientes métodos:

- Entrada de información especifica.

- Emborronado, que nos permite transformar la información cuantitativa en información cualitativa, por medio de variables y modificadores lingüísticos.
- Buscar reglas activas.
- Operar las reglas activas.
- Operadores de implicación.
- Agregación de reglas.

Si a futuro se desea incluir una nueva función de desemborronado, ésta debe hacerse acorde a la programación desarrollada en el código fuente que se presenta a continuación:

```
//********** F U N C I O N E S   D E   D E S E M B O R R O N A D O ************

Function DesCentroide(Serie: TChartSeries):double;
var i: integer;
   suma, suma2:double;
begin
 suma:=0;
 i:=0;
 while i <= (Serie.Count-1)  do
   begin
   suma:= suma+(serie.XValues.Value[i] * serie.YValues.Value[i]);
   i:= i+1;
  end;
 suma2:=serie.YValues.Total;
 if suma2<>0 then  suma:= suma/suma2 else suma:=0;
 desCentroide:= suma;
end;

Function DesCentroSumas(x,y:integer):double; //x,y :identifica la grafica
var i,ki : integer;
    suma, suma2,sumak: double;
begin
 i:=0;   sumaK:=0;
 while i<=(graficas[x,y].series[0].Count-1) do
 begin
  ki:=0;   suma:=0;
 while ki <= (y-1)  do
   begin
   suma:= suma+(graficas[x,ki].series[3].YValues.Value[i]);
   ki:= ki+1;
```

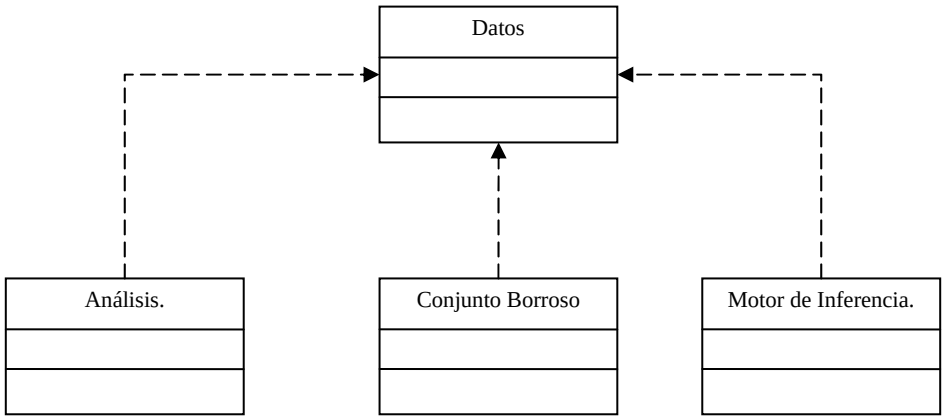
```
  end;
  sumaK:= sumak+(graficas[x,y].series[0].XValues.Value[i]*suma);
  i:=i+1;
 end;
   ki:=0;      suma2:=0;
 while ki <= (y-1)  do
   begin
   suma2:=  suma2 +graficas[x,ki].Series[3].YValues.Total;
   ki:= ki+1;
  end;
  if suma2<>0 then  sumak:= sumak/suma2 else sumak:=0;
  DesCentroSumas:= sumak;
end;

Function DesPMaximo(Serie:TChartSeries):double;
var i: integer;
    Maximo: double;
begin
 DespMaximo:=0;
 Maximo:=serie.YValues.MaxValue;
 i:=0;
 while (i <= (Serie.Count-1)) and (serie.YValues.Value[i]<> Maximo)  do
   begin
   i:= i+1;
  end;
 DesPMaximo:= serie.xValues.Value[i];
end;

Function DesUMaximo(Serie:TChartSeries):double;
var i: integer;
    Maximo: double;
begin
 DesUMaximo:=0;
 Maximo:=serie.YValues.MaxValue;
 i:=(Serie.Count-1);
 while (i >=0) and (serie.YValues.Value[i]<> Maximo)  do
   begin
   i:= i-1;
  end;
 DesUMaximo:= serie.xValues.Value[i];
end;

Function DesMMaximo(Serie:TChartSeries):double;
var i,cont: integer;
    Maximo, suma: double;
```

```
begin
 DesMMaximo:=0;
 cont:=0;
 Maximo:=serie.YValues.MaxValue;
 i:=0;
 while (i <=Serie.Count-1)   do
   begin
    if (serie.YValues.Value[i] = Maximo) then
    begin
    suma:=suma+serie.xValues.Value[i];
    cont:=cont+1;
    end;
    i:= i+1;
  end;
 DesMMaximo:= suma/cont;
end;
```

4.1.1.1.4 PROTOTIPO NO 4.

El diagrama de clases de este prototipo se aprecia en la siguiente figura:

Figura 78 Diagrama de Clases del Prototipo No 4



En este prototipo se agrego la clase Análisis que permite la toma de decisiones del usuario usando como referencia los resultados obtenidos por la herramienta, ya sea en forma gráfica o analítica, también se dará a conocer la solución del proceso paso a paso que es un objetivo del proyecto para dar más flexibilidad y entendimiento a la herramienta. El código fuente que se desarrolló en este prototipo es el siguiente:

```
//********** F U N C I O N  D E  P R E S E N T A C I O N ********************
Function NotNatural(FM:string; esNot: Boolean; conj: string):string;
var cadena: string;
    VNombre, Entonces: string;
begin
 IF (FM = '') Then NotNatural:='' else
 begin
 cadena:= 'Select V.VNombre Nombre, V.Vtipo Tipo from FMiembros.DB M, Variables V
where V.VNombre = M.VNombre  and M.FNombre = '+QuotedStr(FM);
 EjecutarConsulta(DataModulo.Qry_Consulta,Cadena,2,0);
 VNombre:= DataModulo.Qry_Consulta.FieldByName('Nombre').AsString;
 If DataModulo.Qry_Consulta.FieldByName('Tipo').AsString = 'S' then
 begin
 Entonces := ' Entonces (';
 conj:=';'
 end
 else Entonces:= ' (';

 if esNot then
 NotNatural:= Entonces+VNombre+' es No '+FM+') '+conj
 else NotNatural:=Entonces+VNombre+' es '+FM+') '+conj;
 DataModulo.Qry_Consulta.Close;
 end;
end;


unit UndFrmVariables;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, Buttons, TeEngine, Series, TeeProcs, Chart, StdCtrls, Math,
  Mask, DBCtrls, ExtCtrls, RxDBComb, Grids, DBGrids, RXDBCtrl,db, RxCombos,
  OleCtrls, Chartfx3, ImgList, TeeFunci;

type
```

```
TFrmVariables = class(TForm)
  Panel3: TPanel;
  Chart1: TChart;
  Panel2: TPanel;
  Label6: TLabel;
  Label7: TLabel;
  Label8: TLabel;
  Label9: TLabel;
  DBEdit3: TDBEdit;
  DBEd_Param2: TDBEdit;
  DBEd_Param3: TDBEdit;
  DBEd_Param4: TDBEdit;
  DBEd_Param1: TDBEdit;
  RxDBCB_Mamdani: TRxDBComboBox;
  RxDBGrid1: TRxDBGrid;
  Label2: TLabel;
  Label1: TLabel;
  DBEdit2: TDBEdit;
  Label3: TLabel;
  Label4: TLabel;
  DBEdit5: TDBEdit;
  DBEdit6: TDBEdit;
  DBEdit8: TDBEdit;
  DBEdit7: TDBEdit;
  Label5: TLabel;
  DBNavigator1: TDBNavigator;
  Chart_Origen: TChart;
  FastLineSeries1: TFastLineSeries;
  Lbl_Param1: TLabel;
  Lbl_Param2: TLabel;
  Lbl_Param3: TLabel;
  Lbl_Param4: TLabel;
  Image_gauss: TImage;
  Image_triangular: TImage;
  Image_trapezoidal: TImage;
  Image_sigmoidal: TImage;
  Image_bell: TImage;
  Series1: TPointSeries;
  RxDBComboBox2: TRxDBComboBox;
  DBNavigator2: TDBNavigator;
  RxDBCB_Sugeno: TRxDBComboBox;
  procedure SpeedButton1Click(Sender: TObject);
  procedure RxDBGrid1GetCellParams(Sender: TObject; Field: TField;
    AFont: TFont; var Background: TColor; Highlight: Boolean);
  procedure PintarFMiembro;
```

```
    procedure Chart1ClickSeries(Sender: TCustomChart; Series: TChartSeries;
      ValueIndex: Integer; Button: TMouseButton; Shift: TShiftState; X,
      Y: Integer);
   function NoSerie(nombre:string):integer;
   procedure PintarVariable;
   procedure Chart1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
   procedure RxDBComboBox2Change(Sender: TObject);
   procedure ActivarVar;
    procedure RxDBGrid1DrawColumnCell(Sender: TObject; const Rect: TRect;
      DataCol: Integer; Column: TColumn; State: TGridDrawState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmVariables: TFrmVariables;


implementation
Uses UndDataModulo,UndVariables;

{$R *.dfm}


procedure TFrmVariables.SpeedButton1Click(Sender: TObject);
var
serie1:TchartSeries;
i:integer;
begin
PintarFMiembro;
end;


function TFrmVariables.NoSerie(nombre:string):integer;
var i: integer;
 value:integer;
 totalseries: integer;
begin
i:=-1;
value := -1;  // -1 si no existe
totalseries:= chart1.SeriesCount;
```

```
while (i < totalseries-1)and (value = -1) do
 begin
 Inc(i);
 if chart1.Series[i].Name = nombre then value:=i;
 end;
Noserie:=value;
end;

procedure TFrmVariables.ActivarVar;
begin
     if (DataModulo.TVariablesVTipo.AsString = 'S') and
        (DataModulo.TProyectoTipo.AsString = 'Sugeno') then
       begin
          RxDBCB_Sugeno.Enabled:=true;
          RxDBCB_Sugeno.Visible:=true;
          RxDBCB_Mamdani.Enabled:=false;
          RxDBCB_Mamdani.Visible:=false;
       end
       else
        begin
          RxDBCB_Sugeno.Enabled:=false;
          RxDBCB_Sugeno.Visible:=false;
          RxDBCB_Mamdani.Enabled:=true;
          RxDBCB_Mamdani.Visible:=true;
       end;
end;

procedure TFrmVariables.PintarVariable;
var
serieNueva: Tchartseries;
tmpS:TChartSeriesClass;
begin

//pinto serie actual
    tmpS:=TChartSeriesClass(chart_Origen.Series[1].ClassType);
    serieNueva:=tmpS.Create(chart_Origen);
    serieNueva.Assign(Chart_Origen.Series[1]);
    serieNueva.Name:= 'Seleccion';
    serieNueva.Title:= '- Serie Actual -';
    serieNueva.SeriesColor:=Colores[1];
    Chart1.AddSeries(serieNueva);


  if DataModulo.TFMiembros.Active then // si esta abiertos
  begin
```

```
  Chart1.Title.Text.Clear;
  Chart1.Title.Text.Add('Variable: '+DataModulo.TVariablesVNombre.asstring);
  with DataModulo.TFMiembros do
   begin
    first;
    while not (eof)do
     next;
     first;
   end;
   end;


end;

procedure TFrmVariables.PintarFMiembro;
var
serieNueva: Tchartseries;
tmpS:TChartSeriesClass;
i,serieNo:integer;
x,a,c,d,b: double;
min,max,minmin,maxmax, paso: double;
begin
if DataModulo.TFMiembros.Active then
 begin
//hallar la serie
serieNo:= Noserie(DataModulo.TFMiembrosFNombre.asstring);
 if serieNo=-1 then //la serie no existe toca crearla
   begin
    tmpS:=TChartSeriesClass(chart_Origen.Series[0].ClassType);
    serieNueva:=tmpS.Create(chart_Origen);
    serieNueva.Assign(Chart_Origen.Series[0]);
    serieNueva.Name:= DataModulo.TFMiembrosFNombre.asstring;
    serieNueva.Title:= DataModulo.TFMiembrosFNombre.asstring;
    serieNueva.SeriesColor:=Colores[DataModulo.TFMiembros.RecNo];
    Chart1.AddSeries(serieNueva);
    PintarFMiembro;
   end
  else
   begin

  min:= DataModulo.TVariablesVRangoMin.AsFloat;
  max:=DataModulo.TVariablesVRangoMax.AsFloat;
  paso:= (max-min)/100;
  //pintaremos 10 cuerpos hacia delante y hacia atras
  minmin:= min- (10*(max-min));
```

```
  maxmax:= max+ (10*(max-min));;
  chart1.BottomAxis.Maximum:= max;
  chart1.BottomAxis.Minimum:= min;


    //borar la serie
    chart1.Series[SerieNo].Clear;
  //  chart1.series[Noserie('Seleccion')].Clear;
  //triangular
  if (DataModulo.TFMiembrosFTipo.Value ='triangular') then
     begin
      chart1.Series[SerieNo].AddXY(minmin,0);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam1.AsFloat,0);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam2.AsFloat,1);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam3.AsFloat,0);
      chart1.Series[SerieNo].AddXY(maxmax,0);
      chart1.series[Noserie('Seleccion')].Assign(chart1.Series[SerieNo]);
      chart1.series[Noserie('Seleccion')].Clear;

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam1.AsFloat,0);

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam2.AsFloat,1);

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam3.AsFloat,0);
     end;
  //trapezoidal
  if (DataModulo.TFMiembrosFTipo.Value ='trapezoidal') then
      begin
      chart1.Series[SerieNo].AddXY(minmin,0);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam1.AsFloat,0);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam2.AsFloat,1);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam3.AsFloat,1);
      chart1.series[SerieNo].AddXY(DataModulo.TFMiembrosFParam4.AsFloat,0);
      chart1.Series[SerieNo].AddXY(maxmax,0);
      chart1.series[Noserie('Seleccion')].Assign(chart1.Series[SerieNo]);
      chart1.series[Noserie('Seleccion')].Clear;

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam1.AsFloat,0);

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam2.AsFloat,1);

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam3.AsFloat,1);

chart1.series[Noserie('Seleccion')].AddXY(DataModulo.TFMiembrosFParam4.AsFloat,0);
      end;
```

```
  // gbell
  //1/(1+  power(abs((x-c)/d),2*a)) a>0
  if (DataModulo.TFMiembrosFTipo.Value='gbell') then
     begin
   //  chart1.MinXValue()
      c:=DataModulo.TFMiembrosFParam1.Value;
      d:=DataModulo.TFMiembrosFParam2.Value;
      b:=DataModulo.TFMiembrosFParam3.Value;
      if ((b<=0)or (d=0)) then application.messagebox('El valor b debe ser un número > 0 ' +
char(13)+'y el Valor del parámetro d no puede ser 0.'+char(13)+'Cambielos por
favor','Error')
      else
       begin
         x:= minmin;
          while x<= maxmax do
           begin
           chart1.Series[SerieNo].AddXY(x, bell(x,b,c,d));
           x:=x + paso;
           end;
       end;
      {for i:= minmin to maxmax do
      begin
       x:= i/100;
       chart1.Series[SerieNo].AddXY(x, bell(x,b,c,d));
      end;
      }chart1.series[Noserie('Seleccion')].Assign(chart1.Series[SerieNo]);
      chart1.series[Noserie('Seleccion')].Clear;
      chart1.series[Noserie('Seleccion')].AddXY(c,1);
     end;
  //gaussiana
  //e^(-0.5)[(X-c)/d]^2
  if (DataModulo.TFMiembrosFTipo.Value='gaussiana') then
     begin
  //  chart1.MinXValue()
      c:=DataModulo.TFMiembrosFParam1.Value;
      d:=DataModulo.TFMiembrosFParam2.Value;
      if (d=0) then application.messagebox('El valor d debe ser un número diferente de 0.
Cambielo por favor','Error')
      else
        begin
{      for i:= minmin to maxmax do
      begin
       x:= i/100;
       chart1.Series[SerieNo].AddXY(x, gauss(x,c,d));
```

```
      end;
}
        x:= minmin;
         while x<= maxmax do
          begin
          chart1.Series[SerieNo].AddXY(x, gauss(x,c,d));
          x:=x + paso;
         end;
       end;
      chart1.series[Noserie('Seleccion')].Assign(chart1.Series[SerieNo]);
      chart1.series[Noserie('Seleccion')].Clear;
      chart1.series[Noserie('Seleccion')].AddXY(c,1);
     end;
  //sigmoidal
  //1/(1+e^(-a*(x-c)))
   if (DataModulo.TFMiembrosFTipo.Value='sigmoidal') then
     begin
   //  chart1.MinXValue()
      c:=DataModulo.TFMiembrosFParam1.Value;
      a:=DataModulo.TFMiembrosFParam2.Value;
      x:= minmin;
         while x<= maxmax do
          begin
          chart1.Series[SerieNo].AddXY(x, sigmoidal(x,a,c));
          x:=x + paso;
         end;
      chart1.series[Noserie('Seleccion')].Assign(chart1.Series[SerieNo]);
      chart1.series[Noserie('Seleccion')].Clear;
      chart1.series[Noserie('Seleccion')].AddXY(c,0.5);
     end;

  end;
end;
end;


procedure TFrmVariables.RxDBGrid1GetCellParams(Sender: TObject;
  Field: TField; AFont: TFont; var Background: TColor; Highlight: Boolean);
begin
 if ((Sender as TDBGrid).DataSource.DataSet.RecNo mod 2) = 0 then
  begin
  background := $00EDC0C2;
  AFont.Color := clMaroon;
  end;
end;
```

```
procedure TFrmVariables.Chart1ClickSeries(Sender: TCustomChart;
  Series: TChartSeries; ValueIndex: Integer; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
serieNueva: Tchartseries;
tmpS:TChartSeriesClass;
id_existe, id_origen: Integer;
begin
  DataModulo.TFMiembros.Locate('FNombre',Series.Name,[]);
  id_origen:= Noserie(Series.Name);
end;


procedure TFrmVariables.Chart1MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var mousex, mousey: double;
    valy1, valy2, valy3, valy4 : double;
    valparam1,valparam2,valparam3,valparam4: double;
    Id_actual: integer;
begin
Try
//minx:=Chart1.BottomAxis.CalcPosValue(Value: double):integer
//minx:=Chart1.BottomAxis.CalcXPosValue(Value:double):integer;
//minx:=Chart1.BottomAxis.CalcPosPoint(Value: integer) :double;
if Shift= [ssLeft] then

begin
Id_actual:= NoSerie(DataModulo.TFMiembrosFNombre.AsString);

//debo verificar si estoy cerca de un punto parametro
//alpha:= Chart1.BottomAxis.Increment/2;
alpha:=(DataModulo.TVariablesVRangoMax.AsFloat                                    -
DataModulo.TVariablesVRangoMin.AsFloat)/100;
mousex := RoundTo(Chart1.BottomAxis.CalcPosPoint(x),-3);
mousey := Roundto(Chart1.LeftAxis.CalcPosPoint(y),-3);

if (dataModulo.TFMiembrosFTipo.Value ='gbell') or
   (dataModulo.TFMiembrosFTipo.Value = 'gaussiana')  then
 begin
  valparam1 := dataModulo.TFMiembrosFParam1.Value;
  valy1:=1;
 end
 else if (dataModulo.TFMiembrosFTipo.Value ='sigmoidal') then
 begin
```

```
 valparam1 := dataModulo.TFMiembrosFParam1.Value;
 valy1:=0.5;
end
else if (dataModulo.TFMiembrosFTipo.Value ='triangular') then
begin
  valparam1 := dataModulo.TFMiembrosFParam1.Value;
  valparam2 := dataModulo.TFMiembrosFParam2.Value;
  valparam3 := dataModulo.TFMiembrosFParam3.Value;
  valy1:=Chart1.Series[Id_actual].YValue[1];
  valy2:= Chart1.Series[Id_actual].YValue[2];
  valy3:= Chart1.Series[Id_actual].YValue[3];
end
else if (dataModulo.TFMiembrosFTipo.Value ='trapezoidal')then
begin
  valparam1 := dataModulo.TFMiembrosFParam1.Value;
  valparam2 := dataModulo.TFMiembrosFParam2.Value;
  valparam3 := dataModulo.TFMiembrosFParam3.Value;
  valparam4 := dataModulo.TFMiembrosFParam4.Value;
  valy1:=Chart1.Series[Id_actual].YValue[1];
  valy2:= Chart1.Series[Id_actual].YValue[2];
  valy3:= Chart1.Series[Id_actual].YValue[3];
  valy4:= Chart1.Series[Id_actual].YValue[4];
end;

 //todos el parametro uno puede ser tomado
 if (mousex < valparam1+alpha) and (mousex > valparam1-alpha) and (mousey <
valy1+alpha) and (mousey > valy1-alpha)  then
  begin
   DataModulo.TFMiembros.edit;
   DataModulo.TFMiembrosFParam1.Value:=mousex;
  end;

 if (dataModulo.TFMiembrosFTipo.Value ='triangular') or
(dataModulo.TFMiembrosFTipo.Value ='trapezoidal') then
  begin
  if (mousex < valparam2+alpha) and (mousex > valparam2-alpha)and (mousey <
valy2+alpha) and (mousey > valy2-alpha)  then
  begin
   DataModulo.TFMiembros.edit;
   DataModulo.TFMiembrosFParam2.Value:=mousex;
  end
  else
  if (mousex < valparam3+alpha) and (mousex > valparam3-alpha)and (mousey <
valy3+alpha) and (mousey > valy3-alpha)  then
  begin
```

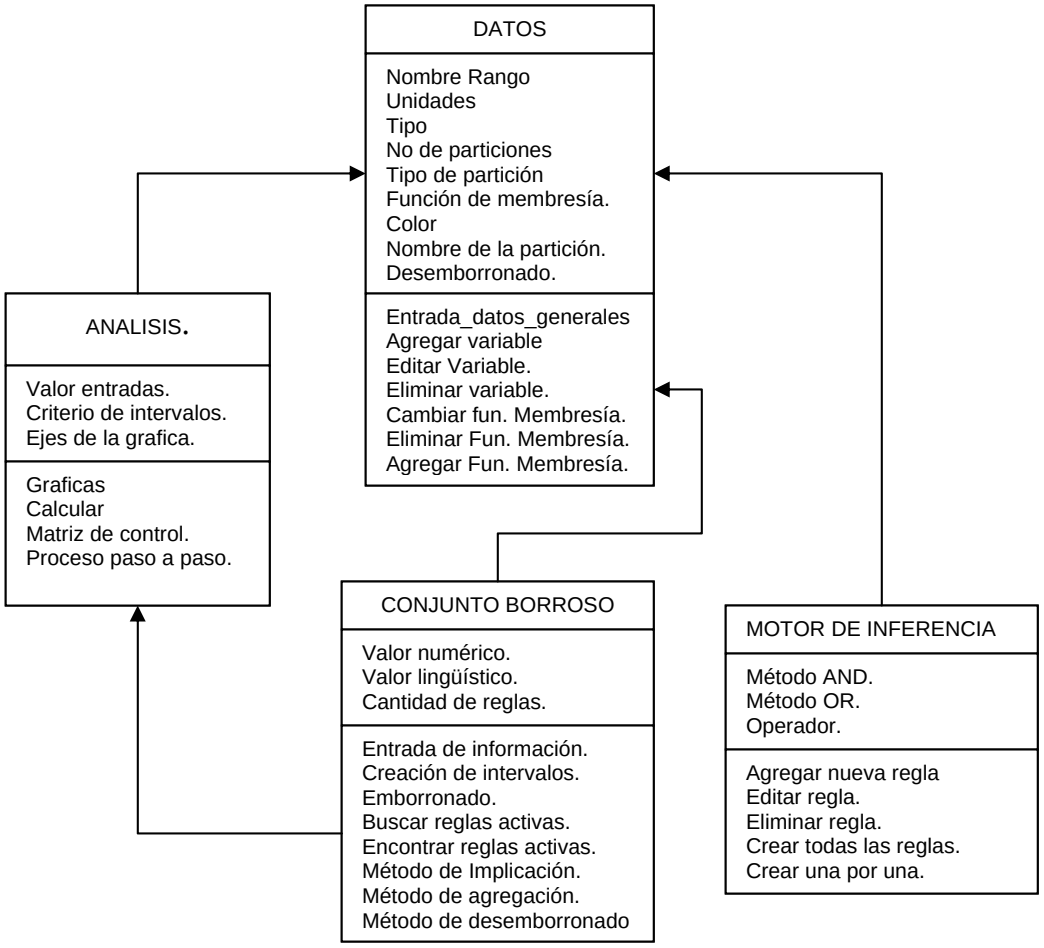
```
   DataModulo.TFMiembros.edit;
   DataModulo.TFMiembrosFParam3.Value:=mousex;
  end
  end;

  if (dataModulo.TFMiembrosFTipo.Value ='trapezoidal') then
  if (mousex < valparam4+alpha) and (mousex > valparam4-alpha) and (mousey <
valy4+alpha) and (mousey > valy4-alpha)  then
  begin
   DataModulo.TFMiembros.edit;
   DataModulo.TFMiembrosFParam4.Value:=mousex;
  end;
end;
except
end;

end;



procedure TFrmVariables.RxDBComboBox2Change(Sender: TObject);
begin
//existen 2 posibilidades Mamdani /sugeno en lo que respecta
//a variables de salida
{if DataModulo.TProyectoTipo.AsString = 'Mamdani' then
   begin
    RxDBCB_Sugeno.Enabled:=false;
    RxDBCB_Sugeno.Visible:=false;
    RxDBCB_Mamdani.Enabled:=true;
    RxDBCB_Mamdani.Visible:=true;
   end
   else
   begin
    RxDBCB_Sugeno.Enabled:=true;
    RxDBCB_Sugeno.Visible:=true;
    RxDBCB_Mamdani.Enabled:=false;
    RxDBCB_Mamdani.Visible:=false;
   end;
}end;

procedure TFrmVariables.RxDBGrid1DrawColumnCell(Sender: TObject;
  const Rect: TRect; DataCol: Integer; Column: TColumn;
  State: TGridDrawState);
begin
  if (datacol=0) then ((Sender as
```

```
DBGrid).Canvas.TextOut(rect.Left+1,rect.Top+1,InttoStr((Sender as
TDBGrid).DataSource.DataSet.RecNo)));
end;

end.
```

4.1.1.2 DIAGRAMA DE CLASES FINAL.

El diagrama de clases final se aprecia en la siguiente figura:

Figura 79 Diagrama de Clases Final.



4.1.3 PRUEBAS DE INGENIERÍA DE SOFTWARE.

Estas pruebas fueron realizadas con el fin de mejorar la calidad del software tanto en facilidad de uso como en eficiencia de su desempeño.

4.1.3.1 PRUEBAS ALFA.

Estas pruebas se llevan a cabo en un entorno controlado, en presencia del desarrollador y del director del proyecto. Para la realización de las pruebas alfa en el sistema COBOR 2.0 se elaboró una guía para que el usuario final desarroll'o con el fin de registrar errores y los problemas de uso de la herramienta.

Guía para pruebas alfa.

1. Se realiza una pequeña exposición sobre el funcionamiento del paquete y el objetivo para el que ha sido diseñado.
2. Entregar varios proyectos con diferentes variables de entradas y salidas al usuario.
3. Solicitar al usuario probar los proyectos variando los conjuntos borrosos y utilizando diferentes funciones de membresía para observar los resultados.
4. Construir con el usuario la base de reglas borrosas para entender la facilidad de cada uno de los métodos dados por la herramienta, y verificar cuál es el más óptimo según sea el caso.
5. Mostrar al usuario los métodos de desemborronado y que él interactúe con cada uno de ellos para que observe cuál es el más eficiente.
6. Construir un controlador borroso basado en un proyecto ya guardado.
7. Introducir parámetros inválidos para comprobar las diferentes validaciones utilizadas.
8. Realizar Observaciones y recomendaciones para todo el proceso de ejecución del sistema.

Se realizaron pruebas indicando a los usuarios en términos generales el funcionamiento del sistema, sus áreas de utilización y las características de construcción de controladores borrosos, dentro de los resultados se obtuvieron las siguientes sugerencias:

- Dar una mejor explicación sobre los conceptos de la lógica borrosa para la creación de los controladores.
- Que el sistema sea un poco más flexible, en cuanto a la construcción de los conjuntos borrosos.
- Seria bueno proveer al sistema de un análisis de sensibilidad con los resultados obtenidos.

En general se observó que el sistema trabaja bien aunque se tienen algunas falencias, la mayoría de ellas de carácter conceptual puesto que los usuarios desconocían las operaciones básicas para trabajar con Lógica borrosa.

4.2 SISTEMA COBOR 2.0.

4.2.1 ARQUITECTURA DE COBOR 2.0.

El sistema se construyó en el ambiente de desarrollo DELPHI versión 7.0, que pertenece a la casa Borland y se presenta como una alternativa para el desarrollo de aplicaciones; Además, se destaca de Borland la velocidad, robustez y eficiencia de los compiladores que desarrolla.

Esto con el fin de desarrollar una herramienta que esté en capacidad de enfrentarse a problemas de gran amplitud como son los controladores borrosos, ya que pueden existir sistemas de muchas entradas y salidas, conllevando a una mayor robustez del programa para soportar este tipo de aplicaciones.

4.2.2 CARACTERÍSTICAS DEL SISTEMA

COBOR 2.0 es una herramienta para el diseño y manejo de controladores inteligentes basados en lógica borrosa, debido a la gran aceptación que estos controladores borrosos han tenido últimamente se trato de implementar un sistema capaz de proveer apoyo al experto de los procesos.

Después de tener las variables de entrada y salida definidas se continúa el proceso normal para la construcción de un controlador borroso, como es la base de reglas, los métodos de desemborronado, y otras características que la herramienta brinda al usuario como:

- Generación de los conjuntos borrosos.
- Elección de la función de membresía.
- Construcción de las reglas borrosas.
- Método de desemborronado.
- Matriz de control.
- Impresión de los datos y el reporte del caso en estudio.

4.2.3 CARACTERÍSTICAS DE LOS COMPONENTES.

Cada uno de los componentes tiene funcionalidades que permiten ayudar a todo el proceso global para la creación de los controladores borrosos.

Componente de carga de datos

Este componente esta formado por la clase Datos permitiendo la carga a partir de archivos planos ya sean propios del sistema o en el formato de COBOR 2.0, tomando esa información recibida para su manipulación por medio de las leyes de la lógica borrosa,

esto con el fin de generar las soluciones requeridas por los usuarios, después de los procesos propios de un controlador borrosos como son:

- ✓ Emborronado.
- ✓ Motor de inferencia.
- ✓ Desemborronado.

Componente de creación de los conjuntos borrosos.

Este componente brinda facilidades para la construcción de los conjuntos borrosos de entrada y salida que son la base para la creación del controlador borroso, permitiendo la manipulación de las funciones de membresía, los implicadores, la construcción de la base de reglas y otras características propias del sistema.

Componente de visualización de resultados.

Para este componente se utilizó un componente grafico que permite la visualización, para una fácil comprensión y asociación de los resultados, realizando todo esto por métodos implementados durante el desarrollo del proyecto.

Motor e Interfaz a Usuarios

En el motor del sistema se centralizan o agrupan instancias de los diferentes componentes base del sistema, permitiendo la realización del proceso en forma global. La interfaz a usuarios comprende todos los componentes y formularios gráficos con los cuales se interactúa, construida de forma amigable y de fácil manejo.

4.2.4 PARÁMETROS ESPECÍFICOS.

Para la construcción de los controladores borrosos se requieren una serie de parámetros que se explican a continuación:

- Mínimo numero de variables: la herramienta exige como mínimo la inclusión de dos variables de entrada y una variable de salida al proceso ya que es la única restricción para operar la herramienta.
- Máximo numero de variables: La herramienta da un rango máximo de variables al controlador que es de quince (15).
- Base de reglas: la base de reglas tendrá la opción para su construcción de generar las reglas una por una, según sea la elección del experto.
- Funciones de membresía: Las funciones de membresía que se implementaron fueron: Triangular, trapezoidal, gaussiana, bell y sigmoidal. Que son las más utilizadas por los expertos en lógica borrosa, debido a su sencillez y buena transformación de los datos numéricos.
- Método de desemborronado: Los métodos de desemborronado implementados son los siguientes, centro de área, Media de máximos, Primer máximo, y ultimo máximo. Son los más eficientes computacionalmente hablando, y además los que tienen más exactitud en los resultados obtenidos, se da la posibilidad de implementar otros métodos.

4.2.5 REQUERIMIENTOS DE HARDWARE Y SOFTWARE.

Para el desarrollo del sistema COBOR 2.0 se utilizó un equipo con las siguientes características:

- Procesador Pentium IV de 2.8 Mhz.
- 128 MB en RAM.

Para la ejecución del sistema COBOR 2.0 se requieren como mínimo 64 MB de memoria RAM, se recomienda tener 128 MB y tener procesador Pentium III o superior. Esta herramienta funciona en sistemas operativos Windows series 9X, NT, 2000, XP.

CONCLUSIONES.

- La ingeniería de Sistemas se preocupa por encontrar soluciones prácticas a los problemas que se presentan en todas las áreas de la ciencia, por esta razón busca nuevas herramientas para cumplir este objetivo, prueba de ello es la utilización de una teoría como la lógica borrosa, que nos sirve de apoyo para el control de procesos de ingeniería agilizando y facilitando la implementación de los controladores.

- La Lógica Borrosa demostró ser una herramienta muy potente y eficaz para la construcción de controladores, y manejo de procesos de elevada complejidad, ya sea por la cantidad de variables que se obtienen como por la no linealidad de sus parámetros.

- Frente a la posibilidad de “modelar” un determinado comportamiento, diferentes personas pueden dar diferentes soluciones expresadas con lenguaje natural dependiendo de la abstracción que cada uno haga de la situación. La lógica borrosa es una rama de la inteligencia artificial que se funda en el concepto "Todo es cuestión de cuanto se cumple”, lo cual permite manejar información vaga o de difícil especificación. Es entonces posible con la lógica borrosa gobernar un sistema por medio de reglas de 'sentido común', que se refieren a cantidades indefinidas. Esto hace que la lógica borrosa de un soporte formal al razonamiento basado en lenguaje natural.

- Cuando se trabaja con una herramienta como COBOR 2.0 se busca obtener como resultado “La matriz de Control”, que indica cuál acción tomar ante las posibles combinaciones de estados en las entradas del sistema.

- Las aplicaciones de los controladores borrosos han ido creciendo rápidamente gracias a que dan una solución eficiente a la mayoría de procesos donde los controladores tradicionales no dan resultados óptimos.

- Se obtuvo una herramienta con una interfaz amigable y didáctica que lleva al usuario a cumplir con su objetivo, la elaboración de un controlador borroso, ésta herramienta esta caracterizada por:

  - ✓ Flexibilidad al cargar datos almacenados en archivos planos.

  - ✓ Tener una completa sección para la elaboración de conjuntos borrosos, con las funciones características más utilizadas y los métodos de desemborronado más eficientes.

  - ✓ Tener un motor de inferencia que permita la posibilidad de creación de la base de reglas ya sea combinando las variables de entrada y salida o creando las reglas una por una.

  - ✓ Contar con un analizador de la información que da varias posibilidades según las necesidades del experto del proceso, ya sea de forma gráfica, numérica, o generando la matriz de control.

- La elección el lenguaje de programación Delphi 7.0 permitió la creación de un motor eficiente y versátil, que absorbió la alta complejidad del proyecto brindando al usuario una transparente y muy fácil manera de crear controladores borrosos.

RECOMENDACIONES FINALES.

- La combinación de la lógica borrosa con otros tipo de técnica de control adaptativo como son las redes neuronales o los algoritmos genéticos dan como resultado herramientas poderosas, que se están implementando hoy en día de una manera eficaz y práctica en la solución de problemas complejos. En este sentido se recomienda Implementar otros métodos de desemborronado combinando la lógica borrosa con redes neuronales.

- Es conveniente entrar a analizar la complejidad de la situación que se desea controlar, además que es muy recomendable verificar la existencia de un modelo matemático de la situación en cuestión. Ya que si éste existe, la solución que da suele ser más exacta que la de un controlador borroso. Por lo general la lógica borrosa se emplea para resolver problemas de alta complejidad y cuyo modelo matemático no es lineal. La lógica borrosa es de gran utilidad en la solución de sistemas con muchas variables.

- Queda abierta la posibilidad de implementar nuevas funciones de membresía o métodos de desemborronado, debido a la flexibilidad de la herramienta, cabe anotar que las implementadas en COBOR 2.0 son las más confiables y eficientes.

BIBLIOGRAFIA

BOOCH, G., RUMBAUGH, J., JACOBSON, I. (1999) El Lenguaje Unificado de Modelado. Addison-Wesley.

DUBOIS D., and H. PRADE Fuzzy sets and Systems: Theory and applications, academis press, New Cork, 1980.

Eco, Humberto. *Cómo se Hace una Tesis*, Barcelona, España, Editorial Gedisa S.A., 1994.

HABER R., Introducción al control borroso, GIALBACOL, Grupo de interés de aplicaciones de la lógica borrosa.TERANO, T., ASAI, K., SUGENO, M., Theory and its applications, 1991.

HELLENDOORM H., Y C. Thomas “Defuzzyfiication in fuzzy controllers”, intelligent and fuzzy Systems, Vol 1, pp 109-123, 1993.

H. R. BERENJI and P. KHEDKAR, Learning and tuning fuzzy logic controllers through reinforcements. IEEE Trans. Neural Networks 3 (1992)

J. J. BUCKLEY and Y. HAYASHI, Fuzzy neural networks: A survey, Fuzzy Sets and Systems 66 (1994).

KOSKO B., Neural networks and Fuzzy systems, Prentice Hall Englewood cliffs, New Jersey, 1992.

LARMAN, C. UML y Patrones.: Introducción al Análisis y Diseño Orientado a objetos. Pearson.

PRESSMAN, Roger. (1995). Ingeniería del Software, McGraw-Hill. 3ª Ed.

RUIZ F., PEREZ M., y COBOS M., Desarrollo de un sistema de ayudas para el diseño de controladores borrosos 1997.

SUGENO M., “An introductory survey on fuzzy control” Vol 36, pp 59-83, 1985.

TEIXEIRA Steve & PACHECO Xavier. Guía de desarrollo Delphi 7. ED Prentice Hall.2004.

VELARDE, J. (1991), Gnoseología de los sistemas difusos.Serv. Publicaciones Univ. De Oviedo, Oviedo.

ZADEH L., “Fuzzy and sets “inf, control, Vol 8, pp 338-353, 1965.

ZADEH, L. A. y KACPRZYK, J. (eds.)(1992), Fuzzy Logic for the Management of Uncertainty. John Wiley, Nueva York.

REFERENCIA BIBLIOGRAFICA.

TEORIA

- [1] TERANO, T., ASAI, K., SUGENO, M., Theory and its applications, 1991.
- [2] ZADEH L., “Fuzzy and sets “inf, control, Vol 8, pp 338-353, 1965.
- [3] DUBOIS D., and H. PRADE Fuzzy sets and Systems: Theory and applications, academis press, New Cork, 1980.
- [4] HABER R., Introducción al control borroso, GIALBACOL, Grupo de interés de aplicaciones de la lógica borrosa.
- [5] SUGENO M., “An introductory survey on fuzzy control” Vol 36, pp 59-83, 1985.
- [6] KOSKO B., Neural networks and Fuzzy systems, Prentice Hall Englewood cliffs, New Jersey, 1992.
- [7] HELLENDOORM H., Y C. Thomas “Defuzzyfiication in fuzzy controllers”, intelligent and fuzzy Systems, Vol 1, pp 109-123, 1993.
- [8] DARWIN, C. (1859). On the Origin of Species by Means of Natural Selection or the Preservations of Favored Races in the Struggle for Life. London: John Murray.

- [9] CORREDOR M., Martha Vitalia. Principios de Inteligencia Artificial y Sistemas Expertos. Ediciones UIS. 1982.

TECNICA

- [10] CANTU M, (2003) Delphi 7. 1ª edición, Prentice Hall.

- [11**]** AYRES J, BOWDEN D, DIEHl L, DORCAS P, (1998) Nucleo API Win 32 McGraw-Hill.

- [12] REISDOROPH, Kent. (2003). Aprendiendo Delphi 7.0 en 21 días. Sams Publishing.

METODOLOGICA

- [13] BOOCH, G., RUMBAUGH, J., JACOBSON, I. (1999) El Lenguaje Unificado de Modelado. Addison-Wesley.

- [14] LARMAN, C. UML y Patrones.: Introducción al Análisis y Diseño Orientado a objetos. Pearson.

- [15] PRESSMAN, Roger. (2000). Ingeniería del Software. McGraw-Hill. 6ª Ed.

ANEXO 1

MANUAL DEL USUARIO PARA MANEJO DEL SOFTWARE COBOR 2.0

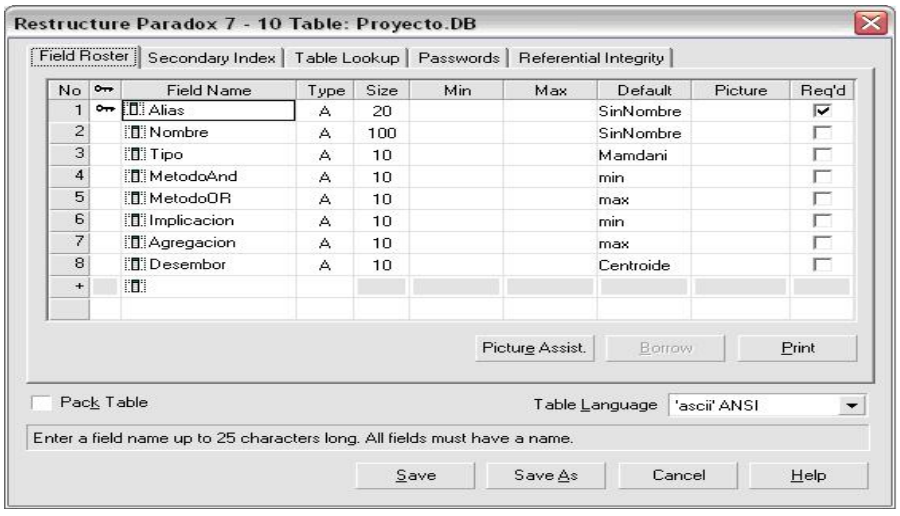
1. GENERALIDADES DEL SOFTWARE

COBOR 2.0 es una herramienta para el apoyo y desarrollo de sistemas de control, utilizando la lógica borrosa como una tecnología adaptativa. Mediante el monitoreo de las variables de control, se determina la acción a tomar por el controlador sobre la inferencia de una base de reglas de conocimiento definida a partir de la experiencia dada por un experto en el control de procesos, tomando un soporte formal de los fundamentos teóricos de los conjuntos difusos.

El sistema se construyó en el ambiente de desarrollo DELPHI versión 7.0 y el almacenamiento de datos de la herramienta a través de la base de datos PARADOX.

El sistema COBOR 2.0 se desarrolló con una Estructura de bases de datos que se realizó en PARADOX, como un manejo de archivos Indexados, a continuación se presenta la estructura de las tablas que se diseñaron:

Figura 80. Ventana Tabla Base de Datos PARADOX



En la Figura 80, Se presenta un pequeño ejemplo del almacenamiento de datos, para ello crea los campos que requiera el proyecto.

2. REQUERIMIENTOS DEL HARDWARE Y SOFTWARE.

Para el correcto funcionamiento del software se requiere sea instalado en equipos que tenga esta configuración:

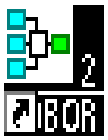
- Procesador AMD ó Pentium IV mayor de 1.7 Mhz.
- Se requieren como mínimo 64 MB de memoria RAM, se recomienda tener 128 MB
- Sistema Operativo: la herramienta funciona en Microsoft Windows series 9X, NT, 2000, XP.

3. INSTALACION DEL SOFTWARE

Para la correcta instalación del software se debe tener en cuenta los siguientes pasos:

a. Inserte el CD de instalación que contiene el software COBOR 2.0 en la unidad de CD.
b. Bajo el explorador de Windows ubicar la unidad de CD, donde se pueden visualizar los diferentes archivos que componen el software.
c. Abrir el archivo ejecutable SetupCobor2.exe, el cual inicia la instalación.
d. El ejecutable despliega una pantalla indicando los pasos a seguir, hasta la correcta instalación.
e. El programa crea una carpeta denominada Cobor2, ubicada en la siguiente dirección: c:\archivos de programa\cobor2 donde quedará instalado.
f. Así mismo el instalador crea un acceso directo al programa en el menú inicio del escritorio de Windows.
g. Si el usuario lo desea puede crear un acceso directo en el escritorio de Windows, para ello se debe ubicar el archivo ejecutable cobor2.exe allí.

Figura 81. Acceso Directo a COBOR 2.0



Cobor2.exe.lnk

h. Debido a que los rangos de números con los que trabaja COBOR 2.0 tienen al punto (.) como configuración del símbolo decimal y separador de miles, se debe revisar la *configuración regional* del equipo de manera que este así. Para verificar esto, se va a la siguiente dirección a través del menú inicio de Windows:
   Inicio / panel de control / configuración regional y de idioma / personalizar
   En las casillas símbolo decimal y separador de miles debe estar el signo punto (.)
i. En los computadores que no tengan instalados programas de la casa Borland a la que pertenece Delphi 7.0, es necesario instalar la librería ***BdeInst.dll***, esta librería se adjunta dentro del CD de instalación. El registro de esta librería se realiza de esta forma: crear una carpeta en la unidad C:\ llamada ***cobor2*** y copiar allí esta librería. La dirección quedaría así: C:\cobor2\BdeInst.dll.
   Luego vamos a registrarla en los archivos de Windows de la siguiente forma:
   inicio / ejecutar, Allí escribimos lo siguiente: **regsvr32 C:\cobor2\BdeInst.dll** y oprimimos aceptar, empezando el sistema operativo el registro. De esta forma queda registrada la librería.

Una vez efectuados los pasos anteriores se puede acceder al software ya sea a través del menú inicio o por al acceso directo creado.

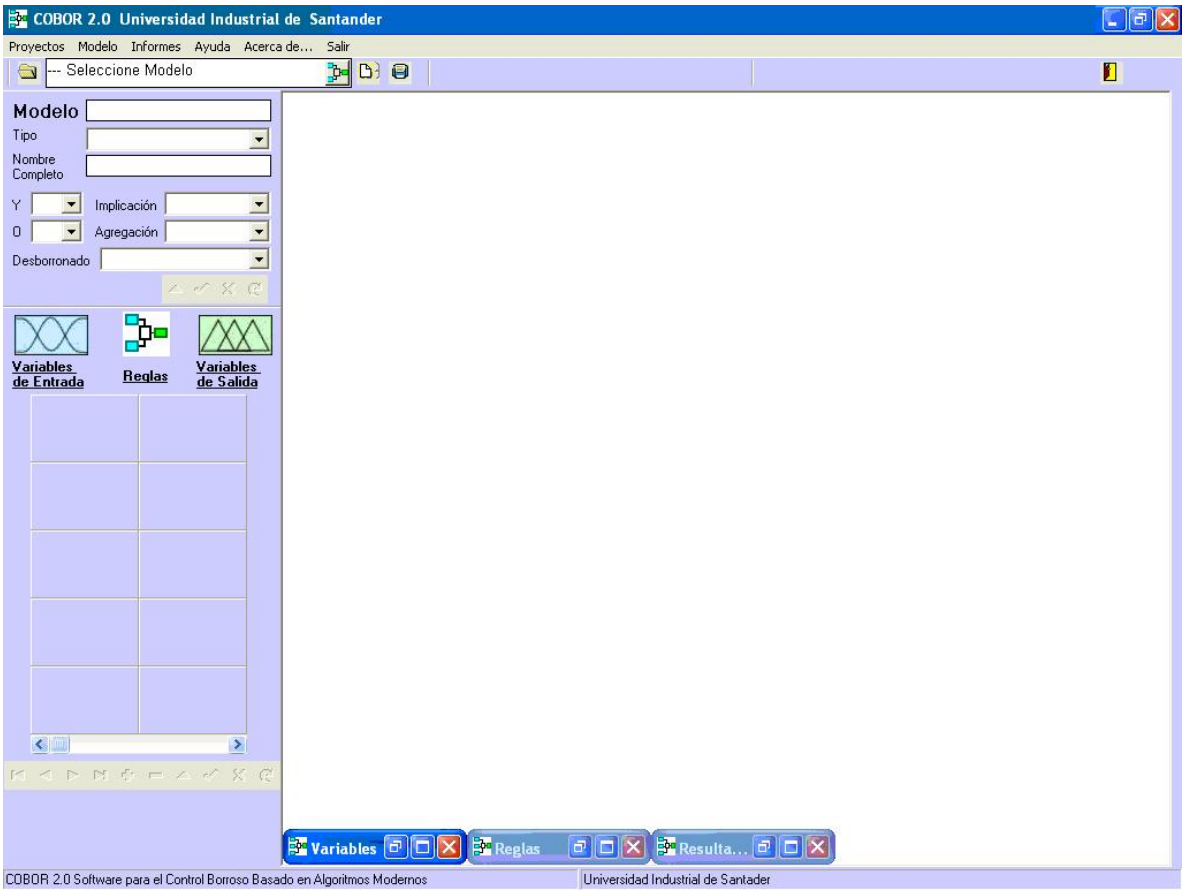
4. USO DE LA HERRAMIENTA Y PRESENTACIÒN DE LOS MENUS.

Al abrir el programa aparece el entorno principal de trabajo. Visualizamos de una forma gráfica todo el entorno de Cobor 2.0, que es de fácil lectura y permite empezar el desarrollo del trabajo.

En este momento puedo Crear un proyecto o seleccionar un proyecto ya existente.

El software COBOR 2.0 presenta el siguiente entorno de trabajo:

Figura 82. Entorno de trabajo COBOR 2.0



A partir de esta pantalla de visualización empieza el desarrollo de la herramienta.

El menú Principal presenta estas opciones:

1. Proyectos
2. Modelo
3. Informes
4. Ayuda
5. Acerca de...
6. Salir

1. Proyectos: Es la primera opción del menú principal y nos brinda el siguiente submenú:

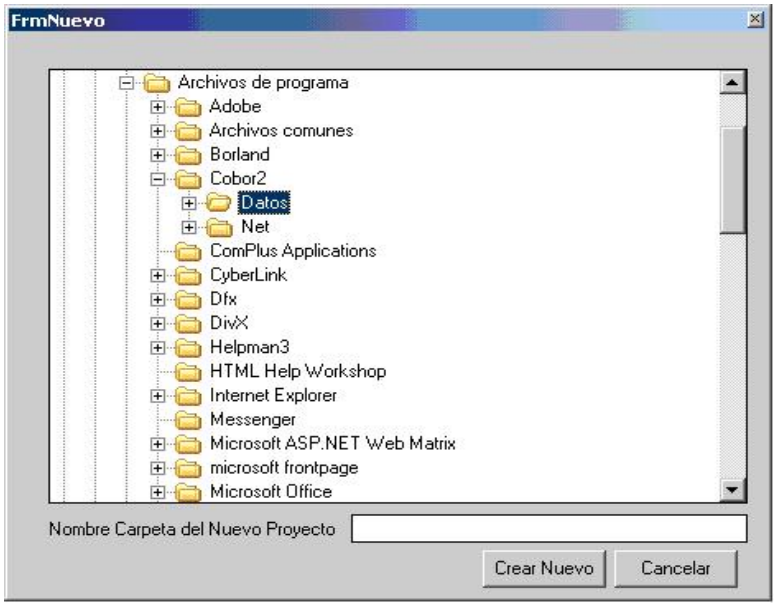
1.1 Nuevo: Permite la creación de un nuevo proyecto. Hay que tener en cuenta que al dar esta opción el programa crea una carpeta por proyecto, la cual ubica en la carpeta Cobor2 donde quedo instalado el programa. El nombre de la carpeta se escribe en la casilla junto al texto: Nombre Carpeta del Nuevo Proyecto. La nueva carpeta se ubica por defecto debajo de la carpeta donde esté el ejecutable del software.

En la carpeta correspondiente a cada proyecto se almacenan unos archivos con extensión .db que son las tablas que se necesitan para generar los datos que requiera el proyecto. Los otros archivos con extensiones .val, .px, manejan la integridad, los tipos y otras cuestiones de la base de datos.

También puede activarse la opción **nuevo**, a través del correspondiente botón de acceso directo ubicado en la pantalla inicial.

Nota: Si existe un proyecto abierto y se va a crear uno nuevo, se debe primero salir del que esta abierto, ya que el software no acepta dos proyectos simultáneamente, y luego si activar la función proyecto nuevo.

Figura 83. Ventana Nuevo Proyecto. Ej. Nuevo proyecto se llama “Datos”



1.2 Abrir: Es la opción que nos permite seleccionar la carpeta en donde tengo almacenado algún proyecto ya creado.

Figura 84. Ventana Buscar Carpeta

También puede activarse la opción abrir, a través del correspondiente botón de acceso directo ubicado en la pantalla inicial.

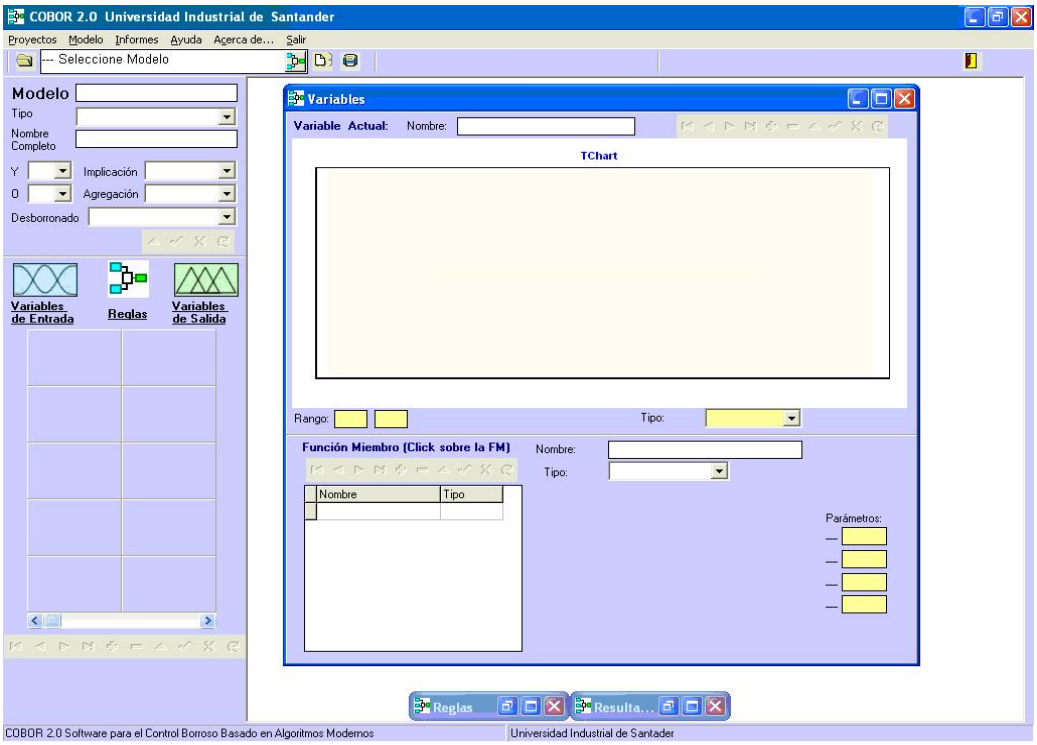
Nota: Si existe un proyecto abierto y se va a abrir otro, se debe primero salir del que esta abierto, ya que el software no acepta dos proyectos simultáneamente, y luego si activar la función abrir proyecto.

1.3 Salvar: Es la opción que me permite guardar todo el proyecto que he creado o modificado, lo ubica en la pantalla principal y hay que indicarle la carpeta en donde se desee salvar la información.

1.4 Salir: Es la opción que termina la ejecución del programa.

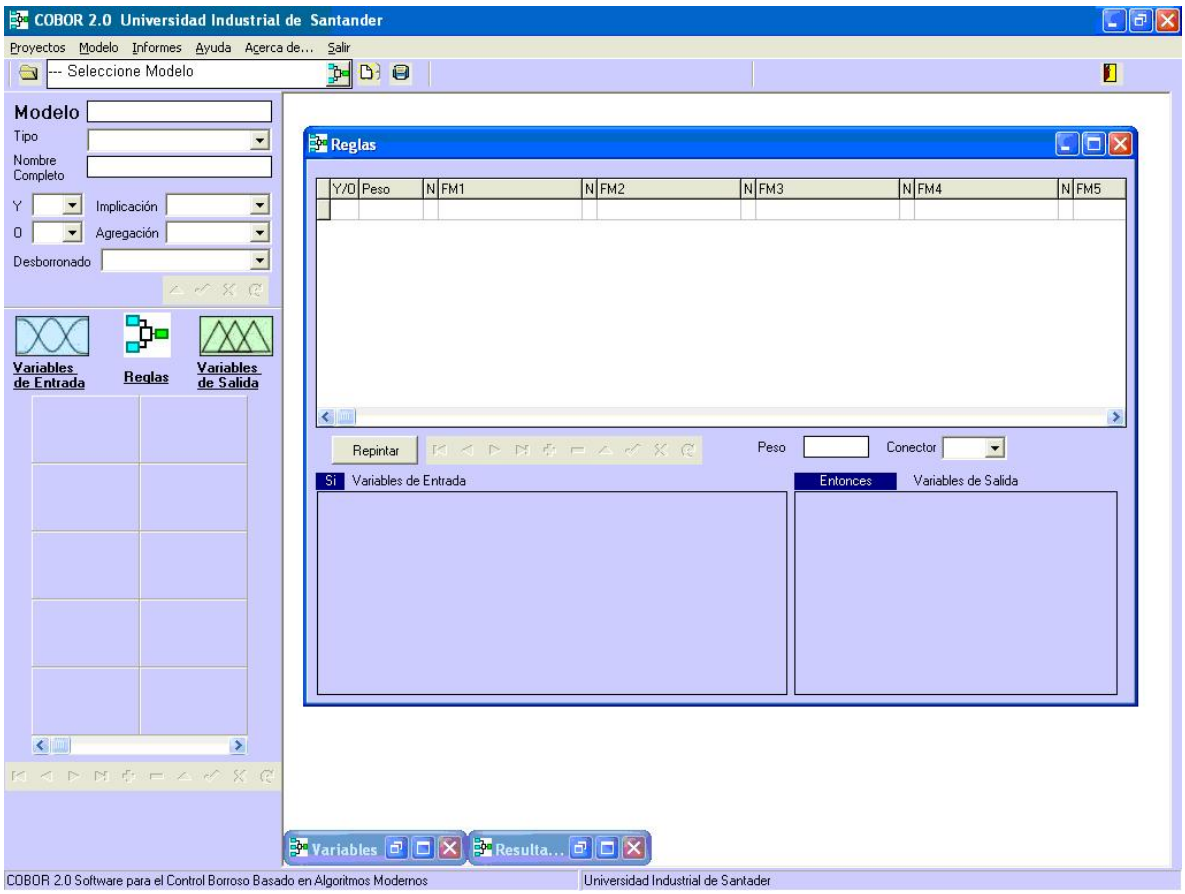
2. Modelo: Es la opción del menú principal que nos permite visualizar las ventanas que se trabajan en el software COBOR 2.0. Al entrar a ésta opción del menú presenta tres alternativas: Variables, Reglas y Resultados. Las siguientes gráficas nos muestran estas ventanas:

Figura 85. Ventana Variables del Proyecto.



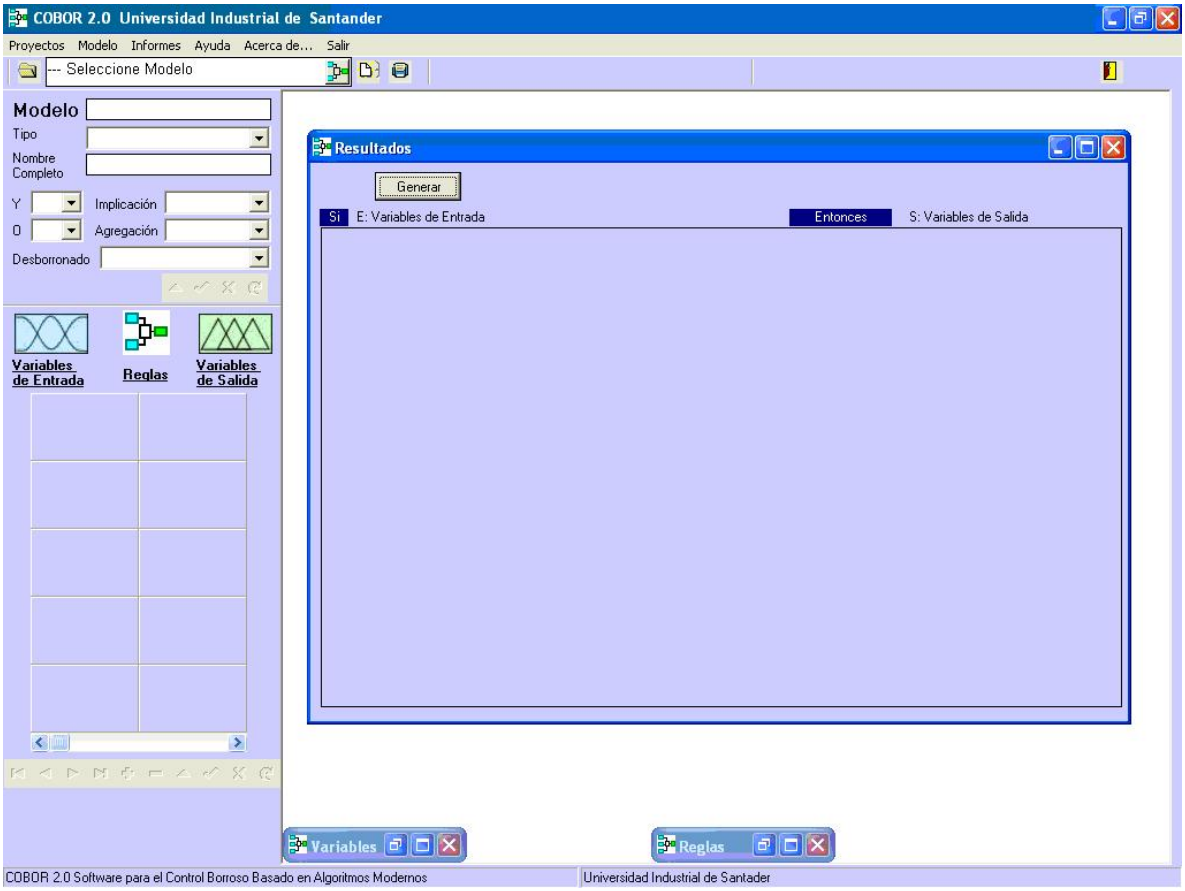
Visualizamos la ventana de las Variables ya sean de entrada o Salida y las especificaciones de las mismas. Detalle del uso de esta se verán el numeral 5 de esta ayuda.

Figura 86. Ventana Reglas del proyecto



Visualizamos la ventana de las Reglas que es muy importante en el controlador Borroso, allí me permite hacer la matriz con las diferentes opciones que el experto desee implementar. Detalle del uso de esta ventana se explicará en el numeral 5 de esta ayuda.

Figura 87. Ventana Resultados del proyecto



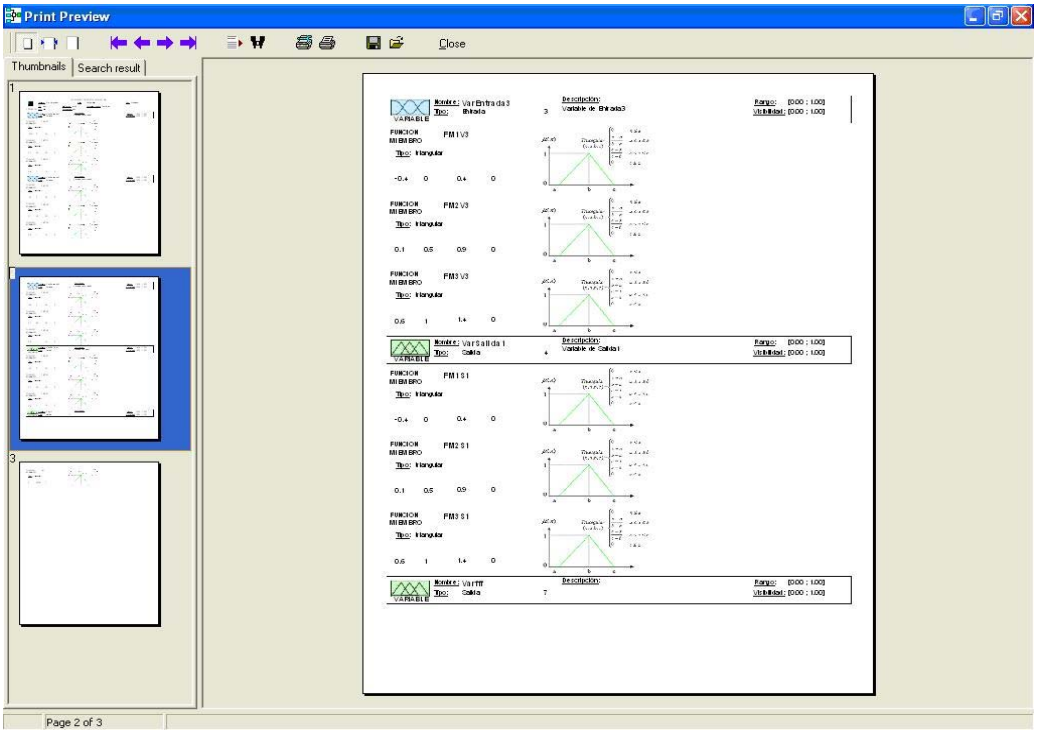
Visualizamos la ventana en donde se generan los resultados de una forma gráfica.

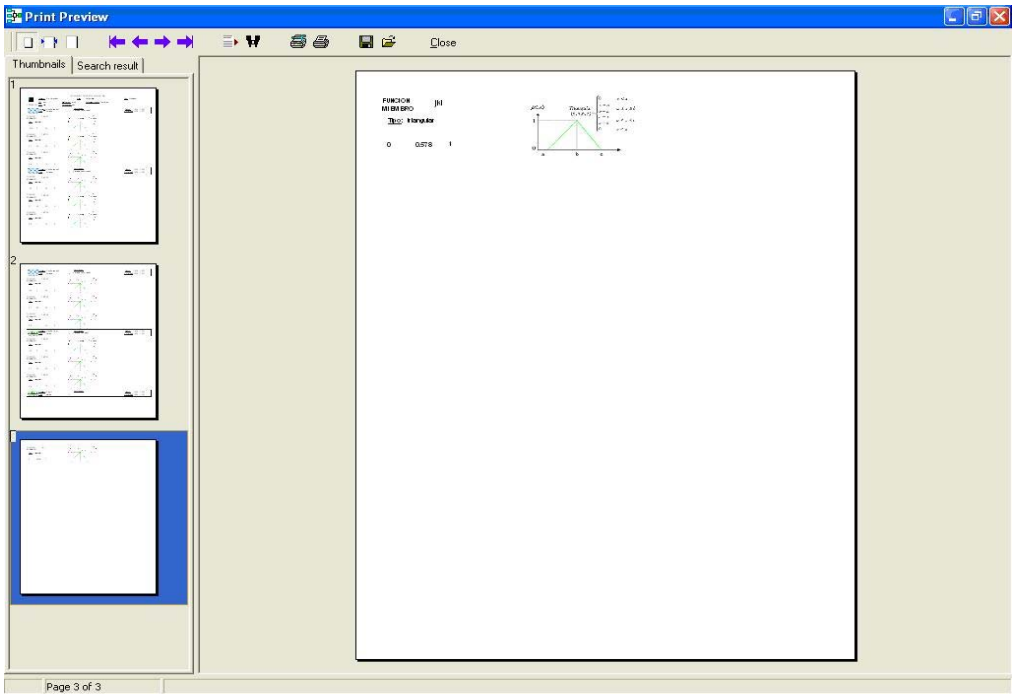
En ésta ventana se generan los procesos que nos permiten observar a través de gráficas todo el proyecto que se ha venido generando. Detalle del uso de esta ventana se explicará en el numeral 5 de esta ayuda.

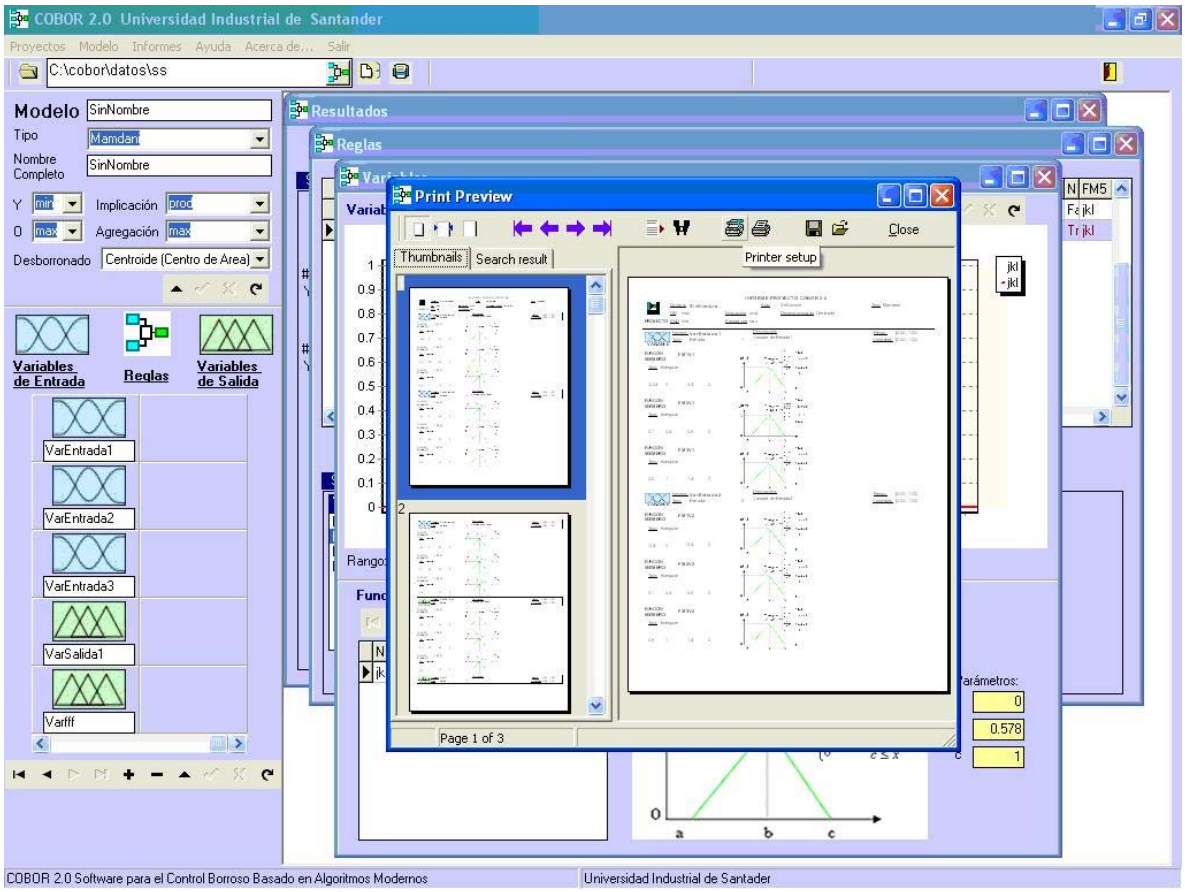
3. Informes: Es la opción del menú que nos presenta el informe detallado de las variables y reglas que conforman el proyecto. Posee el siguiente submenú:

3.1 Proyecto: Visualiza textual y gráficamente los datos de variables y reglas que se utilizaron en el proyecto. Las ventanas que se visualizan son las siguientes:

Figura 88. Ventanas Informes del proyecto





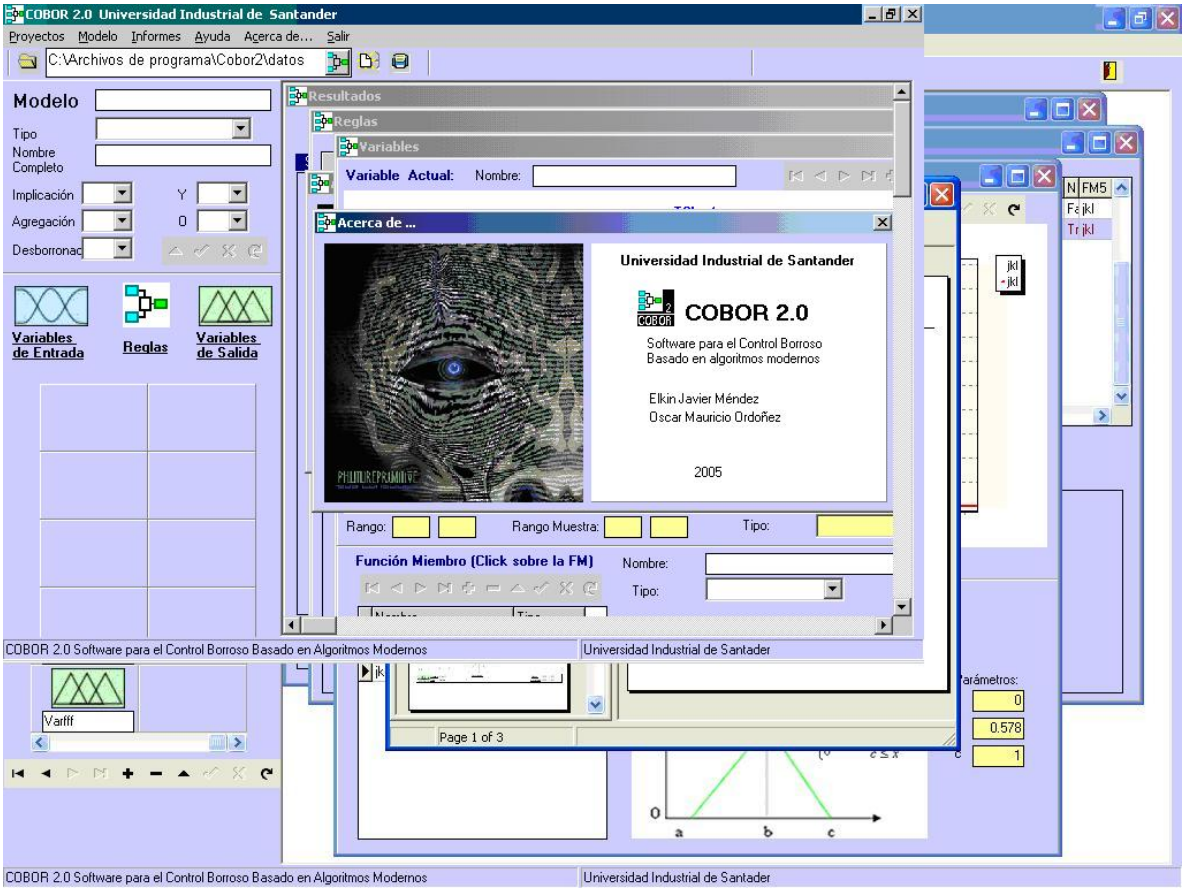


Visualizamos el informe detallado del proyecto que se ha venido desarrollando. Dentro de la visualización del informe, se activa un menú grafico que contiene entre otras las siguientes opciones: pasar a siguiente pagina, imprimir, guardar, etc.

4. Ayuda: Es la opción que nos permite visualizar la ayuda a usuario del software de manera que se pueda consultar aspectos relativos del funcionamiento de éste.

5. Acerca de...: Es la opción del menú que presenta los desarrolladores del software COBOR 2.0

Figura 89. Ventana Autores del proyecto

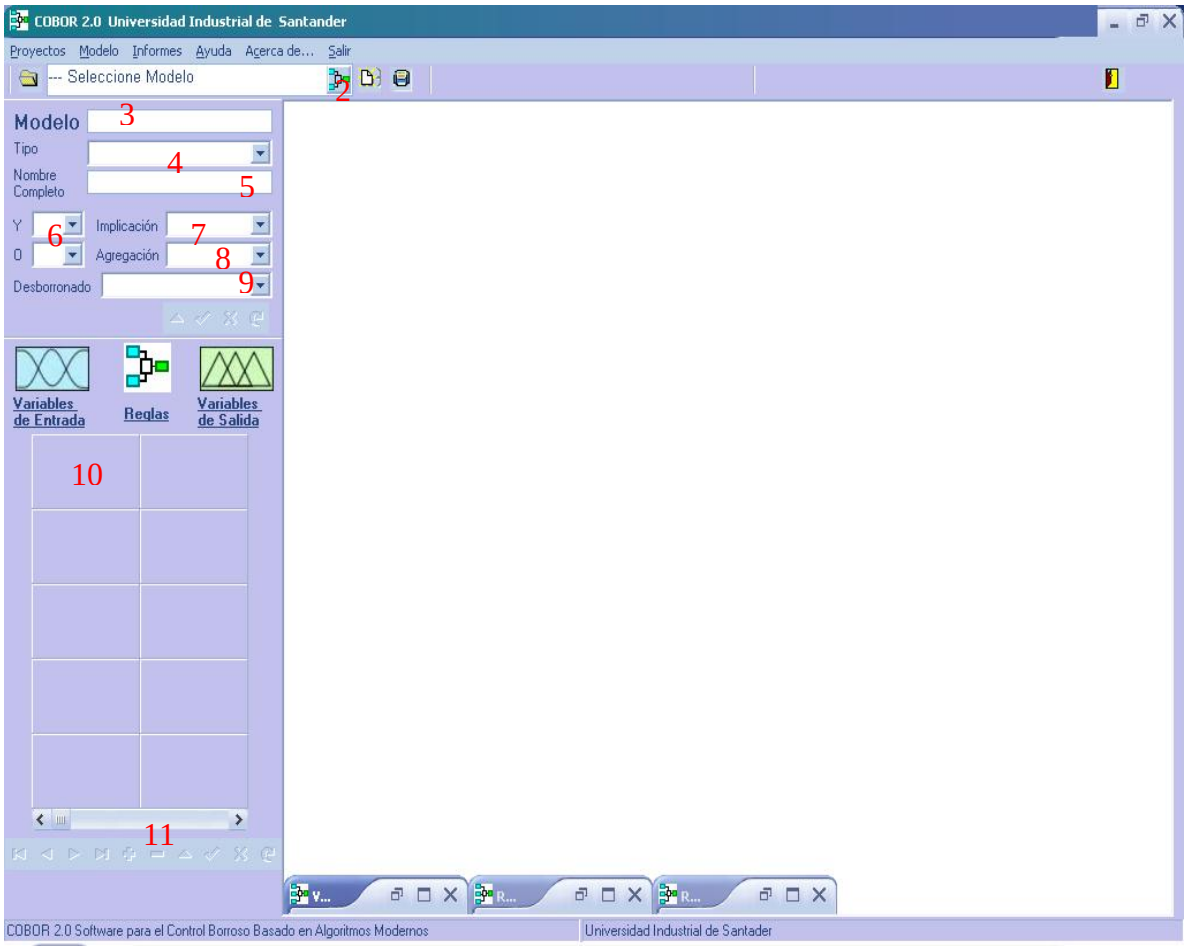


6. Salir: Es la opción que termina la ejecución del software

5. EJEMPLO DETALLADO DE UN PROYECTO EN EL SOFTWARE COBOR 2.0

A continuación vamos a desarrollar un proyecto para explicar de una forma detallada los pasos que se deben tener en cuenta en la realización y manejo del sistema.

Figura 90. Ventana Principal detallada.

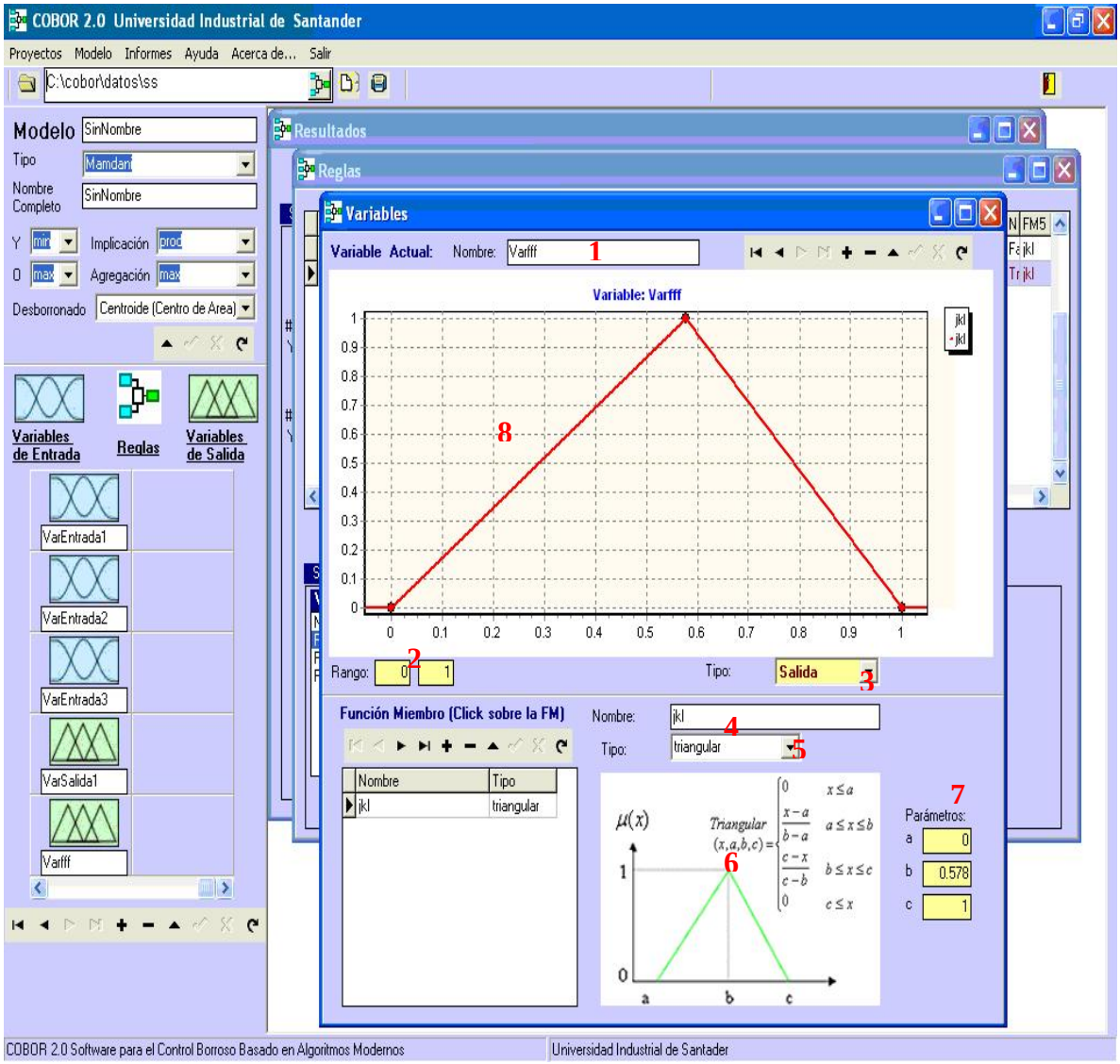


1. El primer aspecto a tener en cuenta es seleccionar la carpeta en donde tengo toda la información, escribimos la dirección de la carpeta en la casilla superior o con el botón que está junto a la casilla seleccionamos la dirección en donde se desea trabajar.

2. Abrimos el proyecto seleccionado.
3. Seleccionamos el nombre del Modelo y se coloca en la casilla adjunta a la palabra MODELO.
4. Se selecciona el tipo de controlador Borroso: Mandami o Sugeno.
5. Se escribe el nombre completo del Proyecto.
6. Y / O: son los métodos que se usarán en las reglas para las operaciones.
7. En la implicación selecciono el método: Mínimo o Producto
8. En la agregación se selecciona: Máximo
9. Selecciono el método de desemborronado que se desee trabajar: Centroide (centro de Área), Centro sumas, Primer Máximo, último Máximo y media máximos.
10. Selecciona la ventana de variables de control y acción (Entrada / salida) a generar. Las de color azul son variables de entrada y las de color verde son variables de salida.
11. Hay que tener en cuenta los botones que se visualizan en el software: el + es para agregar, el – para borrar, el triangulo para modificar, el signo de visto bueno es para Salvar, la Letra X para cancelar, el  para refrescar o actualizar la base de datos.

VENTANA VARIABLES

Figura 91. Ventana Variables detallada



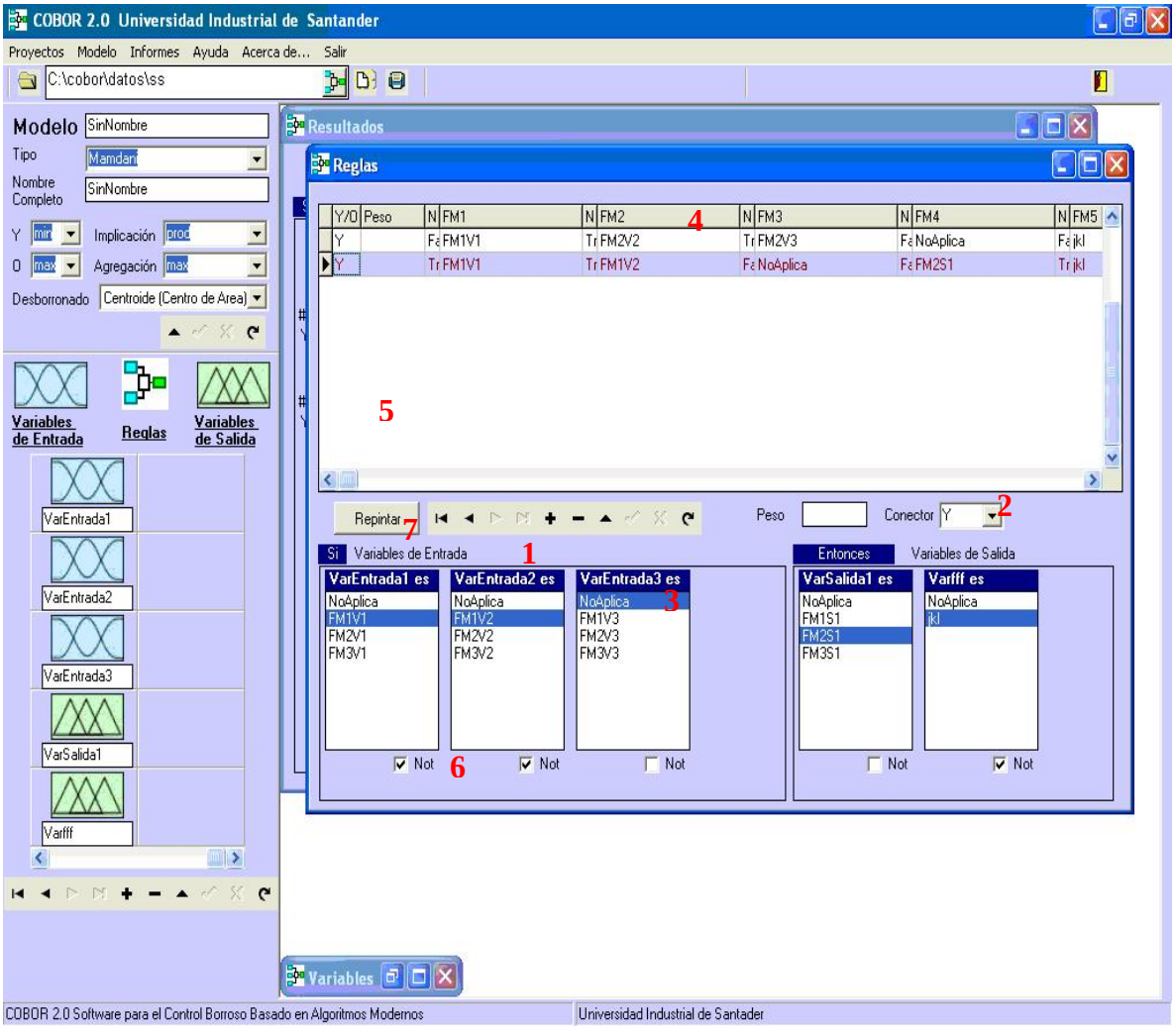
En la ventana de Variables es importante tener en cuenta:

1. El nombre de la Variable
2. El rango de la variable. Generalmente está entre 0 y 1
3. El tipo de Variable, si es de Entrada o Salida.
4. La Función Miembro de la Variable especificando un nombre para ésta función

5. El tipo de función miembro o de pertenencia o de membresía que se desee trabajar, en el sistema COBOR 2.0 se implementaron 5 funciones: Triangular, Trapezoidal, Gbell, Gaussiana y sigmoidal. Después de seleccionada oprimo la tecla TAB para asumir la función de pertenencia.
6. Se Visualiza el esquema de cada función de pertenencia a trabajar.
7. Se escogen los valores de los parámetros para éstas funciones.
8. En la gráfica central se observa las funciones miembros de la variable y si se desea cambiar los parámetros, se ubica en el punto que se desee mover ya sea aumentando o disminuyendo la posición y la gráfica permite que esto genere unos nuevos parámetros de acuerdo a la posición deseada.

VENTANA REGLAS

Figura 92. Ventana Reglas Detallada



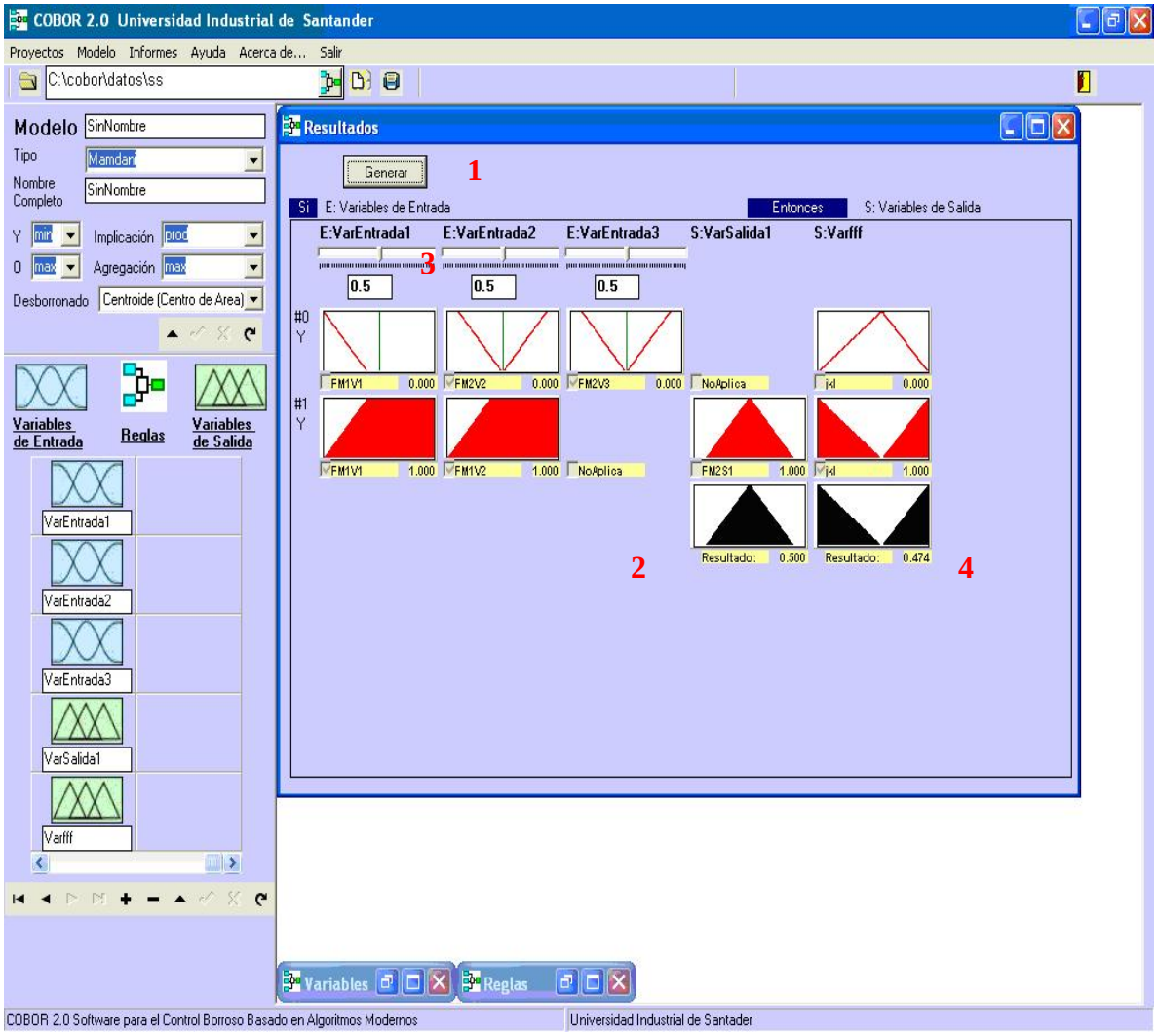
Visualizamos la ventana Reglas es importante tener en cuenta:

1. El experto en el control borroso genera las reglas que él desee implementar de acuerdo a las variables que se generaron.

2. Se selecciona el conector que se utilizará en la regla ya sea Y ó O
3. Es importante que todas las reglas estén bien definidas por ejemplo si hay variables que no aplican se debe seleccionar la opción no aplica de lo contrario generará errores en los resultados. El no aplica significa que en los resultados no se tiene en cuenta y no se genera gráfica.
4. Se hacen todas las posibles combinaciones que se deseen implementar con las funciones miembro de las variables.
5. El sistema trabaja hasta 15 variables.
6. Si selecciona la opción not, quiere decir que la función es negada, es decir que se multiplica la función por –1 y en la gráfica se visualiza al revés.
7. El botón Repintar lo utilizamos cuando se olvido crear una variable y ya estando creado el proyecto se genera. Entonces oprimo el botón y visualizó los cambios con la nueva variable.

VENTANA RESULTADOS

Figura 93. Ventana Resultados Detallada

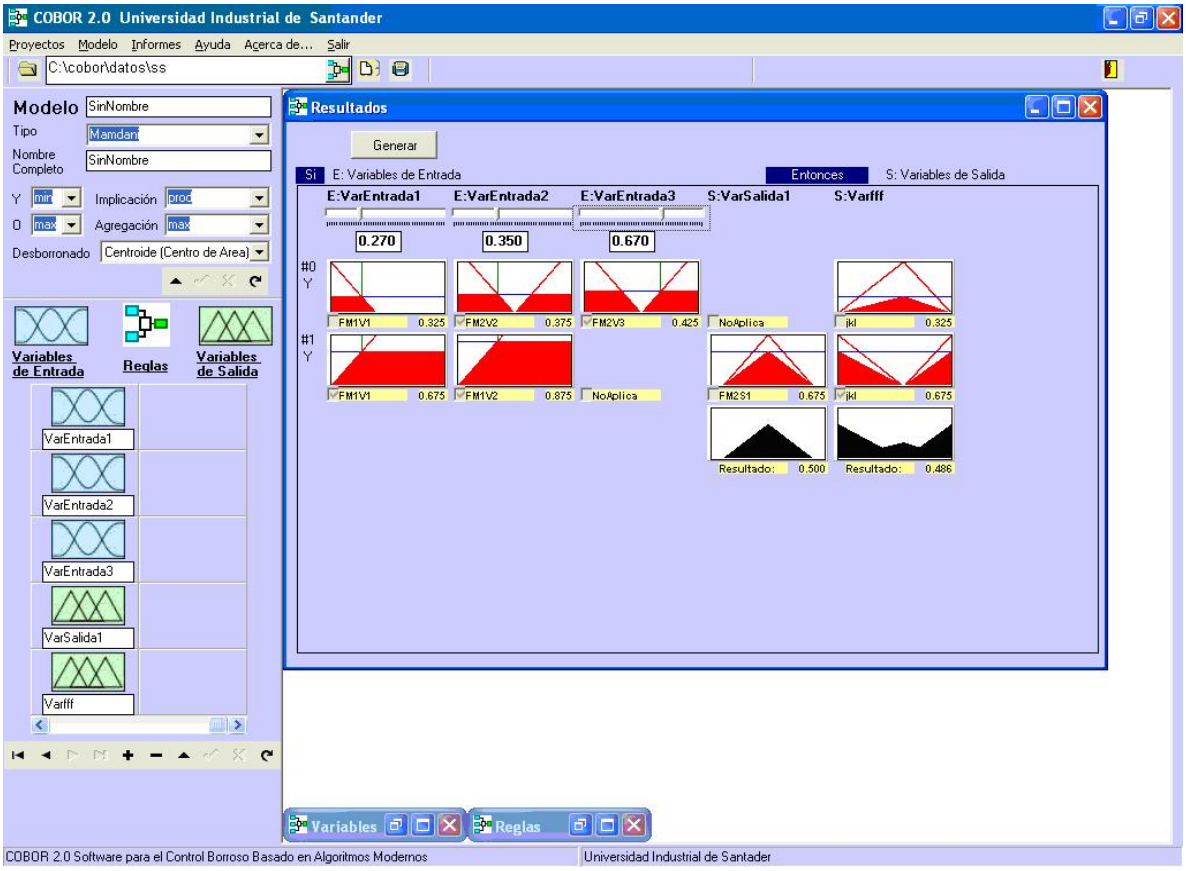


Luego que tengo listas las variables de entrada y Salida así como las Reglas, el experto en el control borroso necesita observar los resultados del proceso, para ello se visualiza la ventana de Resultados

En ésta ventana es importante tener en cuenta:

1. Para visualizar las gráficas oprimo el botón Generar.
2. El sistema permite observar las variables de Entrada y Salida y las gráficas de resultados.
3. En las variables de entrada se observan unas barras de desplazamiento horizontales que me permiten cambiar los valores de las variables de entrada y así obtener diferentes resultados.
4. Todo se observa de manera gráfica lo que permite al experto en control borroso tomar la decisión más acertada.

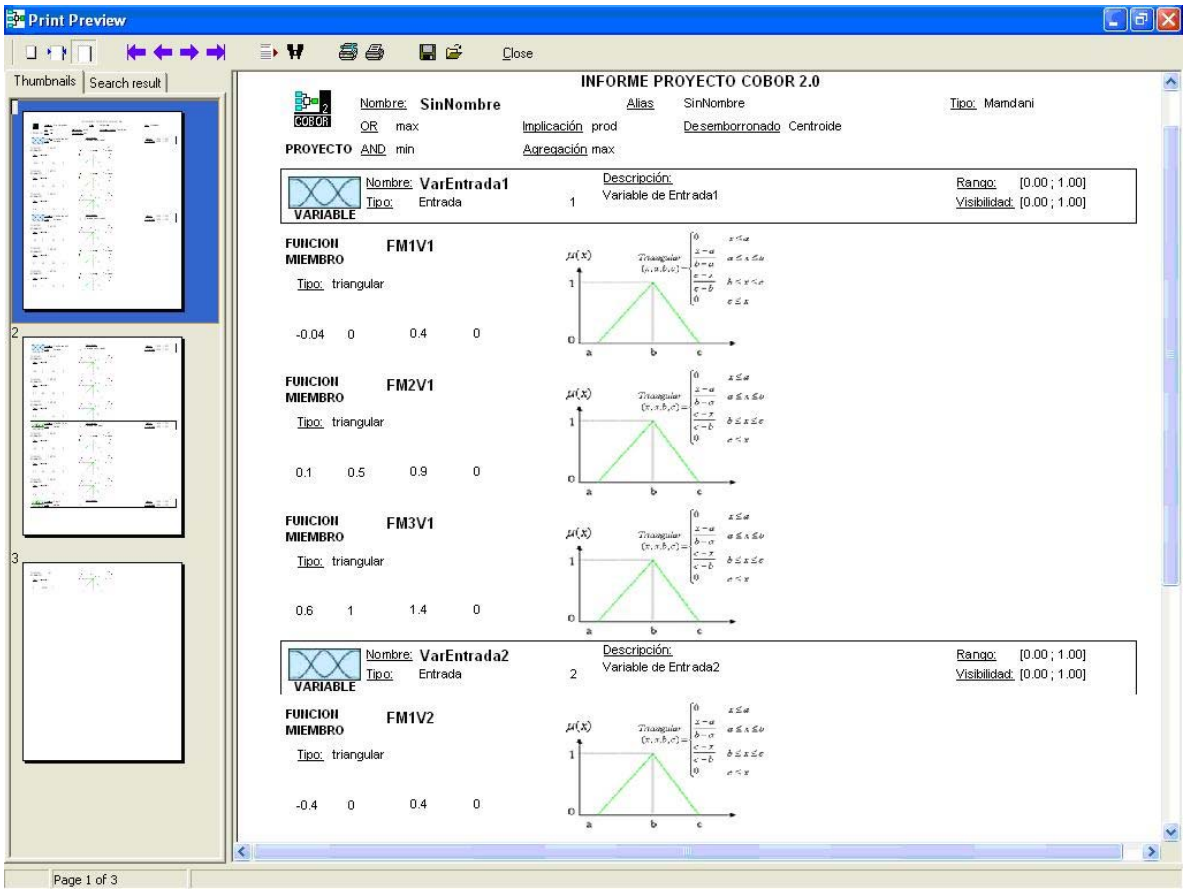
Figura 94. Ventana Resultados con Gráficas



Moviendo las barras de desplazamiento en las variables de entrada, obtengo diferentes resultados que se observan de una manera gráfica.

VENTANA INFORMES

Figura 95. Ventana Informes detallada del Proyecto



Ya generado todo el proceso para el control borroso, visualizamos los informes que se generan a partir de todo el proyecto, de una forma gráfica y útil para el experto, esta opción se encuentra en el menú principal y seleccionamos informes. El Informe generado nos permite en la barra de tareas visualizarlo en diferentes tamaños, avanzar en las diferentes páginas, retroceder, ir a una determinada página, buscar texto, imprimir, guardar el informe, abrir un informe y salir.