CONTROLADORES DIFUSOS. EJEMPLO DE EXTRACCIÓN AUTOMÁTICA DE BASES DE CONOCIMIENTO PARA CONTROLADORES DIFUSOS DESARROLLADOS CON FUZZY-CLIPS

Joaquín Peña, Francisco Leal y Antonio Ruiz
Informe Técnico
Dpto. Lenguajes y Sistemas Informáticos de la Universidad de Sevilla
Avda. de la Reina Mercedes, s/n. Sevilla, 41.012
E-mail: jpena@lsi.us.es

INDICE

1 Introducción: Utilización de la Lógica Difusa en los problemas de Control.

La Lógica Difusa, basada en la Teoría de Conjuntos Difusos [3], hace uso de dichos conjuntos para modelar la incertidumbre y los conceptos imprecisos o ambiguos. Mientras que en la Teoría de Conjuntos Clásica un elemento pertenece o no a un conjunto, la Teoría de Conjuntos Difusa reconoce que puede haber distintos grados de pertenencia a uno o más conjuntos. Los conjuntos difusos permiten, gracias a esto, hacer un tratamiento lingüístico de los elementos y, aunque ésta no sea la manera perfecta de describir la realidad, se aproxima mucho más que cualquier otra existente.

Una de las formas básicas de representar el conocimiento es bajo la forma de reglas de decisión o control del tipo "Si (condición)...Entonces..(decisión)...". Pero, generalmente, el conocimiento de un experto (o la información que se obtiene de un proceso) se explica en términos de conceptos imprecisos e implica operaciones y decisiones no bien definidas o concretas.

Los conjuntos difusos representan, en cierta manera, "valores lingüísticos" de una determinada magnitud. De este modo surge el concepto de "variable lingüística" como aquella que toma valores lingüísticos, por ejemplo, Temperatura="muy baja","alta"... frente a la variable numérica clásica, que sólo admite valores concretos, como Temperatura=20ºC., 40ºC., etc...

El elemento primario de la Lógica Difusa es el Lenguaje Natural, y sus esquemas de razonamiento son esquemas de "razonamiento aproximado" con proposiciones imprecisas, típicamente de carácter lingüístico, como podrían ser las reglas que se obtienen a partir de la expresión lingüística del conocimiento de un operador humano versado en el control de un determinado proceso.

Uno de las principales aportaciones de la aplicación de la lógica difusa en la automatización de procesos es la inclusión de la heurística y de la experiencia acumuladas por un operador experto. Esto ha llevado a pensar en la posible sustitución de leyes de control poco efectivas por soluciones basadas en consideraciones heurísticas sobre el proceso a controlar.

En principio, se podría pensar que los datos que se utilizan en un sistema de control son objetivos y precisos, pero no siempre es así. En suma, la Lógica Difusa, o más adecuadamente la Teoría de Conjuntos Difusos, encuentra sus aplicaciones más importantes en sistemas complejos que no tienen linealidades en su forma de operar, y que por extensión ha encontrado sus principales aplicaciones en el control de:

· Sistemas no lineales.
· Sistemas con perturbación no predecible, o sensores poco precisos, y
· Sistemas donde se necesita incorporar la experiencia humana.

Además, los datos de entrada incluso pueden ser subjetivos y vagos (ej.: los que se obtienen de la observación humana). Es precisamente debido a estas imprecisiones por lo que se incluye la heurística.

El Control de Procesos es el campo en que la lógica difusa ha tenido mayor auge, ya que la inclusión de lógica heurística permite obtener soluciones simples y eficientes para el control de determinados procesos.

La introducción de heurística en forma de reglas puede contemplar y mejorar el comportamiento de un algoritmo de control discreto, introduciendo reglas "clásicas" basadas en la Lógica Binaria clásica. Pero en el diseño y obtención de reglas que involucran los esquemas vagos de razonamiento de un experto, la heurística no puede ser transferida o asimilada, al menos de manera directa, en forma de reglas desarrolladas según la Lógica Clásica y la Teoría Clásica de Conjuntos. La experiencia de un experto puede aparecer recogida en un conjunto de reglas del tipo:

> "SI (la Humedad es BAJA) Y (la Temperatura es MUY ALTA) ENTONCES (aumentar CONSIDERABLEMENTE la Potencia del sistema)".

Estas reglas pueden entenderse como combinaciones de ciertas variables y magnitudes lingüísticas dentro del ámbito de la Teoría de los Conjuntos Difusos y la Lógica Difusa.

Los esquemas de razonamiento aproximado se implementarán a partir de operadores y relaciones construidos según las propiedades y operaciones básicas definidas sobre los conjuntos difusos.

Los Controladores basados en Lógica Difusa (FLCs) [1] pueden considerarse como controladores digitales en los que la señal de control que se aplica sobre el proceso se obtiene tras una "inferencia difusa" a partir de las variables actuales del proceso (normalmente salida y consigna actuales), y una matriz de reglas, obtenida como protocolo lingüístico proporcionado por ingenieros u operadores del proceso.

En un controlador basado en reglas, el principal esfuerzo de diseño reside fundamentalmente en la adquisición adecuada de la heurística del experto en forma de reglas de control. El éxito de los Controladores Difusos no radica en sí en la utilización de conjuntos difusos, sino en su naturaleza basada en reglas. Esto no quiere decir que la teoría de conjuntos difusos no sea importante. Ésta hace que la implementación de un controlador basado en reglas de carácter heurístico sea mucho más fácil que por cualquier otro método.

En definitiva, el desarrollo de los controladores basados en reglas de naturaleza heurística necesita de los conjuntos difusos para modelar los conceptos e informaciones vagas e imprecisas, necesita de una Teoría de Conjuntos Difusos que defina las propiedades y operaciones características de este nuevo tipo de conjuntos, que permitan desarrollar operadores y conectivos lógicos adecuados para combinar la información cualitativa formando proposiciones lógicas que reflejen la heurística de control (reglas), así como implementar los esquemas de inferencia o "razonamiento apropiado" que requiere esta nueva concepción lógica (Lógica Difusa).

2 Controladores Difusos.

2.1 Estructura de un Controlador Difuso

La figura 2.1 muestra la estructura genérica de un FLC. Un FLC es un tipo de Sistema Basado en Reglas Difusas que está compuesto por una **Interfaz de Fuzzificación**, una **Base de Conocimiento**, un **Sistema de Inferencia** y la **Interfaz de Defuzzificación.**

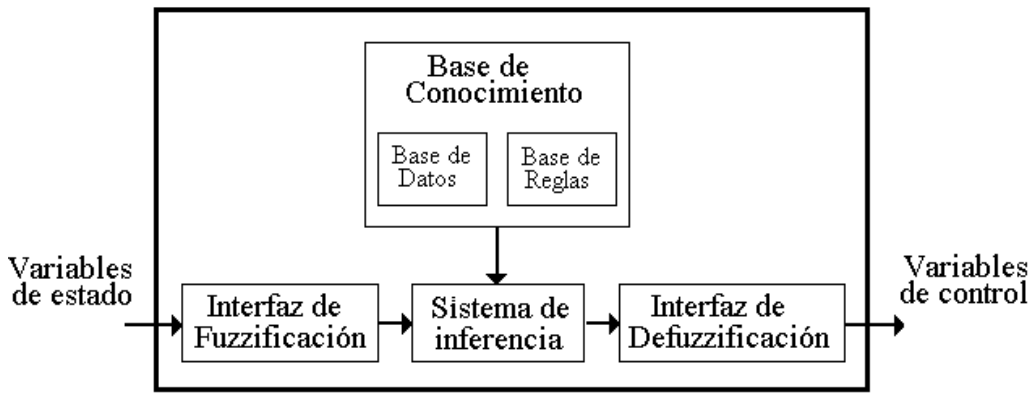

Figura 2.1. Estructura genérica de un FLC.

La **Interfaz de Fuzzificación** realiza un escalado (no siempre necesario) de los valores de las entradas para adecuarlos a los valores típicos para los que se define el controlador, y posteriormente realiza la *fuzzificación*. La *fuzzificación* consiste en la conversión de los datos de la entrada en valores lingüísticos apropiados para ser utilizados como entidades difusas.

La **Base de Conocimiento** está constituída por una *Base de Datos* y una *Base de Reglas*. La Base de Datos contiene las definiciones de las etiquetas lingüísticas, es decir, la funciones de pertenencia de los Conjuntos Difusos en que se dividen las variables de entrada y de control. La Base de Reglas es una colección de reglas de control difuso que representa el conocimiento experto del sistema a controlar.

El **Sistema de Inferencia** inferirá las acciones de control difuso empleando alguna representación de la implicación difusa, así como de los procedimientos de la lógica difusa.

La **Interfaz de Defuzzificación** convierte la acción difusa de control actualmente inferida en una acción concreta susceptible de aplicación sobre el proceso, y realiza un escalado (cuando sea necesario) para adecuar los rangos de salida para los que se ha definido el controlador con las entradas del proceso.

2.2 Base de Conocimiento.

2.2.1 Reglas Difusas. Base de Reglas.

Considérese como ejemplo ilustrativo un problema genérico que puede ser resuelto en términos de razonamiento aproximado por un ser humano experto en la materia, generalmente siguiendo el esquema del modus ponens generalizado. Esto quiere decir que el experto es capaz, mediante un conjunto de reglas o pautas de actuación que constituyen su experiencia sobre el tema, de elaborar unas conclusiones o consecuentes a partir de unos hechos observados o antecedentes. El experto podrá transmitir sus conocimientos, al menos parcialmente, mediante un conjunto de N reglas del tipo "IF-THEN", con antecedentes relacionados por el conectivo "AND" en la mayoría de los casos.

IF	A_1 AND	B_1	THEN	C_1
IF	A_2 AND	B_2	THEN	C_2
		.	.	
		.	.	
		.	.	
IF	A_N AND	B_N	THEN	C_N

Supóngase que A_i, B_i y C_i son números difusos en los universos **A**, **B** y **C**, cuyas funciones de pertenencia estarán determinadas también por el experto.

Cada una de las reglas se puede expresar como una relación difusa definida mediante los conectivos "AND" y "THEN". El conectivo AND es la conjunción y el conectivo THEN representa la implicación. La representación de estos conectivos se puede realizar mediante el uso del mínimo, obteniendo la siguiente expresión:

$$\mu_{Ri}=\min\{\mu_{Ai}(a), \mu_{Bi}(b), \mu_{Ci}(c)\}$$

donde *e*, *de* y *u* son elementos pertenecientes a los universos **A**, **B** y **C** respectivamente. Esta expresión determina el grado de verdad de la variable lógica formada por la proposición lógica (regla).

La experiencia total del experto estará formada por la unión de todas las reglas. Como cada regla heurística de control está representada por una relación difusa, el comportamiento global del Controlador Difuso podrá caracterizarse por una única relación difusa que será una combinación adecuada de las relaciones que constituyen cada regla. La combinación puede realizarse con un conectivo "OR". Es decir, podrá considerarse que es la relación difusa que resulta de aplicar el conectivo "OR" a cada una de las relaciones que expresan las reglas individuales. Simbólicamente,

$$R=OR\ (R_1, R_2,.....,R_N)$$

Desde un punto de vista práctico, los aspectos de eficiencia y computabilidad exigibles a un controlador industrial requieren una simplificación del algoritmo difuso. Se optará como el máximo para la implementación del conectivo "OR":

$$R=R_1 \cup R_2 \cup ... R_N = \cup R_i \ (i=1...n)$$

$$\mu_R(a,b,c)=\max \{\min\{\mu_{Ai}(a), \mu_{Bi}(b), \mu_{Ci}(c)\}\} \quad \text{siendo } 1 \leq i \leq N.$$

La experiencia del ser humano, recogida y almacenada en este tipo de reglas, es lo que se denomina Base de Reglas.

Existen varios modos de derivación de las reglas de control, siendo este punto fuente de numerosas investigaciones actualmente. Entre éstos se pueden destacar:

- Los basados en la experiencia de un experto, generalmente a través de la expresión verbal de su experiencia como experto, o a partir de cuestionarios cuidadosamente organizados.
- Los basados en las acciones de control de un operador, en función de los datos de entrada-salida observados o muestreados.
- Los basados en un modelo difuso del proceso.
- Los basados en el aprendizaje (controladores autorganizados).

Para ilustrar lo explicado anteriormente sobre la Base de Reglas, se propone el siguiente ejemplo:

Supóngase un sistema que controla la potencia de un sistema de aire acondicionado en base a la temperatura y a la humedad ambientes.

Supóngase que el sistema de refrigeración se desea controlar con una serie de reglas definidas mediante la observación de un experto. El experto generará las reglas que determinan los valores de la salida (*potencia*) en función de las variables de entrada (*temperatura* y *humedad).*

La siguiente figura representa la estructura de la *Base de Reglas* en que se apoya el sistema. Esta *Base de Reglas*, como se puede observar en la figura, correspondería al sistema presentado anteriormente, con sus variables de entrada (*temperatura* y *humedad* ambientes). La variable de salida (*potencia* que se le suministra al sistema) no aparece en la tabla expresamente. Su valor se determina mediante la intersección de un valor de *temperatura* y uno de *humedad*. Ahora se explicará el procedimiento para obtener el valor de la salida a partir de cualesquiera valores en las entradas.

Base de Conocimiento:
Base de Reglas

Humedad \ Temperatura	VL	L	H	VH
VL	VL	VL	L	L
L	VL	L	L	H
H	L	L	H	H
VH	H	VH	VH	VH

VL = Muy bajo
L = Bajo
H = Alto
VH= Muy alto

Figura 2.2. Tabla correspondiente a la Base de Reglas.

Los valores que se representan a la derecha de la tabla (VL,L,H,VH) son las etiquetas lingüísticas que se hacen corresponder a los valores de cada una de las variables (de entrada y de salida).

La tabla sintetiza las reglas que ha generado el experto. Por ejemplo, la correspondencia entre el valor de *temperatura* H (temperatura alta) y el de valor de *humedad* L (humedad baja) da lugar, según la tabla, a un valor de la potencia de L (baja potencia para el sistema).

La regla generada sería :

IF *Temperatura* **IS** *H* **AND** *Humedad* **IS** *L* **THEN** *Potencia* **IS** *L*

Otras posibles reglas serían:

IF *Temperatura* **IS** *H* **AND** *Humedad* **IS** *VL* **THEN** *Potencia* **IS** *L*
IF *Temperatura* **IS** *H* **AND** *Humedad* **IS** *VH* **THEN** *Potencia* **IS** *VH*
.
.
.

2.2.2 Funciones de Pertenencia. Base deDatos.

Las reglas de un Controlador Difuso se caracterizan mediante la cuantificación y normalización de los Universos de Discurso, el número de Conjuntos Difusos o categorías lingüísticas de entrada o de control, así como las funciones de pertenencia asociadas a cada uno de estos conjuntos.

El número de niveles de cuantificación determina la exactitud del control obtenido. El número de categorías lingüísticas determina la granularidad del control que se puede obtener con un Controlador Difuso.

A la hora de definir los Conjuntos Difusos y sus funciones de pertenencia existen dos posibilidades:

a) que los universos sean *discretos*: la función de pertenencia de un Conjunto Difuso se representa mediante un vector de números cuya dimensión depende del grado de discretización.

b) que los universos sean *continuos*: las funciones de pertenencia se representan mediante funciones regulares, normalmente triangulares o trapezoidales.

La elección de la forma de la función de pertenencia se decidirá según el proceso a realizar. Por ejemplo, si los datos medibles van a estar afectados por ruídos, las funciones de pertenencia deberían ser suficientemente anchas para poder reducir la sensibilidad del controlador frente a estos. De lo anterior se deduce que la especificación de las funciones de pertenencia será un aspecto fundamental a la hora de proporcionar robustez al controlador. Posteriormente se explicará con más detalle la importancia de que las funciones de pertenencia sean lo suficientemente anchas.

En la figura 2.3 se pueden observar las funciones de pertenencia para diferentes clasificaciones lingüísticas, también llamadas variables lingüísticas o Conjuntos Difusos. Las

variables lingüísticas son, en este caso: *VL,L,H,VH* (ver ejemplo anterior sobre la *Base de Reglas*). Dichas variables lingüísticas están definidas por las funciones trapezoidales definidas por el experto (de ahí su irregularidad) que se observan en la figura 2.3.

La variable particionada en Conjuntos Difusos será la *temperatura* (ver ejemplo anterior sobre la *Base de Reglas*). La figura indica, por tanto, a qué variable o variables lingüísticas corresponde un determinado valor lingüístico de la variable *temperatura*. Un valor lingüístico de la variable puede pertenecer a varios Conjuntos Difusos, pero la función de pertenencia tomará diferentes valores según la variable lingüística que se esté considerando.

Ejemplo: Supóngase que el valor numérico de la variable temperatura es 43º. En la figura 2.3 se observa que dicho valor se encuentra dentro de dos clasificaciones lingüísticas: H y VH. El valor 43º tendrá distintos grados de pertenencia para cada una de estas clasificaciones lingüísticas: se puede observar que el grado de pertenencia al Conjunto Difuso VH es mayor que el grado de pertenencia al conjunto H.

Como se verá posteriormente, el grado de pertenencia determina la actitud del controlador, ya que a cada valor lingüístico de la variable (*temperatura)* se le asignará la clasificación lingüística *(VL,L,H,VH)* cuya función de pertenencia (función trapezoidal en este caso) se ajuste más a dicho valor de variable (es decir, la clasificación lingüística cuya función de pertenencia obtenga el máximo valor para cada valor lingüístico de la variable).

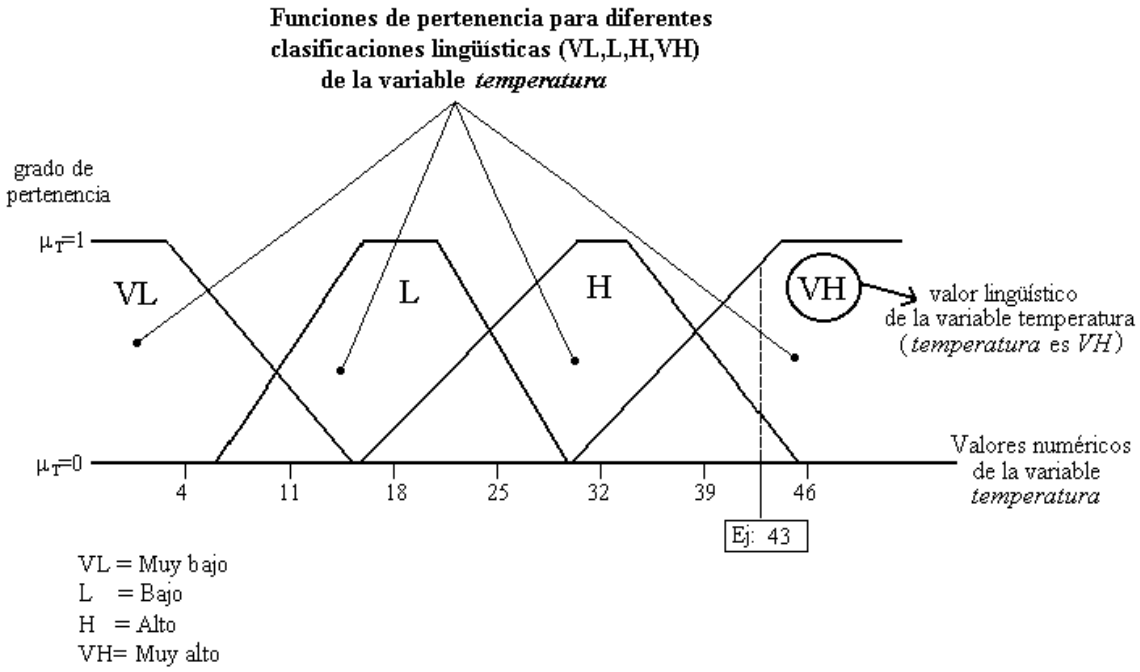

Figura 2.3. Funciones de Pertenencia para diferentes clasificaciones lingüísticas.

2.3 Interfaz de Fuzzificación.

En el momento de plantear la realización de un sistema real, hay que tener en cuenta que los valores que se introduzcan en el sistema no vienen dados en forma de Conjuntos Difusos, sino como valores concretos de las diferentes variables del sistema. Por ejemplo: un termómetro no proporciona la temperatura ambiente con valores como VL (muy baja) o H (alta), sino con valores concretos de la temperatura en grados.

Para controlar cualquier proceso se necesitan valores concretos que introducir al sistema.

Es necesario, por tanto, utilizar algún tipo de interfaz entre los valores concretos y los Conjuntos Difusos de las variables. La interfaz de fuzzificación ha de ser capaz de elaborar Conjuntos Difusos a partir de las entradas concretas del controlador.

Existen dos tipos de Fuzzificación:

- *Fuzzificación Puntual:* El Conjunto Difuso correspondiente se construye con base en un valor concreto. Supóngase que las variables de entrada poseen un valor concreto en el instante actual: los valores serán $t=t_o$ (temperatura) y $h=h_0$ (humedad). Los Conjuntos Difusos de entrada se pueden definir, por tanto, con las siguientes funciones de pertenencia (ver además figura 2.4):

$$\mu_T(t)=\begin{cases} 1 \text{ si } t=t_0 \\ 0 \text{ en cualquier otro caso} \end{cases}$$

$$\mu_H(h)=\begin{cases} 1 \text{ si } h=h_0 \\ 0 \text{ en cualquier otro caso} \end{cases}$$

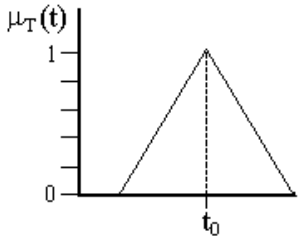

Figura 2.4. Representación de la Fuzzificación Puntual para la variable t.

- *Fuzzificación no Puntual:* En este caso, cuando $t=t_0$, $F(t_0)=1$, y la función de pertenencia del resto de los valores para el universo T decrece a medida que los valores se alejan de t_0. El mismo proceso se sigue para el resto de variables de entrada, creando una función de pertenencia para cada variable (ver figura 2.5)

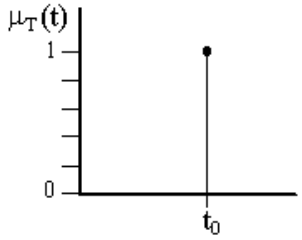

Figura 2.5. Representación de la Fuzzificación no Puntual para la variable t.

2.4 Motor de Inferencia. Inferencia Difusa.

El motor de inferencia representa el núcleo del controlador, y agrupa toda la lógica de inferencia borrosa del controlador, de barrido de las reglas durante ésta, elección refinada de reglas a utilizar, etc.

El Sistema de Inferencia o Motor de Inferencia Difusa se basa en la aplicación del Modus Ponens Generalizado (GMP), una extensión del Modus Ponens clásico, propuesto por Zadeh[33] como sigue:

IF X IS A THEN Y IS B
X IS A'
———————————————
Y IS B'

La sentencia condicional Si X es A entonces Y es B (siendo X e Y variables lingüísticas, A y B conjuntos difusos) representa una relación difusa entre A y B, definida en UxV, (siendo $U=U_1 x\ U_2 x...x\ U_n$ y V los universos de las variables de entrada $X_1,...X_n$ y la salida Y, respectivamente). La relación difusa se expresa mediante un conjunto difuso R cuya función de pertenencia es $\mu_R(x,y)$ viene dada por:

$$\forall\ x \in \mathbf{U},\ y \in \mathbf{V}:\ \mu_R(x,y) = I(\mu_A(x),\ \mu_B(y))$$

siendo $\mu_A(x)$ y $\mu_B(x)$ las funciones de pertenencia de los conjuntos difusos A y B, respectivamente, e I un operador de implicación difuso que modela la relación difusa.

La función de pertenencia del conjunto difuso B´ en el consecuente, obtenida por el MPG, se deduce mediante una proyección sobre **V** en términos de la Regla Composicional de Inferencia (RCI) (presentadas por Zadeh[33]) dada por la siguiente expresión:

$$\mu_{B'}(y)=Sup_{x\in U}\{t'(\mu_{A'}(x),\ I(\mu_A(x),\ \mu_B(y)))\}$$

donde

$$\mu_{A'}(x)=t(\mu_{A'1}(x),\ ...,\ \mu_{A'n}(x)),\ \mu_A(x) = t(\mu_{A1}(x),\ ...,\ \mu_{An}(x))$$

siendo t y t´ t-normas e I un operador de implicación.

Debido al hecho de que la entrada x correspondiente a la variable de estado es un valor concreto, $x=x_0$, entonces A´ es una función puntual, esto es, $\mu_{A'}(x)=1$ si $x=x_0$, , y $\mu_{A'}(x)=0$ si $x\neq x_0$. Por lo tanto, el RCI se reduce a la siguiente expresión:

$$\mu_{B'}(y)=I(\mu_A(x_0) , \mu_B(y))$$

Partiendo de esto, se deduce que depende directamente del operador difuso seleccionado. En literatura específica, se proponen una enorme cantidad de operadores que pueden utilizarse como operadores de implicación en el proceso de control de inferencia difusa. Se han publicado muchos estudios que añaden información para seleccionar este operador. [3,4,6,7,8,9,12,15,17,21,23,27]

Como se ha comentado, el cáculo de $\mu_A(x_0)$ consiste en la aplicación de un operador de conjunto $\mu_{A'i}(x_i)$:

$$\mu_A(x_0)=\mathbf{t} (\mu_{A1}(x_1),...,\mu_{An}(x_n))$$

Al resultado de la aplicación del operador conector **t** se le llama comunmente emparejamiento, y representa los emparejamientos entre valores presentados en las entradas y los conjuntos difusos de los antecedentes de la regla. Se le denominará **h**.

El Motor de Inferencia produce tantos Conjuntos Difusos de salida como reglas existan en la Base de Conocimiento.

El operador de implicación que se utilizará para implemetar el Motor de Inferencia se podrá definir de muchas maneras, como se indica a continuación.

Una clasificación de los operadores de implicación difusa es la propuesta por Dujet y Vincent [11] considerando la extensión que ellos realizan sobre la lógica booleana:

- Aquellos que extienden la Implicación Booleana.

  Dentro de este grupo, se encuentran las *Funciones de Implicación* difusas [26]. Estos operadores satisfacen la siguiente tabla de verdad:

a\b	0	1
0	0	0
1	0	1

- Aquellos que extienden la Conjunción Booleana.

  Las *Force Implication* [11] y *t-normas*, cuando se usan como operadores de implicación,[12] se incluyen en este grupo satisfaciendo la siguiente tabla de verdad:

a\b	0	1
0	1	1
1	0	1

Existen muchos operadores de implicación que no pertenecen a ninguna de estas dos familias. En lo sucesivo, se van a mostrar algunos ejemplos de operadores de implicación [7,8,12,22] pertenecientes a estos tres grupos.

- *Operadores de Extensión de la Implicación Booleana.*

Las *Funciones de Implicación*[26] son los operadores más conocidos que extienden la implicación booleana.

Una función continua I: [0,1]x[0,1]→[0,1] es una *Función de Implicación* difusa sii ∀ x,x' ,y,y',z∈ [0,1] que verifica las siguientes propiedades: [26]

1. Si x≤x' entonces I(x,y)≥ I(x',y)
2. Si y≤y' entonces I(x,y)≤ I(x,y')
3. Elemento Neutro: I(0,x)=1
4. Principio de Identidad: I(1,x)=x
5. Propiedad Asociativa: I(x,I(y,z))=I(y,I(x,z))

Los operadores que extienden la implicación booleana están clasificados en diferentes familias: [26,27]

Strong Implications (S-Implicaciones): (Kleene-Dienes) Corresponden a la definición de la implicación en la lógica Booleana clásica:

$$A \Rightarrow B = \neg A \vee B.$$

Presentan la forma: I(x,y)=**s**(N(a),b), siendo **s** una t-conorma y N una función de negación.

Residual Implications (R-Implicaciones): Obtenidas como residuo de una *t-norma*, tal como sigue: I(x,y) = sup{c:c∈ [0,1]/ **t** (c,x)≤y}

Las *Funciones de Implicación* que deben ser seleccionadas son aquellas que muestren el mejor comportamiento. A continuación se muestran algunos ejemplos de estas funciones.

Implicaciones-S:
Dubois-Prade:

$$\mathbf{I(x,y)} = \begin{cases} 1\text{-}x, & si\ y=0 \\ y, & si\ x=1 \\ 1 & en\ otro\ caso \end{cases}$$

Implicaciones-R:
Goguen:

$$\mathbf{I(x,y)} = \begin{cases} 1, & si\ y \leq x \\ y, & en\ otro\ caso \end{cases}$$

Implicaciones S-R:
Lukasiewicz:

$$\mathbf{I(x,y)}=\text{Min}(1,1-x+y)$$

- *Otras extensiones de la implicación booleana:*

Hay algunos operadores de implicación que extienden la Implicación Booleana pero no verifican las propiedades de las *Funciones de Implicación*. Se ha seleccionado el siguiente operador debido a su buen comportamiento en estudios precedentes [7,8]. Se ha seleccionado el siguiente operador debido a su buen comportamiento en estudios precedentes [7,8].

$$\mathbf{I(x,y)}=\begin{cases} 1, si\ x=y \\ y, en\ otro\ caso \end{cases}$$

- *Operadores de Extensión de la Conjunción Booleana. t-normas*:

(a) *t-normas*:

La *t-norma* se definió en Teoría de Conjuntos. Podemos observar los siguientes ejemplos de *t-normas* como extensión de la implicación:

Producto Lógico (Mínimo):

$$\mathbf{I(x,y)}=\text{Min}(x,y)$$

Producto Algebraico:

$$\mathbf{I(x,y)}=x*y$$

---

b) Operadores de *Force-Implication*.

Los operadores de *Force-Implication* fueron introducidos para "modelar el razonamiento humano de una manera más natural sin necesidad de generar una implicación". [11]

b.1) Operadores de *Force-Implication* basados en operadores de indistinguibilidad:

Están constituídos por la siguiente expresión:

$$I(x,y)=\mathbf{t}\ (x\ ,\ E(x\ ,\ y)\ )$$

Donde t es una *t-norma*, y E es un operador de indistinguibilidad

$$E=\mathbf{t'}(I'(x,y),I'(y,x))$$

Siendo t'una *t-norma*, e I' una *Función de Implicación*.
Existen tres tipos diferentes de operadores de indistinguibilidad, dependiendo de la *t-norma* utilizada para definirlos: [26]

· Relaciones de Similitud: **t'** (x,y)=Min (x,y).
· Relaciones Probabilísticas: **t'** (x,y)=x*y.
· Relaciones de Parecido: **t'**(x,y)=Max (0,x+y-1).

Se van a mostrar los tres operadores de la *Force-Implication* que presentaron el mejor comportamiento basados en la familia de operadores de indistinguibilidad. Han sido obtenidos en términos del operador de indistinguibilidad[26] de E_{Godel} y tres *t-normas*: lógica, algebraica y productos ponderados. Sus expresiones son:

$$\mathbf{I(x , y)}=\text{Min } (x , E_{Gödel}(x , y)$$

Donde $E_{Gödel}$ (x , y) es :

- 1 si x=y
- Min (x , y) en cualquier otro caso.

$$\mathbf{I(x , y)}=x * E_{Gödel}(x , y)$$
$$\mathbf{I(x , y)}=\text{Max } (x + E_{Gödel}(x , y) - 1 , 0)$$

(b.2) *Force Implication* basadas en distancias:

Este segundo grupo obedece a la expresión:

$$I(x,y)=t(x,1-d(x,y))$$

Donde t es una *t-norma*, y *d* es una distancia.
Se considerará un operador de *Force Implication* basado en la *t-norma* del producto ponderado cuya expresión es:

$$\mathbf{I(x , y)}=\text{Max } (x - | x - y| , 0)$$

2.5 Interfaz de Defuzzificación.

Del mismo modo que se necesita un método para pasar los valores concretos de entrada del Controlador Difuso a Conjuntos Difusos, se necesita un método para realizar el paso contrario en las variables de control. Así se podrán obtener valores concretos susceptibles de aplicación sobre el proceso.

Existen dos tipos de métodos de defuzzificación [1,7,31] de acuerdo al modo en que los conjuntos individuales $B_{i'}$ se agregan para ejecutar el trabajo del conector *también*, llamado G:

· ***Modo A***: *Primero Agregación, Después Deffuzificación:* La Interfaz de Defuzzificación realiza la agregación de cada uno de los conjuntos difusos inferidos, $B_{i'}$, para obtener el conjunto difuso final *B'*.

$$\mu_{B'}(y)=G\{\mu_{B'1}(y), \mu_{B'2}(y),..., \mu_{B'n}(y)\}$$

Normalmente, el operador de agregación que modela al conector *también* (G) es el mínimo o el máximo. Tras esto, el conjunto difuso B' se defuzzifica usando cualquier estrategia como puede ser "Media de los Máximos".

Media de la Máxima (normalmente llamada MOM):

$$y_0 = (y_1 + y_2)/2$$

cuando $y_1 = \text{Min}\{z/\mu_{b'}(z) = \text{Max}\ \mu_{b'}(y)\}$; $y_2 = \text{Max}\ \{z/\mu_{b'}(z) = \text{Max}\ \mu_{b'}(y)\}$

· ***Modo B***: *Primero Deffuzificación, Después Agregación:* Evita el cálculo del conjunto difuso final *B'* considerando la contribución de cada salida de la regla individualmente, obteniendo la acción final de control mediante el cálculo (una media, una suma ponderada o una selección de alguna de ellas) de un valor concreto característico asociado a cada uno de ellos.

Por ejemplo, *Valor Máximo ponderado por el emparejamiento*[7,13,28]:

$$y_0 = \Sigma_i (h_i * G_i) / \Sigma_i h_i$$

siendo h_i el grado de emparejamiento y G_i

Históricamente, el modo A fue por primera vez propuesto y utilizado en las aproximaciones iniciales al control difuso [19,20] . Por otro lado, el modo B es uno de los más usados hoy en día en los sistemas de tiempo real más avanzados (que necesitan una respuesta lo más rápida posible) y en muchas otras clases de sistemas, debido a su simplicidad.

2.6 Ejemplo de diseño de un Controlador Difuso

Hasta ahora se han explicado los diferentes pasos que realiza un Controlador Difuso. Este apartado presenta un ejemplo de realización de un Controlador Difuso desde el principio, tratando los diferentes pasos a seguir para su diseño.

El sistema elegido es el del control de la potencia suministrada a un sistema de aire acondicionado propuesto anteriormente.

El primer paso es definir las variables de entrada y de salida del sistema: las entradas serán, como se explicó en apartados anteriores, la *temperatura* y la *humedad*.

La salida del sistema será la *potencia* que se le proporciona al aparato de aire acondicionado.

El muestreo será realizado mediante un operador humano que tome periódicamente muestras de la temperatura ambiente mediante un termómetro y de la humedad mediante un higómetro. La potencia que se le suministra al sistema se medirá mediante un polímetro.

Existen, por tanto, tres Universos de Discurso, uno por cada variable. En cada universo se definirán cuatro Conjuntos Difusos, cada uno correspondiente a una variable lingüística.

Las variables lingüísticas serán VL, L, H y VH (muy bajo, bajo, alto y muy alto).

Una vez definidas tanto las variables del sistema como los Universos de Discurso y sus categorías lingüísticas, se puede pasar a la definición de las funciones de pertenencia de cada uno de los Conjuntos Difusos.

En este caso se suponen las siguientes funciones de pertenencia para las variables de estado T y H (ver figura 2.6)

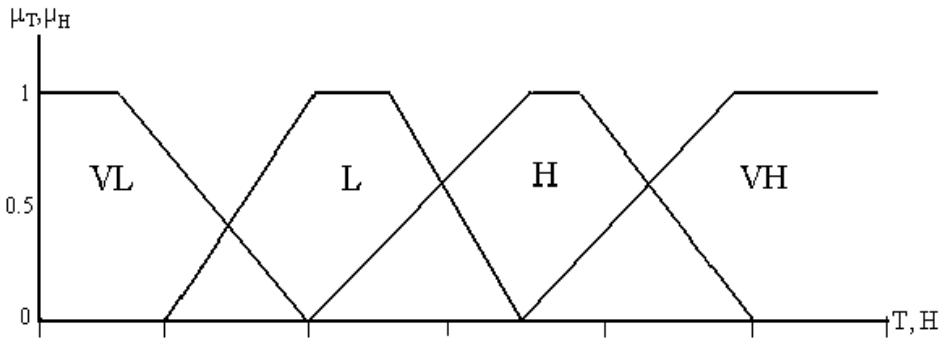

Figura 2.6. Funciones de Pertenencia para las variables de estado T y H

y ésta otra para la variable P (ver figura 2.7):

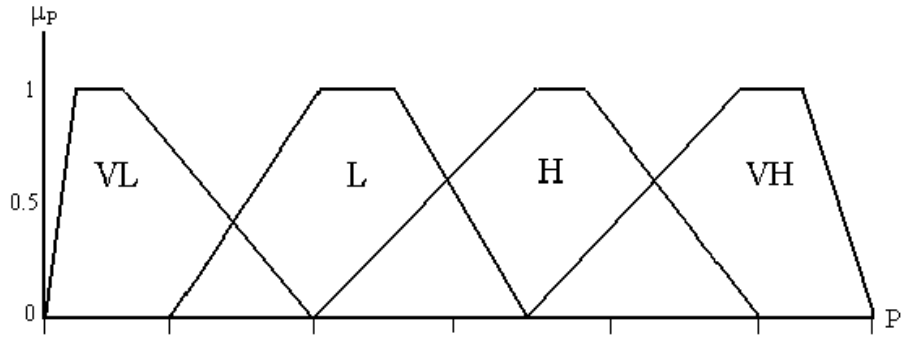

Figura 2.7. Función de Pertenencia para la variable de control P.

En este caso las funciones de pertenencia han sido definidas por el experto. También podrían haber sido definidas arbitrariamente.

Como se puede observar en la figura, los Conjuntos Difusos se solapan entre sí. Esto es así para que no existan zonas muertas entre los Conjuntos Difusos y siempre se activen dos Conjuntos Difusos por universo. Esto provoca la activación de cuatro reglas en el mismo periodo de muestreo, ya que si a un valor de T le corresponden dos Conjuntos Difusos y a un valor de H le corresponden otros dos, se generarán cuatro posibles reglas.

Ejemplo: Para los valores concretos de los sensores de 8.5º de temperatura y 81% de humedad se obtendrían las siguientes reglas:

IF *Temperatura* **IS** *VL* **AND** *Humedad* **IS** *H* **THEN** *Potencia* **IS** *L*
IF *Temperatura* **IS** *VL* **AND** *Humedad* **IS** *VH* **THEN** *Potencia* **IS** *H*
IF *Temperatura* **IS** *L* **AND** *Humedad* **IS** *VH* **THEN** *Potencia* **IS** *L*

IF *Temperatura* **IS** *L* **AND** *Humedad* **IS** *VH* **THEN** *Potencia* **IS** *VH*

ya que t_0=8.5º pertenece a los Conjuntos Difusos VL y L, y h_0=81% a los conjuntos H y VH (ver figuras 2.8 y 2.9).

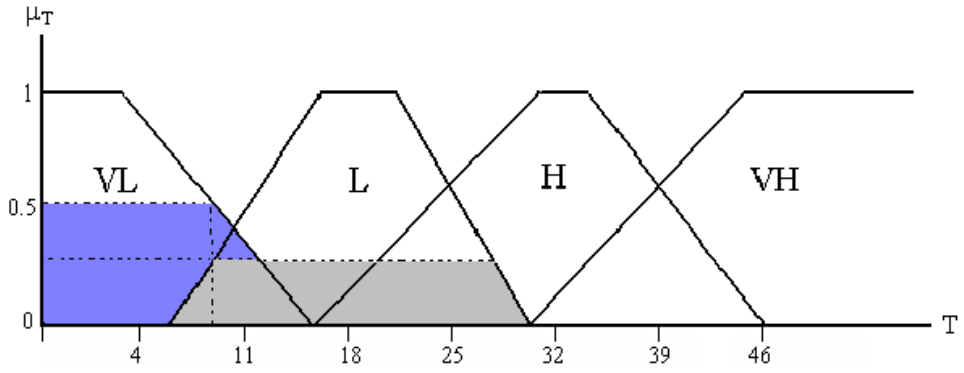

Figura 2.8. Correspondencia del valor de temperatura t_0=8.5º.

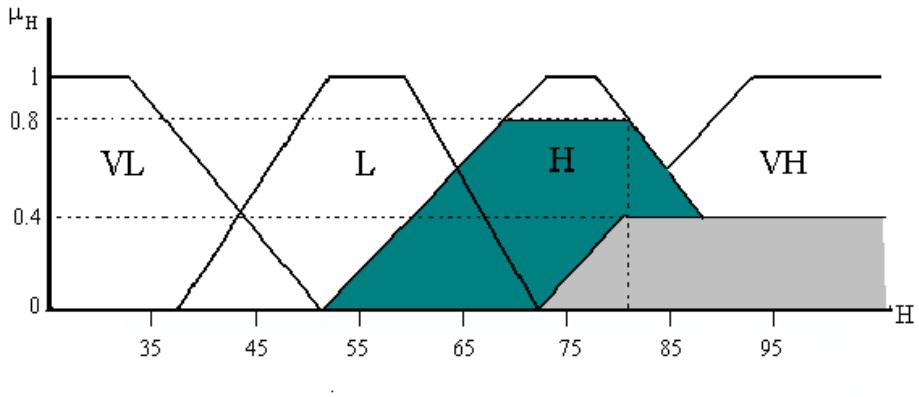

Figura 2.9. Correspondencia del valor de humedad h_0=81%.

Si únicamente se activara un Conjunto Difuso por universo, se generaría una sola regla por cada combinación de valores. La existencia de varias reglas permite realizar una selección de la más adecuada entre las mismas (se elige la de mayor grado).

Si la temperatura es, por ejemplo, de 8.5º, no tiene sentido clasificarla únicamente como "temperatura muy baja" ya que está cerca de ser simplemente "baja". Es esta la razón por la que se realiza el solapamiento.

Hay que hacer notar también que los Conjuntos Difusos VL y VH están acotados para la variable P ya que, si no fuera así, el procedimiento de defuzzificación mediante el método del centro de gravedad o mediante el método del máximo grado de pertenencia no sería posible.

Una vez se han definido las funciones de pertenencia, se procede a generar las reglas de la Base de Conocimiento. Las reglas se obtienen, en este ejemplo, de las acciones del controlador sobre la variable de salida ante las variables de entrada. Se genera una tabla como la siguiente (ver figura 2.10):

Base de Conocimiento:
Base de Reglas

Humedad \ Temperatura	VL	L	H	VH
VL	VL	VL	L	L
L	VL	L	L	H
H	L	L	H	H
VH	H	VH	VH	VH

VL = Muy bajo
L = Bajo
H = Alto
VH= Muy alto

Figura 2.10. Base de Reglas

Posteriormente se generan los diferentes grados de activación de las reglas según los valores de las variables del sistema. Para ello, se recorren cada una de las reglas de la Base de Reglas, realizando la inferencia según la tabla de la figura 2.10.

Aquí se representan dos de las cuatro posibles reglas cuyo grado de emparejamiento no es 0 (ver figura 2.11):

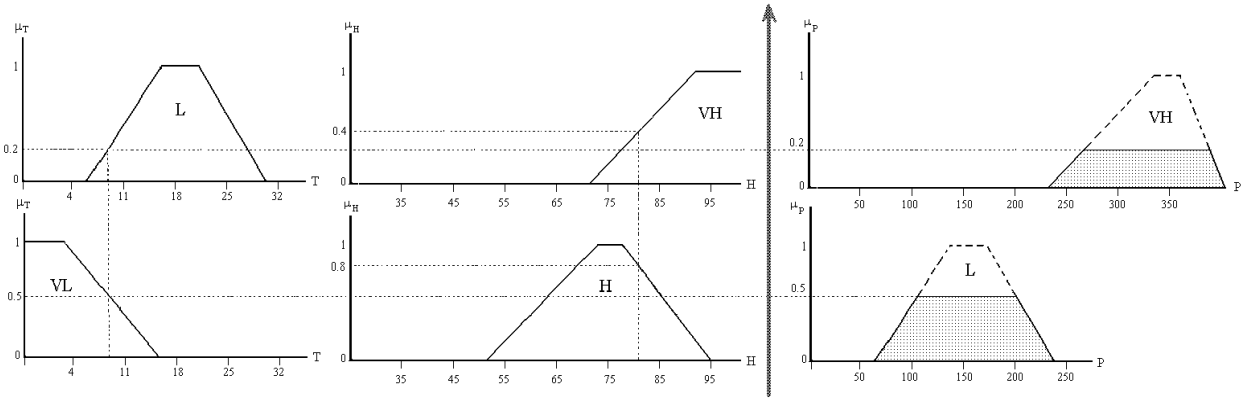

Figura 2.11. Base de Reglas

Las reglas representadas son:

IF *Temperatura* **IS** *L* **AND** *Humedad* **IS** *VH* **THEN** *Potencia* **IS** *VH*
IF *Temperatura* **IS** *VL* **AND** *Humedad* **IS** *H* **THEN** *Potencia* **IS** *L*

Se ha utilizado la t-norma del mínimo como conectivo AND, de forma que $\alpha_i=\min\{\mu_{Ti}(t), \mu_{Hi}(h)\}$. Como operador de implicación se ha utilizado el Mínimo.

Tras esto, se realiza el proceso de defuzzificación. Se utilizará el Modo A (Agregación primero y defuzzificación después), obteniendo el resultado siguiente (ver figura 2.11):

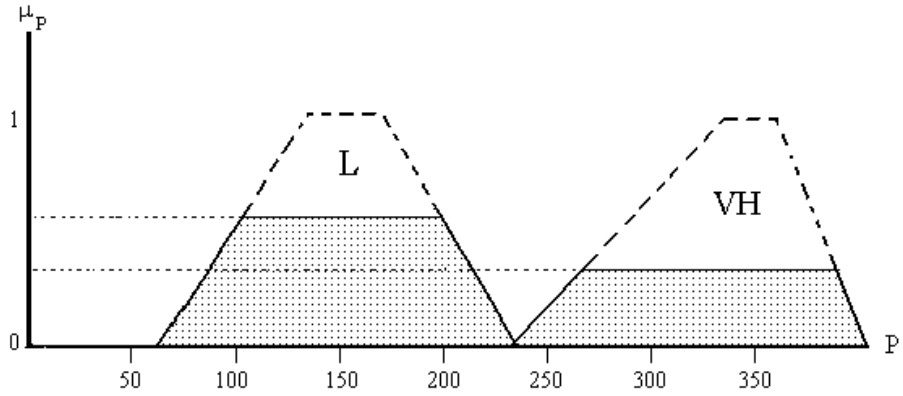

Figura 2.11. Base de Reglas

3 Algoritmo de Wang y Mendel.

Una de las técnicas más conocidas para poder construir las reglas de control de un sistema basado en Lógica Difusa es el algoritmo de Wang y Mendel [4]. Mediante dicho algoritmo, se obtiene una Base de Reglas de control difusas partiendo de un conjunto de ejemplos y, opcionalmente, de la experiencia aportada por un experto humano (en caso de disponer de ella).

Estos dos aspectos se relacionan íntimamente: el conjunto de ejemplos (proporcionado al algoritmo mediante un muestreo del sistema) complementa a las reglas del operador experto, ya que éste puede no ser capaz de expresar todas y cada una de las reglas que realmente utiliza para controlar su sistema. Por otro lado, la obtención de las reglas mediante un muestreo puede ser apoyada mediante reglas concretas creadas por el experto.

En un principio, el algoritmo de Wang y Mendel no surgió con este fin, sino que pretendía ser un Controlador Difuso que aprendiera por sí mismo a generar las reglas que lo controlaran.

Para la generación de la mencionada Base de Reglas Difusas, se parte de un conjunto de ejemplos que consiste en los pares numéricos "estado del sistema" y "acción de control a ejecutar" en tal estado.

Este conjunto de ejemplos es obtenido como una matriz de datos procedente del muestreo en el tiempo de las variables, tanto de estado como de control, del sistema que se desea controlar mientras el experto actúa sobre el sistema de modo satisfactorio.

El algoritmo de Wang y Mendel cubre el vacío producido por la probable incapacidad, por parte del experto humano, de expresar de manera certera y eficiente todas y cada una de las reglas que utiliza realmente a la hora de controlar un sistema determinado.

Además, el algoritmo permite la posibilidad de incorporar la experiencia de un experto humano en la medida de lo posible (o lo que es igual "en qué situación se realizan acciones") expresada en forma de reglas "IF-THEN".

Gracias a esto último se puede complementar la Base de Reglas procedente del conjunto de ejemplos con la Base de Reglas generada por el experto.

Este algoritmo, por tanto, es un método de obtención de Bases de Reglas para un FLC que sustituya a un controlador humano, ya que necesita los ejemplos de actuación del experto humano en forma numérica.

Descripción del Algoritmo:

El Algoritmo de Wang y Mendel se divide en cuatro pasos claramente diferenciados. Tras la ejecución secuencial de cada uno es los pasos se obtendrá la Base de Reglas que contiene las reglas por las que se regirá el Controlador Difuso correspondiente.

- **Paso 1:** *Definición de las particiones difusas de las variables de estado y control.*

La primera operación en la definición de las particiones difusas es tomar la primera variable de estado presente en la matriz. Se realiza entonces un recorrido de todos los valores presentes en la matriz para esta variable, y se tomarán el valor máximo y el valor mínimo (rango de valores de la variable en la matriz).

Una vez hecho esto, se procederá a realizar una división del rango de valores en 2N+1 regiones, correspondiendo cada una de ellas a un Conjunto Difuso. Cada Conjunto Difuso, además, se define mediante una función de pertenencia.

Las anteriores operaciones se repetirán de manera análoga para el resto de las variables de estado y de control presentes en la matriz, pudiendo ser tanto el valor de N como el tipo de función de pertenencia distintos para cada una de las variables.

Del párrafo anterior se puede deducir que el algoritmo de Wang y Mendel no genera por sí sólo la Base de Datos que utiliza el Controlador Difuso, ya que requiere que tanto el parámetro N como los tipos de funciones de pertenencia sean definidos por el usuario.

- **Paso 2:** *Generación de Reglas Difusas desde los ejemplos.*

Una vez obtenida la Base de Datos procedente del paso anterior, se podrán generar las reglas difusas a partir de los ejemplos.

Cada ejemplo, formado por el par *variables de estado* y *variables de control* produce una regla. La etiqueta difusa que corresponde a cada antecedente u consecuente de la regla es la función de pertenencia con la que mejor empareje el valor numérico de esa variable en cada ejemplo, es decir, la variable lingüística cuya función de pertenencia obtenga un mayor valor en ese punto.

- **Paso 3:** *Asignación de grados a cada regla generada.*

A cada regla generada por su correspondiente ejemplo se le asocia un grado (de 0 a 1). Para calcular el grado que le corresponde a cada regla, se realizarán las dos operaciones siguientes:

- Recorrer cada una de las variables de estado y de control que la componen, calculando el valor de la función de pertenencia para la etiqueta difusa que se le ha asignado en el paso anterior.

  De este modo, se obtiene un grado de pertenencia para cada una de las variables que constituyen la regla, tanto antecedentes (variables de estado) como consecuentes (variables de control).

- Una vez ejecutada la operación anterior, el grado de la regla en cuestión se puede calcular como el producto de los grados de pertenencia de todas las variables que componen la regla.

Una vez se ha calculado el grado de todas las reglas, éste se utilizará para Reducir el conjunto de reglas, que en principio contiene tantas reglas como ejemplos.

La reducción se realiza de la siguiente forma:

· Si en el conjunto de reglas hay más de una regla con los mismos antecedentes, se eliminan las que tengan menor grado. Esto resuelve conflictos entre reglas, ya que a iguales antecedentes (parte "IF") con distintos consecuentes (parte "THEN") constituyen acciones contradictorias que son resueltas manteniendo únicamente la regla de mayor grado. Tal elección es razonable en tanto otorga confianza a la regla que cuenta con mayor *emparejamiento* de sus variables a las etiquetas difusas de la partición.

· Cuando existan más de una regla iguales (partes "IF" y "THEN" iguales), permanecerá en la Base de Reglas la de mayor grado asociado, eliminándose las ocurrencias que lo tengan menor.

- **Paso 4:** *Conformación de la Base de Reglas combinada.*

En este paso se procederá a la fusión de las reglas incluidas en la Base de Reglas de naturaleza numérica procedente de los pasos anteriores y las reglas de naturaleza lingüística aportadas por el experto.

Esta fusión se realizará de forma que en la Base de Reglas final se encuentren tanto las reglas de experto como las reglas procedentes de ejemplos.

Si se representa la Base de Reglas en una tabla con una dimensión de entrada por cada antecedente, las reglas obtenidas de los ejemplos ocuparán una sola entrada por cada regla, ya que se trata de reglas AND. Del mismo modo ocurre con las reglas del experto.

El conjunto de ejemplos consiste en los pares numéricos "estado del sistema" y "acción de control a ejecutar" en tal estado. La experiencia del experto se expresa en forma de reglas "IF-THEN".

4 Ejemplo de aplicación: Sistema Experto de Diagnóstico Médico.

Para mostrar la utilidad de la lógica difusa a la hora de realizar un sistema, se partirá de los conceptos anteriormente mencionados para diseñar una Base de Conocimiento que determine las decisiones de un *Sistema Experto de Diagnostico medico para bypass coronario [2]*.

Se generará una Base de Conocimiento a partir de datos proporcionados por médicos especialistas en dicha materia. La Base de Conocimiento se utilizará posteriormente a la hora de decidir que tratamiento es posible aplicar sobre un determinado paciente cuyos datos proporciona el usuario.

4.1 Generación de la Base de Conocimiento.

A la hora de realizar la Base de Conocimiento se partirá de un fichero de ejemplos con el siguiente formato (extraído de las tablas con datos reales ya existentes):

```
A A A  A A A   C C  C C C;

0  1 1 0 0 60  2 2 0 2 2;
0  1 1 0 0 40  2 2 0 2 2;
0  1 1 0 0 20  2 2 0 2 2;
        .
        .
        .
```

Las variables de entrada del sistema están representadas mediante la letra **A** y las de salida mediante la letra **C**. Cada linea del fichero contiene un conjunto de valores (un valor por cada variable). El fichero representa, por tanto, los valores que deben tomar las variables de salida del sistema para unos determinados valores de entrada. Es así como expresa su conocimiento un experto: mediante unas acciones concretas de salida para un estado del sistema que sirve como entrada (en este caso el estado de un paciente).

A continuación se realiza una breve explicación de cada una de las variables que intervienen en este sistema experto:

Variables de entrada:

Riesgo quirúrgico: Puede tomar el valor S o N, dependiendo de si existe riesgo a la hora de realizar la intervención quirúrgica en el paciente.

Resultado del test de esfuerzo: Previamente a la decisión de intervenir al paciente éste debe realizar un test de esfuerzo. Esta variable representa el resultado de

dicho test. Puede tomar el valor P (positivo) o el valor N (negativo) (1 ó 0 en el fichero de ejemplos, respectivamente).

LTCD: Esta variable representa si el paciente padece o no Enfermedad del Tronco Común Izquierdo (Left Common Trunk Disease). (1 ó 0 en el fichero de ejemplos, respectivamente).

BVD: Representa el grado en que el paciente padece Enfermedad de los Vasos Sanguíneos (Blood Vessel Disease). Tomará el valor 0 si no la padece. En caso contrario tomará los valores 1, 2 o 3 según el grado en que la padezca.

PADA: Representa si en la familia del paciente existen o no antecedentes de la enfermedad (1 ó 0 en el fichero de ejemplos, respectivamente).

LFVD: Indica la Fracción de Expulsión del Ventrículo Izquierdo. Esta variable puede tomar cualquier valor entre 0 y 100 (expresa un porcentaje).

Variables de salida:

Posibilidad de aplicar la técnica de Revascularización: Mediante esta variable se indica lo adecuado de aplicar la técnica de Revascularización. Puede tomar los valores + (es una técnica adecuada para el estado del paciente), - (no se debe aplicar la técnica) ó **?** (existen dudas sobre si la técnica puede o no aplicarse).

Nota: El valor + se representa con el valor numérico *2*, el valor – con *0* y las posible duda con *1*.

PTCA: Indica lo mismo que la variable anterior para la técnica de PTCA.

CABG: Idem para la técnica de CABG.

Grado de acuerdo en la aplicación de la Revascularización: Esta variable indica en qué grado están de acuerdo los especialistas médicos a la hora de aplicar o no la técnica de revascularización. Puede tomar los valores **D** (los especialistas están en desacuerdo), **I** (hay una considerable diversidad de opiniones) ó **A** (los especialistas están de acuerdo).

Nota: Los valores D, I y A se representan como *0*, *1* y *2* respectivamente.

Grado de acuerdo en la aplicación del PTCA o CABG: Representa lo mismo que la variable anterior, pero para la aplicación de la técnica PTCA o CABG (sólo es posible aplicar una de estas dos técnicas, de ahí que sólo exista una variable).

Todas estas variables están presentes en cada una de las reglas que genera el sistema en la Base de Conocimiento. Tras generar la Base de Conocimiento se obtiene un fichero .clp con el siguiente formato:

```
(deftemplate A_0
-0.5      1.5
((A_0_0 (-0.5 0)  (0.0 1)  (0.5 0))
(A_0_1  (0.0 0)   (0.5 1)  (1.0 0))
(A_0_2  (0.5 0)   (1.0 1)  (1.5 0))
```

))

Mediante esta plantilla se están definiendo cada uno de los conjuntos difusos en que se divide el rango de valores de la variable de entrada A_0 (Riesgo quirúrgico). En este caso se crearán tres conjuntos difusos (siempre se utiliza un número impar de conjuntos difusos) a pesar de que la variable es binaria (existe o no riesgo quirúrgico). La variable, por tanto, siempre pertenecerá al conjunto difuso A_0_0 (valor de riesgo 0 con grado de pertenencia de 1) o al A_0_2 (valor de riesgo 1 con grado de pertenencia de 1).

Si la variable fuera el grado de BVD, se podrían crear cinco conjuntos difusos, indicando BVDs de 0, 1, 2 y 3 (es necesario crear 5 conjuntos porque el número de conjuntos debe ser 2*N+1 siendo N proporcionado por el usuario)

```
(defrule R0
(A_0 A_0_0)
(A_1 A_1_0)
(A_2 A_2_2)
(A_3 A_3_0)
(A_4 A_4_0)
(A_5 A_5_2)
=>
(assert (C_0 C_0_2))
(assert (C_1 C_1_2))
(assert (C_2 C_2_0))
(assert (C_3 C_3_2))
(assert (C_4 C_4_2)))
```

Las reglas de la Base de Conocimiento se representan mediante el código anterior. Para que se active la regla anterior deben darse las siguientes condiciones:

- La variable A_0 (Riesgo Quirúrgico) debe tener el valor difuso A_0_0.
- La variable A_1 (Valor del Test de Esfuerzo) tiene el valor difuso A_1_0.
- La variable A_2 (Valor de LTCD) tiene el valor difuso A_2_2.

.
.
.

Si se dan todas las condiciones, se introducirán los valores que el sistema ha calculado para las variables de control a partir del fichero de ejemplos del experto:

- La variable C_0 (Posibilidad de aplicar la técnica de Revascularización) tomará el valor difuso 2 (representa el valor + que indica que la técnica es adecuada).
- La variable C_2 (Posibilidad de aplicar la técnica de PTCA) tomará el valor difuso 2 (representa el valor + que indica que la técnica es adecuada).

.
.
.

Por lo tanto, en la Base de Conocimiento está sintetizado, en forma de reglas con valores difusos, el conocimiento que los expertos proporcionaron mediante ejemplos con valores concretos.

4.2 Funcionamiento de la aplicación:

A continuación se explica cómo poner en marcha el sistema experto de manera rápida. Posteriormente se detallará cómo funciona el sistema (cómo se genera la Base de Conocimiento a partir del fichero de ejemplos mediante la aplicación en Java "Generador Clips" y cómo se controla el sistema mediante reglas Clips).

4.2.1 Puesta rápida en funcionamiento.

- Instalar y ejecutar el programa Fuzzy Clips [5].
- Cargar el fichero "diagnostico.clp".

Nota: El fichero *diagnostico.clp* contiene la Base de Conocimiento generada a partir del fichero *ejemplos.ejp*, además de las reglas que constituyen el programa para pedir los datos al usuario y proporcionarle un diagnóstico. Las reglas propias del programa están visiblemente separadas en el fichero *diagnóstico.clp*.

- Hacer "(reset)".
- Ejecutar el código con "(run)".
- Proporcionar cada uno de los valores para las variables de entrada.
- Observar el diagnóstico que proporciona el sistema.

4.2.2 Explicación detallada del funcionamiento de la aplicación.

La aplicación "Generador de Bases de Conocimiento para Clips" es capaz de crear una Base de Conocimiento a partir de un fichero de ejemplos (.ejp). Dicha Base de Conocimiento se almacena en un fichero .clp que admite el programa Fuzzy Clips. A este fichero deben añadirse un conjunto reglas que, junto con las proporcionados por el Generador de Bases de Conocimiento Clips, son capaces de obtener un diagnóstico adecuado según los valores que le proporcione el usuario sobre un determinado paciente. La aplicación Clips se basará en las reglas de la base de conocimiento para proporcionar el diagnóstico.

Para generar la Base de Conocimiento a partir de un fichero de ejemplos se ejecutará la siguiente línea:

Java Fuzzyclips <nombre_entrada.ejp> <nombre_salida.clp> <lista de valores de n>

Un ejemplo de utilización sería:

Java FuzzyClips ejemplos.ejp base_de_conocimiento.clp 1 1 1 2 1 1 1 1 1 1

La línea anterior ejecuta la aplicación generadora de la Base de Conocimiento. Se utilizará "nombre_entrada.ejp" como fichero de ejemplos y "nombre_salida.clp" como fichero de salida (Base de Conocimiento).

Es necesario proporcionar una lista de valores de n (1 1 1 2 1 1 1 1 1 1 1). Cada valor proporcionado corresponde a una de las variables del sistema (A_0, A_1.... C_0, C_1...), y determina el número de conjuntos difusos en que se dividirá su rango (serán 2*n+1 conjuntos).

Para este conjunto de variables del sistema (las comentadas en la sección 1) la lista de valores de n será exactamente la que se ha puesto como ejemplo.

Puede parecer engorroso tener que proporcionar esta lista, pero de este modo se deja abierta la posibilidad de incorporar nuevas variables al sistema o de dividir el rango de alguna de ellas en más conjuntos difusos de una manera bastante sencilla (ej: si en lugar de 1 se usa n=2, el rango de la variable se dividirá en 2*2+1=5 conjuntos difusos).

A continuación se muestra una parte del contenido del fichero generado:

```
;Antecedentes: 6
;Consecuentes: 5
;Numero de reglas: 36

;La Base de Datos es:

(deftemplate A_0
        -0.5    1.5
(       (A_0_0          (-0.5 0)        (0.0 1)         (0.5 0))
        (A_0_1          (0.0 0)         (0.5 1)         (1.0 0))
        (A_0_2          (0.5 0)         (1.0 1)         (1.5 0))
        )
)
(deftemplate A_1
        1.0     1.0
(       (A_1_0          (1.0 0)         (1.0 1)         (1.0 0))
        (A_1_1          (1.0 0)         (1.0 1)         (1.0 0))
        (A_1_2          (1.0 0)         (1.0 1)         (1.0 0))
        )
)
        .
        .
        .


;La Base de Reglas es:

(defrule R0
        (A_0 A_0_0)
        (A_1 A_1_0)
        (A_2 A_2_2)
        (A_3 A_3_0)
        (A_4 A_4_0)
        (A_5 A_5_2)
        =>
        (assert (C_0 C_0_2))
        (assert (C_1 C_1_2))
        (assert (C_2 C_2_0))
        (assert (C_3 C_3_2))
        (assert (C_4 C_4_2))
)

(defrule R1
        (A_0 A_0_0)
        (A_1 A_1_0)
        (A_2 A_2_2)
        (A_3 A_3_0)
        (A_4 A_4_0)
        (A_5 A_5_1)
        =>
        (assert (C_0 C_0_2))
        (assert (C_1 C_1_2))
        (assert (C_2 C_2_0))
        (assert (C_3 C_3_2))
        (assert (C_4 C_4_2))
)

        .
```

.
.
.

Una vez se ha obtenido la Base de Conocimiento en un fichero .clp, hay que añadir a éste el conjunto de reglas que la utilizan para pedir los datos del paciente al usuario y proporcionar el diagnóstico.

Las reglas de control del sistema son:

```
(defrule inicializacion
        =>
        (set-fuzzy-inference-type max-min)
        (set-alpha-value 0.55))

        % ************************************************
        % (set-alpha-value 0.55)
        % Hace que se equiparen aquellos conjuntos difusos
        % que emparejen por encima del valor alpha=0.55

        %  (set-fuzzy-inference-type max-min)
        % Establece el modo de inferencia.
        % ************************************************


(defrule inicio
       =>
       (printout   t   "*****************************************************************"
crlf)
       (printout t "*  Sistema Experto de Diagnostico medico para bypass coronario *"
crlf)
       (printout   t   "*****************************************************************"
crlf)
       (printout t crlf)
       (printout t crlf)
       (assert (obtiene-datos)))


(defrule obtiene-datos
        (obtiene-datos)
       =>
        (assert(resultado))
       (printout t "¿Se produce riesgo quirurgico [S/N]? " crlf)
       (assert(riesgo (read)))

       (printout t "Introduzca el resultado del test de esfuerzo [P/N]: " crlf)
       (assert(etest (read)))

       (printout t "¿Se produce LCTD [S/N]?: " crlf)
       (assert(ltcd (read)))

       (printout t "¿Se produce BVD [S/N]?: " crlf)
       (assert(bvd (read))))


(defrule riesgo-positivo
       ?r<-(riesgo ?vriesgo&S|s)
       =>
       (retract ?r)
       (assert(A_0 A_0_2)))

(defrule riesgo-negativo
       ?r<-(riesgo ?vriesgo&N|n)
       =>
       (retract ?r)
       (assert(A_0 A_0_0)))


(defrule etest-esfuerzo-negativo
       ?r<-(etest ?vtest&N|n)
```

```
        =>
        (retract ?r)
        (assert(A_1 A_1_0)))


(defrule etest-esfuerzo-positivo
        ?r<-(etest ?vtest&P|p)
        =>
        (retract ?r)
        (assert(A_1 A_1_2)))


(defrule ltcd-negativo
        ?r<-(ltcd ?vltcd&N|n)
        =>
        (retract ?r)
        (assert(A_2 A_2_0)))


(defrule ltcd-positivo
        ?r<-(ltcd ?vltcd&S|s)
        =>
        (retract ?r)
        (assert(A_2 A_2_2)))


(defrule bvd-negativo
        ?r<-(bvd ?vbvd&N|n)
        =>
        (retract ?r)
        (assert(A_3 A_3_0))
        (printout t "¿Se produce PADA [S/N]?: " crlf)
        (assert(pada(read)))

        (printout t "¿Cual es el fraccion de expulsion del ventriculo izquierdo (1-
100%)?: " crlf)
        (assert(lvef (read))))


(defrule bvd-positivo
        ?r<-(bvd ?vbvd&S|s)
        =>
        (retract ?r)
        (assert (pregunta-grado-bvd)))

(defrule pregunta-grado-bvd
        (pregunta-grado-bvd)
        =>
        (printout t "¿Cual es el grado de bvd (1 2 ó 3)? " crlf)
        (assert (bvd-grado (read)))
        (printout t "¿Se produce PADA [S/N]?: " crlf)
        (assert(pada(read)))

        (printout t "¿Cual es el fraccion de expulsion del ventriculo izquierdo (1-
100%)?: " crlf)
        (assert(lvef (read)))
        (printout t crlf))

(defrule bvd-positivo-grado
        ?r<-(bvd-grado ?vbvd)
        =>
        (retract ?r)
        (assert(A_3 (?vbvd 0)(?vbvd 1)(?vbvd 0))))


(defrule pada-positivo
        ?r<-(pada ?vpada&S|s)
        =>
        (retract ?r)
        (assert(A_4 A_4_2)))

(defrule pada-negativo
        ?r<-(pada ?vpada&N|n)
        =>
```

```
        (retract ?r)
        (assert(A_4 A_4_0)))


(defrule lvef
        ?r<-(lvef ?vlvef)
        =>
        (retract ?r)
        (assert(A_5 (?vlvef 0)(?vlvef 1)(?vlvef 0))))



(defrule resultado-revascularizacion-0
        ?r<-(C_0 C_0_0)
        ?s<-(C_1 ?var)
         ?h<-(resultado)
        =>
        (retract ?r ?s ?h)
        (printout t crlf)
        (printout t "No se debe aplicar Revascularizacion con grado de acuerdo (0 1 ó 2)
" (moment-defuzzify ?s) crlf))

(defrule resultado-revascularizacion-1
        ?r<-(C_0 C_0_1)
        ?s<-(C_1 ?var)
         ?h<-(resultado)
        =>
        (retract ?r ?s ?h)
        (printout t crlf)
        (printout t "Indecisión a la hora de aplicar Revascularizacion con grado de
acuerdo (0 1 ó 2) " (moment-defuzzify ?s) crlf))

(defrule resultado-revascularizacion-2
        ?r<-(C_0 C_0_2)
        ?s<-(C_1 ?var)
         ?h<-(resultado)
        =>
        (retract ?r ?s ?h)
        (printout t crlf)
        (printout t "Se debe aplicar Revascularizacion con grado de acuerdo (0 1 ó 2) "
(moment-defuzzify ?s) crlf))




(defrule resultado-ptca-0
        ?r<-(C_2 C_2_0)
        ?s<-(C_4 ?var)
        =>
        (retract ?r )
        (printout t crlf)
        (printout t "No se debe aplicar PTCA con grado de acuerdo (0 1 ó 2) " (moment-
defuzzify ?s) crlf))

(defrule resultado-ptca-1
        ?r<-(C_2 C_2_1)
        ?s<-(C_4 ?var)
        =>
        (retract ?r)
        (printout t crlf)
        (printout t "Indecisión a la hora de aplicar PTCA con grado de acuerdo (0 1 ó 2)
" (moment-defuzzify ?s) crlf))

(defrule resultado-ptca-2
        ?r<-(C_2 C_2_2)
        ?s<-(C_4 ?var)
        =>
        (retract ?r)
        (printout t crlf)
        (printout t "Se debe aplicar PTCA con grado de acuerdo (0 1 ó 2) " (moment-
defuzzify ?s) crlf))

(defrule resultado-CABG-0
        ?r<-(C_3 C_3_0)
```

```
        ?s<-(C_4 ?var)
        =>
        (retract ?r ?s)
        (printout t crlf)
        (printout t "No se debe aplicar CABG con grado de acuerdo (0 1 ó 2) " (moment-
defuzzify ?s) crlf))

(defrule resultado-CABG-1
        ?r<-(C_3 C_3_1)
        ?s<-(C_4 ?var)
        =>
        (retract ?r ?s)
        (printout t crlf)
        (printout t "Indecisión a la hora de aplicar CABG con grado de acuerdo (0 1 ó 2)
" (moment-defuzzify ?s) crlf))

(defrule resultado-CABG-2
        ?r<-(C_3 C_3_2)
        ?s<-(C_4 ?var)
        =>
        (retract ?r ?s)
        (printout t crlf)
        (printout t "Se debe aplicar CABG con grado de acuerdo (0 1 ó 2) " (moment-
defuzzify ?s) crlf))

(defrule error
         (resultado)
        =>
         (reset)
        (printout t crlf)
        (printout t "El sistema no puede proporcionar un diagnostico para este paciente,
... !!" crlf)
        (printout t crlf))
```

Mediante este conjunto de reglas de control, el sistema puede responder con un diagnóstico basado en el conocimiento unificado de especialistas médicos al posible estado de los pacientes que el usuario le proporcione. Además, el sistema es fácilmente modificable, siendo posible introducir nuevos tratamientos o parámetros de entrada.

5 Bibliografía

[1] E.H. Mamdani, S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller, Int. J. of Man-Machine Studies 7 (1975) 1-13.

[2] L. M. Laita, E. Roanes-Lozano, V. Maojo. *A Computer Algebra Based Study of Inference and Verificaction. Application to Medical Appropriateness Criteria*. Fuzzy Sets and Systems. 1997.

[3] L.A. Zadeh, *Fuzzy Sets*, Information and Control 8 (1965) 338-353.

[4] L.X. Wang and J. Mendel, *Generating Fuzzy Rules by Learning from Examples*, IEEE Transactions on Systems, Man, and Cybernetics 22, 6 (1992) 1414-1427.

[5] R. A. Orchad. *FuzzyCLIPS version 6.04A*. User's Guide. National Research Council Canada. 1998