TESIS DE LICENCIATURA

VINCULANDO PROGRAMACIÓN LÓGICA Y LÓGICA DIFUSA

AUTOR: Diego Ariel Aizemberg

LU: 117/87

DIRECTOR: Osvaldo Gonzalez

FECHA: Febrero del 1997

1. Introducción

El objetivo de este trabajo es el estudio del estado actual de la lógica difusa y los formalismos clásicos de programación lógica, como resultado del mismo, la generación de un ambiente de programación lógica basado en los principios de la lógica difusa. La recopilación sobre lógica difusa pretende ser una introducción en esta disciplina, sin necesidad de conocimientos previos en el tema, ya que a partir de conocimientos de la teoría clásica de conjuntos extendemos hacia la teoría de conjuntos difusos. Si se desean ampliar conocimientos en este campo, se recomienda el libro "*Fuzzy Sets, Uncertainty, And Information*" de George Klir y Tina Folger [K&F_1992], además de los trabajos de Dubois y Prade "*Outline of fuzzy set theory: an introduction*" [D&P_1979] o de su libro "*Fuzzy Sets and Systems: Theory and Aplications*" [D&P_1980].

1.1 Abstract

The target of this work is to build an IDE (Integrated Development Enviroment) of fuzzy logic programming and fuzzy systems based on the study of the present state of art in Fuzzy Logic and the classical formalism for logic programming.

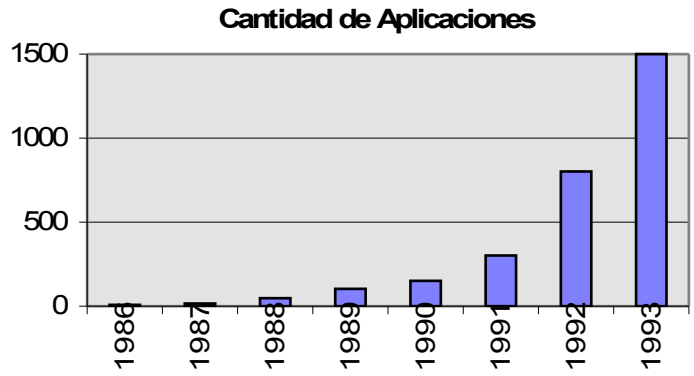
The survey of fuzzy logic shoud be red without any knowledge in this subject, only classical set theory will be needed.

2. Motivaciones de la Lógica Difusa

La lógica difusa es una herramienta potente para resolver problemas de mundo real y ha sido ampliamente usada especialmente en áreas de control y toma de decisiones. En general es usada en problemas que no son fácilmente definibles por modelos matemáticos prácticos, por ejemplo para controlar un sistema de trenes o tomar decisiones con acciones que cotizan en la bolsa de valores. Actualmente se está ampliando su campo de acción, ya que existen muchas aplicaciones que se resuelven al combinar redes neuronales y lógica difusa. Pero antes de hablar de lógica difusa deberíamos introducir el concepto de conjuntos difusos.

La teoría de conjuntos difusos provee una buena herramienta para modelar sistemas imprecisos. Las aplicaciones van desde economía y ciencias sociales hasta inteligencia artificial, desde procesos de control industrial hasta diagnóstico médico computarizado. Como se puede ver el campo de acción es realmente grande.

El profesor L. A. Zadeh por el año 1965 fue quien sentó las bases de los conjuntos difusos [Za_1965], a los cuales no se le prestó atención por más de 10 años. En la actualidad existe un auge alrededor de esta teoría. El siguiente gráfico muestra el número aproximado de aplicaciones industriales y comerciales realizadas con dicha tecnología [To_1993].



La siguiente tabla muestra una serie de acontecimientos históricos importantes para el desarrollo de los sistemas difusos:

1930	Lukasiewicz: Lógicas n-valuadas.
1965	Lotfi A. Zadeh: Creación de la teoría de conjuntos difusos.
1972	M. Sugeno: Medidas difusas.
1974	E.H. Mandani y otros: Aplicaciones de control difuso.
1982	Linkman: La primera aplicación industrial se puso en marcha en Dinamarca.
1985	M. Togai y H. Watanabe: Desarrollo conceptual de un VLSI difuso.
1986	Hitachi: Un controlador difuso se usó para controlar un subterráneo en Sendai después de ocho años de estudio.
1987	T. Yamakawa: Controlador difuso analógico.
1987	IFSA (International Fuzzy System Association) El subterráneo de Hitachi causó un boom industrial en Japón.
1988	M. Togai: Microprocesador digital difuso.
1989	LIFE (Laboratory for International Fuzzy Engineering Research) en Japón
1990	Aplicaciones comerciales en maquinas de uso diario, en Japón.
1992	La primer conferencia internacional sobre "*Fuzzy Systems*" organizada por la IEEE.
1993	*Transactions on Fuzzy Systems* organizada por la IEEE.

3. Introducción a la teoría de Conjuntos Difusos

Conjuntos difusos:

Un conjunto difuso está definido por una función continua de grados de pertenencia, ya no a la manera clásica donde teníamos sólo dos posibles valores, 0 ó 1, sino que pertenencia está dada sobre el intervalo real [0,1].

Las nociones de inclusión, unión, intersección, complemento y relación pueden ser extendidas a este tipo de conjuntos.

Por ejemplo, la clase de los animales claramente incluye perros, caballos, pájaros, etc. como miembros y excluye rocas, fluidos, plantas, etc. No obstante objetos como estrellas de mar, bacteria, ameba, etc. tienen un estado ambiguo con respecto a la clase de los animales. De la misma manera el número "10" en relación con la clase de los números mucho mayores que "1". También la clase de las "mujeres hermosas" o la de los "hombres altos" no constituyen clases o conjuntos a la manera clásica.

De aquí en adelante llamaremos *crisp sets* a los conjuntos clásicos para diferenciarlos de los *fuzzy sets*.

Definiciones:

Def. 3.1: Un conjunto difuso A en un universo de discurso X está caracterizado por su función de pertenencia μA (*membership function*) que toma valores en el intervalo [0,1], es decir:

$$\mu A: X \rightarrow [0,1]$$

quedando representado el conjunto difuso como

$$A = \int \mu A(x)/x \qquad \text{en el caso continuo}$$

ó

$$A = \mu A(x_1)/x_1 + \mu A(x_2)/x_2 + \ldots + \mu A(x_n)/x_n \qquad \text{en caso discreto}$$

Ejemplo: Sea X los números reales y A el conjunto difuso de los números reales cercanos al cero. Existen muchas maneras diferentes de definir este conjunto, depende

el grado de la magnitud que se está midiendo para decir si un elemento esta cerca o lejos. Una posible forma sería mediante la siguiente función:

$$\mu A(x) = \left(\frac{1}{1+10x^2} \right)^2$$

Ahora podemos extender todas las definiciones de la teoría clásica de conjuntos.

Def. 3.2: Un conjunto difuso *A* es ***vacío*** si y sólo si su función de pertenencia asigna 0 a todos los elementos de *X*.

$$A = \phi \quad sii \quad \forall x \in X, \mu A(x) = 0$$

Def. 3.3: Dos conjuntos difusos son iguales, y lo notaremos *A(x)=B(x)* si y sólo si $\mu A(x) = \mu B(x)$ para todo *x* en *X*. (En adelante sólo escribiremos $\mu A = \mu B$).

Def. 3.4: Inclusión.

A está incluido en *B* ó *A* está contenido en *B* si y sólo si la función de pertenencia de *A* es menor o igual que la de *B* para todo elemento.

$$A(x) \subseteq B(x) \; sii \; \forall x \in X, \mu A(x) \leq \mu B(x)$$

Comentarios:

- Se puede ver que esta noción de inclusión esta íntimamente relacionada con la implicación ($\Rightarrow$).
- Después fueron estudiadas las medidas de inclusión, ya que esta definición era categórica, es decir un conjunto difuso está incluido o no lo está.

Def. 3.5: Intersección.

La intersección de dos conjuntos difusos *A* y *B* con sus respectivas funciones de pertenencia $\mu A(x)$ y $\mu B(x)$ es un nuevo conjunto difuso *C* escrito como $C(x) = A(x) \cap B(x)$ de manera tal que:

$\mu C = \mu A \cap \mu B$ ó $\mu C(x) = \mathbf{min}[\mu A(x), \mu B(x)]$

Def. 3.6: Unión.

La unión de dos conjuntos difusos *A* y *B* con sus respectivas funciones de pertenencia $\mu A(x)$ y $\mu B(x)$ es un nuevo conjunto difuso *C* escrito como $C(x) = A(x) \cup B(x)$ de manera tal que:

$\mu C = \mu A \cup \mu B$ ó $\mu C(x) = \mathbf{max}[\mu A(x), \mu B(x)]$

Def. 3.7: El complemento de un conjunto difuso *A*, denotado por A' se define como:

$\mu A' = 1 - \mu A$

Notar que esta definición de complemento se comporta en los casos extremos igual que en la lógica clásica, es decir que si para algún *x*, $\mu A(x)=1$ entonces $\mu A'(x)=0$ y viceversa. Más adelante se verán otras maneras de definir el complemento, pero siempre se exigen estas condiciones denominadas condiciones de frontera o límite; como la lógica difusa es una extensión de la lógica clásica, en los casos límites se conservan todas las propiedades de la lógica clásica. Lo mismo sucede con la unión y la intersección, existen también otras maneras de definirlas, pero el fundador de la lógica difusa Lotfi Zadeh [Za_1965] las definió de esa manera y se ha probado en trabajos posteriores que es la forma en la cual se respetan la mayor cantidad de propiedades provenientes de la lógica clásica.

Propiedades de la unión, intersección y complemento

Leyes de De Morgan

$(A \cup B)' = A' \cap B'$

$(A \cap B)' = A' \cup B'$

Leyes distributivas

$C \cap (A \cup B) = (C \cap A) \cup (C \cap B)$

$C \cup (A \cap B) = (C \cup A) \cap (C \cup B)$

Estas igualdades también se podrían expresar con respecto a las funciones de pertenencia. Por ejemplo:

$$1\text{-}\mathbf{max}[\mu A,\mu B] = \mathbf{min}[1\text{-}\mu A, 1\text{-}\mu B]$$

Def. 3.8: Soporte de un conjunto difuso:

El soporte de un conjunto difuso A en el conjunto universal X, es un conjunto *crisp* que contiene todos los elementos de X donde el grado de pertenencia a A es mayor que cero; es decir:

$$\mathbf{soporte}(A) = \{ x \in X \mid \mu A(x) > 0 \}$$

Se puede definir que un conjunto difuso es vacío cuando el soporte es vacío.

Def. 3.9: α-Corte:

El α-corte $A\alpha$ de un conjunto difuso A es un conjunto clásico definido de la siguiente manera:

$$A\alpha = \{ x \in X \mid \mu A(x) \geq \alpha \} \qquad \text{(débil)}$$

o veces se lo define como:

$$A\alpha = \{ x \in X \mid \mu A(x) > \alpha \} \qquad \text{(fuerte)}$$

Propiedades:

$$(A \cup B)\,\alpha = A\alpha \cup B\alpha$$

$$(A \cap B)\alpha = A\alpha \cap B\alpha$$

$$(A')\alpha \neq (A\alpha)'$$

Def. 3.10: Cardinalidad de un conjunto difuso:

La cardinalidad escalar de un conjunto difuso A es definida por:

$$|A| = \sum_{x \in X} \mu A(x) \qquad \text{cuando } A \text{ tiene un soporte finito}$$

Def. 3.11: Entropía:

Muchos autores hablan de medidas de difusidad, la entropía sirve para evaluar cuan difuso es un conjunto difuso. Por ejemplo *e(A)=0* sii *A* es un conjunto no difuso, totalmente *crisp*.

Si *X* es finito la entropía se calcula de la siguiente manera:

$$e(A) = -\sum_{x \in X} \left(\mu A(x) \cdot \log(\mu A(x)) \right) + \left(1 - \mu A(x) \cdot \log(1 - \mu A(x)) \right)$$

Comentario: Como la función logaritmo sólo está definida para los reales mayores que cero, notar que la sumatoria se hace sólo sobre los *x* que pertenecen a *X*, por lo tanto los *x* que la función $\mu A(x)$ asigna 0 no son tomados en cuenta, ya que no pertenecen.

4. Relaciones Difusas [D&P_1979]

Una relación difusa *R*, es un conjunto difuso en el producto cartesiano $X \times Y$ en el universo de *X* y de *Y*. $\mu R(x,y)$ es la función de pertenencia que muestra el grado de fuerza que tiene la relación *R* entre los pares *x* e *y*.

Las relaciones difusas generalizan el concepto de relaciones ordinarias.

Ejemplo clásico:

Sean *X* e *Y* dos conjuntos, con los siguientes elementos:

$X=\{a,b,c\}$ $Y=\{1,2,3\}$

y la siguiente relación:

$R(x,y) = \{(a,1),(a,2),(b,2),(b,3)\}$

Ejemplo difuso:

Sean *X* e *Y* dos conjuntos, con los mismos elementos del ejemplo anterior pero la relación *R* ahora es difusa, es decir, hay una función de grados de pertenencia para la relación, definida de la siguiente manera:

$\mu R(a,1) = 0.7$ $\mu R(a,2) = 0.3$

$\mu R(b,2) = 1$ $\mu R(b,3) = 0.5$

en general:

$\mu R(x,y) \rightarrow [0,1]$

También se podría especificar como una matriz de pesos, de la siguiente manera:

	1	**2**	**3**
a	0.7	0.3	0
b	0	1	0.5
c	0	0	0

o también como un grafo que rotulamos los arcos con los pesos.

Definiciones:

Def 4.1: *Composición*:

Sean R y S dos relaciones difusas. $R: X \times Y$ y $S: Y \times Z$ entonces la relación $R \circ S: X \times Z$ está definida por:

$$\mu R \circ S(x,z) = \underset{y \in Y}{Sup}\{ min[\mu R(x,y), \mu S(y,z)]\}$$

Queda establecido de esta manera un camino de x a z a través de cualquier y. La conexión global entre x y z es el mejor camino x-y-z.

Def 4.2: *Proyección*:

La proyección de una relación difusa R sobre X, es un conjunto difuso $Px(R)$ donde la función de pertenencia queda definido por:

$$\mu \Pi R(x) = \underset{y \in Y}{Sup} \{\mu R(x,y)\}$$

Def 4.3: Una relación difusa R es *separable* sii $\forall x, \forall y\ \mu R(x,y) = \min[\mu \Pi R(x), \mu \Pi R(y)]$

Def 4.4: Una relación difusa R sobre $X \times Y$ es:

reflexiva	sii	$\forall x \in X, \mu R(x,x) = 1$
simétrica	sii	$\forall x,x' \in X, \mu R(x,x') = \mu R(x',x)$
antisimétrica	sii	$\forall x,x' \in X, x \neq x', \mu R(x,x') > 0, \mu R(x',x) = 0$
transitiva	sii	$\forall x,x',x'' \in X$, *Si* $\mu R(x,x')$ & $\mu R(x',x'') \Rightarrow \mu R(x,x'')$
Δ-transitiva	sii	$\forall x,x',x'' \in X, \mu R(x,x'') \geq \mu R(x,x')\ \Delta\ \mu R(x',x'')$

- Las relaciones que son reflexivas y simétricas son llamadas relaciones de proximidad.
- Cuando R es transitiva, entonces su clausura se define por:

$R^* = R \cup R^2 \cup R^3 \cup R^m \cup \ldots$ donde $R^m = R \cup R^{m-1}$

usando la composición sup-min. Como R es transitiva, entonces R^* existe y es min-transitiva.

- Cuando R es reflexiva, simétrica y min-transitiva las relaciones difusas son llamadas **similaridades**. Podemos obtener como clausura transitiva de proximidades. Mas aún, si R es una relación de similaridad, entonces R^* es una ultramétrica.

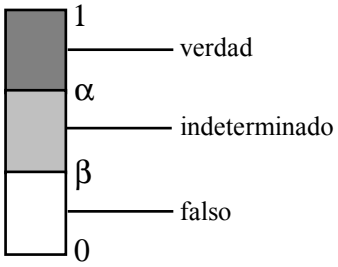
- Las relaciones reflexivas, simétricas y producto-transitivas son más débiles que las similaridades.

5. Lógica Difusa

Historia de las lógicas multivaluadas:

La noción de "pertenencia" que juega un papel fundamental en el caso de conjuntos ordinarios, no tiene el mismo rol en el caso de los conjuntos difusos. Si introducimos dos niveles α y β $(0<\alpha<1,\ 0<\beta<1,\ \beta<\alpha)$ y decimos que "*x* pertenece a *A*" si $\mu A(x) \geq \alpha$ ó "*x* no pertenece a *A*" si $\mu A(x) \leq \beta$ nos queda un rango entre α y β en el cual los *x* serán indeterminados. Esto motivó a la lógica trivaluada de Kleene en el año 1952, utilizando tres valores de verdad.

Verdad	si	$\mu A(x) \geq \alpha$
Falso	si	$\mu A(x) \leq \beta$
Indeterminado	si	$\beta < \mu A(x) < \alpha$



Lógica Difusa

La Lógica Difusa es un superconjunto de la lógica convencional o clásica, que ha sido extendida para manejar el concepto de verdad parcial. Estos valores de verdad recorren el intervalo real [0,1] donde el 0 representa "falso" y 1 "verdadero"

La suposición de que toda proposición es verdadera o falsa, es decir, esté basada en la lógica clásica, ha sido discutida desde los tiempos de Aristóteles. Por ejemplo, el valor de verdad de proposiciones que involucran al futuro, no son ni verdaderas ni falsas, pueden ser potencialmente verdaderas o potencialmente falsas. Por ejemplo: "Mañana va a llover" o "El próximo mes no va a salir el sol". A ese tipo de proposiciones se les asignó un tercer valor de verdad llamado indeterminado.

Surgieron muchas lógicas trivaluadas, la operación común entre ellas es el complemento que se definió de la siguiente manera:

La lógica clásica puede ser extendida sobre una lógica con tres valores de verdad (trivaluada) de varias maneras. Es muy común definir la negación *a'* como *1-a*. El siguiente cuadro muestra las primitivas en varias lógicas trivaluadas.

		Lukasievwicz				Bochvar				Kleene				Heyting				Reichenbach			
a	***b***	∧	∨	⇒	⇔	∧	∨	⇒	⇔	∧	∨	⇒	⇔	∧	∨	⇒	⇔	∧	∨	⇒	⇔
0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	½	0	½	1	½	½	½	½	½	0	½	1	½	0	½	1	0	0	½	1	½
0	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
½	0	0	½	½	½	½	½	½	½	0	½	½	½	0	½	0	0	0	½	½	½
½	½	½	½	1	1	½	½	½	½	½	½	½	½	½	½	1	1	½	½	1	1
½	1	½	1	1	½	½	½	½	½	½	1	1	½	½	1	1	½	½	1	1	½
1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
1	½	½	1	½	½	½	½	½	½	½	1	½	½	½	1	½	½	½	1	½	½
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Existen otras lógicas trivaluadas, como la de Bochvar, Kleene, Heyting o Reichenbach. Las primitivas como conjunción, disyunción, implicación y equivalencia difieren de una lógica a otra, en el caso que una proposición tenga el valor de verdad 0.5, en los casos extremos todas se comportan igual a la lógica clásica. También se extendieron las lógicas trivaluadas a n-valuadas, la primer lógica n-valuada fue propuesta por Lukasiewicz. Las operaciones primitivas en la lógica n-valuada de Lukasiewicz fueron definidas así:

- $a' = 1 - a$
- $a \wedge b = \mathbf{min}[a,b]$
- $a \vee b = \mathbf{max}[a,b]$
- $a \Rightarrow b = \mathbf{min}[1,1+b-a]$
- $a \Leftrightarrow b = 1 - | a - b |$

En realidad Lukasiewicz usó sólo negación e implicación para definir las otras operaciones.

- $a \vee b = (a \Rightarrow b) \Rightarrow b$
- $a \wedge b = (a' \vee b')'$
- $a \Leftrightarrow b = (a \Rightarrow b) \wedge (b \Rightarrow a)$

Se puede observar que en este tipo de lógicas no vale la ley de la contradicción $(a \wedge a' = 0)$ ni la ley del tercero excluido $(a \vee a' = 1)$. Para ver que no se cumplen estas leyes sólo hace falta asignarle un valor de verdad distinto de 0 y 1.

Si tomamos $a=½$ entonces a' también será $½$ y podemos observar cómo estas propiedades no se cumplen

$$½ \wedge ½ = 0 \quad \mathbf{min}[½ , ½] = ½ \neq 0$$

$$½ \vee ½ = \mathbf{max}[½ , ½] = ½ \neq 1$$

Operadores y operaciones con conjuntos difusos

Previamente comentamos que la unión, intersección y complemento pueden ser definidas de diferente manera. Por ahora vamos a mantener la definición de complemento fija, como $1-\mu A$ y ver distintas formas de definir la unión e intersección.

Sean A y B dos conjuntos difusos y μA y μB sus respectivas funciones de pertenencia.

	Intersección	**Unión**
Zadeh	$\mathbf{min}[\mu A,\mu B]$	$\mathbf{max}[\mu A,\mu B]$
Producto	$\mu A.\mu B$	$\mu A+\mu B - (\mu A.\mu B)$
Dubois y Prade	$\mathbf{max}[0,\mu A+\mu B-1]$	$\mathbf{min}[1,\mu A+\mu B]$

La intersección de Zadeh la notaremos como $\mu A \cap B$ y la unión como $\mu A \cup B$. Los operadores de Dubois y Prade $\mu A \cap\!\!\!- B$ y $\mu A \cup\!\!\!- B$ respectivamente. Y los provenientes de las probabilidades (producto) como $\mu A.B$ para la intersección y $\mu A+B$ para la unión.

$\mu A+B$ también es conocido como *suma algebraica,* así como también a $\mu A.B$ se lo conoce como *producto algebraico.* Zadeh llamó al operador de unión de Dubois y Prade *suma limitada*. Y para completar con esta sección de nombres de operaciones faltaría nombrar la *diferencia limitada* que se define como $\mathbf{max}[0,\mu A(x)-\mu B(x)]$ y a la *diferencia simétrica* definido como $|\mu A(x)-\mu B(x)|$.

Se puede ver las siguientes relaciones de inclusión:

$$A \dot{\cap} B \subseteq A.B \subseteq A \cap B$$

y también se puede observar que:

$$A \cup B \subseteq A + B \subseteq A \dot{\cup} B$$

Notar que el enfoque de Zadeh propone la unión más pequeña y la intersección más amplia. Lo contrario ocurre con los operadores de Dubois y Prade.

Estos operadores para computar la intersección son llamados también **Normas Triangulares** porque respetan los siguientes propiedades:

$$(A \cap B)(x) = A(x) \otimes B(x)$$

- $\otimes: [0,1] \times [0,1] \rightarrow [0,1]$
- Conmutabilidad: $\alpha \otimes \beta = \beta \otimes \alpha$
- Asociatividad: $(\alpha \otimes \beta) \otimes \gamma = \alpha \otimes (\beta \otimes \gamma)$
- Monotonía: si $\alpha \geq \alpha', \beta \geq \beta'$, entonces $\alpha \otimes \beta \geq \alpha' \otimes \beta$, $\alpha \otimes \beta \geq \alpha \otimes \beta'$
- $\alpha \otimes 1 = \alpha, \quad \alpha \otimes 0 = 0$

y los operadores definidos para la unión son llamados **Conormas Triangulares** y respetan estas propiedades:

$$(A \cup B)(x) = A(x) \oplus B(x)$$

- $\oplus : [0,1] \times [0,1] \rightarrow [0,1]$
- Conmutabilidad: $\alpha \oplus \beta = \beta \oplus \alpha$
- Asociatividad: $(\alpha \oplus \beta) \oplus \gamma = \alpha \oplus (\beta \oplus \gamma)$
- Monotonía: si $\alpha \geq \alpha', \beta \geq \beta'$, entonces $\alpha \oplus \beta \geq \alpha' \oplus \beta$, $\alpha \oplus \beta \geq \alpha \oplus \beta'$

- $\alpha \oplus 1 = 1, \qquad \alpha \oplus 0 = \alpha$

Para la función complemento también hay varias alternativas, pero como la lógica difusa es una extensión de la lógica clásica, en los casos extremos (0 y 1) esta función debe comportarse igual, es decir que satisfaga las condiciones de frontera (*boundary conditions*). Además se pide que sea una función monótona decreciente.

$$c(0) = 1, \qquad c(1) = 0$$

el complemento estándar se define como:

$$c(\alpha) = 1 - \alpha$$

Sugeno y Yager definen el complemento como una familia de funciones de la siguiente manera:

Sugeno: $c_\lambda(\alpha) = \dfrac{1-\alpha}{1+\lambda\alpha}, \quad \lambda > -1$

Yager: $c_w(\alpha) = \sqrt[w]{1-\alpha^w}, \quad w > 0$

Comentarios:

La familia de funciones la generamos al asignarle distintos valores a λ y a w.

Estas funciones se comportan igual al complemento standard cuando $\lambda = 0$ y $w = 1$.

Implicación:

Manteniendo la negación de A como 1-μA el operador de implicación **(A→B)** debe ser definido de la siguiente manera:

	A→B
Zadeh	**max[1-μA,μB]**
Producto	**(1-μA)+(μA.μB)**
Dubois y Prade	**min[1,1+μB-μA]**

Vemos que la implicación que surge de los operadores propuestos por Dubois y Prade corresponde con la implicación de Lukasiewicz.

6. Aplicaciones

Aplicaciones de la teoría de conjuntos difusos:

Los sistemas difusos han sido exitosamente utilizados en muchas aplicaciones comerciales. En los últimos años Japón ha tenido el liderazgo en el tema, llevado a un extremo tal que la mayoría de los electrodomésticos tienen el sello de la tecnología "neuro-fuzzy". La aplicación mas importante de Japón es la del control del subterráneo de Sendai, pero podemos recordar también otras como: foco automático en cámaras fotográficas y vídeo, control de aire acondicionado, ascensores inteligentes, mezcladoras de cemento, lavarropas, reconocedores de caracteres, etc.

No sólo en el terreno práctico es donde se ha avanzado con los sistemas difusos, hay además un gran espectro de estudios teóricos sobre esta disciplina. Podríamos trazar la siguiente clasificación de las aplicaciones a la teoría de conjuntos difusos:

I) Sistemas formales

- i) Sistemas difusos
- ii) Gramáticas y lenguajes difusos
- iii) Algoritmos difusos

II) Lenguaje natural y modelos de razonamiento aproximado

III) Aplicaciones a sistemas e Inteligencia Artificial

- i) Investigación de operaciones
- ii) Análisis de decisión
- iii) Clasificación y clustering
- iv) Control
- v) Diagnóstico
- vi) Aprendizaje

Control	Subterráneos, ascensores, control de tráfico, automóviles, lavarropas, heladeras, cámaras de vídeo, hardware (VLSI fuzzy), etc.
Reconocimiento de patrones	Imágenes, audio, procesamiento de señales.
Inferencia	Sistemas expertos para diagnóstico, planeamiento y predicción; procesamiento de lenguaje natural; interfaces inteligentes; robots inteligentes; ingeniería de software.
Acceso a la información	Bases de datos, interfaces de lenguaje natural, consultas difusas.

Es este trabajo sólo vamos a tratar algunos de estos puntos, como ser el manejo de incertidumbre en modelos de razonamiento aproximado, sistemas expertos, aplicaciones en control y medicina.

6.1 Manejo de incertidumbre en IA

Modelos de Razonamiento Aproximado

Informalmente, razonamiento aproximado ó difuso es el proceso o procesos mediante el cual una conclusión imprecisa se deduce a partir de una colección de premisas también imprecisas.

Consideremos las reglas del siguiente tipo:

$$Si\ A_1\ \&\ ..\ \&\ A_n \rightarrow C\ con\ FC.$$

Este tipo de regla utiliza el Factor de Certeza *FC* para concluir *C*, es decir debilita la implicación y le da un carácter de inseguridad a la regla, este *FC* debe estar expresado en el rango *[0..1]*. Si el *FC = 1* entonces la implicación equivale a la clásica.

Si hay varias reglas que concluyen *C* con distinto *FC*, como cualquier camino puede ser utilizado para la demostración se debería tomar el máximo entre todos los factores de certeza.

¿Cómo calcular cuando hay un encadenamiento de reglas en la deducción?

Podemos plantear el siguiente problema, sean las reglas:

$$\begin{array}{ll} A \rightarrow B & fc1 \\ \underline{B \rightarrow C \quad fc2} & \\ A \rightarrow C & fc3 \end{array}$$

Claramente se deduce $A \rightarrow C$ pero ¿con que factor de certeza?. Vamos a reescribir el ejemplo de dos maneras diferentes, primero como proposiciones con cuantificadores difusos, y después para generalizar lo trataremos como relaciones difusas:

Si tomamos el ejemplo anterior como proposiciones con cuantificadores difusos lo deberíamos reescribir de la siguiente manera:

$$\begin{array}{l} P_1 \equiv Q_1 (A \rightarrow B) \\ \underline{P_2 \equiv Q_2 (B \rightarrow C)} \\ P_3 \equiv Qx (A \rightarrow C) \end{array}$$

Si suponemos que $Q_1 = Q_2$ es el cuantifiador *todos* entonces claramente Qx también será *todos*. Pero si los factores de certeza ya no están en los casos extremos, deberíamos combinarlos de alguna manera, Zadeh en su trabajo [Za_1983], propuso usar el producto de $Q_1 \otimes Q_2$ para calcular el valor de Qx.

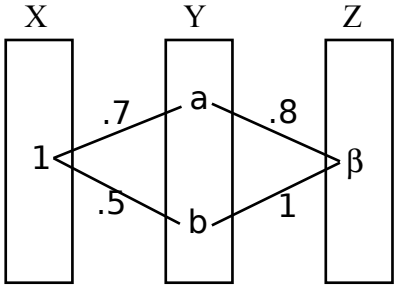
Notar que Zadeh en este caso no propuso el mínimo para calcular el valor de Qx, como vimos antes en la relación entre los operadores de Zadeh, Probabilistico y Dubois y Prade, el operador producto es un poco mas *pesimista* que el mínimo.

Ejemplo:

Sean $P(x,y)$ y $Q(y,z)$ relaciones difusas con los siguientes elementos:

$$P = \{(1,\mathrm{a})/0.7\,,\,(1,\mathrm{b})/0.5\}$$
$$Q = \{(\mathrm{a},\beta)/0.8,\,(\mathrm{b},\beta)/1\}$$

ó más claramente en este diagrama:



Si queremos saber, por ejemplo *R(1,β)* deberíamos calcular *R* como la composición entre *P* y *Q* de la siguiente manera:

Composición **max-min**

$$R(x,z) = P(x,y) \circ Q(y,z)$$
$$\mu R(x,z) = \max_{y \in Y} \{ min[\mu P(x,y), \mu Q(y,z)] \}$$
$$\therefore$$
$$\mu R(1,\beta) = max[min(.7,.8), min(.5,1)] = max[.7,.5] = 0.7$$

ó también podemos computar la relación *R(1,β)* como:

Composición **max-prod**

$$R(x,z) = P(x,y) \otimes Q(y,z)$$
$$\mu R(x,z) = \max_{y \in Y} \{ \text{prod}\, [\mu P(x,y), \mu Q(y,z)] \} = \max_{y \in Y} \{ \mu P(x,y) \otimes \mu Q(y,z) \}$$
$$\therefore$$
$$\mu R(1,\beta) = max[0.7 \times 0.8, 0.5 \times 1] = max[0.56, 0.5] = 0.56$$

La composición **max-min** puede ser vista como el grado de la fuerza que tiene el encadenamiento; si un eslabón es muy débil entonces la fuerza de la cadena será muy débil. Si tuviéramos que extender a *n* composiciones, no habría problema, de cada rama se obtiene el mínimo de los encadenamientos y de todos los mínimos se elige el máximo. También se puede calcular utilizando la composición **max-producto** (⊗), lo cual se logra cambiando en cada rama el mínimo por el producto, de esta manera, todas la conexiones ayudan a formar el valor del camino, en cambio cuando usamos el mínimo, solamente la conexión más pequeña es la que da el valor al encadenamiento.

La forma para calcular la composición de relaciones puede variar dependiendo del tipo de problema que se quiere modelar.

6.2 Control Difuso

Los controladores difusos han sido introducidos por Mamdani [Ma_1974] y por Mamdani y Assilian [M&A_1975] para el control de procesos complejos, como el de plantas industriales, especialmente cuando no hay un modelo preciso del proceso que se quiere controlar.

Tipos de reglas en modelos aproximados:

En los **modelos clásicos** las reglas tienen la siguiente forma:

Si Presión = 10 ATM, entonces Volumen = 2.5 m³

donde se plantea una relación entre una variable de entrada o un conjunto de variables de entrada y la regla determina una conclusión asignando algún valor sobre las variables de salida (generalmente una). No siempre se tienen modelos precisos, por lo tanto en ese caso se pueden modelar la reglas usando **modelos imprecisos** como muestra la siguiente regla:

Si Presión ≥ 5 ATM, entonces Volumen ≤ 6 m³

También se pueden plantear relaciones usando factores de certeza o probabilidades, a estos modelos se los denominan **modelos inciertos** (uncertain models). Ej.:

Si Presión ≥ 5 ATM, entonces Prob(Volumen = 6 m³) = 0.9

ó

Si Presión ≥ 5 ATM, entonces Volumen = 6 m³ con FC=0.9

Pero los modelos que nos interesan en este trabajo son los **modelos vagos**. Donde las reglas describen relaciones entre conjuntos difusos.

Si Presión = Alta, entonces Volumen = Bajo

Estas reglas pueden ser vistas como parches aproximados para tratar de mapear una relación de compatibilidad entre estados o variables.

Modus Ponems Generalizado

El ***modus ponems*** usado en la lógica clásica es un mecanismo de inferencia que permite por ejemplo deducir lo siguiente: "todos los hombres son mortales" y sabemos que "Sócrates es un hombre" por lo tanto podemos concluir que "Sócrates es mortal" ó formalmente:

$$\frac{P \rightarrow Q \wedge P}{Q}$$

Cuando trabajamos con modelos vagos, la inferencia sobre este tipo de reglas esta regida por el ***modus ponems generalizado*** que se puede enunciar de la siguiente manera:

$$\frac{P \rightarrow Q \wedge P'}{Q'}$$

donde si P' es similar a P entonces podremos deducir un Q' que será también similar a Q. Este mecanismo de inferencia se comporta igual al clásico si P y P' son iguales, entonces Q' será igual a Q.

Entre las aplicaciones en donde podemos encontrar este tipo de regla de inferencia se encuentran los sistemas expertos y los procesos de control en ingeniería, entre otros.

Estructura de un controlador difuso:

El propósito de los controladores es el de computar valores para variables de acción (system output) a partir de la observación de las variables de entrada (system input).

La relación entre las variables de entrada y las variables de acción pueden ser vistas como un conjunto de reglas lógicas. Si esta relación es conocida solamente en forma cualitativa, una estrategia razonable es la de representar estas reglas como reglas difusas. Un ejemplo de una regla difusa podría ser: "Si la temperatura es muy alta entonces el cambio de presión debe ser pequeño". Como se puede ver en el ejemplo, "muy alta" y "pequeño" son conjuntos difusos, pero definidos sobre universos diferentes. Esta regla se puede ver como una relación difusa entre temperatura x presión.

Como el o los valores de entrada son valores exactos, sujetos a los errores de medición, son leídos a través de dispositivos externos, y las reglas están expresadas en forma cualitativa, a los datos se les hace un proceso de ***fuzzyficación*** a conjuntos difusos y luego para producir una acción se debe ***desfuzzyficar*** los resultados obtenidos.

En el siguiente ejemplo, está representado un prototipo que resuelve el problema del péndulo invertido, sus reglas y el motor de inferencia general, que se puede utilizar para otro problema de control. Los datos de entrada son el ángulo y la velocidad angular. Como respuesta el sistema devuelve la fuerza a ser aplicada. En este prototipo los datos se leen desde el teclado. Se podría extender esta versión a otro con una simulación completa.

Las reglas tienen la siguiente forma:

regla *i* ::= si ((X = A) & (Y = B)) → (Z = C)

ejemplo:

regla 8 ::= si (velocidad = baja) & (aceleración = alta) → (frenado = mediano)

El motor de inferencias obtendrá que el "frenado debe ser mediano" en función de los antecedentes, por lo tanto computara la funciones de pertenencia para los valores de velocidad y aceleración sobre los conjuntos difusos "baja" y "alta".

$$\mu C(Z) = \mu Ri(X,Y) = \min[\mu A(X), \mu B(Y)]$$

Cuando *n* reglas están disponibles para ser usadas, el resultado de la relación difusa *R* es la unión de las *n* relaciones elementales. (*i* : *1..n*)

$$\mu C(Z) = \mu R(X,Y) = \coprod_{i=1}^{n} \mu Ri(X,Y) = max[\mu Ri(X,Y)] \quad 1 \le i \le n$$

Este tipo de razonamiento es similar a la composición de relaciones difusas. Sin embargo, puede ser una decisión dependiendo del contexto, cambiar la función **min** por el producto, o cualquier otra. Asimismo con la función **max**.

En el **Apéndice A** se encuentra el código completo para implementar sistemas expertos para control utilizando reglas difusas.

6.3 Aplicaciones en Medicina

6.3.1 Diagnóstico médico utilizando lógica difusa:

E. Sánchez escribió un trabajo llamado "Medical diagnosis and composite fuzzy relations" [Sa_1979] en donde formalizaba el proceso de diagnóstico médico y lo interpretaba como una composición de relaciones difusas.

Para una patología dada, vamos a denotar ***S*** el conjunto de síntomas, ***D*** al conjunto de diagnósticos y ***P*** al conjunto de pacientes. Vamos a llamar "conocimiento médico" a la relación difusa, generalmente denotada por ***R***: ***S→D*** que expresa asociaciones entre síntomas y diagnósticos o un grupo de diagnósticos.

Sea ***A*** un subconjunto difuso de ***S***, es decir los síntomas del paciente actual. ***R*** una relación difusa de ***S*** en ***D***, entonces el cómputo de la composición ***sup-min*** sobre ***B*** es igual a ***RoA*** y define el estado del paciente en términos del diagnóstico como un subconjunto difuso ***B*** de ***D***, caracterizado por la siguiente función de pertenencia:

$$\mu B(d) = \sup_{s \in S} \{ min[\mu A(s), \mu R(s,d)] \} \quad d \in D$$

6.3.2 Especificaciones difusas y diagnóstico automático:

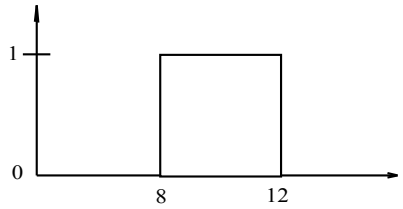
El problema que se tiene al querer hacer diagnóstico automático a partir de las especificaciones de los especialistas es que éstas están plagadas de ambigüedades, experiencia e inconsistencias. En los siguientes párrafos se van a mostrar unas especificaciones extraídas de [R&K_1968] el cual trata sobre el análisis de las diferentes fases encontradas durante el sueño. Lo importante en esta sección es poder observar el grado de ambigüedad en las definiciones y por ello no es necesario explicar conceptos médicos como "complejos K", "Spindles", etc.

Generalmente en un sueño normal se pueden determinar siete fases diferentes. Una de las tareas de los especialistas en estudios del sueño consiste en analizar el trazado encefalográfico (EEG) y clasificarlo según las fases en que se encuentra. Las fases están definidas de la siguiente manera:

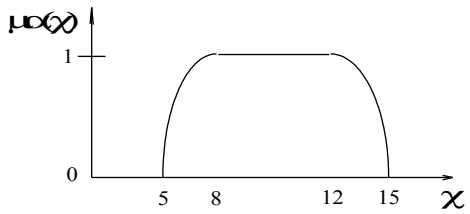
Fase V (Vigilia)	El electroencefalograma (EEG) contiene actividad alfa y/o actividad rápida de bajo voltaje.
Fase 1	Un voltaje relativamente bajo, con frecuencias entremezcladas, sin movimientos rápidos oculares (REM).
Fase 2	"Spindles" de sueño a 12 c/sg y complejos K, sobre un trazado de fondo de relativamente bajo voltaje y frecuencia mixta.
Fase 3	Moderada cantidad de ondas lentas de gran amplitud.
Fase 4	Gran cantidad de ondas lentas de gran amplitud
Fase NREM	Fases 1,2,3 y 4 combinadas.
Fase REM	Bajo voltaje, ritmos semejantes a la fase 1, episodios de movimientos oculares rápidos y registro electromiográfico (EMG) de baja amplitud.

Vemos que en las definiciones hay frases como: bajo voltaje, relativamente bajo, moderada cantidad, gran cantidad, semejante a otra fase, etc. que nos estarían dando un gran indicio que este problema sería factible resolverlo mediante el uso de lógica difusa. Inclusive las definiciones categóricas se podrían definir como conjuntos difusos de la siguiente manera:

Las ondas alfa se las definen generalmente como un EEG con ondas en el rango de los 8 a los 12 Hertz (Hz). Por lo tanto la manera clásica de representar las ondas alfa sería:



de esta manera si el trazado tiene ondas con 7.8 Hz no tendría ondas alfa. Es evidente que a este concepto habría que definirlo de otra manera y no en forma tan categórica. Por ejemplo como muestra la siguiente función de pertenencia:



De esta manera la función de pertenencia $\mu\alpha(x)$ asignará un cierto grado de pertenencia a las frecuencias que se encuentren en el rango 5 a 8 Hz y 12 a 15 Hz que antes no estaban tenidas en cuenta.

Definiendo así los posibles conjuntos difusos y también asignando valores correctos a las relaciones difusas entre estos conjuntos, se podría resolver un problema con técnicas de reconocimiento automático usando computadoras como un sistema de ayuda para el análisis de EEG.

6.4 Futuro de los Sistemas Difusos - Sistemas Híbridos.

Últimamente se han destacado varias formas de sistemas híbridos que combinan lógica difusa con redes neuronales y algoritmos genéticos.

6.4.1 Neuro-Fuzzy Systems

Hay muchos trabajos de investigación y desarrollo industrial que vinculan redes neuronales y la lógica difusa. Usando estas dos técnicas en forma complementaria se logran mejores resultados dado que cada una aporta su mejor característica creando así una nueva forma de resolver problemas.

Se utiliza a las redes neuronales en los sistemas híbridos, para determinar las funciones de pertenencia, o para validar la base de conocimiento. Las redes neuronales funcionan muy bien en los casos en donde no se conocen las reglas del problema. Cuando estas reglas se conocen y son vagas, no hay que dudar en utilizar lógica difusa.

Es muy difícil conocer cuantos conjuntos difusos necesitará el sistema difuso que se desarrolla, o cual va a ser la forma de sus funciones de pertenencia, o como se interpretarán los conectivos lógicos. Las primeras dos preguntas las responden las redes neuronales.

Además los sistemas difusos no tienen capacidad de aprendizaje, problema que se solucionó con los sistemas híbridos que mencionamos.

6.4.2 Algoritmos Genéticos Difusos

Las aplicaciones sobre algoritmos genéticos combinados con control difuso han sido investigados no sólo en el terreno académico, sino también a nivel comercial. Los algoritmos genéticos son buenos para ajustar las funciones de pertenencia del sistema, encontrando soluciones adecuadas acorde a la función de evolución.

6.4.3 Sistemas de control Fuzzy-PID

Los sistemas PID para control son usados para problemas convencionales, generalmente problemas lineales. La sigla PID representa Proportional-Integral-Differential.

Para ciertas aplicaciones, los sistemas fuzzy y PID son empleados juntos como un controlador híbrido. El controlador PID puede ser usado para aproximar y generar una respuesta rápida, mientras tanto el controlador difuso ajusta la respuesta para ganar performance.

7. Definición del Lenguaje

En esta sección se presentara la propuesta de un nuevo lenguaje difuso, haciendo también un análisis de la semántica de dicho lenguaje.

Sea L_A el siguiente lenguaje difuso definido por: $L_A \equiv < O , C , P >$

donde ***O*** son los objetos, ***C*** los conjuntos difusos, ***P*** los predicados y ***U*** el umbral.

7.1 Objetos

$O \equiv \{ o1, o2, \dots , on \}$

Ej.: ***O*** ≡ {sensor, caldera, válvula}

Estos objetos tienen *características* o *propiedades*, como la temperatura del sensor, la presión de la caldera o la apertura de la válvula. Y los posibles valores que pueden tomar serán calificaciones difusas, como alta, normal, grande, etc.

7.1.1 Propiedades

$P(o) = \{ p1, p2, \dots , pn \}$

Ej.:. o_1 = ***'sensor'*** , $p_1(o_1)$ = ***'temperatura'***

7.1.2 Valores

$V(o,p) = \{ v_1, v_2, \dots , v_n \}$

Ej.:. o_1 = ***'sensor'*** , $p_1(o_1)$ = ***'temperatura'*** , $v_1(o_1,p_1)$ = ***'alta'***

7.2 Conjuntos difusos

C es una familia de funciones que van desde el universo del discurso en el [0,1]. Estas funciones representan las características o las propiedades difusas de los objetos, ***D*** representa el dominio de los objetos.

Dominio: $D \equiv Dominio(O)$

Conjuntos difusos: $C \equiv \{ f_1 , f_2 , \dots , f_n \}$ donde $f_i(D) \rightarrow [0,1]$ $\quad 1 \leq i \leq n$

7.3 Predicados

Los predicados serán aserciones sobre conjuntos difusos o relaciones entre los objetos.

Predicados: $P = P_h \cup P_r$

Hechos: $P_h = \{ p_o(v) / n \ , \ p_o \in P(o) \ \wedge \ v \in V(o,p) \ \wedge \ n \in [0,1] \}$

Negación de hechos: *Si* $x \in P_h$ *entonces* $\neg x \in P_h$

Antecedente o Consecuente: $S = \{ p_o(v) / p_o \in Po \wedge v \in V \}$

Negación: *Si* $x \in S$ *entonces* $\neg x \in S$

Conjunción de antecedentes: *Si* $a \in S$ *y* $b \in S$ *entonces* $a \wedge b \in S$

Reglas: $P_r = \{ a \Rightarrow b / a \in S$ y $b \in S \}$

Ej. de P_h = { temperatura(alta) / 0.8 , presión(normal) / 0.3 }

Ej. de P_r = { temperatura(alta) $\Rightarrow$ presión(baja) , p(a) $\wedge$ q(b) $\wedge$ r(c) $\Rightarrow$ s(d) }

Las aserciones las llamaremos hechos difusos y son equivalentes a los cortes-α vistos en la Def 3.9 y las relaciones serán reglas, estas reglas corresponden a las reglas para **modelos vagos** desarrollados en la sección 6.2.

7.4 Semántica

En esta sección intentaremos extrapolar conceptos de la teoría de modelos clásica o *crisp* aplicada al lenguaje L_A y llevarla al terreno difuso.

Supongamos que tenemos un programa lógico P definido por las siguientes sentencias:

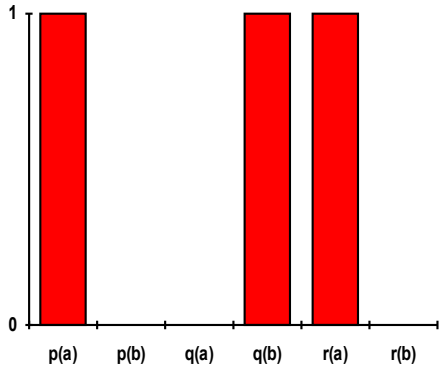
$P = \{ p(a), q(b), p(X) \wedge q(Y) \rightarrow r(X) \}$

El universo de Herbrand es: $U_h = \{ a, b \}$

la Base de Herbrand es: $B_h = \{ p(a), p(b), q(a), q(b), r(a), r(b) \}$

y las consecuencias lógicas $Cn(P)$ o significado de este programa lógico es:

Cn(P) = { p(a), q(b), r(a) }



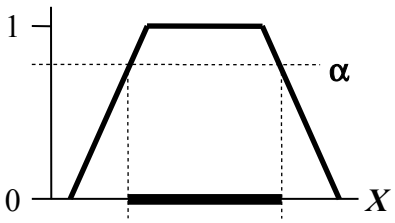
Cn(P) en forma gráfica.

Ahora nuestro programa lógico ***P*** tendrá extensiones difusas acordes a las definiciones del lenguaje difuso L_A.

Sea ***P = { p(a) / 0.8, q(b) / 0.7, p(X) ∧ q(Y) → r(X) }***

En el programa lógico anterior ***p(a)*** era definitivamente cierto, y ***p(b)*** falso, en este programa ***p(a)*** vale con un cierto nivel de verdad, este nivel es de ***0.8***, por lo tanto ***p(a) / 0.5*** no se podría deducir de ***P***, pero en cambio ***p(a) / 0.9*** si. Para entender esto podríamos revisitar la definición de α-corte.

El α-corte $A\alpha$ de un conjunto difuso ***A*** es un conjunto clásico definido de la siguiente manera: $A\alpha = \{ x \in X \mid \mu A(x) \geq \alpha \}$



Apoyándonos en este gráfico, podemos ver que todos los cortes que superen a α serán mas chicos en ***X***, por lo tanto estarán incluidos en el corte original.

El universo de Herbrand seria el mismo que en el ejemplo *crisp*: U_h ***= { a, b }***

La Base de Herbrand la llamaremos *Base Difusa de Herbrand* y quedara definida de la siguiente manera:

B_h ***= { p(a)/0.8$^+$, p(b)/0$^+$, q(a)/0$^+$, q(b)/0.7$^+$, r(a)/0.7$^+$, r(b)/0$^+$ }***

Notamos $p(a)/0.8^{+}$ queriendo indicar que vale $p(a)/n$ para $0.8 \leq n \leq 1$

Notar que B_h es infinita, aunque no contenga símbolos de función.

Una interpretación en este contexto se lograría asignando valores entre *0* y *1*, por ejemplo sea $I_1 = \{ p(a)/0.5, p(b)/ 0.5, q(a)/1, q(b)/0, r(a)/0.75, r(b)/0.25 \}$

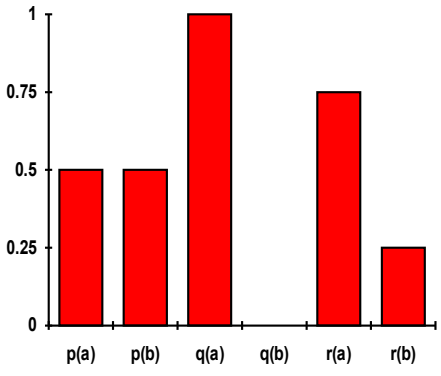


Gráfico de I_1.

Luego para saber si una interpretación es modelo de un programa de L_A deberíamos extender algunas definiciones clásicas, por ejemplo que es una interpretación en L_A .

7.4.1 Interpretación en L_A

Sea P un conjuntos de sentencias de L_A, I una interpretación de P, diremos que I se satisface si y solo si el valor de verdad de cada termino ground de I es mayor o igual al modelo mínimo de Herbrand de P.

7.4.2 Modelo mínimo de Herbrand en L_A

Supongamos ahora que $S = \{p(a),q(b)\}$ y $M_1 = \{p(a),q(b),p(b)\}$, $M_2 = \{p(a),q(b),q(a)\}$, como estamos trabajando con cláusulas de Horn, la intersección de los modelos también será modelo de S, entonces podemos definir al significado de un programa lógico como la intersección de todos los modelos de Herbrand.

$Cn(S) = \cap M_i$

En el caso clásico, la cantidad de modelos de Herbrand son finitos, si la base es finita. Pero hay infinitos modelos de Herbrand difusos para una base de Herbrand difusa. Definiremos intersección de modelos difusos de la siguiente manera:

Veamos el siguiente ejemplo difuso:

$S = \{p(a)/0.8 , q(b)/0.7\}$
$M_1 = \{p(a)/0.8 , q(b)/1 , p(b)/1\}$
$M_2 = \{p(a)/0.9 , q(b)/0.75 , q(a)/1\}$

se puede ver M_1 y M_2 son modelos de S, también lo será la intersección y utilizaremos para computar la intersección el operador ***min***, sea $M_3 = M_1 \cap M_2$

$M_3 = \{ p(a)/0.8, q(b)/0.75, p(b)/0, q(a)/0 \}$

Si computáramos todos los modelos de S, la intersección de estos no daría el modelo mínimo de S.

Sea $P = \{ p(a) / 0.8, q(b) / 0.7, p(X) \wedge q(Y) \rightarrow r(X) \}$

Las consecuencias lógicas $Cn(P)$ o significado de este programa lógico es:

$Cn(P) = \{ p(a) / 0.8, q(b) / 0.7, r(a) / 0.7 \}$

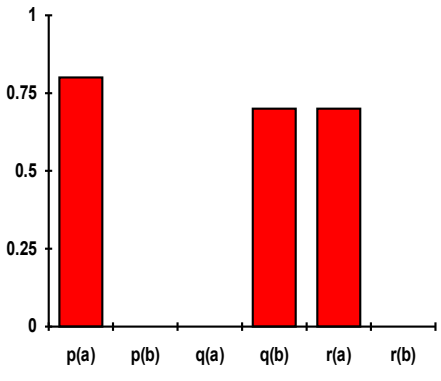


Gráfico de $Cn(P)$.

7.4.3 α-Modelos

Def.: Sea P un programa lógico difuso de L_A y M_1 un modelo de P, si todos los niveles de verdad de los términos ground que componen a M_1 superan a α entonces M_1 es un α-modelo y llamaremos "*umbral*" a este nivel α.

8. Características de la herramienta

La herramienta propuesta como conclusión de este trabajo, intenta destacar las características mas sobresalientes de la lógica difusa, generando un ambiente cómodo y amigable para desarrollar aplicaciones que resuelvan problemas reales.

Esta herramienta puede ser personalizada según las preferencias del usuario o constructor del sistema. Entre las opciones a personalizar podemos encontrar distintos métodos para evaluar la conjunción, disyunción, negación, umbral, defusificación y la ubicación de los módulos de conjuntos difusos, reglas asociadas y motor de inferencia. Entre las características generales de la herramienta podemos destacar:

- Múltiples operadores.
- Redefinición por parte del usuario de los operadores.
- Umbral de propagación.
- Posibilidad de defusificar los resultados.
- Múltiples proyectos y preferencias.
- Visualización de conjuntos difusos.
- Un ambiente integrado para la construcción del motor de inferencia, reglas y conjuntos difusos.

Los distintos métodos implementados para evaluar la conjunción y disyunción se detallan a continuación:

	Conjunción	**Disyunción**
Zadeh	$\min[a,b]$	$\max[a,b]$
Lukasievicz	$\max[0,a+b-1]$	$\min[1,1-a+b]$
Producto	$a.b$	$(a+b)-(a.b)$
Definida por el usuario	...	...

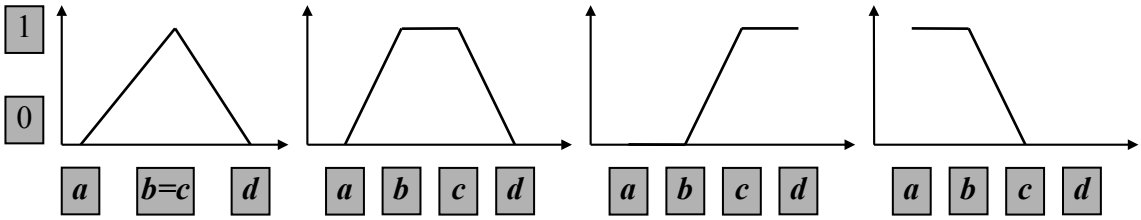
Para evaluar la negación se puede elegir entre:

Estándar	$1-a$
Sugeno	$(1-a)\big/(1+\lambda a)$
Yager	$\sqrt[w]{1-a^w}$
Definida por el usuario	**...**

Además se debe especificar donde residen los módulos de definiciones de conjuntos difusos, reglas y motor de inferencia.

Conjuntos difusos

Los conjuntos difusos se pueden representar de varias formas: triangular, trapezoidal, tipo S ó tipo Z como muestra la siguiente figura. Para ello solo hay que especificar el tipo y los valores ***a, b, c*** y ***d***.



Por ejemplo el conjuntos difuso 'joven' se podría representar de la siguiente manera:

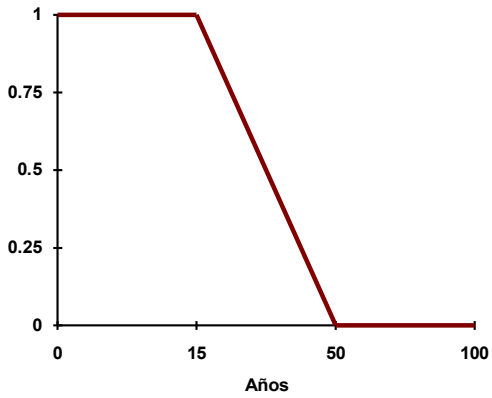
fs(persona,joven,tipo-z,[a(0),b(15),c(50),d(100)]).

donde los x's intermedios son interpolados en forma lineal.

fs(OBJETO,VALOR,TIPO,[a(A),b(B),c(C),d(D)]

dominio(OBJETO,DOMINIO)

TIPO ≡ { triangular, trapezoidal, tipo-s o tipo-z }

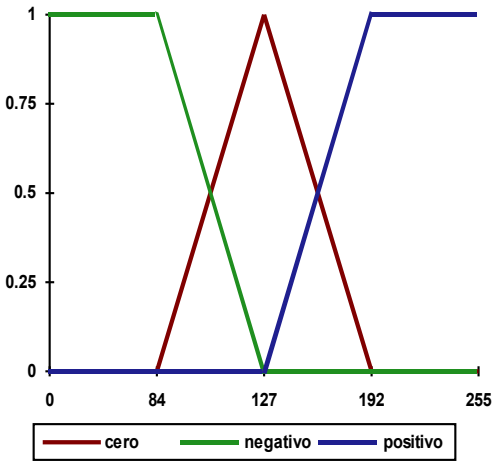


FS edad(joven) ≡ [a(0),b(15),c(50),d(100)]

dominio(edad) ≡ { años }

rango(edad) ≡ [0..100]

TIPO ≡ { tipo-z }



En este caso se han definidos tres conjuntos difusos, típicos en los ejemplos de *control*, si el conjunto difuso fuera la *fuerza*, donde los posible valores son: cero, negativo y positivo. En el primer caso, el conjunto difuso es triangular, en el segundo tipo-z y el ultimo tipo-s.

Reglas

Las reglas son representadas con la forma descripta en la sección 6.2 de modelos vagos:

Ej.: **si** presión = alta **y** temperatura = normal **entonces** volumen = bajo.

para lograr mayor declarativid se le ha agregado un nombre o código a cada regla.

regla1 **::= si** presión **es** alta **&** temperatura **es** normal **=>** volumen **es** bajo.

donde los operadores fueron definidos de la siguiente manera:

```
op(1050,xfx,::=).

op(1000,fx,si).

op(900,xfx,=>).

op(670,xfy,&).

op(600,xfx,es).
```

Donde estas reglas describen relaciones entre conjuntos difusos. El operador (=) puede ser leído como 'es' y no como una relación de igualdad. De esta manera no hace falta tener operadores del tipo (>) mayor o (<) menor dado que estos conceptos pueden ser definidos en el conjunto difuso.

Hechos:

Los hechos son representados de la siguiente manera:

presión **es** alta **fc** 0.8

ó

no(apertura **es** alta **fc** 0.2).

En el primer caso estamos especificando que la presión es muy alta, o casi definitivamente alta. Técnicamente estamos haciendo un α-corte de 0.8 sobre el conjunto difuso presión es alta. En el segundo ejemplo estamos definiendo un concepto pero por su complemento, el complemento será evaluado según las preferencias definidas.

Como se puede ver, es un ambiente general para la construcción de sistemas difusos, cambiando el modulo del motor de inferencia se puede hacer razonamiento hacia atrás (backward-chaining) o hacia adelante (forward-chaining).

Una de las características interesantes en dicho ambiente es la figura del umbral de propagación, este umbral tiene puede ser definido como un número real en el intervalo [0,1]. Si el umbral de propagación esta fijado en 1, todas las reglas para poder ser disparadas, sus antecedentes deberán ser totalmente ciertos. Es decir, estaríamos en un entorno de trabajo del estilo de lógica clásica. En cambio si fijamos el umbral en 0, todas las reglas van a poder ser disparadas. Por lo que se puede observar este umbral juega la función de controlador de la anchura de los arboles de inferencia.

9. Aportes Originales

Quiero destacar de este trabajo, los siguientes aportes originales:

- *Una herramienta general*. La herramienta construida es definida luego de hacer una recopilación de trabajos sobre lógica difusa, y no toma partido por ninguno de ellos en particular, sino que trata de tomar las mejores ideas.

- *Umbral de propagación*. Este nuevo concepto es utilizado en el proceso de la inferencia, generalmente se utilizaban valores difusos para escribir las reglas o los hechos, en este caso este umbral fija cuanto se van a expandir los arboles de búsqueda, o debilitar las inferencias.

- *Sintaxis fija, semántica variable*. Las sintaxis para la escritura de reglas es fija, pero cambiando en las preferencias podemos interpretar a los operadores de conjunción, disyunción y negación de manera diferente.

- *Ambiente de construcción*. El ambiente ha sido utilizado para la construcción del motor de inferencia, las reglas y los conjuntos difusos.

10. Conclusiones y Trabajo Futuro

No hace falta probar la efectividad que ha tenido la utilización de la lógica difusa en resolver problemas reales de algún modo inteligente, se pueden solucionar problemas complejos con técnicas sencillas.

Los *métodos convencionales* son buenos para problemas simples, mientras que los *sistemas difusos* son adecuados para problemas complejos o aplicaciones que requieren pensamiento intuitivo.

El próximo paso en esta disciplina seguramente estará dado por la sinergía lograda entre diferentes paradigmas que ayudaran a resolver problemas desde diferentes enfoques, por ejemplo el de redes neuronales, algoritmos genéticos, etc. Esta claro que cuando hablamos de razonamiento inteligente no nos podemos restringir a solo un tipo de razonamiento, deductivo o inductivo, o al tratamiento simbólico o numérico, no debemos ser extremistas o minimalistas y encontrar una buena forma de globalizar dichas técnicas o enfoques para poder construir maquinas que *razonen* un poco mejor.

Como continuación de este trabajo, sugerimos utilizar el ambiente para la creación de otros motores de inferencia o la implementación de soluciones a problemas reales.

11. Bibliografía:

[D&P_1979] Dubois, Didier & Prade, Henri: "*Outline of fuzzy set theory: an introduction*" Advances in fuzzy set theory and applications. pp 27-48. 1979.

[D&P_1980] Dubois, Didier & Prade, Henri: "*Fuzzy Sets and Systems: Theory and Applications*", Academic Press, New York, 1980.

[K&F_1992] Klir George J. & Folger Tina A.: "*Fuzzy Sets, Uncertainty, And Information*", Prentice-Hall International Editions. 1992.

[Ma_1974] Mamdani, E.H.: "*Application of fuzzy algorithms for control of simple dynamic plant*" Proc. Inst. Electr. Eng. 121, pp 1585-1588. 1974.

[M&A_1975] Mamdani, E.H. & Assilian, S.: "*An experiment in linguistic synthesis with a fuzzy logic controller*" Electron. Lett. 11, pp 625-626. 1975.

[KHJ_1996] Mark Kantrowitz, Erik Horstkotte, and Cliff Joslyn, "*Answers to Frequently Asked Questions about Fuzzy Logic and Fuzzy Expert Systems*", comp.ai.fuzzy, 02/1996, ftp://ftp.cs.cmu.edu/user/ai/pubs/faqs/fuzzy/fuzzy.faq

[Rus_1992] Ruspini Enrique: "*Lógica Difusa*" Escuela de Ciencias Informáticas. 1992.

[Rus_1994] Ruspini Enrique: "*Introduction to Fuzzy Set Theory and to Fuzzy Logic*" Curso de temas teóricos y prácticos de lógica difusa. 1994.

[Sa_1979] Sánchez E.: "Medical diagnosis and composite fuzzy relations" Advances in fuzzy set theory and applications. pp 437-444. 1979.

[TAS_1992] Toshiro Terano, Kiyoji Asai, Michio Sugeno. "*Fuzzy Systems Theory and its Applications*" 1992.

[To_1993] Toshinori Munakata, Yashvant Jani. "*Fuzzy Systems: An Overview*". Communications of the ACM. pp 69-76. 1994.

[Yoichi_1991] Yoichi Tsuji, Takefumi Usui, Kazuyuki Nagasawa y Misao Itoi. "*Automatic detection of sleep onset by means of fuzzy logic*". Sleep Research 20A, 1991.

[Za_1965] Zadeh L.A.: "*Fuzzy Sets*" Information and Control. Vol. 8, pp 338-353. 1965.

[Za_1975] Zadeh L.A.: "*Fuzzy Logic And Approximate Reasoning*" Synthese 30, pp 407-428. 1975.

[Za_1983] Zadeh L.A.: "*The role of fuzzy logic in the management of uncertainty in expert systems*" Fuzzy Set And Systems 11, pp 199-227. 1983.

11.1 Bibliografía médica:

[R&K_1968] Rechtschaffen & Kales. "*A normal of standardized terminology, techniques and scoring system for sleep stages of human subjects*". 1968.

12. Apéndice A

12.1 Implementación Prolog de un Controlador Difuso.

```
%%%%%%%
%%%%%%% Definición de operadores
%%%%%%%
:- reset_op.
:- op(1050,xfx,::=).
:- op(1000,fx,if).
:- op(900,xfx,then).
:- op(670,xfy,&).
:- op(600,xfx,=).
:- op(500,xfy,v).

%%%%%% Gramática de una regla
%%%%%%       regla        ###    nro. ::= if Antecedentes then Consecuente
%%%%%%       Antecedentes ###    Antecedente
%%%%%%       Antecedentes ###    Antecedente & Antecedentes
%%%%%%       Antecedente  ###    (X = A)
%%%%%%       Consecuente  ###    (Y = B)

r01 ::= if (angulo = nl) & (velocidad = ze) => (fuerza = pl).
r02 ::= if (angulo = ze) & (velocidad = nl) => (fuerza = pl).
r03 ::= if (angulo = nm) & (velocidad = ze) => (fuerza = pm).
r04 ::= if (angulo = ze) & (velocidad = nm) => (fuerza = pm).
r05 ::= if (angulo = ns) & (velocidad = ze) => (fuerza = ps).
r06 ::= if (angulo = ze) & (velocidad = ns) => (fuerza = ps).
r07 ::= if (angulo = ns) & (velocidad = ps) => (fuerza = ps).
r08 ::= if (angulo = ze) & (velocidad = ze) => (fuerza = ze).
r09 ::= if (angulo = ze) & (velocidad = ps) => (fuerza = ns).
r10 ::= if (angulo = ps) & (velocidad = ze) => (fuerza = ns).
r11 ::= if (angulo = ps) & (velocidad = ns) => (fuerza = ns).
r12 ::= if (angulo = ze) & (velocidad = pm) => (fuerza = nm).
r13 ::= if (angulo = nm) & (velocidad = ze) => (fuerza = nm).
r14 ::= if (angulo = ze) & (velocidad = pl) => (fuerza = nl).
r15 ::= if (angulo = pl) & (velocidad = ze) => (fuerza = nl).

%%%%%%%%%%%%
%%%%%%%%%%%% Definición de conjuntos difusos
%%%%%%%%%%%%

system_input([angulo,velocidad]).
system_output([fuerza]).

fs(angulo,[nl,nm,ns,ze,ps,pm,pl]).
fs(velocidad,[nl,nm,ns,ze,ps,pm,pl]).
fs(fuerza,[nl,nm,ns,ze,ps,pm,pl]).

traduccion(nl,'Negative Large').
traduccion(nm,'Negative Medium').
traduccion(ns,'Negative Small').
traduccion(ze,'Zero').
traduccion(ps,'Positive Small').
```

```
traduccion(pm,'Positive Medium').
traduccion(pl,'Positive Large').

upper_limit(255).

fs(angulo,nl,000,031,031,063).
fs(angulo,nm,031,063,063,095).
fs(angulo,ns,063,095,095,127).
fs(angulo,ze,095,127,127,159).
fs(angulo,ps,127,159,159,191).
fs(angulo,pm,159,191,191,223).
fs(angulo,pl,191,223,223,255).

fs(velocidad,nl,000,031,031,063).
fs(velocidad,nm,031,063,063,095).
fs(velocidad,ns,063,095,095,127).
fs(velocidad,ze,095,127,127,159).
fs(velocidad,ps,127,159,159,191).
fs(velocidad,pm,159,191,191,223).
fs(velocidad,pl,191,223,223,255).

fs(fuerza,nl,000,031,031,063).
fs(fuerza,nm,031,063,063,095).
fs(fuerza,ns,063,095,095,127).
fs(fuerza,ze,095,127,127,159).
fs(fuerza,ps,127,159,159,191).
fs(fuerza,pm,159,191,191,223).
fs(fuerza,pl,191,223,223,255).

%%%%%%%
%%%%%%% Motor de inferencia para control difuso
%%%%%%%

member(X,[X|Xs]).
member(X,[Y|Ys]):- member(X,Ys).

append([],X,X).
append([X|Xs],Y,[X|Z]):- append(Xs,Y,Z).

maximo(X,L):- member(X,L), mayor_todos(X,L).

mayor_todos(X,[]).
mayor_todos(X,[Y|Ys]):- X >= Y, mayor_todos(X,Ys).

minimo(X,L):- member(X,L), menor_todos(X,L).

menor_todos(X,[]).
menor_todos(X,[Y|Ys]):- X =< Y, menor_todos(X,Ys).

%%%%%%
%%%%%%  get_system_inputs
%%%%%%
get_system_inputs:- system_input(Inputs), abolish('/'(si,2)), leer(Inputs).

%%%%%%
```

```
%%%%%% El leer se hace desde teclado, pero podria ser una lectura desde ports
%%%%%%
leer([]).
leer([X|Xs]):-  write('Cual es el valor de '), write(X), write(' ? '),
  read(Y), asserta(si(X,Y)), leer(Xs).

%%
%% Fuzzificacion:
%%% Para cada system input, computar el grado de pertenencia a cada conjunto
difuso.
%%%
fuzzification:-
  system_input(Inputs),
  abolish('/'(mt,3)),
  computar_grado_de_pertenencia(Inputs).

computar_grado_de_pertenencia([]).
computar_grado_de_pertenencia([X|Xs]):-
  computar_grado(X),
  computar_grado_de_pertenencia(Xs).

computar_grado(X):-
  findall(fs(X,FS,A,B,C,D),fs(X,FS,A,B,C,D),Todos),
  computar(Todos).

computar([]).
computar([fs(X,FS,A,B,C,D)|Xs]):-
  si(X,Input),

  Point_1 is A,
  Point_2 is D,
  Slope_1 is 255 / (B - A),
  Slope_2 is 255 / (D - C),

  Delta_1 is Input - Point_1,
  Delta_2 is Point_2 - Input,

  case(
    [ (Delta_1 =< 0) -> (Value is 0),
      (Delta_2 =< 0) -> (Value is 0),

      true -> (S1D1 is Slope_1 * Delta_1,
              S2D2 is Slope_2 * Delta_2,
              upper_limit(UL),
              minimo(Value,[S1D1,S2D2,UL]))
    ]),

  asserta(mt(X,FS,Value)),    % mt = memoria de trabajo
  computar(Xs).

rule_evaluation:-
  findall(R,(R::=X),L), abolish(re/4), evaluar(L),
  buscar_maximos.

buscar_maximos:-
  re(X,A,B,V1), re(Y,A,B,V2), X \== Y, maximo(V,[V1,V2]), retract(re(X,A,B,V1)),
```

```
  retract(re(Y,A,B,V2)), assert(re(X v Y,A,B,V)), buscar_maximos.
buscar_maximos.

evaluar([]).
evaluar([X|Xs]):- evaluo(X), evaluar(Xs).

evaluo(X):-
  (X ::= if A => Y=B),  % Busco la regla X
  eval_antec(A,V),      % Evaluo los antecedentes
  assert(re(X,Y,B,V)).  % La regla X concluye B con valor V

eval_antec((A = B),V):- mt(A,B,V).
eval_antec(A & B,V):- eval_antec(A,V1), eval_antec(B,V2), minimo(V,[V1,V2]).

%
% hacer la defuzzification para cada system output
%
defuzzification:-
  findall(A,(re(A,B,C,D),D > 0),L), defusificar(L,[],[],Valor),
  abolish(so/2), assert(so(fuerza,Valor)).

defusificar([],N,D,V):- sumo_lista(N,SN), sumo_lista(D,SD), V is SN / SD.
defusificar([X|Xs],Numerador,Denominador,Valor):-
  re(X,B,C,D), centroide(B,C,V), Num is D*V,
  append([Num],Numerador,N1), append([D],Denominador,D1),
  defusificar(Xs,N1,D1,Valor).

sumo_lista([],0).
sumo_lista([X|Xs],Y):- sumo_lista(Xs,Z), Y is X + Z.

centroide(X,Y,Valor):- fs(X,Y,A,B,C,D), Valor is (A+D) / 2.

put_system_outputs.

main:-
  get_system_inputs,listing(si), nl,
  fuzzification, listing(mt), nl,
  rule_evaluation, listing(re), nl,
  defuzzification, write('System output = '), listing(so), nl,
  put_system_outputs,
  main.
```

13. Apéndice B

En esta sección se brindara documentación técnica para extender la funcionalidad de la herramienta implementada.

13.1 Documentación Técnica

El ambiente de programación para el desarrollo de sistemas expertos difusos ha sido implementado en Visual Basic Professional versión 3.0 con llamadas a una biblioteca de funciones prolog implementada para la materia Lab. VII-GA.

Los archivos que conforman la biblioteca de funciones prolog son: `WAPI.DLL`, `WAPI.IDB` y `ARITY.DLL`. Estos archivos se deben instalar en el mismo directorio que el programa ejecutable (`FUZZY.EXE`).

Esta bibliteca de funciones puede ser utilizada desde cualquier lenguaje de programación en el ambiente Windows tipo Visual Basic, Delphi, Borland o Microsoft C es decir cualquier lenguaje que soporte la declaración de funciones externas.

Los prototipos de las declaraciones son los siguientes:

```
Declare Function Init Lib "wapi.dll" (ByVal v as Integer) as Integer
Declare Function Done Lib "wapi.dll" () as Integer
Declare Function AddClauses Lib "wapi.dll" (ByVal aStr$) as Integer
Declare Function ProveGoal Lib "wapi.dll" (ByVal aStr$) as Integer
Declare Function AnotherProve Lib "wapi.dll" () as Integer
Declare Function GetPrologStr Lib "wapi.dll" () as String
Declare Function VBGetPrologStr Lib "wapi.dll" () as String
```

Además se ha implementado un predicado especial para permitir la comunicación entre prolog y la UI (user interfase), este predicado se lo denoto **"writeInUI(X)"**. Este predicado deposita en una memoria temporaria el resultado obtenido y cuando se invoca a la funcion **`GetPrologStr`** o **`VBGetPrologStr`** se vacia el resultado. Si hay varios resultados almacenados se los retira en forma de pila mediante la función **`AnotherProve`** hasta que ésta este vacia, el siguiente ejemplo ilustra la utilización de estas funciones y predicados en un ambiente procedural y un ambiente lógico.

Visual Basic

```
Dim s as string
Dim value as integer
value = Init(False) //inicio el interprete prolog
if value <> False then
  value = AddClauses("demo.ari") //Consult(demo.ari)
  if value <> False then //si todo OK
    value = ProveGoal("member(X,[1,2,3]),writeInUI(X)")
    while value <> False //mientras haya otros
      s = VBGetPrologStr() //obtengo los siguientes valores
      Debug.Print s //imprimo el resultado
      value = AnotherProve() //intento hacer otra prueba
    wend
  end if
  value = Done() //fin del uso del interprete prolog
end if
```

Prolog (demo.ari)

```
member(X,[X|Xs]).
member(X,[Y|Ys]):-
  member(X,Ys).
```

Utilizamos la función **`VBGetPrologStr()`** en vez de **`GetPrologStr()`** dado que los strings en Visual Basic no son del tipo ASCZ (strings terminados con un caracter nulo o "\0"), si utilizaramos otro ambiente de desarrollo tipo Visual C++ o Delphi se debería usar la función **`GetPrologStr()`**.

A continuación se describiran algunos predicados importantes que deben estar definidos

`objetivo(X)`	El usuario define cual es el objetivo que desea probar, ej. objetivo(apertura). Para que el objetivo pueda ser probado, alguna regla debe concluir en X ó X es un hecho.
`run(X,L)`	El motor de inferencia trata de probar el objetivo definido por el usuario. La lista L devuelta contiene los posibles resultado. El formato de L puede variar (si se utiliza defusificación o no), pero siempre viene ordenada en base a los valores de verdad.
`member(X,L)`	Definido en Common.ari, es utilizado para separar los resultados cuando vienen en formato de lista.
`umbral(X)`	Definido en las preferencias, $0 \leq X \leq 1$.
`expl(E-regla)`	Predicados que contienen las reglas utilizadas.
`expl(H-hecho)`	Predicados que contienen los hechos utilizados.
`donde(fsets,X)`	X define el sendero a los conjuntos difusos.
`donde(motor,X)`	X define el sendero al motor de inferencia.
`donde(reglas,X)`	X define el sendero a las reglas.
`conjuncion(X)`	$X \in$ {zadeh , lukasiewicz , producto , definida-por-el-usuario}
`disyuncion(X)`	$X \in$ {zadeh , lukasiewicz , producto , definida-por-el-usuario}

`negacion(X)`	X ∈ {standard , sugeno , yager , otra }
`sugeno(λ)`	Parametro λ de Sugeno.
`yager(ω)`	Parametro ω de Yager.
`desfuzzifico(X)`	X ∈ { true , false }
`dominio(C,D)`	C≡Clase, D≡Dominio. Ejemplo: dominio(temperatura,grados).
`fsd(C,P,T,L)`	Definición de un conjunto difuso donde: C≡Clase, P≡Propiedad, T≡Tipo, L≡Lista de valores, donde T ∈ {trapezoidal,triangular,s,z}, ejemplo: fsd(temperatura,alta,0,[a(159),b(191),c(223),d(255)]).
`fs(S,T,L)`	Otra posible definicion de un fuzzy set, esta forma fue la primera utilizada y solo cntemplaba formas trapezoidales, luego se agrego la posibilidad de 4 formas descriptas en el punto anterior, ejemplo: fs(temperatura,alta,[[159,0],[191,1], [223,1],[255,0]]).
`fuzzy_sets(L)`	L = Lista de los posibles conjuntos fuzzy.

El motor de inferencia implementado (`motor.ari`) ha sido programado integramente utilizando los predicados prolog estándar, su predicado principal de entrada es **`run(X,L)`**, donde **`X`** es el objetivo y **`L`** es la lista resultado. Este resultado puede ser una lista de conjuntos difusos con sus respectivos valores de certeza o el resultado de la defusificación.

INDICE