ALGUNAS EXPERIENCIAS DIDÁCTICAS EN EL CAMPO DE LA LÓGICA DIFUSA

Joan S. Domingo **Eusebi Realp**

EUETIB
C/Comte d'Urgell 187, Dpt. d'Electrònica
08036 Barcelona Tf.: (93)430 16 04 Fax: (93)430 97 07
e-mail: jdom@ei.euetib.upc.es **e-mail: erealp@ei.euetib.upc.es**

RESUMEN

Desde 1990, el Grupo de Lógica Difusa y Redes Neuronales de la EUETI de Barcelona viene trabajando en distintos proyectos que hasta la fecha se han traducido en distintas realizaciones que se han presentado en diferentes foros nacionales. En la presente ponencia, se exponen los trabajos realizados con el doble ánimo de darse a conocer y de establecer nexos con grupos similares a nivel nacional. Todos los trabajos persiguen hasta la fecha la eleboración de aplicaciones prácticas de regulación con finalidad docente, sobre diferentes plataformas.

INTRODUCCIÓN

Desde 1990, el Grupo de Lógica Difusa y Redes Neuronales Artificiales de la Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona, EUETIB, estableció tres líneas de trabajo, una sobre Autómatas Programables (PLC), otra sobre microprocesadores y otra sobre lógica programable (PLD). El objetivo del grupo ha venido siendo la difusión de estas técnicas entre su alumnado, con el propósito de formar a los nuevos técnicos en el dominio de estas materias. Desde entonces, se han llevado a cabo varias implementaciones dentro del marco de los Proyectos de Fin de Carrera con la colaboración de los alumnos de primer ciclo de ingeniería en electrónica industrial del Plan 1972.

1 CONTROL DE TEMPERATURA CON PLC [5]

En la línea PLC, se ha realizado un control de temperatura con el módulo fuzzy FZ001 que Omron propone para sus autómatas de la serie C-200H. La aplicación está basada en el diagrama de bloques que se presenta en la figura 1. En ella se puede apreciar que la estructura es, básicamente, la típica de un control clásico, pero con la salvedad de que el controlador convencional ha sido sustituido por el controlador difuso.

La aplicación, consiste en controlar la temperatura de un recinto de reducidas dimensiones con una consigna fijada entre 0 y 100 °C (0 a 10 V). Para el aporte energético se utiliza el calor suministrado por 4 lámparas de incandescencia de 40W cada una, utilizando, además, su luminosidad para dar una idea visual de la aportación de calor en función de la luz. Como elemento refrigerador se utiliza un pequeño ventilador alimentado a 12 Vdc.

La unidad difusa C200H-FZ001 aporta al PLC C200H un avanzado circuito de proceso rápido, basado en lógica difusa, con 8 entradas y 4 salidas; cada regla puede tener hasta 8 condiciones y 2 conclusiones, lo que permite un amplio rango de aplicación. La CPU del PLC trata esta unidad como a cualquier otro módulo de E/S y puede accederse con instrucciones tipo movimiento de datos.

La programación del PLC C200H se ha reducido al simple control de la unidad difusa FZ001 en concordancia con el objetivo didáctico que subyace en todas las aplicaciones llevadas a cabo por el grupo de trabajo citado. La ocupación en código del programa, es muy reducida.

En la aplicación considerada, se han tomado siete funciones de pertenencia de las dos variables de entrada (ERROR y DERR) y se han distribuido de forma equitativa entre los 4.095 puntos posibles (2^{12}), esto es, entre 0 V y +10 V. Para las funciones de pertenencia de las variables de salida (CALOR y FRÍO), se sigue el mismo procedimiento anterior, teniendo en cuenta que el programa FSS opera en base a singletones. El número de funciones de pertenencia para las salidas es de 4, pudiéndose aumentar el número de las mismas para el caso de desear una mayor precisión.

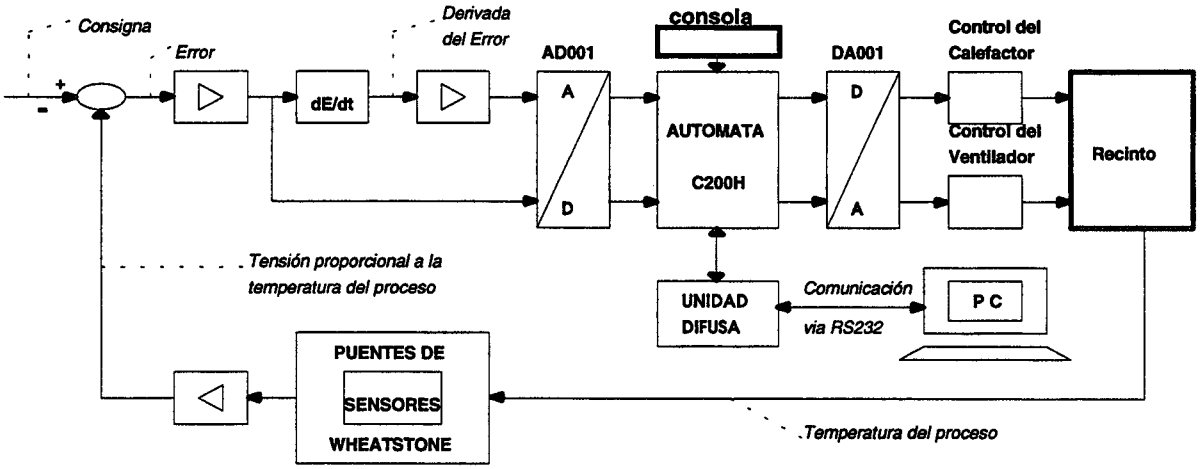


Figura 1: Diagrama de bloques del sistema de regulación.

Mediante el software FSS, se pueden transmitir las FAM al módulo de control difuso, a la vez que monitorizar su ejecución y suministrar consignas; el recinto sujeto a variación térmica, puede asimismo afectarse de un aporte externo de energía adicional o de una sobreventilación, de forma que se pueda variar asíncronamente con el control del proceso, los parámetros del mismo, para poder mejor evaluar la bondad de respuesta del control.

2 CONTROL DE TEMPERATURA CON μP RISC [4]

Como extensión de la aplicación anterior, se ha realizado un segundo control de temperatura, en las mismas condiciones allí expuestas pero con la variante de utlizar un microcontrolador en vez de un PLC.

La consigna, al igual que en la aplicación con PLC, es modificable desde el exterior, y la temperatura del proceso, dada a través de sensores de temperatura (PT-100), es la variable de entrada al proceso. A partir de ella y de forma analógica se calcula el error y la derivada del mismo, de forma que se disponga de ambas magnitudes como variables de entrada concreta al regulador. Con estas variables el controlador difuso realiza las tres fases clásicas de un proceso de control fuzzy, fuzzificación, evaluación de reglas, y defuzzificación.

El microcontrolador utilizado para esta variante es el PIC 16C71 de Microchip, que se ha programado de forma que, una vez recibidos los valores de la temperatura del proceso y la consigna (en 8 bits cada uno), calcule el error y la derivada del mismo, tomada como el valor incremental sufrido por la temperatura entre dos ciclos de programa consecutivos. A partir de estas variables de entrada se calculan los valores de las variables de salida, que son las que controlan la etapa de potencia de las lámparas de incandescencia y la del ventilador.

La implementación de la base de conocimiento en memoria se realiza utilizando 4 puntos para representar cada una de las funciones de pertenencia. Esta estrategia, posibilita realizar funciones triangulares o trapezoidales en un reducido espacio, dada su simplicidad. Para cada una de las dos variables de entrada, por lo tanto, para almacenar esta información se necesitan 28 posiciones de memoria.

Respecto a los 4 puntos que definen un conjunto difuso, el primero y el cuarto tienen asociada una ordenada nula, mientras que a los dos puntos intermedios les corresponden un valor de 255. En cuanto a las funciones de pertenencia de las variables de salida, se han tomado funciones de tipo singleton. Se ha previsto hacer frente a un número máximo de siete singletons para cada variable de salida, por lo que se reservan 14 posiciones de memoria para contener la definición de las variables de salida en el microcontrolador.

2.1 Conclusiones

Los ensayos prácticos realizados muestran un buen funcionamiento del regulador. Las diferentes situaciones a las que se le sometió, proporcionaron una respuesta lo suficientemente acotada dentro de las posibilidades del mismo. Aparte del interés práctico del sistema desarrollado, por su reducida complejidad es un módulo muy útil para introducirse en el estudio del control basado en la lógica difusa. Por otra parte el algoritmo del programa, estructurado en forma de módulos, puede servir de guía para futuras ampliaciones en la misma línea de trabajo.

3 REGULACIÓN DE VELOCIDAD DE UN MOTOR TRIFÁSICO [6]

Como continuación a los desarrollos citados en los apartados precedentes, la presente aplicación regula un motor de corriente alterna trifásico de potencia fraccionaria.

El sistema de control se ha organizado en torno al mC de Microchip, PIC17C44, cuya programa-ción puede obtenerse a partir de la herramienta software FUZZYTECH-MP, que facilita la generación del programa compilado a partir de las funciones de pertenencia y base de reglas de las variables de control entradas a dicho software.

La utilización del PIC17C44 permite un buen tiempo de ejecución del programa, al tener implementado un multiplicador hardware que realiza, en un solo ciclo de instrucción, una multiplicación de dos variables de 8 bits.

3.1 Descripción del hardware

Los elementos que foman el sistema hardware, son el µC, un convertidor A/D MAX120 con un tiempo de conversión inferior a los 10 µs y otro D/A MAX530, de 12 bit ambos, un PC y un ondulador que controla la velocidad del motor por variación de frecuencia. El control del ondulador se basa en una tensión contínua de entre 0 y 10 V.

A partir de la consigna de velocidad entrada al sistema de regulación y de la velocidad actual (medida con una dínamo tacométrica acoplada mecánicamente al motor), el µC calcula el error de velocidad (ERR) y la derivada de éste respecto al tiempo (DERR) de acuerdo con las expresiones utilizadas también para la aplicación de los apartados anteriores. Estas dos variables concretas serán las entradas al sistema de control difuso.

A cada una de las variables se le asignan cinco funciones de pertenencia con etiquetas lingüísticas NL,NS,ZR,PS,PL, distribuidas dentro del universo de discurso. Dicho número de funciones de pertenencia es el máximo que permite el software empleado. Para las funciones de pertenencia de la variable de salida, se emplean cinco singletons.

3.2 Prueba y puesta a punto del programa de control

La obtención del programa para el control difuso óptimo, exige necesariamente la realización de pruebas con diferentes conjuntos de funciones de pertenencia de cada una de las variables de entrada y salida. Por otra parte, la base de reglas óptima se obtendrá, también, a través de la realización de ensayos con distintos contenidos de la tabla que la define.

Todas las pruebas con diferentes conjuntos de funciones de pertenencia y bases de reglas se pueden realizar facilmente, de forma dinámica sobre el sistema, con la ayuda del software FUZZYTECH-MP sobre una plataforma PC, conectada via RS-232 al µC que gobierna el sistema de control.

Una vez obtenidas todas las especificaciones óptimas que definen al sistema, mediante la opción de compilación del FUZZYTECH-MP, se consigue la subrutina de control difuso de la velocidad. Esta subrutina se incorporará al programa de servicio general del µC y finalmente, con el paquete de software MPASM se consigue el programa objeto que tiene que ser grabado posteriormente en la memoria EPROM interna del µC.

4 IMPLEMENTACIÓN MEDIANTE FPGA DE UNA CÉLULA BÁSICA DE REGULACIÓN MEDIANTE LÓGICA DIFUSA [7]

La finalidad de dicha alicación es la de resolver en una versión monochip los mismos conceptos expuestos en la aplicación anterior, a la vez que ampliarla con un soporte hardware y software para poder testear y verificar el comportamiento de cada etapa de dicha célula mediante Boundary Scan, puesto que el acceso a todos los puntos del circuito, se ve imposibilitada por la cantidad de pins de la FPGA utilizada.

4.1 Estructura de la célula

La célula consta de dos entradas, una salida y nueve reglas de inferencia que se procesan en paralelo y está formada por los mismos bloques funcionales que se han señalado en la aplicación anterior.

Para cada una de las dos variables de entrada expresadas en 5 bits cada una de ellas, se han definido tres conjuntos difusos: N, ZE y P. El universo de discurso de ambas variables comprende el intervalo [0,31], y las funciones de pertenencia toman valores dentro del intervalo real [0,1], lo cual se refleja en la figura 2.

Nótese que tan sólo se utilizan 8 de las 32 combinaciones para el intervalo en que una entrada puede pertenecer simultáneamente a dos conjuntos; ello responde a criterios docentes, buscando la simplicidad, puesto que ésta aplicación forma parte de una práctica de introducción.

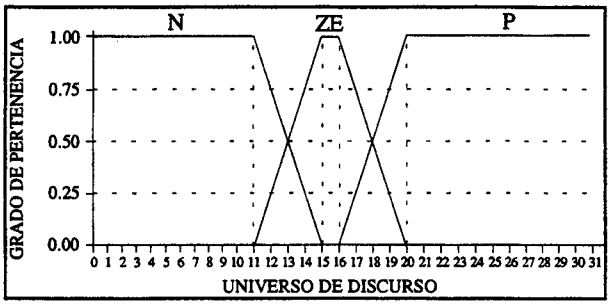


Figura 2: Conjuntos difusos asociados a las entradas.

Para poder tratar los valores obtenidos en esta primera etapa se utiliza la codificación que se describe en la tabla 1 (nótese que no se optimiza el código de 3 bits utilizado):

Tabla 1: Codificación de los grados de pertenencia.

Grado de pert.	Valor binario
0	000
0.25	001
0.5	010
0.75	011
1	100

De las posibles geometrías que pueden adoptar las funciones de pertenencia se han elegido las trapezoidales debido a la simplificación que representa su codificación y las operaciones que implica.

Se utilizan reglas simples, y una de las características de la célula consiste, como se ha indicado, en que el usuario pueda modificar estas reglas de un modo sencillo. Ello se realiza de forma externa codificando el consecuente de cada una de las nueve reglas que intervienen en el proceso de control.

Dado que se obtienen seis conjuntos difusos de entrada (como resultado de la fuzzificación de cada una de las variables de entrada en tres conjuntos difusos), se obtienen nueve reglas, las cuales pueden representarse mediante una matriz de 3x3 donde la intersección de cada fila con cada columna determina el consecuente de las reglas. Con dos bits es suficiente para determinar el consecuente: N (01), ZE (10) y P (11). La combinación 00 se reserva para el caso en que no se codifique ningún consecuente, con lo que dicha regla no intervendrá en el proceso.

Para el cálculo del valor máximo de los consecuentes pertenecientes al mismo conjunto de salida, realizado mediante la operación anterior, se ha diseñado un circuito combinacional que realiza el máximo para cada uno de los conjuntos difusos de salida de forma independiente y en paralelo.
A efectos de determinar la función asociada a los conjuntos difusos de salida, se ha empleado la inferencia correlación-producto, determinado el centro de gravedad de las figuras que forman cada uno de los conjuntos en función del grado de pertenencia obtenido como resultado del circuito de obtención del máximo.

4.2 Implementación y conclusiones

El diseño se ha implementado sobre dos FPGAs 4003A de Xilinx. Uno de los problemas que se ha afrontado ha sido la partición de la célula en dos chips, puesto que el punto ideal de separación en dos mitades prácticamente iguales, seria justo antes de determinar los máximos y este es justo el punto donde existen más conexiones, con el consiguiente problema de *pinout*. Al efecto, cabe señalar una aportación del mismo Grupo sobre la incidencia de la cantidad de conexiones que se tienen en cada punto de una aplicación de estas características [1].

La frecuencia máxima real de funcionamiento conseguida es de 24 Mhz, lo cual demuestra que la implementación mediante FPGAs da una velocidad de respuesta muy superior a la que obtendríamos frente a cualquier solución basada en microprocesador, puesto que en este caso, se alcanzan los 200 MFlips. En cualquier caso, el factor que limita la frecuencia de operación como es de entender, son los convertidores A/D, que en el mejor de los casos siguen teniendo un tiempo de respuesta inferior al de la célula fuzzy.

El control de un sistema basado en reglas de inferencia, reconfigurables por el usuario en todo momento, de forma dinámica, aporta un elevado grado de flexibilidad al prototipo. Así, la utlización de esta célula será idónea para experimentar en el control de sistemas en los que intervienen como entradas de control del proceso el error y la variación del error.

Además, dicha aplicación tiene accesibles las entradas TDI, TDO, TCK y TMS definidas por el estándar IEEE 1149.1 e implementadas en las FPGAs utilizadas, con la finalidad de poder efectuar un test interno de la célula e incluso reconfigurarla, lo cual da aún más flexibilidad a la aplicación. La implementación software del Boundary Scan Test se ha realizado en PC de forma que es posible comprobar de una forma muy simple y didáctica el funcionamiento de la célula.

5- APLICACIÓN DE UN CONTROL FUZZY A UNA SILLA DE RUEDAS [2]

La paraplejia, es una disminución física que afecta a una parte de la población y que impide en mayor o menor grado la suficiencia motriz; la presente aplicación trata de ofrecer una solución fuzzy al comportamiento que debe presentar una silla de ruedas para este dispar colectivo. Se utiliza la idea de controlar dicha silla mediante FPGAs que implementen un regulador difuso paralelo (PFLC).

Cabe considerar que la cantidad de variables a tener en cuenta en el proceso es relativamente elevada, de forma que se adoptará la idea de simplificar las mismas en base a reducir las reglas compuestas en reglas simples según el principio de asociatividad [1].

5.1 Variables consideradas

Las variables de entrada consideradas, son la velocidad, la vibración, la inclinación respecto al plano de rodamiento y la consigna. Como variable de salida, tan sólo se considera la velocidad. La entrada de la consigna de velocidad de avance y magnitud de giro vienen dadas por un *joystick* potenciométrico. La velocidad máxima se establece entorno a los 6 km/h y es la que puede alcanzar la silla cuando se actúa máximamente sobre dos motores, acoplados a las dos ruedas mayores.

5.2 Fuzzificación y Desfuzzificación

La fuzzificación de las entradas se realiza en base a cuatro conjuntos difusos, (NL, NM, PM y PL). Todas las variables concretas de entrada se expresan en 4 bits y los conjuntos difusos son de geometría trapezoidal, lo que permite sistematizar bastante el diseño a la vez que simplificarlo. Para representar los distintos grados de pertenencia a cada conjunto, se toman 2 bits, y otros 2 bits para codificar dichos conjuntos, de forma que la información final de los grados de pertenencia de cada variable de entrada, se expresa en 4 bits.

Normalmente, se toma la posición central de la matriz de inferencia para dar a la salida una acción de control ZE, cuando el número de subconjuntos difusos es impar; cuando, no obstante, es par, existen cuatro posiciones centrales en la matriz de inferencia, que deben hacer referencia las unas a las otras de forma que la salida vaya pasando continuamente de una de estas posiciones a otra. En la presente aplicación se ha optado por éste caso al poderse así aprovechar al máximo los bits que codifican a los conjuntos.

La velocidad de operación que las FPGA ofrecen, supera adecuadamente al tiempo de respuesta de los motores que accionan la silla, de forma que aunque se dé una situación de casi-estabilidad, al suministrarle continuamente órdenes al motor, éste, compensa tal cantidad de órdenes por el efecto de integración que ofrece.

La desfuzzificación se realiza mediante una técnica basada en LUT y el resultado de la misma se devuelve al espacio concreto en formato binario natural de 4 bits. No existe un tratamiento especial en este sentido, y se opera según métodos estándar.
La inferencia se realiza por cableado directo de todos los grados de pertenencia a cada conjunto difuso de salida.

5.3 Implementación y conclusiones

La implementación se ha realizado sobre dos FPGA de Xilinx, del tipo XC4003A; en la figura 3, puede apreciarse la agrupación por bloques que se ha alojado en cada una de las dos piezas, atendiendo a la densidad de cableado que un regulador lógico difuso como el utilizado exige.

Esta, es una de las más recientes aplicaciones que ha emprendido el Grupo, y a la fecha, se está en fase de montaje de los drivers para los dos motores y de la electrónica asociada a los sensores; ello no obstante, el diseño del control y de los acondicionadores tanto de las señales de entrada como de salida, ha superado con éxito todas las fases de simulación. Los resultados finales a nivel experimental se obtendrán en los próximos meses, de forma que a la fecha aún no es posible establecerlos.

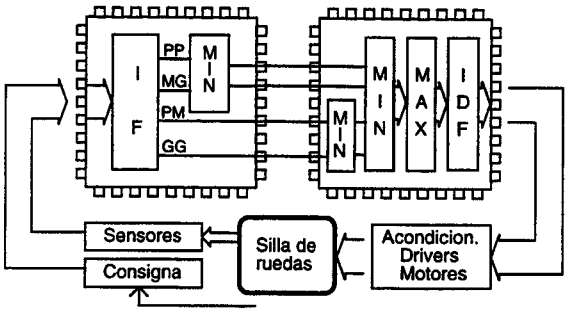


Figura 3: Partición del regulador en 2 FPGAs.

6 CONTROL DIFUSO DE UNA MORDAZA PARA MECANIZADOS [3]

La presente aplicación, persigue la obtención de un control difuso para una mordaza de banco de taller a efectos de sujetar piezas para su mecanizado; se persigue controlar la fuerza ejercida por la mordaza sobre la pieza, de forma que no se ejerzan presiones innecesarias que la puedan desgastar, a la vez que garantizando su sujeción durante el proceso de mecanizado, todo ello a la velocidad adecuada para operar en tiempo real.

6.1 Sistema global

La mordaza mecánica de sujeción viene accionada por un tornillo sin fin sobre el que actúa un motor paso a paso. La fuerza de sujeción de la pieza, con el control que se presenta, es el adecuado en cada instante y viene dada por la acción que un controlador difuso paralelo efectúa sobre la pieza sujetada en función del esfuerzo que las operaciones de mecanizado ejerzan sobre ella y de la naturaleza del material. El diagrama de bloques del sistema es el que se describe en la figura 4.

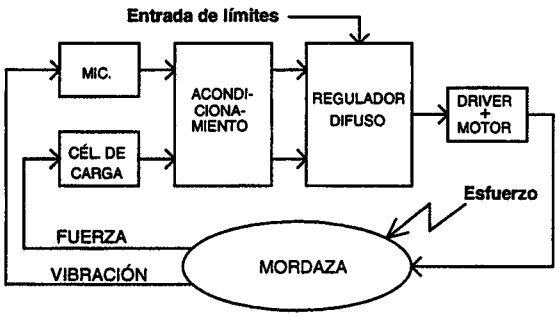


Figura 4: Diagrama de bloques.

6.2 Fuzzificación, Inferencia y Desfuzzificación

El regulador opera en base a 5 conjuntos difusos por cada variable de entrada, con geometría trapezoidal. A efectos de realizar una equivalencia directa entre la combinación de entrada y el grado de pertenencia a cada conjunto, se destinan 16 combinaciones para cada flanco de cada conjunto, lo que supone un código de 4 bits. La distribución de combinaciones de las variables concretas de entrada son: S ⇒ [000000-001111], SM ⇒ [000000-011111], M ⇒ [010000-101111], ML ⇒ [100000-111111] y L ⇒ [110000-111111].

A diferencia de la mayoría de reguladores difusos, en los que el punto de estabilidad se encuentra en el centro de la FAM, en la presente aplicación (véase la tabla 2), tal punto se fija en el extremo superior izquierdo, por considerarse que sobre el sistema no se van a ejercer fuerzas negativas y que el esfuerzo, por consiguiente, siempre va a ser positivo.

Tabla 2: FAM para la mordaza.

	S_1	SM_1	M_1	ML_1	L_1
S_2	Z	R	R	R	R
SM_2	MLP	SMP	SP	SP	R
M_2	LP	MLP	SMP	SP	R
ML_2	LP	LP	MLP	SMP	Z
L_2	LP	LP	LP	MLP	Z

El proceso de concretización se basa en singletones que representan a seis conjuntos de salida, uno para aflojar R (Reverse), uno de acción estable Z, y cuatro para aumentar la presión SP (Small Press), SMP (Medium), MLP y LP. El número de bits destinado a cubrir la salida es de 5, de forma que se dispone de 32 combinaciones, de las cuales se utilizan tan sólo 21 debido a la distribución que sobre los distintos conjuntos se efectúa.

De las 21 combinaciones posibles, una es la de equilibrio (00100, correspondiente a Z) y 4 corresponden al retroceso (00000, 00001, 00010 y 00011, sobre R), con lo que restan aún 16 combinaciones destinadas al avance (SP,SMP,MLP y LP). Nótese que la técnica utilizada, aún basarse en la de singletones, no es exactamente ésta.

6.3 Implementación y conclusiones

La implementación se ha realizado sobre una FPGA Xilinx 4005E-3, y permite una velocidad de evaluación de reglas de 50 Mflips. El grado de ocupación lógica de la FPGA está sobre el 80%, lo que permite alojar, además del regulador, el circuito de control de una etapa previa al driver micropaso del motor a efectos de traducir a cantidad de impulsos la combinación entregada por el regulador. Asimismo, la FPGA aloja la electrónica de tratamiento de las órdenes que el operario suministre al sistema.

REFERENCIAS

[1] Domingo, J. S. *"Consideraciones a la implementación de FLCs en FPGAs"*. SAAEI '96. Zaragoza.

[2] Domingo, J.S., A.Herms, I.Escoda, M.Martínez y D.Marín. *"Aplicación de un control fuzzy a una silla de ruedas." SAAEI-97.* Valencia *(en proceso de presentación)*

[3] Domingo, J.S., A. Herms, I. Escoda, A. Grau y J. Viñas. *"Control difuso de una mordaza para mecanizados." SAAEI-97.* Valencia *(en proceso de presentación)*

[4] Roca, G., O. Lapeña, E. Realp y A. Subirats. *"Módulo de control fuzzy con procesador RISC." SAAEI-95.* Tarragona.

[5] Subirats, A., E. Realp y H. Martínez. *"Lógica difusa: una aplicación con autómata programable". SAAEI-94.* Tarragona.

[6] Subirats, A., E. Realp y J. Escofet. *"Aplicación de la lógica difusa a la regulación de un motor de c.a. trifásico." SAAEI-96.* Zaragoza.

[7] Vilà, F., J.S. Domingo, A. Herms y I. Escoda. *"Implementación mediante FPGA de una célula básica de regulación mediante lógica difusa." SAAEI-97.* Valencia *(en proceso de presentación)*

PRINCIPALES CÁLCULOS SOBRE INCERTIDUMBRE DIFUSA A TRAVÉS DEL PROGRAMA C.I.D.

Jorge Jiménez
Dept. Matemáticas
Universidad de Oviedo
e-mail: meana@etsiig.uniovi.es

Susana Montes
Dept. Estadística
Universidad de Oviedo
e-mail: smr@pinon.ccu.uniovi.es

Mercedes Díez
Dept. Matemáticas
Universidad de Oviedo

Resumen

El propósito de este trabajo era el de realizar un programa de manejo sencillo que realizase las principales operaciones algebraicas entre conjuntos difusos finitos: unión, intersección, complementario, permitiendo su visualización gráfica. Además se requirió que calculase alfa-cortes, tanto débiles como fuertes, y especialmente la probabilidad, la borrosidad, según algunas de las definiciones más usuales y la incertidumbre de cualquier suceso difuso.

Palabras Clave: Conjuntos difusos, probabilidad, medida de borrosidad y medida de incertidumbre.

1 INTRODUCCIÓN

Una parte del grupo de investigación de la Universidad de Oviedo que trabaja en temas relacionados con la teoría de los conjuntos difusos, estaba y está especialmente interesada en el desarrollo de una teoría de la incertidumbre tanto difusa como probabilística para sucesos difusos. En las investigaciones realizadas hasta el momento, aunque principalmente teóricas, se buscaban en muchas ocasiones contraejemplos que probasen que un resultado era falso, o se realizaba el experimento un número considerable de veces con el fin de poder empezar a pensar que tal resultado era cierto. Esto suponía una gran cantidad de cálculos que eran realizados "manualmente", evidentemente no se hacían con lápiz y papel, pero como durante mucho tiempo no se disponía de un paquete especializado para el trabajo con difusos, lo que se hacía era realizar un pequeño programa en alguno de los lenguajes de programación conocido por los investigadores: Basic, Fortran, C, ... que era necesario modificar en cada caso.

Al empezar a aparecer programas con paquetes específicos para difusos, por ejemplo Mathematica y Matlab, en muchas ocasiones tampoco recogían alguno de los conceptos necesarios. Es por ésto que empieza a surgir la idea de realizar un programa que cumpliese las siguientes exigencias:
(1) De manejo sencillo, y ésta era una de las cualidades principales que se le exigía, porque si era necesario aprender muchos comandos o algún tipo de lenguaje, no merecería la pena el esfuerzo, y tendría más cuenta seguir con las herramientas de las que se disponía hasta el momento.
(2) Que se adaptase exactamente a las necesidades que existen en este momento, cabiendo la posibilidad de ampliarlo si se necesitaba algún otro tipo de cómputo en posteriores investigaciones.

El primer punto llevaba a pensar en la realización del programa en algún lenguaje que programase bajo Windows, para así poder aprovechar todas las ventajas de la filosofía de este tipo de programación, como por ejemplo la multitarea, aparte de que este tipo de aplicaciones tiene un interface mucho más "amigable" tanto para el programador como para el usuario. El segundo punto llevó a la imposición de realizar este software en un lenguaje de programación bastante universal y entre los numerosos que verificaban los requisitos del punto anterior: Borland C++, Delphi, Visual Basic, etc., se dejó la decisión en manos del programador que consideró conveniente el uso del Visual Basic.

Con el fin de cumplir todas estas exigencias surgió el programa C.I.D., cuyas posibilidades y funcionamiento serán explicados en el resto de las secciones.

2 EMPEZANDO CON C.I.D.

El programa C.I.D. consta de un disquete que contiene un programa de instalación ya que los archivos están comprimidos, así como el ejecutable. Antes de instalar

la aplicación es necesario asegurarse de que el equipo cumple los requisitos mínimos para el funcionamientos de la misma; dichos requisitos mínimos son: Mp 386, 4 MB. de memoria RAM, un ratón (que aunque no es imprescindible, si que es muy recomendable) y Windows versión 3.1.

Si el sistema cumple los requerimientos anteriores se puede comenzar con la instalación, la cual debe realizarse desde "Windows", ejecutando el fichero instalar.exe, que copia en el disco duro los ficheros necesarios para el perfecto funcionamiento del programa, además de incluir en pantalla un icono que permitirá hacer uso del programa simplemente con su selección.

Una vez instalado, aparece (durante el tiempo que tarda en cargar el programa, tiempo que evidentemente está directamente relacionado con el tipo de máquina con la que se esté trabajando) una pantalla de presentación del programa como la que se reproduce a continuación:

Figura 1: Pantalla de presentación.

pantalla que da paso a la principal del programa que consta de las siguientes opciones:

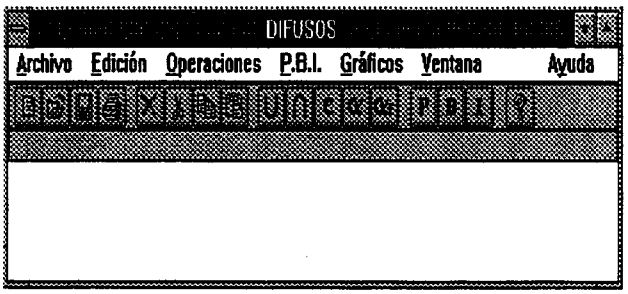


Figura 2: Pantalla principal.

La base del programa es esta pantalla de menús, con una barra de herramientas y una barra de estado. Su manejo es idéntico a cualquier programa de este tipo que se ejecute bajo "Windows". Mediante el uso del ratón o teclas predefinidas se va accediendo a las diferentes operaciones de los menús y los submenús.

Para activar cualquiera de los menús el método es el habitual, sólo hay que hacer clic con el botón izquierdo del ratón sobre su título, presionar la combinación de teclas ALT + letra subrayada de la opción deseada, o bien situarse sobre la barra de menús con ALT y desplazarse por ella mediante los cursores. Una vez activado el menú, éste se despliega mostrando las opciones que lo componen.

Esta pantalla consta de varios elementos: menú, barra de herramientas y barra de estado.

En el menú se visualiza cada una de los temas que el usuario puede tratar, los cuales están constituidos a su vez de varios submenús.

En la barra de herramientas aparecen una serie de botones que al ser pinchados con el ratón van a provocar el desencadenamiento de varias tareas (Guardar, Imprimir, Unión, etc), tareas que el usuario puede visualizar en la barra de estado situada en la parte superior de la pantalla y que le orientan sobre el efecto que puede producir el pinchar el botón sobre el que se sitúa la flecha del ratón en cada momento.

A continuación se va a hacer un breve repaso de cada uno de estos menús:
(1) ARCHIVO: la principal utilidad de esta opción es la de permitir la introducción de nuevos conjuntos difusos para su posterior estudio, así como abrir otros conjuntos ya utilizados anteriormente. Además de ésto, es esta opción la que permite abrir fichero ya guardados con datos y todas aquellas operaciones que en tal fichero se hayan realizado, guardar los ficheros actuales, imprimirlos y salir del programa.
(2) EDICIÓN: como en la mayoría de los programas, esta opción permite borrar, cortar, copiar y pegar cualquier texto del fichero con el que se esté trabajando. También permite buscar cualquier palabra en el fichero. Por otro lado se ha incluido aquí la posibilidad de modificar la definición, es decir, los valores de su función de pertenencia, de un conjunto difuso cualquiera. Además de la función de pertenencia, el programa permite modificar los valores de la probabilidad asignados a cada uno de los elementos del referencial.
(3) OPERACIONES: permite el cálculo de la unión, la intersección, y el complementario de cualquier subconjunto difuso de un referencial dado, así como el cálculo de cualquier α-corte, tanto débil como fuerte, para cualquier valor de α.
(4) P.B.I.: esta sección será desplegada si se desea calcular la probabilidad de un conjunto cualquiera, o bien una de las posibles medidas de borrosidad, o de incertidumbre que este programa lleva implementadas. Esta herramienta será posteriormente estudiada con más detalle.
(5) GRÁFICOS: permite mostrar la representación gráfica de cualquiera de los conjuntos mediante diagramas de barras, guardarla o imprimirla.

(6) VENTANA: contiene los comandos de manejo de las distintas subventanas que se tengan abiertas, permitiendo que éstas se coloquen horizontalmente, verticalmente, etc; además permite organizar los iconos si las ventanas están minimizadas, e incluso cerrarlas todas.
(7) AYUDA: permite acceder de forma rápida y sencilla a la comprensión de los principales términos del programa mediante el Índice, y el Buscar ayuda sobre. También ofrece ayuda sobre el manejo de la propia ayuda en Uso de la ayuda y sobre el manejo del propio programa en el Tutorial. Evidentemente también consta de un Acerca de, que ofrece información acerca del programa.

A continuación se van a explicar más detalladamente las principales opciones que permite, acompañándolas de un pequeño desarrollo teórico.

3 INTRODUCCIÓN DE LOS DATOS

Un nuevo conjunto difuso se puede introducir en un fichero (evidentemente dentro de un mismo fichero siempre se está trabajando con el mismo referencial, es decir, con el mismo cardinal) de dos formas, o bien directamente, accionando la opción de Nuevo dentro de Archivo, para introducir un nuevo conjunto difuso, cuyos valores de la función de pertenencia serán introducidos mediante una ventana como la que sigue

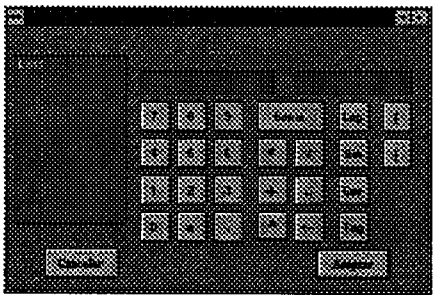


Figura 3: Fichero de salidas.

Como se puede observar esta pantalla se asemeja mucho a una calculadora y su funcionamiento va a ser también muy similar. Debido a que los valores de la función de pertenencia se pueden dar como una expresión matemática que lleva funciones, paréntesis, operaciones, etc, los valores serán el resultado de dicha expresión matemática. En cada momento la pantalla de resultado muestra el operando introducido o los resultados parciales acumulados. Si a continuación la expresión sigue con otro operador o bien a este resultado se le aplica una función, la pantalla mostrará el resultado.

Una vez que los valores son introducidos correctamente y se ha pulsado el botón Aceptar aparecerá una ventana con el título de Nuevo1.dif, éste es en realidad un nuevo documento en el que se irán almacenando los sucesivos estudios que se hagan sobre el conjunto introducido, o sobre aquellos otros que se vayan definiendo dentro del mismo referencial.

La forma inicial que presentaría dicho documento sería:

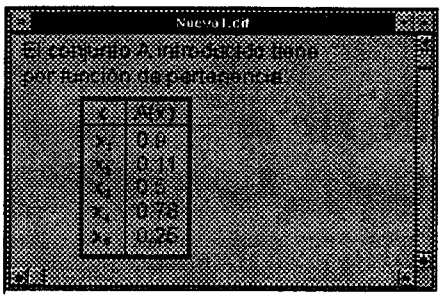


Figura 4: Fichero de salidas.

Otra forma de introducir un conjunto sería cogiendo uno definido en un fichero con el que se ha trabajado en otra sesión. Primero se elige la opción de abrir, se selecciona el fichero del que se desea extraer el conjunto (esta ventana es totalmente análoga a la de cualquier programa que funcione bajo Windows, así que no se va a presentar aquí la figura, debido a las limitaciones de espacio que existen). Una vez elegido el fichero, de entre todos los conjuntos en el definido, se puede elegir aquellos que se desee. seleccionándolos como sigue

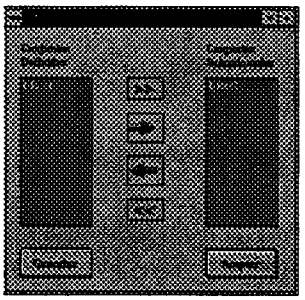


Figura 5: Selección de los conjuntos.

Para todo conjunto que haya sido definido, puede visualizarse, si así se desea, su representación gráfica seleccionando la opción de Gráficos. Para el ejemplo de la figura 4, se obtendría una ventana como la que se muestra en la figura 6.

Este tipo de representaciones gráficas pueden realizarse también para cualquiera de los conjuntos resultantes de alguna de las operaciones que se relatan a continuación.

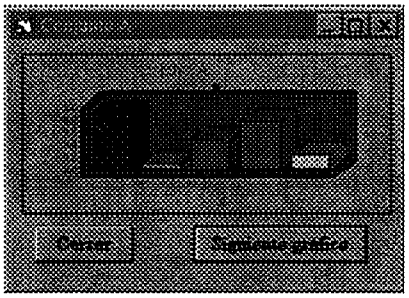


Figura 6: Representación gráfica del conjunto.

4 OPERACIONES BÁSICAS

Este submenú contiene las opciones que permiten el cálculo de las principales operaciones sobre subconjuntos difusos como son la unión, la intersección y el complementario, así como el cálculo de los subconjuntos nítidos asociados al difuso seleccionado, es decir, de los α-cortes, fuertes ó débiles.

Para la realización de cualquiera de estas operaciones es necesario en primer lugar seleccionar el conjunto o conjuntos con los que se desea trabajar, mediante una ventana similar a la mostrada en la figura 5.

Las definiciones que se han manejado para realizar las tres primeras operaciones fueron las iniciales propuestas por Zadeh ([14]), por ser las que habitualmente se usaban en las investigaciones, puesto que verifican las leyes de Morgan. Por lo tanto se consideran los conjuntos difusos unión, intersección y complementario definidos por

$$\begin{aligned}(\widetilde{A}\cup\widetilde{B})(x) &= \max\{\widetilde{A}(x),\widetilde{B}(x)\} \quad \forall\widetilde{A},\widetilde{B}\subset X\\ (\widetilde{A}\cap\widetilde{B})(x) &= \min\{\widetilde{A}(x),\widetilde{B}(x)\} \quad \forall\widetilde{A},\widetilde{B}\subset X\\ (\widetilde{A}^c)(x) &= 1-\widetilde{A}(x) \quad \forall\widetilde{A}\subset X\end{aligned} \tag{1}$$

para cualquier elemento x de X.

Evidentemente los α-cortes, son ([2]) los conjuntos nítidos formados por los puntos de X que pertenecen en grado mayor o igual (o simplemente mayor, si se trabaja con el α-corte fuerte) que α, por lo tanto, para un subconjunto difuso $\widetilde{A}$ cualquiera, su α-corte, también llamado α-corte débil, estará formado por

$$\widetilde{A}_\alpha = \{x\in X/\mu_{\widetilde{A}}(x)\geq\alpha\} \tag{2}$$

y su α-corte fuerte será el conjunto

$$\widetilde{A}_{\overline{\alpha}} = \{x\in X/\mu_{\widetilde{A}}(x)>\alpha\} \tag{3}$$

Una vez realizada una cualquiera de las operaciones anteriores, el resultado obtenido se visualizará en pantalla, al ser añadido al fichero de salidas, y cualquiera de estos conjuntos podrá ser reutilizado como si hubiese sido introducido directamente, permitiendo hacer con él cualquiera de las distintas operaciones que el programa lleva implementadas.

5 MEDIDAS DE PROBABILIDAD, BORROSIDAD E INCERTIDUMBRE

En este submenú se puede calcular la probabilidad de que ocurra cualquier subconjunto difuso, así como medir la incertidumbre que existe en él, asociada al desconocimiento de qué puntos lo forman realmente (borrosidad) o bien asociada al desconocimiento de qué elemento dentro de él ocurriría en un experimento aleatorio (incertidumbre probabilística).

Incluye pues las opciones para poder calcular la probabilidad, borrosidad o incertidumbre de los conjuntos difusos que se seleccionen, siempre bajo una pantalla como la representada en la figura 5. A continuación se van a analizar un poco más en detalle estas opciones y las definiciones que en ellas han sido manejadas.

5.1 PROBABILIDAD

Esta opción permite calcular la probabilidad de uno cualquiera de los conjuntos que son manejados en la sesión, a través de la definición que Zadeh [15] dió de probabilidad de un suceso difuso

$$p(\widetilde{A}) = \int_X \widetilde{A}(x)dP = E(\mu_{\widetilde{A}}) \tag{4}$$

que en el caso de un referencial nítido tomaría la forma

$$p(\widetilde{A}) = \sum_{x\in X}\widetilde{A}(x)p(x) \tag{5}$$

que es precisamente la fórmula empleada para su evaluación en este programa.

A la vista de la ecuación anterior 5, parece evidente como será necesario conocer la probabilidad de ocurrencia de cada uno de los puntos del referencial, que el programa pedirá, por tanto, en caso de que no hubiese sido introducidos con anterioridad.

5.2 BORROSIDAD

Calcula la borrosidad de un conjunto difuso según algunas de las más conocidas definiciones. No son evidentemente las únicas ([3], [6], [7], [11], [13],...), pero si eran las más utilizadas en la actualidad por el grupo que requirió este programa, y su selección se hace mediante una ventana como la que sigue en la que el usuario elige al tipo de medida con la que desea evaluar la borrosidad del conjunto o conjuntos que previamente ha elegido.

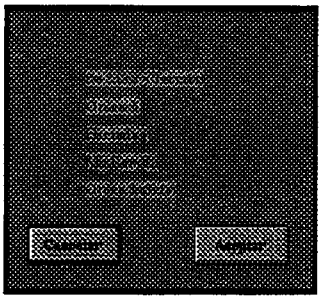


Figura 7: Tipos de medidas de borrosidad.

Las ecuaciones empleadas para estos cálculos han sido:
Medida de borrosidad de De Luca y Termini:[8]

$$f(\widetilde{A}) = -\sum_{x \in X} g(\widetilde{A}(x)) \quad \forall \widetilde{A} \subset X \tag{6}$$

siendo g la función real definida por $g(x) = -(x\log(x) + (1-x)\log(1-x))$.
Medida de borrosidad de orden s:[1]

$$f_s(\widetilde{A}) = \frac{1}{2^{1-s}-1} \sum_{x \in X} [((\widetilde{A}(x))^s - \widetilde{A}(x)) + ((\widetilde{A}^c(x))^s - \widetilde{A}^c(x))] \tag{7}$$

con $s > 0$ y $s \neq 1$.
Medida de borrosidad de Hamming:[5]

$$f(\widetilde{A}) = \sum_{x \in X} |\widetilde{A}(x) - Z(x)| \tag{8}$$

siendo Z el nítido más cercano a $\widetilde{A}$, es decir,

$$\begin{aligned} Z(x) &= 0 \text{ si } \widetilde{A}(x) < 1/2 \\ Z(x) &= 1 \text{ si } \widetilde{A}(x) > 1/2 \end{aligned} \tag{9}$$

Medida de borrosidad de Minkowski:[5]

$$f_w(\widetilde{A}) = \left(\sum_{x \in X} |\widetilde{A}(x) - Z(x)|^w \right)^{1/w} \tag{10}$$

siendo Z el nítido más cercano a $\widetilde{A}$ definido anteriormente y $w \in [1, \infty)$.
Medida de borrosidad de orden infinito:[5]

$$f_{infty}(\widetilde{A}) = \max_{x \in X} |\widetilde{A}(x) - Z(x)| \tag{11}$$

siendo Z el subconjunto nítido definido anteriormente.

Evidentemente, para la clase de orden s, o para la clase de Minkowski será necesario especificar el orden que se desea.

De nuevo los datos obtenidos con presentados en la pantalla de salidas, y podrán ser guardados para su visualización posterior.

5.3 INCERTIDUMBRE

El funcionamiento de esta sección es totalmente análogo al anterior. Aquí se calcula la incertidumbre probabilística ([4]) de un conjunto difuso según algunas de las principales definiciones que este grupo de investigación está manejando actualmente. Si se selecciona esta opción se obtiene en pantalla: Como se ob-

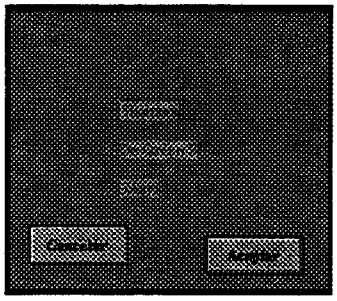


Figura 8: Tipos de medidas de incertidumbre.

serva, permite obtener la incertidumbre a partir de extensión de las definiciones dadas de este concepto para sucesos nítidos por Shannon, Havrda-Charvt y Rényi, que en el caso difuso tomarían la forma siguiente:
Medida de incertidumbre de Shannon:[10]

$$H(\widetilde{A}) = \frac{-\sum_{x \in X} \widetilde{A}(x)P(x)log(\widetilde{A}(x)P(x))}{P(\widetilde{A})} \tag{12}$$

Medida de incertidumbre de orden s de Havrda-Charvt:[12]

$$H_s(\widetilde{A}) = \frac{1}{2^{1-s}-1} \frac{\sum_{x \in X} ((\widetilde{A}(x)P(x))^s - \widetilde{A}(x)P(x))}{P(\widetilde{A})} \tag{13}$$

con $s > 0$ y $s \neq 1$.
Medida de incertidumbre de Rényi:[9]

$$f(\widetilde{A}) = \frac{1}{1-r} \log \left(\frac{\sum_{x \in X} (\widetilde{A}(x)p(x))^r}{p(\widetilde{A})} \right) \tag{14}$$

con $r > 0$ y $r \neq 1$.

6 LA AYUDA DE C.I.D.

Se consideró fundamental la inclusión de una ayuda de manejo sencillo, pero que permitiese solucionar los principales problemas que pudiese encontrarse el usuario.

La ayuda sigue el funcionamiento general de todas las ayudas de "Windows". Al leerla, el usuario se encontrará palabras de color verde subrayadas o sin subrayar, al pasar la flecha del ratón sobre ellas se cambia

la flecha por una mano, esto indica que hay más ayuda referente a ese término que puede ser leída sin más que pinchar sobre la palabra en cuestión.

Consta de un "índice" que presenta todos con conceptos ordenados alfabéticamente y un "buscar" con el funcionamiento de esta herramienta, que permite teclear la palabra sobre la que se desea encontrar ayuda.

Por otro lado esta ayuda también consta de un tutorial que hace un breve recorrido por el programa explicando su manejo y funcionamiento; y una "Ayuda" acerca del uso de la ayuda.

7 CONCLUSIONES Y PUNTOS ABIERTOS

Como conclusión principal recalcar el hecho, que evidentemente se hace palpable a lo largo del trabajo, de que este software ha sido concebido especialmente para subsanar un vacío que existía en las investigaciones de un reducido número de personas, y entre sus pretensiones principales estaba especialmente el fácil acomodo a los requisitos de éstos, más que el conseguir un software comercial. A pesar de ésto, no se descarta la idea de proseguir mejorándolo, añadiéndole otras herramientas que puedan resultar interesantes. Entre los objetivos más inmediatos está el de realizar el tutorial interactivo, cuestión que de momento, por falta de tiempo, aún no ha sido solucionada.

Agradecimientos

La investigación de este trabajo está subvencionada en parte a través del Proyecto DGICYT PB94-1328. Queremos agradecer al Ministerio de Educación y Ciencia dicha ayuda económica.

Referencias

[1] I. Couso, P. Gil and S. Montes (1996). Measures of fuzziness and Information Theory. *IPMU'96* **1**:501-505.

[2] D. Dubois and H. Prade (1980). Fuzzy Sets and Systems. Theory and Applications. *Academic Press, Inc.*

[3] Emptoz (1981). Nonprobabilistic entropies and indetermination measures in the setting of fuzzy sets theory. *Fuzzy Sets and Systems* **5**:307-317.

[4] P. Gil (1981). Teoría matemática de la información. *I.C.E. Madrid.*

[5] G.J. Klir and T.A. Folger (1988). Fuzzy Sets, Uncertainty, and Information. *Prentice-Hall International (UK) Limited*

[6] J. Knopfmacher (1975). On measures of fuzziness. *J. Math. Analysis and Applications* **49**:529-534.

[7] S.G. Loo (1977). Measures of fuzziness. *Cybernetica* **20**:201-210.

[8] A. De Luca and S. Termini (1972). A definition of a Nonprobabilistic Entropy in the Setting of Fuzzy Sets Theory. *Inf. and Control* **20**:301-312.

[9] A. Rényi (1961). On measure of Entropy and Information. *Proc. 4th. Symp. Math. Stat. Prob. Berkeley* **1**:547-561.

[10] C.E. Shannon (1948). A mathematical Theory of Communication. *Bell. Syst. Tech. J.* **27**:379-423,623-656.

[11] E. Trillas and T. Riera (1978). Entropies in Finite Fuzzy Sets. *Inf. Sc.* **15**:159-168.

[12] I. Vadja (1968). Axioms of α-entropy of a generalized probability scheme. *Kybernetica* **4**:105-112.

[13] R.R. Yager (1995). Measures of Entropy and Fuzziness Related to Agregation Operators. *Information and Control* **82**:147-166.

[14] L.A. Zadeh (1965). Fuzzy sets. *Information and Control* **8**:338-353.

[15] L.A. Zadeh (1968). Probability measures of fuzzy events. *J. of Math. Analysis and Applications* **23**:421-427.

XFUZZY 2.0: ENTORNO DE DISEÑO PARA SISTEMAS BASADOS EN LÓGICA DIFUSA

D. López[*], S. Sánchez-Solano[**], A. Barriga[**]

[*] Centro Informático Científico de Andalucía
[**] Instituto de Microelectrónica de Sevilla - CNM
Edificio CICA, Avda. Reina Mercedes s/n, 41012-Sevilla.
email: drlopez@cica.es

Abstract

Xfuzzy es una herramienta de CAD para el desarrollo de sistemas basados en lógica difusa. Está compuesto por un conjunto de módulos o programas que comparten un lenguaje de especificación común y facilitan las diferentes etapas del proceso de diseño: descripción, verificación y síntesis.

1 Flujo de diseño con Xfuzzy

La figura 1 muestra el flujo general de diseño de un sistema difuso utilizando Xfuzzy. La descripción del sistema se realiza mediante el lenguaje de especificación XFL (*Xfuzzy Language*). Una especificación XFL contiene información sobre la base de conocimiento (funciones de pertenencia de antecedentes y consecuentes y estructura de la base de reglas) y sobre el mecanismo de inferencia utilizado (conectivas, función de implicación y método de defuzzificación). La introducción de una especificación puede hacerse mediante un editor de texto o con ayuda de las facilidades gráficas que proporciona Xfuzzy. La descripción XFL del sistema difuso constituye el punto de partida de los diferentes módulos del entorno.

La verificación del sistema difuso se lleva a cabo mediante **Xfsim** (*Fuzzy Systems Simulator*). Este módulo combina la implementación C generada por **Xfc** (*XFL to C translator*) con otra serie de programas que introduce el usuario para comprobar la funcionalidad del sistema en cadena abierta o en lazo cerrado.

El comportamiento del sistema difuso puede refinarse con ayuda de **Xfbpa** (*Xfuzzy Backpropagation Algorithms*), un módulo de aprendizaje supervisado basado en diferentes algoritmos de retropropagación de errores. Xfbpa es capaz de manejar todos los tipos de datos de funciones de pertenencia admitidos por XFL, así como un gran número de operadores difusos y métodos de defuzzificación. El cálculo de las derivadas parciales utilizadas para modificar los parámetros en el proceso de aprendizaje se realiza en tiempo de ejecución del programa.

Es sin embargo en los aspectos de síntesis donde Xfuzzy se diferencia más de otras herramientas fuzzy disponibles. Además de contemplar la síntesis software del motor de inferencia difuso mediante un programa C (Xfc), Xfuzzy incluye dos módulos que facilitan la síntesis hardware de los sistemas difusos descritos en XFL.

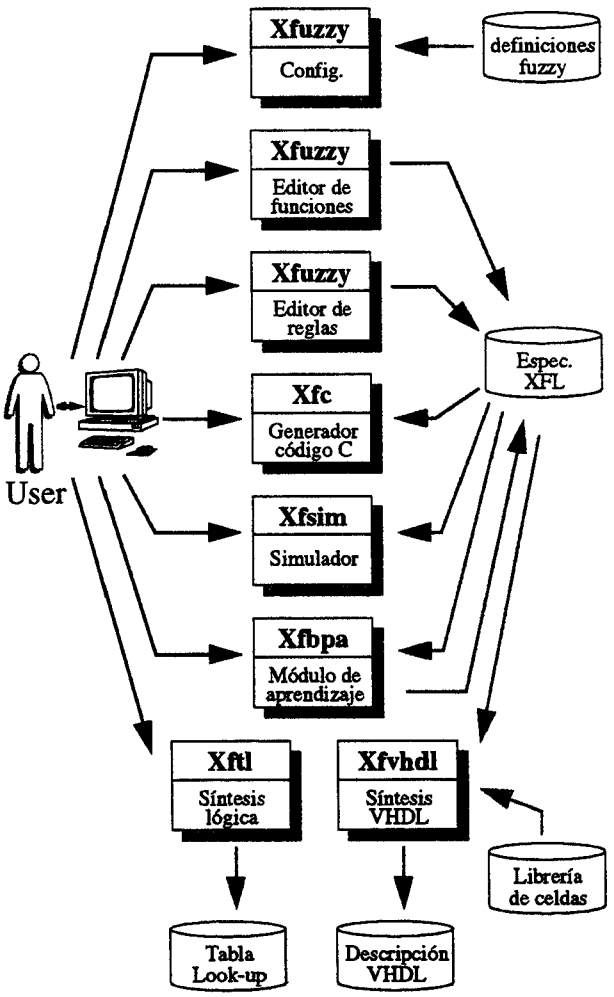


Fig. 1: Flujo de diseño con Xfuzzy 2.0

Xftl (*XFL to Table Look-Up*) traslada la descripción XFL en una tabla entrada/salida equivalente al sistema difuso. Dicha tabla puede ser implementada en una memoria RAM, como bloques combinacionales, o mediante FPGAs. En este último caso Xftl proporciona un fichero de comandos que dirige el proceso de síntesis con las herramientas de Synopsys y Xilinx.

Como una segunda alternativa, **Xfvhdl** (*XFL to VHDL*) traslada la especificación XFL en una descripción VHDL de acuerdo con una determinada arquitectura de implementación de sistemas difusos mediante hardware dedicado. El código VHDL puede ser sintetizado como un circuito integrado de aplicación específica (ASIC) o como una FPGA. Xfvhdl también proporciona un fichero de comandos para las herramientas de síntesis anteriores.

2 Características de Xfuzzy

Xfuzzy ha sido codificado en C sobre sistema operativo Unix. Su interfaz gráfica de usuarios utiliza el entorno de ventanas X-windows. Los módulos de simulación, aprendizaje y síntesis pueden funcionar aisladamente, invocándolos desde la línea de comandos, o conjuntamente a partir de la ventana principal de la aplicación (figura 2a). Xfuzzy puede instalarse en cualquier sistema operativo tipo Unix que disponga de compilador de C y X-windows. La versión β del kit de distribución proporciona *Makefiles* para SunOs, Solaris y Linux.

3 Aplicaciones de Xfuzzy

Las definiciones de los diferentes operadores difusos (conectivas, funciones de implicación, métodos de defuzzificación, etc.) usados por una especificación XFL pueden ser modificados por el usuario (figura 2b). De esta forma las herramientas de descripción y simulación de Xfuzzy proporcionan un banco de pruebas ideal para analizar y comparar nuevos formalismos difusos.

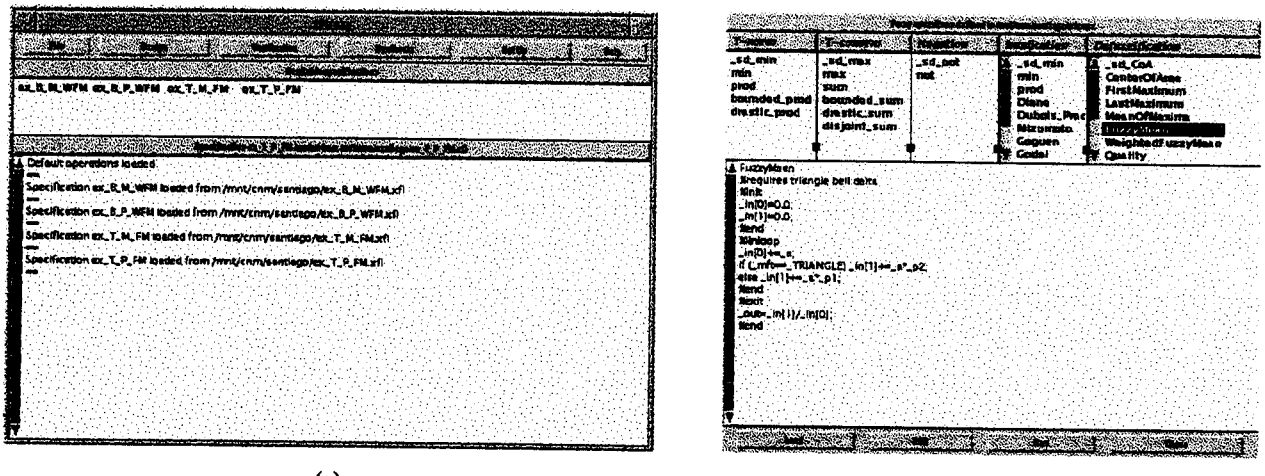
El módulo de aprendizaje permite seleccionar qué funciones de pertenencia deben ser ajustadas, combinar diferentes algoritmos de retropropagación de errores y definir distintas estrategias de actualización de parámetros. Todo ello lo convierte en una herramienta de gran valor didáctico.

Por último, las herramientas de síntesis permiten pasar rápidamente desde una especificación XFL a su implementación hardware, proporcionando un mecanismo valioso para explorar el espacio de diseño de sistemas de inferencia difusa siguiendo diferentes técnicas de implementación.

4 Futuros trabajos

Xfuzzy 2.0 ha sido diseñado como un entorno abierto al que se pueden incorporar nuevas facilidades. Las líneas de progresión que estamos considerando actualmente son:

- Desarrollo de un editor gráfico que facilite la descripción de sistemas con estructura jerárquica.
- Inclusión de métodos de aprendizaje basados en algoritmos genéticos.
- Incorporación de herramientas de síntesis para implementaciones basadas en técnicas analógicas.



(a) (b)

Fig. 2: a) Ventana principal de Xfuzzy. b) Definición de operadores difusos.