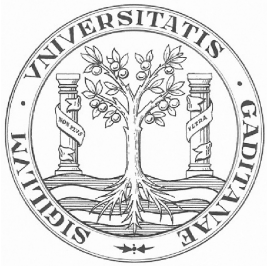


UNIVERSIDAD DE CÁDIZ

FACULTAD DE CIENCIAS

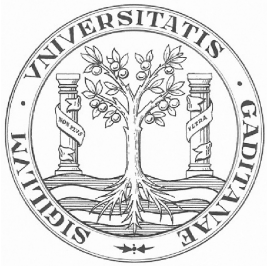
GRADO EN MATEMÁTICAS

RELACIÓN ENTRE LOS MÉTODOS DE INFERENCIA DIFUSA Y LA PROGRAMACIÓN LÓGICA MULTIADJUNTA

Francisco Alonso Fernández

Tutora: Dra. M. Eugenia Cornejo Piñero

Tutora: Dra. Eloísa Ramírez Poussa



UNIVERSIDAD DE CÁDIZ

FACULTAD DE CIENCIAS

GRADO EN MATEMÁTICAS

RELACIÓN ENTRE LOS MÉTODOS DE INFERENCIA DIFUSA Y LA PROGRAMACIÓN LÓGICA MULTIADJUNTA

Francisco Alonso Fernández

Tutora: Dra. M. Eugenia Cornejo Piñero

Tutor: Dra. Eloísa Ramírez Poussa

Firma del alumno | Firma de la tutora | Firma de la tutora

Puerto Real, Cádiz, septiembre de 2019

Abstract

In the following dissertation we carry out a theorical and practical study, about fuzzy inference systems and multiadjoint logic programming. We will end up comparing both frameworks.

Firstly, Chapter 1 introduces the historical context that encompasses both fuzzy logic, in general, and Mamdani's inferences and multi-joint logic programming, in particular. In addition, a brief summary of the contents of the work is presented.

Afterwards, Chapter 2 presents basic concepts about fuzzy logic whose are neccesary for a better understanding of the following chapters.

In Chapter 3 rule-based systems are studied in a theoretical way. After that, our efforts will be focused towards Mamdani and Sugeno inference systems. Chapter 3 with an illustrative example about both Sugeno and Mamdani's methods.

Chapter 4 presents multi-adjoint logic programming, being our main task is to prove results about monotonicity and continuity of the immediate consequences operator. Those results will allow us to work with a practical application in this context.

To finish this work, Chapter 5 establishes a comparative study where the similarities and differences found between Mamdani's inference method and multiadjoint logic programming are shown.

A mi familia.

Resumen

En el presente trabajo se hará un estudio tanto teórico como práctico de los sistemas de inferencia difusos así como de la programación lógica multiadjunta, para terminar haciendo una comparativa de ambas teorías.

En primer lugar, en el Capítulo 1 se introduce el contexto histórico que envuelve tanto a la lógica difusa en general como a las inferencias de Mamdani y la programación lógica multiadjunta en concreto. Además se presenta una pequeña descripción de los contenidos del trabajo.

A continuación, en el Capítulo 2 se exponen conceptos básicos sobre lógica difusa que son necesarios para un buen entendimiento de los capítulos posteriores.

En el Capítulo 3 se estudian los sistemas basados en reglas de manera teórica para después particularizar en las inferencias de Mamdani y Sugeno, terminando con un ejemplo ilustrativo utilizando ambos métodos.

Pasamos al Capítulo 4 donde se expone una introducción a la programación lógica multiadjunta con el objetivo de dar resultados sobre la monotonía y continuidad del operador de consecuencias que nos permitirán una ver una aplicación práctica de esta teoría.

Para terminar, en el Capítulo 5 se establece una comparativa donde se exponen las similitudes y diferencias encontradas entre el método de inferencias de Mamdani y la programación lógica multiadjunta.

Agradecimientos

A mis padres, Francisco y Carmen por apoyarme en cada paso durante estos años y saberme poner los pies en la tierra cuando ha sido necesario. A mi hermana Carmen, por aportar esa pizca de sensatez que tantas veces me hace falta y aconsejarme siempre con tan buen criterio.

A mis amigos del pueblo, de la universidad y del fantástico año de Erasmus, porque con ellos he disfrutado esta etapa universitaria de una manera increíble.

También a todos los docentes que han motivado mi interés por las matemáticas, mención especial a Nieves que hizo un trabajo excepcional inculcando las matemáticas a aquella generación del instituto. Así como a mis tutoras Eloísa y María Eugenia por su disposición y por aceptar tutorizar este trabajo incluso desde la distancia.

septiembre 2019

Índice general

CAPÍTULO

1

Introducción

El raciocinio es la principal diferencia entre los seres humanos y el resto de los seres vivos. Los seres humanos somos capaces de obtener conclusiones a partir de una serie de hipótesis. Por ejemplo, si suponemos cierta la hipótesis "la camiseta es verde" y falsa "las matemáticas no son útiles", podemos deducir, entre otras cosas, que la afirmación "la camiseta es verde y las matemáticas no son útiles" es falsa o que "la camiseta es verde o las matemáticas no son útiles" es verdadera. En este tipo de razonamientos estamos haciendo uso de operadores lógicos. Dichos operadores juegan un papel fundamental en la denominada lógica clásica. Esta lógica es muy útil para trabajar con enunciados que son o bien ciertos, o bien falsos.

Pero, si nos paramos a pensar en el lenguaje cotidiano de los seres humanos, vemos que trabajamos muy a menudo con expresiones que no son totalmente ciertas ni totalmente falsas. Por ejemplo, si consideramos un vaso, sabremos que está vacío cuando no tiene ninguna gota de agua en su interior, y a su vez, que está lleno cuando no quepa ninguna gota de agua más. Pero, ¿cómo está el vaso si solo tiene una gota de agua en su interior?, ¿y si le faltan dos gotas para rebosar? De este tipo de afirmaciones ambiguas ya dio cuenta Aristóteles en la antigua Grecia, pero no fue hasta 1965 cuando Lofti Zadeh, considerado el padre de la lógica difusa, formalizó esta extensión de la lógica clásica con un artículo titulado "*Fuzzy sets*" (22). Zadeh, natural de Azerbaiyán y posteriormente profesor en la Universidad de California presentó en este artículo unos conjuntos sin límites precisos."*La*

lógica difusa es una lógica precisa de la imprecisión" (Lofti Zadeh). Posteriormente, otros matemáticos como Boole o De Morgan han realizado contribuciones a este campo de investigación, que hoy en día es objeto de estudio de muchos científcos.

El alcance de altos niveles de precisión tiene un coste importante de tiempo, dinero o incluso ambas. Parafraseando de nuevo a Zadeh, "*tenemos que explotar nuestra tolerancia a la imprecisión*", y en la mayoría de los problemas que tratamos, si hacemos uso de lógica difusa podemos hacer una importante optimización de costes. Es por eso que, en este trabajo nos centraremos en el estudio de los fundamentos matemáticos subyacentes en la lógica difusa y ver sus aplicaciones como herramienta para automatizar procesos de inferencia.

Una de dichas herramientas que trataremos será el control difuso. En concreto haremos un estudio teórico-práctico de los sistemas de inferencias de Mamdani, así como los de Sugeno. Assilian Mamdani en 1974 desarrolló del primer controlador difuso para una máquina de vapor. Este hecho constituyó un hito remarcable en el desarrollo de la lógica difusa, pero la primera implantación real de un controlador de este tipo fue realizada en 1983 por F.L. Smidth & Co, en una planta cementera de Dinamarca. Durante la década de los ochenta en paralelo al desarrollo de aplicaciones, como controladores en plantas cementeras o trenes, se sigue realizando investigación teórica. Es así como Takagi y Sugeno desarrollan la primera aproximación para construir reglas difusas a partir de datos empíricos.

Otra herramienta que se estudiará en este trabajo es la programación lógica multiadjunta, la cual es un marco lógico general introducido recientemente y que está recibiendo una atención considerable. El marco multiadjunto se originó como una generalización de diversos marcos de programación lógica no clásicos. La idea principal es la de abstraer los detalles particulares de cada paradigma y centrarse solamente en los mínimos elementos matemáticos que posibiliten su ejecutabilidad. Fue propuesta originalmente en (12) y desde entonces se han llevado a cabo numerosos avances como (3, 4, 5, 6) entre otros. Se introdujo como aplicación al trabajo de Robinson (16), quien introdujo la regla de resolución, una regla de inferencia especialmente válida para ser implementada en un ordenador. Desde entonces ha habido numerosos estudios en este ámbito, podemos destacar a Robert A. Kowalski con su formulación del lenguaje de programación (9) o Keith L. Clark con su investigación sobre la relación entre la negación y el fracaso finito (1).

Por último en el presente trabajo estudiaremos la relación existente entre la programación lógica multiadjunta y los sistemas de inferencia de Mamdani, analizando similitudes

y diferencias.

Esta memoria queda organizada de la siguiente manera. En el Capítulo 2 daremos nociones básicas sobre la lógica difusa que se basarán en conjuntos difusos, funciones de pertenencia y conectivos lógicos.

A continuación, en el Capítulo 3 empezamos estudiando los sistemas basados en reglas de manera genérica, de su formación y sus propiedades. Esto nos da el *background* necesario para tratar en concreto los sistemas de inferencia de Mamdani y Sugeno. Veremos como actúa cada uno de ellos, así como sus diferencias, incluyendo un ejemplo práctico de cada sistema.

En el Capítulo 4 pasamos a estudiar la programación lógica multiadjunta. Al ser un campo de estudio nuevo, partimos desde la base dando definiciones básicas sobre conjuntos ordenados, retículos y puntos fijos. Tras ello, se trata la sintaxis y semántica proposicional. Terminamos el preludio teórico con los conectivos lógicos generalizados. Una vez dada la base teórica tenemos el contexto suficiente para poder introducir el operador de consecuencias, en concreto se darán resultados sobre su monotonía y continuidad. Finalmente, gracias a estas condiciones obtenidas en el operador de consecuencias, llevaremos a cabo un ejemplo práctico sobre el funcionamiento de un motor.

Para terminar el trabajo, en el Capítulo 5, tratamos de establecer una comparativa entre el sistema de inferencia de Mamdani y la programación lógica multiadjunta. A lo largo del capítulo trataremos de buscar analogías y diferencias, así como de ver qué ventajas o desventajas presenta una teoría sobre la otra. A lo largo del capítulo, se recurrirá a ejemplos ya tratados en capítulos anteriores además de algún nuevo ejemplo ilustrativo.

En la última sección hablaremos sobre las conclusiones obtenidas a lo largo del trabajo y esbozaremos un camino a seguir para el trabajo o investigaciones futuras.

CAPÍTULO

2

Preliminares

En este capítulo introduciremos definiciones básicas sobre lógica difusa, siendo los primeros estudios en este campo de trabajo realizados por L.A. Zadeh (22). A lo largo de este capítulo nos apoyaremos en nociones vistas en (11)

2.1 Conceptos básicos sobre conjuntos difusos

En la lógica clásica (binaria) para cuantificar la pertenencia de un elemento a un conjunto solo existen dos posibles valores: 0 para indicar que un elemento no pertenece a un conjunto clásico ($x \notin X$) o 1 para indicar que sí pertenece ($x \in X$). En cambio, en lógica difusa, tenemos que la pertenencia de un elemento al conjunto es gradual, variando dicho *grado de pertenencia* entre 0 (el elemento no pertenece al conjunto difuso) y 1 (el elemento está totalmente dentro del conjunto). Si es otro valor del intervalo (0,1), se dice que el elemento pertenece con cierto grado al conjunto.

Definición 2.1. Definimos un *conjunto difuso* A sobre un universo de discurso X (dominio) como un conjunto de pares de la forma:

$$A = \{(x, f_A(x)) \mid x \in X, f_A(x) \in [0, 1]\}$$

donde f_A es el *grado de pertenencia* del elemento x al conjunto A.

A continuación, se introducirán las definiciones concretas de cada noción.

Definición 2.2. Llamamos *universo de discurso* al conjunto de todos los elementos que se consideran en un determinado contexto. Se distinguen dos tipos de universo: los discretos y los continuos. Para los primeros, los conjuntos difusos se representan como un conjunto de pares de valores y, para los segundos, los conjuntos difusos vienen representados por la llamada función de pertenencia al conjunto en cuestión.

Definición 2.3. Llamamos variable lingüística a aquellas variables cuyos valores son palabras o sentencias. Su utilidad es trasladar conceptos lingüísticos a descripciones numéricas.

Definición 2.4. Una etiqueta lingüística es aquella palabra, en el lenguaje natural, que expresa o identifica a un conjunto difuso. Podemos encontrar tantas etiquetas como conceptos abstractos se pueden cuantificar en mayor o menor medida.

Ejemplo 2.1. Vemos en la Tabla 2.1 un ejemplo sobre los conceptos que acabamos de definir.

Variable lingüística	Etiquetas lingüísticas
Velocidad	Alta, media, baja, etc
Calidad del servicio	Pobre, decente, buena, excelente, etc
Altura	Bajo, mediano, alto, etc

Tabla 2.1: Variables y etiquetas lingüísticas

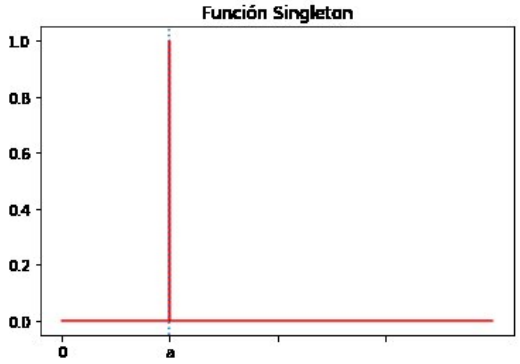
Veremos en la siguiente sección algunas de las funciones de pertenencia más usadas.

2.2 Funciones de pertenencia

En principio, cualquier función con imagen en $[0, 1]$ sería válida para definir conjuntos difusos. Pero en la práctica hay algunas que son más usadas que otras debido principalmente a su facilidad de computación y su estructura lógica. Presentaremos aquí algunas de las más famosas.

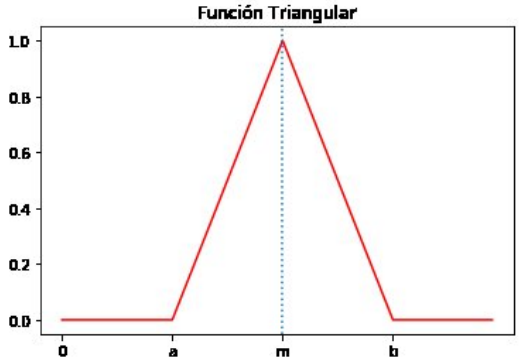
- **Función singleton**: Tiene un único valor cuando $x = a$.

$$f_A(x) = \begin{cases} 0 & \text{si } x = a \\ 1 & \text{si } x \neq a \end{cases}$$



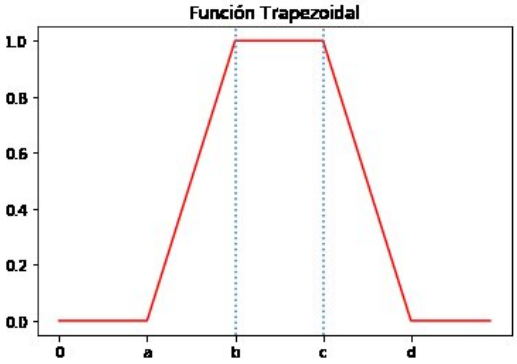
- **Función triangular**: Esta función está definida mediante el límite inferior a, el superior b y el valor modal m, tal que $a < m < b$. La función no tiene porqué ser simétrica.

$$f_A(x) = \begin{cases} 0 & \text{si } x \leq a \\ \frac{x-a}{m-a} & \text{si } a < x \leq m \\ \frac{b-x}{b-m} & \text{si } m < x < b \\ 0 & \text{si } b \leq x \end{cases}$$



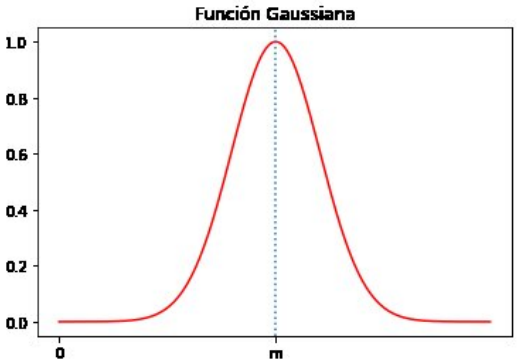
- **Función trapezoidal**: Definida por sus límites inferior a, superior d, y los límites de soporte inferior b y superior c, tal que $a < b < c < d$.

$$f_A(x) = \begin{cases} 0 & \text{si } x \leq a \\ \frac{x-a}{b-a} & \text{si } a < x \leq b \\ 1 & \text{si } b < x \leq c \\ \frac{d-x}{d-c} & \text{si } c < x < d \\ 0 & \text{si } d \leq x \end{cases}$$



- **Función Gaussiana**: Definida por su valor medio m y el parámetro $k > 0$. Esta función es la típica campana de Gauss y cuanto mayor es el valor de k, más estrecha es dicha campana.

$$f_A(x) = e^{-k(x-m)^2}$$



2.3 Caracterización de un conjunto difuso

Un conjunto difuso presenta características que nos pueden resultar útiles a la hora de estudiarlos.

Definición 2.5. La *altura* de un conjunto difuso A definido sobre X se define como:

$$H(A) = \text{máx}\{f_A(x) \mid x \in X\}$$

Definición 2.6. El *soporte* de un conjunto difuso A definido sobre X es el subconjunto que satisface:

$$S(A) = \{x \in X \mid 0 < f_A(x)\}$$

Un conjunto se dice que es *singleton* si su soporte es un único punto.

Definición 2.7. El *núcleo* de un conjunto difuso A definido sobre X viende dado por:

$$Ker(A) = \{x \in X \mid f_A(x) = 1\}$$

Un conjunto difuso se dice *normal* si su núcleo es no vacío. Una *normalización* es una aplicación que convierte un conjunto difuso no normal en un conjunto difuso normal.

Definición 2.8. Llamamos α*-corte* de un conjunto difuso A, a un subconjunto no difuso de elementos de X que satisface:

$$A_\alpha = \{x \in X \mid \alpha \leq f_A(x)\}$$

Vemos en la Figura 2.1 un resumen de las características que acabamos de definir, considerando un conjunto difuso representado por la función de pertenencia Gaussiana. Nótese que en este ejemplo, al tratarse de una función gaussiana sería $Ker(A) = a$.

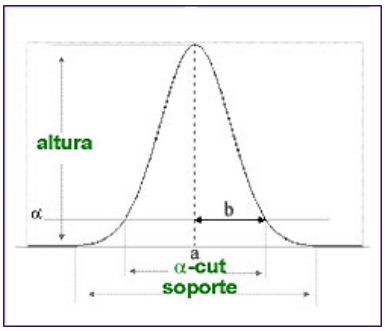


Figura 2.1: Características de un conjunto difuso

Teorema 2.1. ***(Teorema de Representación)*** *Todo conjunto difuso A puede descomponerse en una familia de conjuntos no difusos, es decir, puede reconstruirse a partir de la familia de sus α-cortes:*

$$f_A(x) = \text{máx}\{\alpha \cdot A_\alpha(x) \mid \alpha \in [0, 1]\}$$

para todo $x \in X$, y considerando A_α como la aplicación característica[1].

En la teoría de conjuntos difusos también se introduce la noción de convexidad.

Definición 2.9. Decimos que un conjunto difuso A es *convexo* cuando todos sus α-cortes lo son, es decir, dados $x, y \in X$ se satisface

$$\text{mín}(f_A(x), f_A(y)) \leq f_A(\lambda \cdot x + (1 - \lambda) \cdot y)$$

[1]Esta aplicación se define como $A_\alpha = \begin{cases} 1 & \text{si } x \in A_\alpha \\ 0 & \text{si } x \notin A_\alpha \end{cases}$

2.4 Comparación de conjuntos difusos

Lo primero que debemos de tener en cuenta, es que solo podremos comparar dos conjuntos difusos, A y B, si dichos conjuntos están definidos sobre el mismo universo discurso X. Dentro de este contexto, podemos dar las siguientes definiciones:

Definición 2.10. ***(Inclusión)*** Un conjunto difuso A está incluido en otro conjunto difuso B, y lo denotaremos como $A \subseteq B$, si y y solo si $f_A(x) \leq f_B$, para todo $x \in X$.

Definición 2.11. ***(Igualdad)*** Dos conjuntos difusos A y B se dicen iguales, $A = B$, si y solo si $f_A(x) = f_B(x)$, para todo $x \in X$.

2.5 Conectivos lógicos difusos

A continuación, presentamos las definiciones de las operaciones de intersección, unión, complemento e implicación en el marco de la lógica difusa.

- **Conjunción** ($\wedge$)
  Dados dos conjuntos difusos A y B sobre los universos X e Y respectivamente, y sea el par $(x, y) \in X \times Y$, la *conjunción* $A \wedge B$, viene dada por

$$f_{A \wedge B}(x, y) = \text{mín}\{f_A(x), f_B(y)\}$$

- **Disyunción** ($\vee$)
  Dados dos conjuntos difusos A y B sobre los universos X e Y respectivamente, y sea el par $(x, y) \in X \times Y$, la *disyunción* $A \vee B$, viene dada por

$$f_{A \vee B}(x, y) = \text{máx}\{f_A(x), f_B(y)\}$$

- **Negación** ($\neg$)
  Sea A un conjunto difuso sobre el universo discurso X, la *negación* se corresponde con el cálculo del complementario.

$$f_{\neg A}(x) = 1 - f_A(x)$$

- **Implicación** ($\rightarrow$)**(IF-THEN)**
  Dados dos conjuntos difusos A y B sobre los universos X e Y respectivamente, la *implicación* $A \rightarrow B$ es una aplicación entre A y B tal que $f_{A \rightarrow B}(x, y)$ representa

el grado de verdad de la implicación entre x e y. Existen múltiples definiciones para la implicación, mostraremos las más comunes, aunque cabe mencionar que en este trabajo nos centraremos principalmente en la implicación de Mamdani:

- **Implicación de Kleene:** $f_{A\to B}(x,y) = \max\{1 - f_A(x), f_B(y)\}$
- **Implicación de Mamdani:** $f_{A\to B}(x,y) = \min\{f_A(x), f_B(y)\}$
- **Implicación de Larsen:** $f_{A\to B}(x,y) = f_A(x) \cdot f_B(y)$

Es importante mencionar que todos estos operadores que hemos definido hasta ahora, presentan problemas de pérdida de valores (ya que en su mayoría trabajan con máximos y mínimos). Es por ello que surgen los siguientes conceptos:

Definición 2.12. Una *norma triangular* o *t-norma* es una operación binaria sobre el intervalo unidad $T : [0,1] \times [0,1] \to [0,1]$ tal que para todo $x, y, z \in [0,1]$ se verifican las siguientes propiedades:

1. T es conmutativa; es decir, $T(x,y) = T(y,x)$.
2. T es monótona; es decir, si $x \leq y$, entonces $T(x,z) \leq T(y,z)$.
3. T satisface la condición de frontera con 1: $T(x,1) = x$.
4. T es asociativa, $T(T(x,y),z) = T(x,T(y,z))$

Las t-normas generalizan el operador conjunción.

Ejemplo 2.2. Las tres t-normas más usuales son la de Gödel, producto y Łukasiewicz. Estas t-normas se definen, para todo $x, y \in [0,1]$, como:

$$\begin{aligned} T_G(x,y) &= \min\{x,y\} \\ T_L(x,y) &= \max\{0, x+y-1\} \\ T_P(x,y) &= x \cdot y \end{aligned}$$

Definición 2.13. Una *conorma triangular* o *t-conorma* es una operación binaria sobre el intervalo unidad $S : [0,1] \times [0,1] \to [0,1]$ tal que para todo $x, y, z \in [0,1]$ se verifican las siguientes propiedades:

1. S es conmutativa; es decir, $S(x,y) = S(y,x)$.
2. S es monótona; es decir, si $x \leq y$, entonces $S(x,z) \leq S(y,z)$.
3. S satisface la condición de frontera con 0: $S(x,0) = x$.

4. S es asociativa, $S(S(x,y),z) = S(x,S(y,z))$

Las t-conormas generalizan el operador disyunción.

Ejemplo 2.3. Las tres t-conormas más usuales son la de Gödel, producto y Łukasiewicz. Estas t-conormas se definen, para todo $x, y \in [0,1]$, como:

$$\begin{aligned} S_G(x,y) &= \text{máx}\{x,y\} \\ S_L(x,y) &= \text{mín}\{1, x+y\} \\ S_P(x,y) &= x + y - x \cdot y \end{aligned}$$

Existe un mecanismo para obtener t-conormas a partir de t-normas, pero para ello necesitamos un operador negación.

Definición 2.14. Un operador unario sobre el intervalo unidad $n : [0,1] \to [0,1]$ se dice que es una *negación* si satisface:

1. Si $x \leq y$, entonces $n(y) \leq n(x)$.
2. $n(0) = 1, n(1) = 0$

Diremos que la negación es *fuerte o involutiva* si además se satisface $x = n(n(x))$, para todo $x \in [0,1]$.

Ejemplo 2.4. El operador de negación más usado es el que se define como $n(x) = 1 - x$, para todo $x \in [0,1]$. Dicho operador es un ejemplo de negación fuerte.

Por último veremos la generalización para la implicación.

Definición 2.15. Una *implicación difusa* es un operador binario $\leftarrow$: $[0,1] \times [0,1] \to [0,1]$ que es creciente en el primer argumento, al que llamaremos *consecuente*, y decreciente en el segundo argumento, al cual nos referiremos como *antecedente*.

Resulta sin embargo, que este tipo de implicaciones no presenta propiedades para poder operar eficientemente en ambientes en los que haya algún proceso de deducción, pues no generalizan la propiedad del *modus ponens clásico*. Para que una implicación cumpla dicho *modus ponens* clásico, esta implicación debe tener asociada a una t-norma con la que satisfaga la *propiedad de adjunción*.

Definición 2.16. Data una t-norma T: $[0,1] \times [0,1] \to [0,1]$, si existe una implicación difusa $\leftarrow$: $[0,1] \times [0,1] \to [0,1]$, tal que

$$T(x,y) \leq z, \text{ si y solo si, } x \leq z \leftarrow y \tag{2.1}$$

se dice que $\leftarrow$ es una implicación residuada de T, se escribe como $\leftarrow_T$, y al par $(T, \leftarrow_T)$ se le llama *par adjunto* o *par residuado*. La propiedad dada por la Ecuación (2.1) recibe el nombre de *propiedad de adjunción*.

2.6 Fuzzificación y defuzzificación

La *fuzzificación* es un proceso que permite asociar un valor numérico a un conjunto difuso. Este proceso responde a un conjunto de normas preestablecidas, conceptualizadas a partir del razonamiento humano.

La *defuzzificación* se trata del proceso inverso, permite asociar a un conjunto difuso un valor numérico. El más común y ampliamente usado es el método del centroide. Dada la función salida $f : X \to [0, 1]$, se define el centroide de f en X como

$$Centroide(f, X) = \frac{\int f(x) \cdot x dx}{\int f(x) dx}$$

si f es continua, en el caso de que f sea discreta sería

$$Centroide(f, X) = \frac{\sum_{i=0}^{n} f(x_i) \cdot x_i}{\sum_{i=0}^{n} f(x_i)}$$

Existen otros métodos de desfuzzificación que en casos particulares pueden ser más eficientes como el método de la bisectriz, el método del máximo central, del máximo más pequeño, etc.

CAPÍTULO

3

Métodos de inferencia difusa

En el novedoso campo de la inteligencia artificial, encontramos varios métodos para representar el conocimiento. La mayoría de las ideas en las que se basa este capítulo provienen de (7) y (17). Posiblemente una de las maneras más usadas para la representación del conocimiento sean las expresiones del tipo

IF condición (antecedente), THEN conclusión (consecuente)

a las que llamaremos reglas canónicas. Esta expresión da lugar a los métodos de inferencia basados en reglas del tipo IF-THEN.

Normalmente con el uso de esta expresión tratamos de, mediante un hecho al que hemos llamado antecedente, inferir una conclusión a la que hemos llamado consecuente. Esta forma de representación del conocimiento es bastante apropiada en el contexto de la lingüística porque expresa el conocimiento empírico y heurístico humano en nuestro lenguaje de comunicación. En cambio, no es tan válida para formas más profundas de conocimiento asociadas con la intuición, estructura y comportamiento de los objetos a nuestro alrededor, ya que estas no se encuentran fácilmente asociadas a frases lingüísticas.

Mediante el uso de las propiedades básicas para conjuntos difusos, definidas en el Capítulo 2 , cualquier estructura compuesta de reglas puede ser reducida a un número finito de reglas canónicas. Dichas reglas están basadas en representaciones del lenguaje natural y modelos, que en sí mismos están basados en conjuntos y lógica difusa.

Veremos en este capítulo todo este proceso, empezando por la formación de reglas y terminando en los sistemas de inferencia difusa.

3.1 Formación de reglas

Para cualquier variable lingüística hay tres formas diferentes de establecer dichas reglas canónicas:

1. **Sentencias de asignación:** Son aquellas en las que a las variables se les asigna un único valor mediante el operador "=". No caigamos en el equívoco de que el operador que se asigna es necesariamente un número, puede ser un término lingüístico. Algunos ejemplos de este tipo de sentencias son:

$$\begin{aligned} x &= \text{rápido}, \\ \text{pH} &= \text{neutro}, \\ a &= \pi, \\ l &= n + m. \end{aligned}$$

2. **Sentencias condicionales:** En este tipo de sentencias se imponen condiciones. Si dichas condiciones se cumplen entonces se aplican las llamadas restricciones. Por ejemplo:

   IF $a = b$ THEN ambos son iguales,
   IF Velocidad > 120 THEN multa.

   Estas sentencias reciben el nombre de sentencias lógicas condicionales y su estructura general es IF condición C THEN restricción R.

3. **Sentencias incondicionales:** En este caso no hay una condición específica que debe satisfacerse. A modo ilustrativo:

   Acelera,
   Disminuye una unidad.

   La orden se puede transferir sin ninguna condición. La forma general de este tipo de sentencias es la siguiente:

$$
\begin{array}{rcl}
R^1 & : & \text{Sentencia es } B^1, \\
 & & \text{AND} \\
R^2 & : & \text{Sentencia es } B^2, \\
 & & \text{AND} \\
 & & \ldots, \text{etc.}
\end{array}
$$

donde R^i son las restricciones y B^i son consecuentes difusos.

Es posible combinar todos los tipos de sentencias mediante conectores como "AND", "OR", "ELSE" (no se suelen traducir debido a su uso habitual en todos los lenguajes de programación). El consecuente de las reglas viene definido por restricciones $R^1, ..., R^n$. Por tanto, un sistema basado en reglas con un conjunto de reglas condicionales viene dado como se muestra en la Tabla 3.1:

Regla 1:	IF	condición C^1	THEN	restricción R^1
Regla 2:	IF	condición C^2	THEN	restricción R^2
...				
Regla n:	IF	condición C^n	THEN	restricción R^n

Tabla 3.1: Forma canónica - Sistemas difusos basados en reglas

3.2 Descomposición de las reglas

Es bastante común encontrarse con estructuras de reglas compuestas. Un ejemplo de dichas composiciones puede ser:

IF $x = y$ THEN ambos son iguales

ELSE

IF $x \neq y$

THEN

IF $x > y$ THEN x es mayor
ELSE IF $y > x$ THEN y es mayor
ELSE

Cualquier estructura de reglas compuestas, como la que acabamos de mostrar, puede descomponerse y reducirse en reglas simples. Hay varios métodos para llevar a cabo esta descomposición:

1. **Antecedentes de conjunciones múltiples:** Este método de descomposición, como su nombre indica, se basa en la definición básica de la operación intersección difusa, representada lingüísticamente por AND. La forma general sería la siguiente:

   $$\text{IF } x \text{ es } A^1 \text{ AND } A^2 \text{ ... AND } A^n \text{ THEN y es } B^r$$

   Si asumimos el nuevo conjunto difuso A^r como

   $$A^r = A^1 \wedge A^2 \wedge ... \wedge A^n$$

   donde la función de pertenencia viene dada por:

   $$f_{A^r}(x) = \text{mín}\{f_{A^1}(x), f_{A^2}(x), ..., f_{A^n}(x)\}$$

   por tanto, la regla compuesta que escribimos inicialmente, se puede reducir a la siguiente regla simple:

   $$\text{IF } A^r \text{ THEN } B^r$$

2. **Antecedentes de disyunciones múltiples:** En este caso el método se basa en la operación unión difusa, lingüísticamente representada por OR. En general:

   $$\text{IF } x \text{ es } A^1 \text{ OR } A^2 \text{ ... OR } A^n \text{ THEN y es } B^r$$

   que puede ser reescrita como:

   $$\text{IF } x \text{ es } A^r \text{ THEN } y \text{ es } B^r$$

   donde el conjunto difuso A^r se define como:

   $$A^r = A^1 \vee A^2 \vee ... \vee A^n$$

   con función de pertenencia:

   $$f_{A^r}(x) = \{f_{A^1}(x), f_{A^2}(x), ..., f_{A^n}(x)\}.$$

3. **Sentencias condicionales con ELSE:** Podemos distinguir dos tipos:

   - Las reglas compuestas de la forma:

IF A^1 THEN (B^1 ELSE B^2)

Si consideramos esta regla como una declaración compuesta, podemos dividirla en dos reglas canónicas, obteniendo:

IF A^1 THEN B^1 OR IF NOT A^1 THEN B^2

- Si tenemos una regla compuesta de la forma:

  IF A^1 THEN (B^1 ELSE A^2 THEN (B^2))

  la descomposición vendrá dada por:

  IF A^1 THEN B^1 OR
  IF NOT A^1 AND A^2 THEN B^2.

4. **Reglas IF-THEN anidadas:** Son reglas compuestas de la forma:

   IF A^1 THEN (IF A^2 THEN (B^2))

   se pueden descomponer como:

   IF A^1 AND A^2 THEN B^2

   De manera que estas reglas pueden ser reducidas en un número finito de reglas canónicas.

3.3 Agregación de reglas difusas

Los métodos de inferencia basados en reglas difusas pueden incluir más de una regla. Se llama *agregación de reglas difusas* al proceso de obtener la conclusión (o consecuente) general a partir de los consecuentes de cada regla implicada en el sistema. A la hora de determinar una estrategia de agregación existen dos métodos:

1. **Sistema conjuntivo de reglas:** Se trata de sistemas en el que todas las reglas deben verificarse conjuntamente, por tanto, las reglas están conectadas por el conector AND. En este caso, el consecuente agregado (conclusión), y, viene dado por la conjunción difusa de todos los consecuentes, y^i, de reglas individuales donde $i = 1, 2, .., n$. La función de pertenencia se define como:

$$f_y(x^1, x^2, ..., x^n) = \text{mín}\{f_{y^1}(x^1), f_{y^2}(x^2), ..., f_{y^n}(x^n)\}.$$

2. **Sistema disyuntivo de reglas:** Nos encontramos ahora con sistemas donde solo se requiere que se verifique al menos una regla, por tanto el conector usado es OR. En este caso, el consecuente agregado viene dado por la disyunción difusa de los consecuentes de cada regla, con la siguiente función de pertenencia:

$$f_y(x^1, x^2, ..., x^n) = \text{máx}\{f_{y^1}(x^1), f_{y^2}(x^2), ..., f_{y^n}(x^n)\}.$$

3.4 Propiedades de los conjuntos de reglas

En esta sección presentaremos las cuatro propiedades que debe presentar todo conjunto de reglas: *completitud, consistencia, continuidad e interacción*.

- **Completitud:** Diremos que un conjunto de reglas del tipo IF-THEN es *completo* si cualquier combinación de valores de entrada da como resultado un valor de salida apropiado.

- **Consistencia:** Se dice que un conjunto de reglas del tipo IF-THEN es *no consistente* cuando hay dos reglas con el mismo antecedente pero diferentes consecuentes. Es decir, si queremos un conjunto *consistente*, no deben existir dos reglas que para una misma entrada devuelvan diferente variable de salida.

- **Continuidad:** Diremos que un conjunto de reglas del tipo IF-THEN es *continuo* si no tiene reglas contiguas tal que sus variables de salidas sean conjuntos difusos cuya intersección es vacía.

- **Interacción:** En esta propiedad, supondremos que se trata de una regla del tipo, "IF x es A THEN y es B", esto se representa mediante una relación difusa R^2, así, la composición de A y R no resulta B:

$$A \circ R \neq B$$

3.5 Métodos de inferencia difusa

El objetivo de un sistema de inferencia difusa es tomar valores de entrada y, a partir de reglas preestablecidas, inferir una respuesta o valor de salida. Debido a su naturaleza multidisciplinar, también se llama a estos sistemas: *sistemas difusos basados en reglas difusas,*

sistema experto difuso, sistema de control difuso... La toma de decisiones es una parte fundamental en este tipo de sistemas, para ello el sistema formula unas reglas en las que se basa la decisión. Tanto la construcción de las reglas como el sistema de inferencia, se basan principalmente en conceptos vistos anteriormente como: reglas IF-THEN, operadores difusos... Remarcar que este tipo de sistemas pueden tomar como entrada tanto valores difusos como valores clásicos, pero los valores de salida son casi siempre conjuntos difusos. En cambio, cuando el sistema es usado como un controlador difuso el valor de salida debe ser definido, por tanto han de usarse, en la mayoría de los casos, métodos de *defuzzificación*.

3.5.1 Construcción y funcionamiento del sistema de inferencia

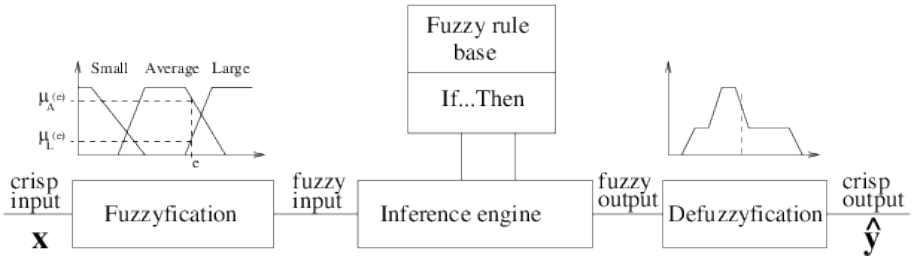


Figura 3.1: Sistema de inferencia difuso

Como podemos observar en la Figura 3.1, el sistema de inferencia difuso está constituido por: un proceso de *fuzzificación*, un sistema de reglas junto con una base de datos (que forman la estructura de conocimiento y forman la unidad de toma de decisiones) y por último, un proceso de *defuzzificación*. Veremos ahora qué función le corresponde a cada bloque:

- **Sistema de *fuzzificación*:** Tiene el papel de transformar las entradas definidas en valores lingüísticos (variables difusas). Para ello, se usan las funciones de pertenencia almacenadas en la estructura de conocimiento.

- **Base de datos**: Lugar donde se encuentran definidas las funciones de pertenencia de los conjuntos difusos que se usarán en el sistema de reglas.

- **Sistema de reglas:** Contiene el conjunto de reglas difusas IF-THEN que instan a la toma de decisiones.

- **Sistema de *defuzzificación*:** Se encarga de transformar los resultados difusos de la inferencia de nuevo en valores de salida clásicos. Para ello se usan métodos como el de centroide, visto en el Capítulo 2 u otros que veremos más adelante.

Una vez vistos los componentes de un sistema de inferencia difusa, vamos a ver los pasos que sigue el *razonamiento difuso* llevado a cabo en el sistema de inferencia:

1. Comparar los valores de entrada con las funciones de pertenencia de los antecedentes para obtener las funciones de pertenencia de cada variable lingüística. Este paso es llamado *fuzzificación*.

2. Combinar (mediante una t-norma normalmente producto o mínimo) los valores de pertenencia para conseguir el valor de verdad[1] de cada regla.

3. Generar los consecuentes adecuados, ya sean difusos o clásicos.

4. Agregar dichos consecuentes para producir la salida del valor clásico. Este paso es llamado *defuzzificación*.

3.5.2 Tipos de métodos de inferencia difusa

En este apartado trataremos de ilustrar los procedimientos matemáticos más usados para realizar la inferencia de los sistemas difusos. Estos procedimientos pueden ser, y son, normalmente implementados en un ordenador por razones obvias. Sin embargo, a veces es conveniente realizar ejemplos reducidos manualmente para comprobar el funcionamiento de los programas o para verificar las operaciones.

Los dos métodos más importantes y más utilizados en la inferencia difusa son el *método de Mamdani* y el *método de Sugeno*. La principal diferencia entre los métodos de Mamdani y Sugeno reside en el consecuente de las reglas lógicas. Por un lado, los métodos de inferencia difusa de Mamdani usan conjuntos difusos como consecuentes para las reglas, mientras que los sistemas de Sugeno usan funciones lineales. Pasaremos a explicar dichos métodos en profundidad y mostraremos un ejemplo práctico de cada uno.

[1] Llamamos valor de verdad al resultado de combinar mediante una t-norma las funciones de pertenencia correspondientes a cada regla evaluadas en los valores de entrada. Por ejemplo en la Figura 3.6 podemos observar que el valor de verdad es 0,7.

3.5.3 Método de inferencia difusa de Mamdani

Este primer método, el más usado en la práctica y más citado en la bibliografía, se lo debemos a Mamdani y Assilian. Este método fue uno de los primeros sistemas de control que se construyó usando lógica difusa. Fue propuesto por Mamdani (1975) en un intento de controlar un sistema de calderas y motores mediante un conjunto de reglas obtenidas empíricamente. El trabajo de Mamdani se basó principalmente en el artículo sobre algoritmos difusos para sistemas complejos y toma de decisiones publicado por Zadeh (1973).

Cuando usamos el método de Mamdani, se espera que las funciones de pertenencia dadas como variable de salida sean conjuntos difusos. Después del proceso de agregación, tenemos entonces un conjunto difuso para cada variable de salida que necesita *defuzzificación*. Es posible, y en muchos casos más eficiente, utilizar la función *singleton* como la función de pertenencia de salida en lugar de un conjunto difuso. Se le conoce normalmente como valor de salida *singleton* y puede ser considerado como un conjunto difuso *pre-defuzzificado*. Obviamente este procedimiento aumenta la eficiencia del método porque se acelera mucho computacionalmente la *defuzzificación* ya que, en el proceso estándar la defuzzificación se lleva a cabo encontrando el centroide, lo que supone la realización de integrales en $\mathbb{R}^2$.

En la Figura 3.2 se muestra un ejemplo de un sistema de inferencia de Mamdani. Los seis pasos que realizan los sistemas de inferencia de Mamdani desde que reciben la variable de entrada hasta que proporcionan la variable de salida son los siguientes:

1. Determinar las reglas difusas que se utilizarán para la inferencia;
2. *Fuzzificar* los valores de entrada mediante las funciones de pertenencia;
3. Combinar los valores *fuzzificados* de acuerdo a las reglas;
4. Encontrar las consecuencias de dichas reglas mediante los valores obtenidos y las funciones de pertenencia para los valores de salida;
5. Encontrar la distribución de las consecuencias;
6. *Defuzzificar* dicha distribución (solo cuando necesitamos que el valor de salida sea clásico).

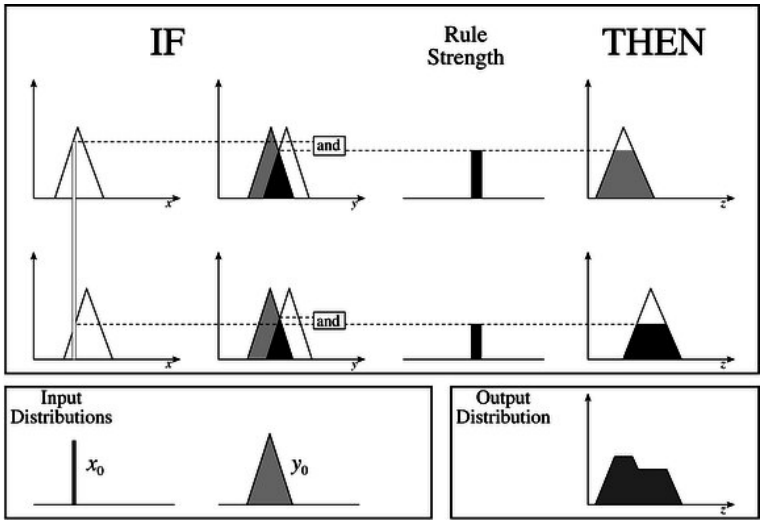


Figura 3.2: Sistema de inferencia Mamdani con dos reglas y dos variables de entrada.

Explicaremos ahora con más detenimiento el procedimiento:

- **Creación de reglas difusas:** Las reglas difusas vienen dadas de la siguiente manera:

  IF *(variable de entrada 1 es función de pertenencia 1)* ***AND/OR*** *(variable de entrada 2 es función de pertenencia 2)* ***AND/OR*** *...* ***THEN*** *(variable de salida n es función de pertenencia n).*

  La creación de reglas difusas se ilustrará en el siguiente ejemplo:

  IF *velocidad es alta* ***AND*** *distancia corta con el siguiente vehículo* ***THEN*** *frena*

  Debe haber funciones de pertenencia que definan que los valores “velocidad alta” (primera variable de entrada) y “distancia corta” (segunda variable de entrada), y lo mismo para el valor de salida “frena”. Este proceso de tomar una entrada como la velocidad y procesarla a través de una función de pertenencia para determinar la “velocidad alta”, recibe el nombre de *fuzzificación* y se explica a continuación. Es importante mencionar que los conectivos lógicos “AND” / “OR” en la regla difusa deben ser definidos por el experto. La aplicación de estos conectivos para obtener la consecuencia de la regla se conoce como combinación difusa, concepto que también se presenta en el siguiente ítem.

- ***Fuzzificación:*** El objetivo de este proceso es el asignar un valor de verdad entre cero y uno, mediante funciones de pertenencia, a los valores de entrada recogidos.
  En el ejemplo de la Figura 3.2 vemos que hay dos valores de entrada, x_0 que es un valor clásico e y_0 que es difuso. En primer lugar, explicaremos el procedimiento centrándonos en el valor de entrada clásico x_0. Puede observarse cómo se calcula el grado de pertenencia al conjunto de x_0, se dibuja una recta hasta encontrar el punto de corte con la función. Estas funciones de pertenencia se establecen previamente y, como hemos visto en los preliminares, pueden representar conceptos como "velocidad alta", "frío", "adulto", etc.

- **Consecuencia:** Para obtener la consecuencia de una regla difusa han de seguirse los siguientes pasos:

  1. Cálculo de la *fortaleza de la regla* mediante la combinación de las entradas *fuzzificadas* utilizando el proceso descrito en el ítem anterior. En el ejemplo que se ilustra en la Figura 3.2, se usa el conectivo difuso AND para combinar las funciones de pertenencia y obtener dicha *fortaleza de la regla*.

  2. El resultado de la evaluación del antecedente se relaciona con el consecuente aplicando un **recorte** (Figura 3.3) o un **escalado** (Figura 3.4. El método más común (más sencillo computacionalmente) es el recorte (clipping) que corta el consecuente con el valor de verdad del antecedente. En cambio el escalado proporciona un resultado más preciso y preserva la forma original del conjunto difuso, se obtiene multiplicando todos los valores por el valor de verdad del antecedente. En el método de inferencia de Mamdani se utiliza el recorte cuyo procedimiento consiste en cortar la función de pertenencia del valor de salida a la altura del valor de verdad obtenido de los antecedentes.

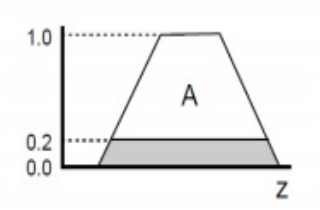


Figura 3.3: Conjunto recortado

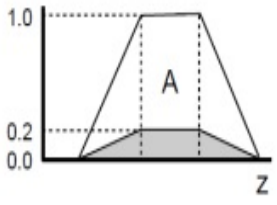


Figura 3.4: Conjunto escalado

- **Combinación de las variables de salida para encontrar su distribución:** Este paso normalmente se realiza usando el conector difuso OR. En la Figura 3.2 vemos un ejemplo de cómo las dos figuras de la derecha se combinan usando dicho operador y se obtiene la distribución que se muestra en la esquina inferior derecha. Aclarar, que para evaluar la disyunción normalmente se emplea el operador máximo.

- ***Defuzzificación:*** Este proceso ha de llevarse a cabo siempre que queramos como salida un valor clásico en vez de uno difuso. Ya vimos en las nociones básicas en qué consiste la *defuzzificación*. Añadiremos en esta sección algunos métodos de *defuzzificación* usuales en las inferencias de Mamdani:

  1. **Media del máximo o máximo central:** Básicamente esta técnica toma los valores donde las funciones de pertenencia alcanzan su máximo y calcula la media como sigue:

     $$z = \sum_{j=1}^{l} \frac{z_j}{l}$$

     Nótese que z es la medida del máximo, z_j es el punto en el que la función de pertenencia alcanza el valor máximo y l es el número de veces que se alcanza el valor máximo.

  2. **Máximo más pequeño:** La salida es el mínimo valor de todos aquellos que alcanzan el valor más alto de la función de pertenencia.

  3. **Máximo más grande:** La salida se corresponde con el máximo valor de todos aquellos que alcanzan el valor más alto de la función de pertenencia.

  Observemos en la siguiente imagen (Figura 3.5) los resultados obtenidos con cada uno de de todos los métodos de *defuzzificación* vistos hasta el momento:

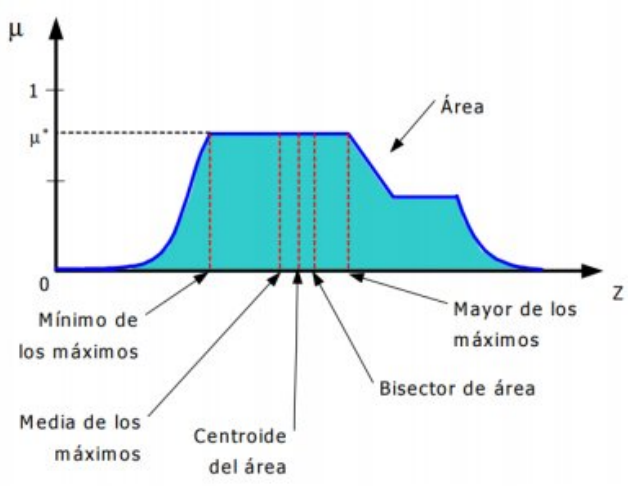


Figura 3.5: Métodos de *defuzzificación*

Ahora consideraremos el caso en el que el valor de entrada es un conjunto difuso. Concretamente, estamos considerando el valor y_0 que aparece en la Figura 3.2. En este caso la *fuzzificación* se lleva a cabo intersecando el valor clásico con la función de pertenencia. En este proceso, como se puede observar, se usa el operador AND para combinar los dos conjuntos *fuzzificados*. El resto del procedimiento es igual al explicado para valores clásicos.

En resumen, la Figura 3.2 muestra un ejemplo gráfico del método de inferencia difusa de Mamdani con dos reglas y dos variables de entrada. Se fuzzifican las dos entradas encontrando la intersección del valor de entrada clásico con la función de pertenencia. Se utiliza el operador mínimo (AND) para combinar las dos entradas difusas y obtener la fortaleza de regla. Se aplica la técnica de recorte usando fortaleza de la regla y finalmente, utiliza el operador máximo (OR) para combinar las salidas de las dos reglas.

A continuación, vamos a ver cómo funciona el método de Mamdani en la práctica con un ejemplo que resolveremos usando la aplicación de lógica difusa que proporciona el *software Matlab*, de manera que todos los conceptos explicados anteriormente quedarán mucho más claros.

Ejemplo 3.1. Supongamos que embarcamos un proyecto sobre coches autónomos. Nos encontramos con la tarea de programar un controlador para este tipo de coche, con el objetivo de saber cuánto tiene que frenar o acelerar un vehículo en carretera en función a la distancia con el vehículo que tenga delante y la velocidad que llevemos.

Ahora vamos a preestablecer los conjuntos difusos así como las funciones de pertenencia para cada variable:

1. Velocidad: Para la primera variable de entrada asumen cinco etiquetas lingüísticas:

   - Muy baja (MB): función de pertenencia trapezoidal $(20, 20, 30, 35)$[1].
   - Baja (B): función de pertenencia triangular $(35, 40, 60)$.
   - Normal (N): función de pertenencia triangular $(60, 75, 80)$.
   - Alta (A): función de pertenencia triangular $(75, 90, 100)$.
   - Muy alta (MA): función de pertenencia trapezoidal $(90, 100, 110, 120)$.

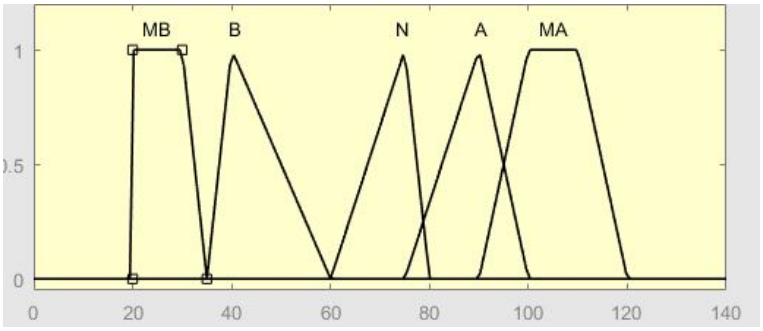


Figura 3.6: Funciones de pertenencia variable Velocidad

2. Distancia con el vehículo de delante: se asumen cinco etiquetas lingüísticas.

   - Muy corta (MC): función de pertenencia trapezoidal $(5, 5, 10, 15)$.
   - Corta (C): función de pertenencia triangular $(10, 15, 20)$.
   - Normal (N): función de pertenencia triangular $(20, 25, 30)$.
   - Larga (L): función de pertenencia triangular $(30, 40, 50)$.
   - Muy larga (ML): función de pertenencia trapezoidal $(45, 50, 60, 70)$.

[1]Los números en los paréntesis no son más que los valores de las constantes correspondientes a cada función de pertenencia (ver Capítulo 2)

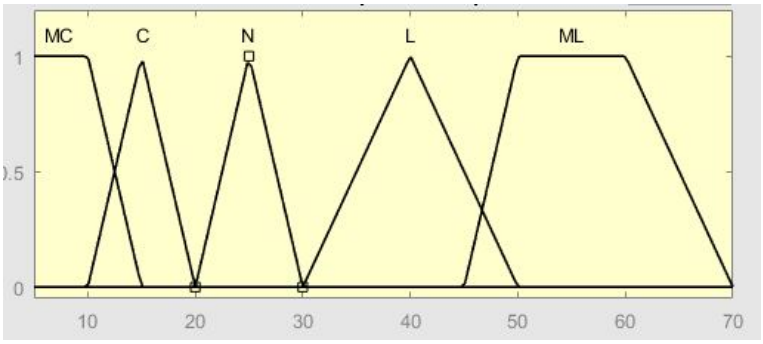


Figura 3.7: Funciones de pertenencia variable Distancia

3. Frenada: A la variable de salida se le asocian las siguientes etiquetas:

   - Frenada inminente (FI): función de pertenencia triangular $(-120, -100, -90)$.
   - Frenada fuerte (FF): función de pertenencia triangular $(-90, -70, -50)$.
   - Frenada moderada (FM): función de pertenencia triangular $(-50, -30, -20)$.
   - Frenada leve (FL): función de pertenencia triangular $(-20, -10, -5)$.
   - Prosiga a su ritmo (PR): función de pertenencia triangular $(-5, 0, 5)$.
   - Aceleración leve (AL): función de pertenencia triangular $(5, 30, 50)$.
   - Aceleración fuerte (AF): función de pertenencia triangular $(40, 75, 100)$.

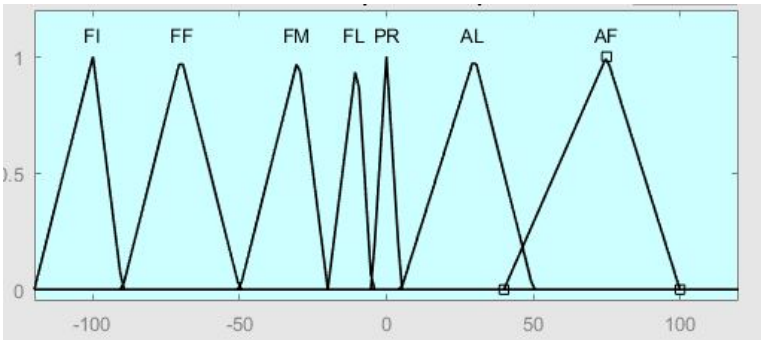


Figura 3.8: Funciones de pertenencia variable Frenada

Las reglas difusas en las que se basará el sistema se muestran en la Tabla 3.2

	Velocidad				
Distancia	***Muy Baja***	***Baja***	***Normal***	***Alta***	***Muy Alta***
Muy Corta	FL	FM	FM	FF	FI
Corta	FL	FL	FM	FF	FI
Normal	PR	PR	PR	FL	FM
Larga	AF	AL	AL	FM	FL
Muy Larga	AF	AF	AL	FM	FM

Tabla 3.2: Reglas IF-THEN del sistema.

Una vez establecidos los conjuntos difusos y las reglas, podemos proceder a la inferencia. Supondremos que los valores de entrada son una velocidad de 110km/h y una distancia con el coche de delante de 40.5m.
Como ejemplo, de las veinticinco reglas que tenemos, tomaremos la siguiente:

If Velocidad es Muy Alta AND Distancia es Larga THEN Frenada es Leve.

Ahora corresponde preguntarnos, de acuerdo con esta regla, ¿cuánto pertenece una velocidad de 110 a una velocidad muy alta? y ¿cuánto pertenece una distancia de 40.5 a distancia larga?. Para comprobarlo no tenemos más que evaluar las funciones de pertenencia correspondientes.

$$f_{MA}(x) = \begin{cases} 0 & \text{si } x \leq 90 \\ \frac{x-90}{100-90} & \text{si } 90 < x \leq 100 \\ 1 & \text{si } 100 < x \leq 110 \\ \frac{120-x}{120-110} & \text{si } 110 < x < 120 \\ 0 & \text{si } 120 \leq x \end{cases}$$

$$f_{L}(x) = \begin{cases} 0 & \text{si } x \leq 30 \\ \frac{x-30}{40-30} & \text{si } 30 < x \leq 40 \\ \frac{50-x}{50-40} & \text{si } 40 < x < 50 \\ 0 & \text{si } 50 \leq x \end{cases}$$

Por tanto:

$$f_{MA}(110) = 1 \text{ y } f_{L}(40.5) = 0.95$$

En el proceso de fuzzificación usaremos el conectivo OR para obtener valor de verdad de la regla, es decir nos quedamos con el máximo de los dos valores, que en este caso es 1.
Como vimos anteriormente, hay dos formas posibles para elaborar el conjunto difuso de salida a partir de los valores de verdad de cada regla, recorte y escalado. En este caso *Matlab* usa *recorte*. En el caso de la *defuzzificación*, la realizamos por el método del centroide. Tras dichos procedimientos, obtenemos los resultados que se muestran en la Figura 3.9.

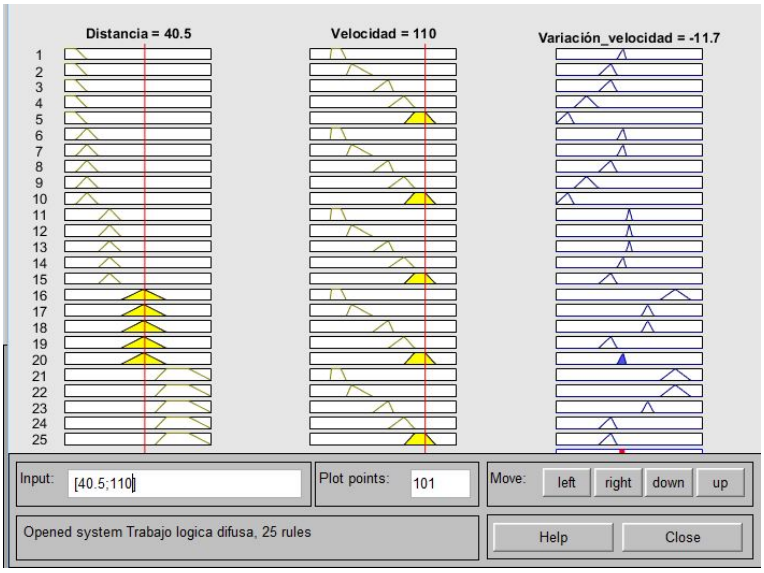


Figura 3.9: Resultados finales

Podemos observar que el controlador nos indica que si vamos a una velocidad de 110km/h y el coche de delante se encuentra a una distancia de 40.5m, entonces debemos bajar 11.7km/h la velocidad.

3.5.4 Sistema de inferencia de Takagi-Sugeno

Este método fue propuesto en 1985 por Takagi y Sugeno (19) como alternativa al método de Mamdani. Este método resulta muy útil cuando tratamos sistemas complejos y de dimensiones mayores que los que podemos resolver mediante el método de Mamdani.

La peculiaridad de este método reside en que el consecuente no es un conjunto difuso sino una función lineal, por tanto la forma general de las reglas es la siguiente:

$$\text{IF } x \text{ es } A \text{ AND } y \text{ es } B \text{ THEN } z = f(x, y)$$

donde A y B son conjuntos difusos. Como hemos mencionado, $f(x, y)$ es una función lineal, por lo tanto no se necesita de un proceso de *defuzzificación*.

Es usual, aunque no obligatorio, que las funciones del consecuente sean polinómicas. Se dirá que tenemos modelo difuso de Sugeno de orden n si $f(x, y)$ es un polinomio de orden n. Podemos destacar algunos casos concretos. Si $f(x, y)$ es un polinomio de orden cero, es decir, una constante, el sistema puede considerarse o bien un caso especial de los sistemas de inferencia de Mamdani donde los consecuentes de cada regla viene dado por una función *singleton*.

De esta manera, una regla típica en los modelos de Sugeno tendría la siguiente forma:

$$\text{IF Entrada } 1 = x \text{ AND Entrada } 2 = y, \text{ THEN Salida es } z = ax + by + c$$

Nótese, que si el sistema fuera de orden cero entonces debería ser $a = b = 0$. Al ser una función lineal, y ahorrarnos la *defuzzificación*, la salida se obtiene directamente mediante la siguiente expresión:

$$z = \frac{\sum_{i=1}^{n} w_i f_i(x_i, y_i)}{\sum_{i=1}^{n} w_i}$$

donde el valor w_i viene dado por el mínimo de los valores de entrada en cada regla R_i, como se observa en la siguiente figura:

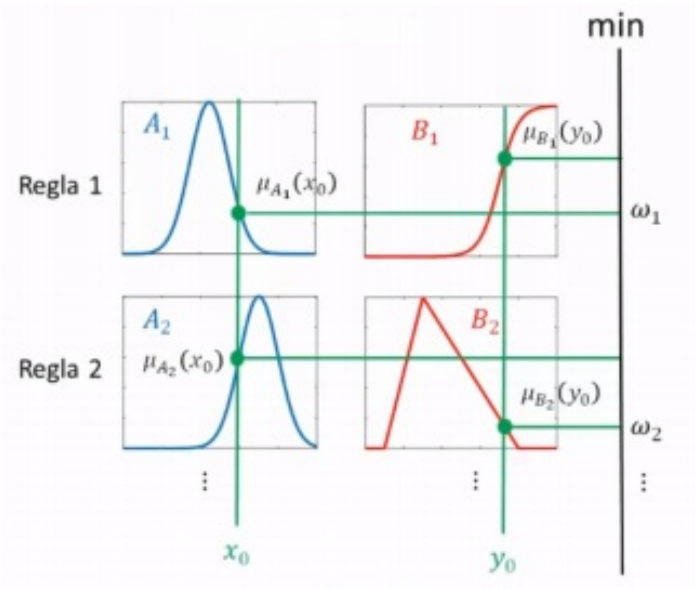


Figura 3.10: Calculo del peso de cada regla en el sistema de inferencia de Takagi-Sugeno.

Vamos a ver cómo funciona el método de Sugeno con un ejemplo muy sencillo.

Ejemplo 3.2. Un equipo de fútbol de élite desea evaluar el riesgo de lesión de sus jugadores en función de los minutos que juegan cada semana, para así poder prevenir alguna de ellas. Teniendo en cuenta que lo normal en este tipo de equipos es jugar tres partidos por semana de media, que supone un total de 360 minutos, se definen las siguientes etiquetas lingüísticas para la variable *Tiempo de juego* (medida en minutos por semana):

- *Tiempo de juego Escaso*: función de pertenencia triangular truncada (0,60).
- *Poco tiempo de juego*: función de pertenencia triangular truncada (0,120).

- *Tiempo de juego Normal*: función de pertenencia triangular truncada (90,120,270).
- *Mucho tiempo de juego*: función de pertenencia trapezoidal truncada a la derecha (200,320).
- *Demasiado tiempo de juego*: función de pertenencia trapezoidal truncada a la derecha (225,450).

Podemos ver dichas funciones de pertenencia en la Figura 3.11:

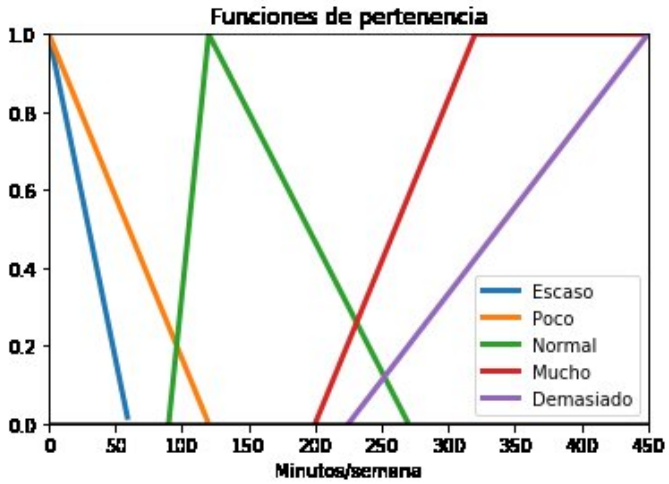


Figura 3.11: Funciones de pertenencia correspondientes a Tiempo de juego.

Además en la Figura 3.12 viene representada la relación entre los minutos jugados y la probabilidad de lesión.

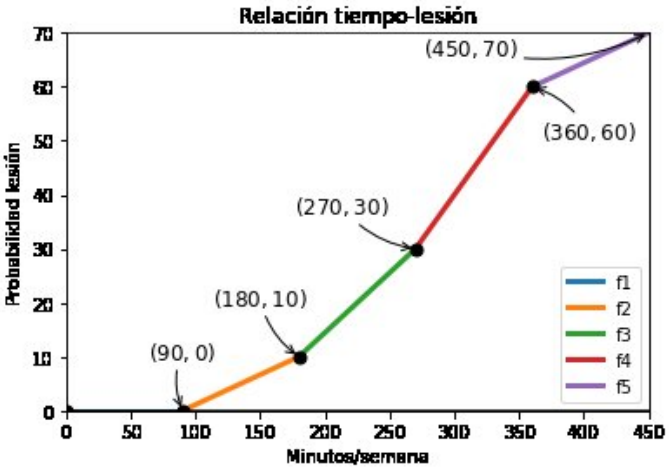


Figura 3.12: Relación entre probabilidad de lesión y tiempo jugado

A partir de dicha relación y de los conjuntos difusos podemos establecer las reglas del sistema:

IF	Tiempo de juego Escaso	THEN	$z = f_1(x) = 0$
IF	Tiempo de juego Poco	THEN	$z = f_2(x) = \frac{x}{9} - 10$
IF	Tiempo de juego Normal	THEN	$z = f_3(x) = \frac{2x}{9} - 30$
IF	Tiempo de juego Mucho	THEN	$z = f_2(x) = \frac{x}{3} - 60$
IF	Tiempo de juego Demasiado	THEN	$z = f_5(x) = \frac{x}{9} + 20$

Tabla 3.3: Reglas IF-THEN para el método de Sugeno

Supongamos ahora que queremos evaluar el riesgo de lesión de un futbolista que ha jugado 210 minutos en una semana. Evaluamos en primer lugar las funciones de pertenencia:

$$\begin{aligned} f_E(210) &= 0 \\ f_P(210) &= 0 \\ f_N(210) &= 0,4 \\ f_M(210) &= 0,08333 \\ f_D(210) &= 0 \end{aligned}$$

Puesto que solo tenemos un valor de entrada y w_i viene determinado por el mínimo de las variables de entrada es inmediato que:

$$\begin{aligned} w_1 &= 0 \\ w_2 &= 0 \\ w_3 &= 0,4 \\ w_4 &= 0,08333 \\ w_5 &= 0 \end{aligned}$$

Por lo tanto:

$$z = \frac{0 \cdot f_1(210) + 0 \cdot f_2(210) + 0.4 \cdot f_3(210) + 0.08333 \cdot f_4(210) + 0 \cdot f_5(210)}{0 + 0 + 0.4 + 0.08333 + 0} = 15,52$$

Por tanto, de acuerdo al sistema de inferencia de Sugeno, un futbolista que ha jugado 210 min en una semana tendrá una probabilidad de lesión del $15,52\,\%$ aproximadamente.

3.5.5 Ejemplo práctico con los métodos de Mamdani y Sugeno

A continuación, presentamos un ejemplo práctico en el que se aplican los dos métodos de inferencia difusa estudiados en este capítulo. El problema en cuestión se trata de un sistema

de inferencia para la evaluación del riesgo de padecer cáncer de mama (18). Dicho sistema de inferencia tendrá dos variables de entrada que serán *Edad del paciente* y *Tamaño del tumor*. La variable de salida será el *Riesgo de cáncer de mama*.

En primer lugar llevaremos a cabo el proceso común de ambos métodos, que será la *fuzzificación* y la formación de reglas.

Para empezar, definiremos los conjuntos difusos así como las funciones de pertenencia de cada variable:

- Edad: La medida de esta variable vendrá dada en un rango de 0 a 80 años. Le asociaremos tres conjuntos difusos a los que les corresponderán las etiquetas lingüísticas "Joven", "Adulto" y "Anciano". Todos estos conjuntos tienen una función de pertenencia gaussiana que se presenta en la Figura 3.13.

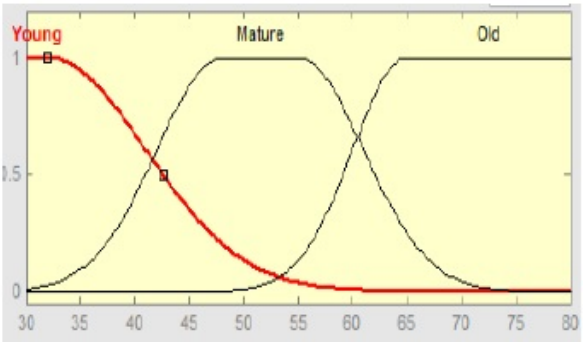


Figura 3.13: Funciones de pertenencia de la variable Edad.

- Tamaño: El tamaño vendrá medido entre 0 y 8000 píxeles. Las etiquetas lingüísticas en este caso son "Pequeño", "Medio" y "Grande". Al igual que en el caso anterior a las etiquetas se les asocian las funciones gaussianas, representadas en la Figura 3.14.

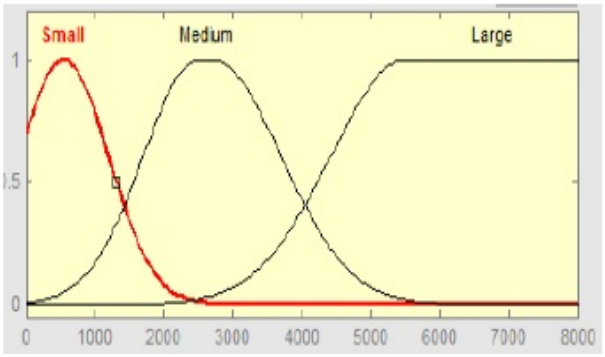


Figura 3.14: Funciones de pertenencia de la variable Tamaño.

- Riesgo: Por último, la variable de salida tendrá las etiquetas lingüísticas “Bajo”, “Medio” y “Alto” que de nuevo tendrán asociadas funciones de pertenencia gaussianas, mostradas en la Figura 3.15.

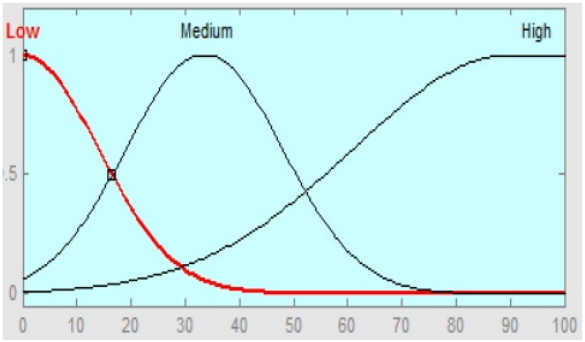


Figura 3.15: Funciones de pertenencia de la variable Riesgo

A continuación presentaremos las reglas difusas en la Tabla 3.4.

IF	Edad es Joven	AND	Tamaño es Pequeño	THEN	Riesgo es Bajo
IF	Edad es Adulta	AND	Tamaño es Pequeño	THEN	Riesgo es Bajo
IF	Edad es Anciano	AND	Tamaño es Pequeño	THEN	Riesgo es Bajo
IF	Edad es Joven	AND	Tamaño es Medio	THEN	Riesgo es Medio
IF	Edad es Adulta	AND	Tamaño es Medio	THEN	Riesgo es Medio
IF	Edad es Anciano	AND	Tamaño es Medio	THEN	Riesgo es Medio
IF	Edad es Joven	AND	Tamaño es Grande	THEN	Riesgo es Alto
IF	Edad es Adulta	AND	Tamaño es Grande	THEN	Riesgo es Alto
IF	Edad es Anciano	AND	Tamaño es Grande	THEN	Riesgo es Alto

Tabla 3.4: Reglas del sistema de inferencia

Nótese, que en el caso del método de Sugeno, la salida solo puede ser una función constante o lineal. Además será un valor entre 0 y 1 por lo que asociaremos un valor de este intervalo a cada etiqueta lingüística de la variable riesgo como se muestra en la Tabla 3.5.

Riesgo de cáncer	**Valor constante**
Bajo	0
Medio	0.5
Alto	1

Tabla 3.5: Valores de las etiquetas lingüísticas de la variable Riesgo.

Comentaremos ahora los resultados obtenidos una vez simulados ambos métodos mediante la aplicación de lógica difusa que ofrece el software *Matlab*.

En la Figura 3.16 podemos ver la relación existente entre la edad y el tamaño del tumor del paciente con el riesgo de padecer cáncer. Mediante cortes adecuados entre hiperplanos y planos podemos ver la relación directa de cada variable de entrada con el riesgo de cáncer. En la Figura 3.17 se observa que en principio la edad no es un factor determinante ya que la función es constante. Era de esperar que fuera así debido a la forma en la que están tomadas las reglas (si nos fijamos en la Tabla 3.4 observamos que la etiqueta solo cambia al cambiar el tamaño del tumor mientras que la edad no influye) y por la sencillez del problema para que sea más fácil la comprensión del mismo.

En cambio, como podemos ver en la Figura 3.18, la relación entre el tamaño del tumor y el riesgo sí que es directamente proporcional.

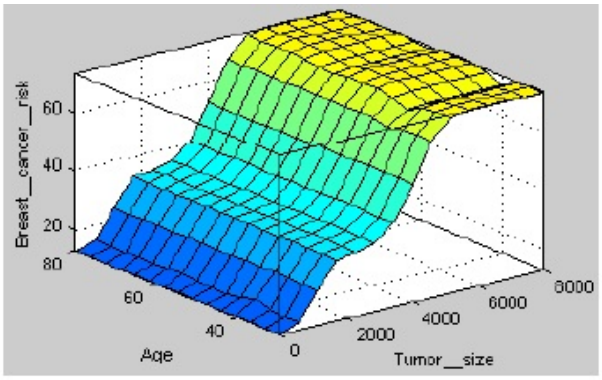


Figura 3.16: Relación entre variables de entrada y salida en método de Mamdani

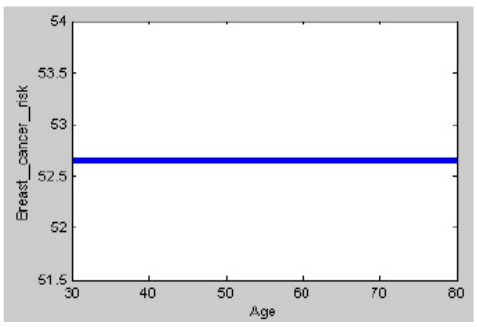


Figura 3.17: Relación entre la variable Edad y la variable Riesgo según método de Mamdani

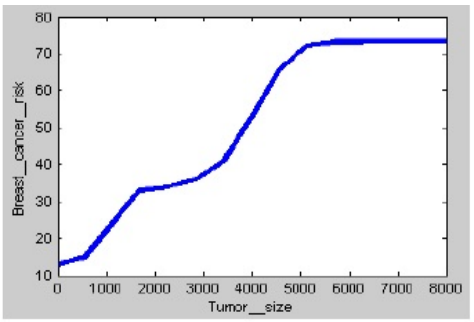


Figura 3.18: Relación entre la variable Tamaño y la variable Riesgo según método de Mamdani

Veremos ahora exactamente las mismas relaciones pero esta vez obtenidas mediante el método de Sugeno:

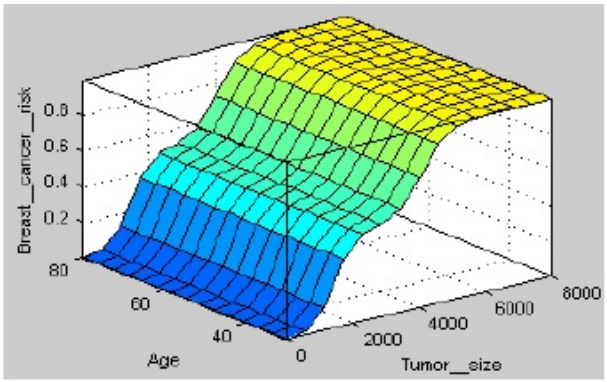


Figura 3.19: Relación entre variables de entrada y salida en método de Sugeno

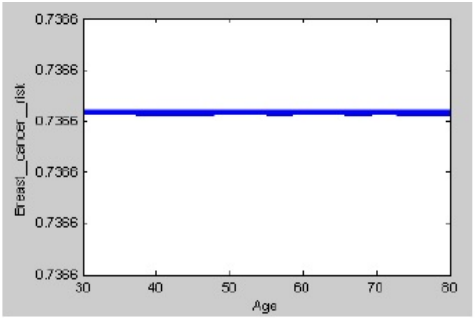


Figura 3.20: Relación entre la variable Edad y la variable Riesgo según método de Sugeno

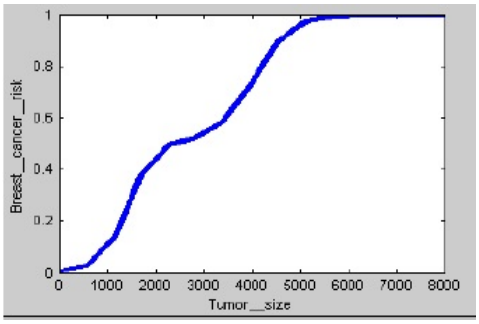


Figura 3.21: Relación entre la variable Tamaño y la variable Riesgo según método de Sugeno

De nuevo, como era de esperar, obtenemos que no hay relación entre edad y riesgo de cáncer y que en cambio sí que la hay (directamente proporcional) entre tamaño del tumor y riesgo de cáncer.

En general podemos observar resultados bastante similares, quizá podamos destacar que en la relación entre tamaño y riesgo aparecen menos picos mediante el método de Sugeno que mediante el método de Mamdani (la Figura 3.21 nos muestra una curva más "cercana a ser diferenciable" que la de la Figura 3.18).

3.5.6 Comparativa entre los métodos de Sugeno y de Mamdani

La mayor diferencia entre los métodos de inferencia difusa estudiados reside en que en el método de Sugeno encontramos funciones lineales en los consecuentes. Aunque también difieren ligeramente, al tener diferentes consecuentes, en los procesos de *fuzzificación* y *defuzzificación*. El número de valores de entrada que se necesitan en el método de Sugeno depende del número y localización de los extremos de las funciones de pertenencia. En el método de Sugeno se debe emplear una gran cantidad de reglas difusas para aproximar funciones periódicas u oscilatorias. Normalmente los controladores que usan el método de Sugeno tienen parámetros mas ajustables en los consecuentes, además el número de parámetros crece exponencialmente con respecto al número de variables de entrada. Si hablamos de resultados matemáticos, son muchos más los que existen para el método de Mamdani en comparación a los que existen para el método de Sugeno. También se puede decir que el método de Mamdani es más fácil de formular que el de Sugeno.

Estableceremos ahora una lista con las ventajas que conlleva usar cada método:

- **Ventajas del método de Sugeno:**

  1. Es computacionalmente eficiente.
  2. Funciona bien con técnicas lineales como por ejemplo los controladores PID[1].
  3. Funciona bien con técnicas adaptativas y de optimización.
  4. Garantiza continuidad en la superficie de salida.
  5. Se adapta bien al análisis matemático.

- **Ventajas del método de Mamdani:**

  1. Es bastante intuitivo.
  2. Está ampliamente reconocido, lo cual facilita la búsqueda de información relacionada.
  3. Encaja muy bien con el pensamiento humano. Es decir, permite la formulación de problemas cotidianos.

Es interesante destacar también algunos de los inconvenientes de cada método:

- **Inconvenientes del método de Sugeno:**

  1. Es menos estudiado en la biliografía.
  2. Su uso es imposible en la práctica cuando no podemos trabajar con conjuntos de datos discretos para lograr el diseño de funciones lineales.

- **Inconvenientes del método de Mamdani:**

  1. El número de reglas aumenta exponencialmente con el número de variables de entrada.
  2. Cuantas más reglas más difícil será saber sin son adecuadas.
  3. Si el número de variables de entrada es elevado, será complicado averiguar la relación entre antecedentes y consecuentes de manera que, las reglas serán difíciles de construir.

En general, la elección del método va a venir determinada por el acceso a los datos que tengamos.

[1]Controlador de acción proporcional, integral y derivativa. Es un mecanismo de control simultáneo por realimentación ampliamente usado en sistemas de control industrial. Este calcula la desviación o error entre un valor medido y un valor deseado.

CAPÍTULO

4

Programación lógica multiadjunta

La programación lógica multiadjunta surge como una generalización de las diferentes aproximaciones no clásicas a la programación lógica, obviando en ellas sus particularidades y preservando únicamente los componentes matemáticos básicos para garantizar la operabilidad. En general para este capítulo seguiremos el esquema del artículo donde se introdujo por primera vez la programación lógica multiadjunta (12). También nos apoyaremos en tesis doctorales y trabajos que han investigado en este campo ((15) (2) (14)).

4.1 Conjuntos ordenados, retículos y puntos fijos

Debido a que los operadores algebraicos con los que trabajaremos posteriormente se definen sobre conjuntos parcialmente ordenados y sobre retículos, vamos a proceder a introducir dichas nociones. Pero primero recordemos qué es un orden parcial.

Definición 4.1. Decimos que una relación binaria $\leq$ sobre un conjunto arbitrario P es un *orden parcial* si se satisfacen las siguientes propiedades:

1. Reflexiva: $x \leq x$, con $x \in P$;
2. Antisimétrica: Si $x \leq y$ e $y \leq x$ entonces necesariamente $x = y$, con $x, y \in P$;
3. Transitiva: Si $x \leq y$ e $y \leq z$ entonces $x \leq z$, para $x, y, z \in P$.

Además, diremos que el *orden es total* si todos los elementos del conjunto están relacionados entre sí.

Definición 4.2. Dado un conjunto arbitrario P y un orden parcial $\leq$ sobre dicho conjunto, llamaremos *conjunto parcialmente ordenado (poset)* al par $(P, \leq)$.

Ejemplo 4.1. El par $(\mathbb{N} \setminus \{0\}, \leq)$ es un conjunto parcialmente ordenado donde la relación de orden parcial $\leq$ se define como sigue: dados $x, y \in \mathbb{N} \setminus \{0\}$, decimos que $x \leq y$ si y solo si x es divisor de y.

Definición 4.3. Sea $(P, \leq)$ un conjunto parcialmente ordenado.

- Si para cualquier par de elementos $x, y \in P$ existe el supremo y el ínfimo del conjunto $\{x, y\}$, entonces $(P, \leq)$ es un *retículo*.
- Si para cualquier subconjunto $K \subseteq P$ existen el supremo y el ínfimo de K, entonces $(P, \leq)$ es un *retículo completo*.

Nótese que todo retículo completo tiene un elemento máximo y un elemento mínimo, los cuales se denotarán como $\top$ y $\bot$, respectivamente.

Lo veremos más claro con el ejemplo que presentamos a continuación.

Ejemplo 4.2. Los *diagramas de Hasse*[1] mostrados en la Figura 4.2 representan diferentes ordenaciones del conjunto $\{a, b, c, d, e\}$.

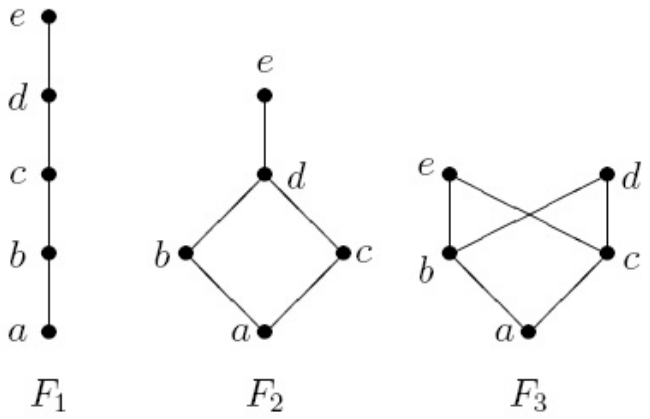


Figura 4.1: Diagramas de Hasse del Ejemplo 4.2

[1]Estos diagramas se usan a menudo para representar conjuntos parcialmente ordenados finitos. Consisten en vértices conectados mediante aristas. Los vértices representan elementos del conjunto parcialmente ordenado y cada arista ascendente entre dos vértices x e y se interpreta entendiendo que $x \leq y$.

Si nos fijamos en los diagramas F_1 y F_2 podemos comprobar que existen tanto el supremo como el ínfimo de cualquier par de sus elementos, por tanto ambos son retículos, más aun, son retículos completos. Sin embargo, esto no se cumple en el diagrama F_3. Atendiendo a los elementos b y c de F_3, observamos que éstos tienen ínfimo pero no supremo. Por tanto, podemos concluir que F_3 no es un retículo.

También podemos observar que F_1 es el único diagrama qu no contiene elementos incomparables, por tanto se trata de un conjunto totalmente ordenado.

Es conveniente ahora exponer una serie de definiciones y resultados relacionados con la teoría de puntos fijos, ya que nos serán útiles cuando estudiamos la semántica de los programas lógicos multiadjuntos. En primer lugar, introducimos una definición que generaliza el concepto de monotonía.

Definición 4.4. Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación. Decimos que T es *monótona* si para todo $x, y \in L$, tales que $x \preceq y$ se tiene que $T(x) \preceq T(y)$.

Para definir la noción de aplicación continua sobre retículos, necesitamos considerar el concepto de conjunto dirigido.

Definición 4.5. Sea $(L, \preceq)$ un retículo y $D \subseteq L$. Decimos que D es un *conjunto dirigido* si cada uno de los subconjuntos finitos de D tiene cota superior en D.

Definición 4.6. Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación. Decimos que T es *continua* si para cada subconjunto dirigido D de L se verifica que

$$T(sup(D)) = sup(T(D))$$

Una vez generalizados los conceptos de monotonía y continuidad, podemos pasar a centrarnos en la teoría de puntos fijos.

Definición 4.7. Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación. Decimos que: $a \in L$ es el *punto fijo mínimo* o el *menor punto fijo* de T si es un punto fijo (es decir $T(a) = a$) y para cualquier otro punto fijo b de T se cumple $a \preceq b$.
Análogamente, dado un punto fijo $a' \in L$ tal que para cualquier otro elemento $b' \in L$ verifica que $b' \preceq a'$, entonces a' recibe el nombre de *punto fijo máximo* o *mayor punto fijo* de T.

El siguiente resultado caracteriza los puntos fijos mínimos y máximos como ínfimos y supremos, respectivamente, de ciertos conjuntos. Concretamente, la siguiente proposición fue demostrada por Tarski en 1955 (20).

Proposición 4.1. *Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación monótona. Entonces T tiene un punto fijo mínimo, al que denotamos lfp(T), y un punto fijo máximo, gfp(T) (esta notación viene del inglés least/greatest fix point). Además se verifica*

$$\mathit{lfp}(T) = \mathit{ínf}\{x \in L \mid T(x) = x\} = \mathit{ínf}\{x \in L \mid T(x) \leq x\}$$

$$\mathit{gfp}(T) = \mathit{sup}\{x \in L \mid T(x) = x\} = \mathit{sup}\{x \in L \mid x \leq T(x)\}$$

Proposición 4.2. *Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación monótona. Supongamos que $a \in L$ es tal que $a \leq T(a)$. Entonces existe un punto fijo $a' \in L$ de T tal que $a \leq a'$.*
De igual forma, si $b \in L$ verifica $T(b) \leq b$, entonces existe un punto fijo $b' \in L$ tal que $b' \leq b$.

Con el objetivo de dar una visión constructivista de tales puntos fijos como límites ordinales, introduciremos el concepto de potencia ordinal de T y recordamos algunas propiedades elementales de los números ordinales.

Comencemos presentando ahora el concepto de ordinal. Intuitivamente los ordinales son los que usamos para contar. Es claro entonces que cada uno de los números enteros no negativos es un número ordinal (denota posición). Recordemos la siguiente definición de números naturales:

- $0 = \emptyset$
- $1 = \{\emptyset\}$
- $2 = \{\emptyset, \{\emptyset\}\}$

y así sucesivamente. De aquí que se pueda definir el orden $<$ en el conjunto de todos los ordinales finitos como sigue: decimos entonces que dados dos ordinales finitos n y m, se tiene que $n < m$ si y sólo si $n \in m$. Más aun, el primer ordinal infinito es el conjunto de todos los enteros no negativos $\omega = \{0, 1, 2, 3...\}$. Denotamos, de ahora en adelante, a cualesquiera dos ordinales, ya sean finitos o no, como α y β. Estamos ahora en disposición de definir los siguientes conceptos:

Definición 4.8. Se define el *ordinal sucesor* de un ordinal α dado como $\alpha + 1 = \alpha \cup \{\alpha\}$. Se puede concluir entonces que dado un ordinal sucesor $\alpha = \beta + 1$, denotamos a β por $\alpha - 1$.
Por ejemplo, $1 = 0 + 1$, $2 = 1 + 1$, $3 = 2 + 1$, y así sucesivamente.

Definición 4.9. Decimos que un ordinal dado α es un *ordinal límite* si no es sucesor de ningún ordinal.
Ciñéndonos a lo anterior es inmediato comprobar que 0 y ω son los ordinales límite más pequeños, después viene $\omega + 1 = \omega \cup \{\omega\}$, y así sucesivamente.

Es momento de introducir la definición de potencia de ordinal de una aplicación monótona.

Definición 4.10. Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación monótona. Se define:

$$\begin{aligned}
T \uparrow 0 &= \bot \\
T \uparrow \alpha &= T(T \uparrow (\alpha - 1)), \text{ si } \alpha \text{ es un ordinal sucesor} \\
T \uparrow \alpha &= sup\{T \uparrow \beta \mid \beta < \alpha\}, \text{ si } \alpha \text{ es un ordinal límite} \\
T \downarrow 0 &= \top \\
T \downarrow \alpha &= T(T \downarrow (\alpha - 1)), \text{ si } \alpha \text{ es un ordinal sucesor} \\
T \downarrow \alpha &= sup\{T \downarrow \beta \mid \beta < \alpha\}, \text{ si } \alpha \text{ es un ordinal límite}
\end{aligned}$$

donde $\top$ y $\bot$ son el elemento máximo y mínimo de L.

Ahora ya estamos en disposición de enunciar el siguiente teorema de Knaster-Tarsky, que proporciona una caracterización de los puntos fijos máximo y mínimo.

Teorema 4.1. *Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación monótona. Entonces, para cualquier ordinal α, $T \uparrow \alpha \leq lgp(T)$ y $gfp(T) \leq T \downarrow \alpha$. Además, existen dos ordinales β_1 y β_2 tales que para cualquier γ_1 verificando $\beta_1 \leq \gamma_1$, se tiene que $T \uparrow \gamma_1 = lfp(T)$ y para cualquier γ_2 verificando $\beta_2 \leq \gamma_2$, se tiene que $T \downarrow \gamma_2 = gfp(T)$.*

Para terminar con la teoría de puntos fijos veremos una proposición que nos permite encontrar el menor punto fijo de una aplicación continua en un número finito de pasos. Este resultado será de utilidad cuando trabajemos con el operador de consecuencias inmmediata del marco de trabajo de programación lógica multiadjunta.

Proposición 4.3. *Sea $(L, \preceq)$ un retículo completo y sea $T\colon L \to L$ una aplicación continua. Entonces lfp(T) = $T \uparrow \omega$.*

Curiosamente, no es cierto el resultado análogo pero en términos del mayor punto fijo.

4.2 Sintaxis y semántica del lenguaje proposicional

En esta sección se introducen algunas nociones y resultados relacionados con el lenguaje proposicional utilizado en la programación lógica multiadjunta. Concretamente, la sintaxis del lenguaje proposicional usado en programación lógica multiadjunta se basa en los conceptos del alfabeto y las expresiones del lenguaje. Estos conceptos requieren el uso de algunas definiciones de álgebra universal como las que se muestran a continuación.

Definición 4.11. Llamaremos *conjunto graduado* a un conjunto Ω con una función que asigna a cada elemento $\omega \in \Omega$ un número $n \geq 0$ al que llamaremos *aridad* de ω.

Considerando un conjunto graduado, podemos generalizar las nociones de estructura algebraica y substructura de una estructura algebraica mediante las siguientes definiciones.

Definición 4.12. Dado un conjunto graduado Ω, una *Ω-álgebra* $\mathfrak{A}$ es un par $\langle A, K\rangle$ donde A es un conjunto no vacío al que llamaremos soporte e K es una aplicación que asigna una función a cada uno de los elementos de Ω como sigue:

1. Cada operador $\omega \in \Omega_n$, $n > 0$ se interpreta como una aplicación $K(\omega) : A^n \to A$, que se denota como $\omega_{\mathfrak{A}}$.
2. Cada elemento $c \in \Omega_0$, es decir c es una constante, será interpretado como un elemento $K(c) \in A$, y lo denotamos como $c_{\mathfrak{A}}$.

Definición 4.13. Dada una Ω-álgebra $\mathfrak{A} = \langle A, K\rangle$, llamaremos *$\Omega$-subálgebra* $\mathfrak{B}$ al par $\langle B, K\rangle$ tal que $B \subseteq A$ y además

1. $J(c) = K(c)$ para todo $c \in \Omega_0$.
2. Dado $\omega \in \Omega_n$, se tiene que $J(\omega) : B^n \to B$ es la restricción de $K(\omega) : A^n \to A$

Estamos en disposición de dar la definición de *alfabeto asociado*.

Definición 4.14. Sea Ω un conjunto graduado y Π un conjunto infinito numerable, cuyos elementos recibirán el nombre de *átomos*. Definiremos el *alfabeto asociado* a Ω y Π, denotado como $A_{\Omega,\Pi}$ como la unión disjunta de ambos conjuntos y el conjunto S de símbolos auxiliares "(", ")" y ",".

En lo que sigue, nos referiremos a un alfabeto simplemente como A_Ω.

Definición 4.15. Dados un conjunto graduado Ω y un alfabeto A_Ω. Se define una *Ω-álgebra de expresiones*, denotada por $\mathfrak{E} = \langle A^*_\Omega, I\rangle$ de la siguiente forma:

1. El soporte A^*_Ω, es el conjunto formado por por todas las cadenas con elementos de A_Ω.

2. La función I recibe el nombre de *función de interpretación* y viene dada para cualesquiera $c, \omega \in \Omega$ y $a_1, ..., a_n \in A^*_\Omega$ de la siguiente manera:

   - $c_{\mathfrak{E}} = c$, para cualquier constante $c \in \Omega_0$.
   - $\omega_{\mathfrak{E}}(a_1) = \omega a_1$, para cualquier operador $\omega \in \Omega_1$.
   - $\omega_{\mathfrak{E}}(a_1, a_2) = (a_1 \omega a_2)$, para cualquier operador $\omega \in \Omega_2$.
   - $\omega_{\mathfrak{E}}(a_1, a_2, ..., a_n) = \omega(a_1, a_2, ..., a_n)$, para cualquier operador $\omega \in \Omega_n$, con $n > 2$.

Las definiciones anteriores nos ponen en contexto para definir un álgebra de fórmulas.

Definición 4.16. Sean Ω un conjunto graduado, Π un conjunto numerable de átomos y $\mathfrak{E}$ el álgebra de expresiones correspondientes al alfabeto A_Ω. Entonces el lenguaje de primer orden generado por Ω sobre Π es un subálgebra del álgebra de expresiones, a la que denotaremos $\mathfrak{F}$. Ésta se llama *Ω-álgebra de fórmulas bien formadas* o simplemente *Ω-álgebra de fórmulas*, y naturalmente sus elementos reciben el nombre de fórmulas.

4.3 Conectivos lógicos generalizados

La programación lógica admite multitud de conectivos lógicos. Pero como cabe esperar, empezaremos generalizando el conjuntor y el disyuntor.

Definición 4.17. Un *operador conjuntor* definido sobre un conjunto parcialmente ordenado $(P, \leq)$ es un operador $\wedge\colon P^2 \to P$ tal que, para todo $x, y, z \in P$, se verifican las siguientes propiedades:

1. $(x \wedge y) \wedge z = x \wedge (y \wedge z)$. El operador cumple la *propiedad asociativa*.
2. Si $x \leq y$, entonces se cumplen $(x \wedge z) \leq (y \wedge z)$ y $(z \wedge x) \leq (z \wedge y)$. El operador cumple la *propiedad de monotonía*.
3. $(x \wedge \top) = (\top \wedge x) = x$. El *elemento neutro* es $\top$.
4. $(x \wedge \bot) = (\bot \wedge x) = \bot$. El *elemento anulador* es $\bot$.

Definición 4.18. Un *operador disyuntor* definido sobre un conjunto parcialmente ordenado $(P, \leq)$ es un operador $\vee : P^2 \to P$ tal que, para todo $x, y, z \in P$, se verifican las siguientes propiedades:

1. $(x \vee y) \vee z = x \vee (y \vee z)$. El operador cumple la *propiedad asociativa*.
2. Si $x \leq y$, entonces se cumplen $(x \vee z) \leq (y \vee z)$ y $(z \vee x) \leq (z \vee y)$. El operador cumple la *propiedad de monotonía*.

3. $(x \vee \bot) = (\bot \vee x) = x$. El *elemento neutro* es $\bot$.

4. $(x \vee \top) = (\top \vee x) = \top$. El *elemento anulador* es $\top$.

Los operadores t-norma y t-conorma que vimos en el Capítulo 2 son casos particulares de los operadores conjuntor y disyuntor. Como ya mencionamos entonces, la función de estos operadores es la de evitar la pérdida de información. No obstante, no son los únicos operadores con estas características, aunque sí los más comunes. Presentaremos ahora una generalización de los operadores con estas características, los cuales reciben el nombre de operadores de agregación.

Definición 4.19. Sea $(P, \leq, \bot, \top)$ un conjunto parcialmente ordenado y acotado, un *operador de agregación n-ario* $@\colon P^n \to P$ es un operador monótono satisfaciendo que $@(\top, ..., \top) = \top$ y $@(\bot, ..., \bot) = \bot$.

Por tanto un operador de agregación puede ser cualquier aplicación monótona cuyos argumentos son valores de un conjunto de un conjunto parcialmente ordenado acotado.
En el siguiente ejemplo veremos los operadores de agregación más usuales.

Ejemplo 4.3. La *media aritmética* M, la *media geométrica* G y la *media armónica* H, definidas en el intervalo unidad como:

$$\begin{aligned} M(x_1, ..., x_n) &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ G(x_1, ..., x_n) &= (\prod_{i=1}^{n} x_i)^{\frac{1}{n}} \\ H(x_1, ..., x_n) &= \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \end{aligned}$$

son operadores de agregación.

4.4 Programación lógica multiadjunta

El marco multiadjunto surge como una generalización de distintos ambientes de programación lógica no clásica cuya estructura semántica es la retículo multiadjunto ((4) (5) (12)). Para definir un retículo multiadjunto, necesitamos introducir el concepto de par adjunto que fue presentado por primera vez en un contexto lógico por Pavelka (13).

Definición 4.20. Sea $(P, \leq)$ un conjunto parcialmente ordenado y sea $(\&, \leftarrow)$ un par de operaciones binarias en P tales que verifican las siguientes propiedades:

- $\&$ es creciente en ambos argumentos, esto es, dados $x, y, z \in P$ tal que $x \leq y$, entonces se verifican $(x\&z) \leq (y\&z)$ y $(z\&x) \leq (z\&y)$
- $\leftarrow$ es creciente en el primer argumento (que recibe el nombre de consecuente) y decreciente en el segundo (que se llama antecedente), es decir, dados $x, y, z \in P$ tal que $x \leq y$, entonces se verifican $(x \leftarrow z) \leq (y \leftarrow z)$ y $(z \leftarrow y) \leq (z \leftarrow x)$
- Para cualesquiera $x, y, z \in P$ se tiene que

$$x \leq (y \leftarrow z) \text{ si y solo si } (x\&z) \leq y$$

Entonces decimos que $(\&, \leftarrow)$ forma un *par adjunto* en $(P, \leq)$

Como ejemplo de pares adjuntos podemos remitirnos al Capítulo 2 y comprobar que las t-normas continuas a la izquierda y sus implicaciones residuadas son un caso particular de los de los pares adjuntos.

Observe que, la monotonía de los operadores $\&$ y $\leftarrow$ está justificada porque se interpretarán como conjunciones e implicaciones generalizadas. Es importante resaltar que $\&$ no necesita ser conmutativo o asociativo, y no se requieren tampoco condiciones de frontera.

Además de las propiedades exigidas en la definición anterior, debemos asumir la existencia de los elementos máximo y mínimo en el conjunto parcialmente ordenado de valores de verdad, y la existencia de uniones para cada subconjunto dirigido, es decir, asumiremos un retículo completo.

El uso de distintas implicaciones y el modus ponens como reglas de inferencia para extender la teoría desarrollada en (6) (21) a un entorno más general, da lugar a considerar varios pares adjuntos en el retículo.

Definición 4.21. Sea $(L, \leq)$ un retículo. Se define un *retículo multiadjunto* $\mathcal{L}$ como una tupla $(L, \preceq, \&_1, \leftarrow_1, ..., \&_n, \leftarrow_n)$ que satisface las siguientes condiciones:

- $(L, \preceq)$ está acotado, lo que quiere decir que tiene elemento mínimo ($\bot$) y elemento máximo ($\top$).
- $(\&_i, \leftarrow_i)$ es un par adjunto en $(L, \preceq)$ para todo $i \in \{1, ..., n\}$.
- $\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$ para todo $\vartheta \in L$ y para cada $i \in \{1, ..., n\}$.

La estructura algebraica que se muestra en la siguiente definición aumenta el poder expresivo del retículo multiadjunto mediante el uso de operadores adicionales.

Definición 4.22. Sea Ω un conjunto graduado conteniendo los operadores $\leftarrow_i$ y $\&_i$, formando pares adjuntos, para $i \in \{1, ..., n\}$ y posiblemente algunos operadores extra. Sea $\mathcal{L} = \langle L, \leq \rangle$ una Ω-álgebra cuyo soporte L, es un retículo completo bajo la relación de orden $\leq$.
Diremos que $\mathcal{L}$ es una *Ω-álgebra multiadjunta* con respecto a los pares $(\&_i, \leftarrow_i)$, con $i \in \{1, ..., n\}$, si $\mathcal{L} = (L, \preceq, I(\&_1), I(\leftarrow_1), ..., I(\&_n), I(\leftarrow_n))$ es un retículo multiadjunto.

En la práctica normalmente tendremos que asumir algunas propiedades en los operadores extra que consideremos. Dichos operadores extra se asumirán como conjuntores, disyuntores o agregadores.

Ejemplo 4.4. Sea $\Omega = \{\leftarrow_P, \&_P, \leftarrow_G, \&_G, \wedge_L, @\}$ y el intervalo unidad, $U = [0, 1]$, en $\mathbb{R}$ (es un retículo). Se define la función I como:

$$\begin{aligned}
I(\leftarrow_P)(x, y) &= \text{mín}(1, x/y) \\
I(\&_P)(x, y) &= x \cdot y \\
I(\leftarrow_G)(x, y) &= \begin{cases} 1 & \text{si } x \leq y \\ 0 & \text{en caso contrario} \end{cases} \\
I(\&_G)(x, y) &= \text{mín}(x, y) \\
I(@)(x, y, z) &= \frac{1}{6}(x + 2y + 3z) \\
I(\wedge_L)(x, y) &= \text{máx}(0, x + y - 1)
\end{aligned}$$

Entonces (U, I) es una Ω-álgebra multiadjunta con dos operadores extra que son un agregador y un conjuntor adicional.

4.4.1 Sintaxis y semántica de los programas lógicos multiadjuntos.

La definición de programa lógico multiadjunto es dada usualmente como un conjunto de reglas y hechos de un lenguaje dado $\mathfrak{F}$. El conjunto de símbolos proposicionales de un progrma lógico multiadjunto se denota usualmente por Π. La sintaxis particular de estas reglas y hechos se introduce a continuación.

Definición 4.23. Sea $(L, \preceq)$ un retículo multiadjunto. Un *programa lógico multiadjunto*, denotado como $\mathbb{P}$, es un conjunto de reglas de la forma $\langle (A \leftarrow_i \mathcal{B}), \vartheta \rangle$ tal que:

- $A \leftarrow_i \mathcal{B}$ es una fórmula de $\mathfrak{F}$ que recibe el nombre de *regla*.
- $\vartheta \in L$ recibe el nombre de *grado de verdad* o *factor de confianza de la regla.*

- El consecuente de la regla, A, es un símbolo proposicional de Π al que llamaremos *cabeza de la regla*.
- El antecedente de la regla, $\mathcal{B}$ recibe el nombre de *cuerpo de la regla* y es una fórmula de $\mathfrak{F}$ formada por símbolos proposicionales $B_1, ..., B_n$ con $n > 0$ mediante el uso de conjuntores $\&_1, ..., \&_n$ y $\wedge_1, ..., \wedge_k$, disyuntores $\vee_1, ..., \vee_l$ y agregadores $@_1, ..., @_m$.

En particular, las reglas cuyo cuerpo es el elemento máximo $\top$ del retículo, reciben el nombre de *hechos* del programa.
Nos referiremos al par $\langle (A \leftarrow_i \mathcal{B}), \vartheta \rangle$ como *regla ponderada* o simplemente como regla cuando no haya lugar a equívoco. También se puede usar la notación $A \xleftarrow{\vartheta}_i \mathcal{B}$.

Nos disponemos ahora a presentar la teoría correspondiente a la semántica de los programas lógicos multiadjuntos, en esta parte muchas ideas vienen inspiradas por (8).
Como estamos trabajando con dos Ω-álgebras, para esclarecer la notación y a qué álgebra pertenece cada operador, en vez de seguir usando $\omega_{\mathfrak{L}}$ o $\omega_{\mathfrak{F}}$, si ω es un operador en Ω, su interpretación en $\mathcal{L}$ se denotará en lo que sigue como $\dot{\omega}$, mientras que su interpretación en F se denotará $\omega_{\mathfrak{F}}$, siempre que no haya lugar a dudas.

Definición 4.24. Sea $(L, \preceq)$ un retículo y Π un conjunto infinito numerable. Una *interpretación* es una aplicación $I \colon \Pi \to L$. El conjunto formado por todas las interpretaciones de las fórmulas definidas por la Ω-álgebra $\mathfrak{F}$ en la Ω-álgebra $\mathcal{L}$ se denota como $\mathfrak{I}_{\mathcal{L}}$.

En otras palabras, una interpretación es una asignación de valores de verdad a cada símbolo proposicional de un lenguaje.

Definición 4.25. Sean dos interpretaciones $I_1, I_2 \in \mathfrak{I}_{\mathfrak{L}}$. Entonces $(\mathfrak{I}_{\mathfrak{L}}, \sqsubseteq)$ es un *retículo de interpretaciones* donde la relación de orden $\sqsubseteq$ verifica $I_1 \sqsubseteq I_2$ si y solo si $I_1(p) \leq I_2(p)$ para todo $p \in \Pi$.
Denotaremos además como $\triangle$ a la *interpretación mínima*, la cual envía cualquier fórmula atómica básica al menor elemento de L, es decir a $\bot$.

En un programa lógico multiadjunto diremos que una regla se satisface siempre y cuando el valor de verdad de la regla sea mayor que el factor de confianza asociado a dicha regla. Formalmente esto se expresa como:

Definición 4.26. Decimos que una interpretación $I \in I_{\mathcal{L}}$ satisface una regla $\langle (A \leftarrow_i \mathcal{B}), \vartheta \rangle$ si y solo si $\vartheta \leq \hat{I}(A \leftarrow_i \mathcal{B})$, donde $\hat{I}$ denota la evaluación de la regla, de la cabeza o del cuerpo de una regla, bajo la interpretación I. Se define un *modelo* de un programa lógico multiadjunto $\mathbb{P}$ como una interpretación I que satisface todas las reglas ponderadas de dicho programa.

Notemos que la siguiente cadena de igualdades

$$\hat{I}(A \leftarrow_i \mathcal{B}) = \hat{I}(A) \dot{\leftarrow}_i \hat{I}(\mathcal{B}) = I(A) \dot{\leftarrow}_i \hat{I}(\mathcal{B})$$

y la evaluación de $\hat{I}(\mathcal{B})$ continúan inductivamente de manera usual, hasta que todos los símbolos proposicionales en $\mathcal{B}$ se alcancen y se evalúen bajo I. En el caso particular de un hecho $(\langle A \leftarrow_i \top), \vartheta\rangle$ que se satisfaga, significa

$$\vartheta \leq \hat{I}(A \leftarrow_i \top) = I(A) \dot{\leftarrow}_i \top$$

por la tercera propiedad de los pares adjuntos esto es equivalente a $\vartheta \dot{\&}_i \top \leq I(A)$ y a su vez por la tercera propiedad de los retículos multiadjuntos se tiene que $\vartheta \leq I(A)$.

Definición 4.27. Un elemento $\lambda \in L$ es una *respuesta correcta* al programa $\mathbb{P}$ y la pregunta $?A$ si para una interpretación arbitraria $I : \Pi \to L$ la cual es un modelo de $\mathbb{P}$ se verifica $\lambda \leq I(A)$.

El operador de consecuencias fue introducido por Emden y Kowalski en (9). Es posible hacer una generalización de dicho operador al contexto de la programación lógica multiadjunta de la siguiente manera:

Definición 4.28. Sea $\mathbb{P}$ un programa lógico multiadjunto, I una interpretación y A un símbolo proposicional, entonces podemos definir el siguiente elemento de L:

$$T^{\mathfrak{L}}_{\mathbb{P}}(I)(A) = \sup\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\}$$

El operador $T^{\mathfrak{L}}_{\mathbb{P}} : \mathcal{I}_{\mathcal{L}} \to \mathcal{I}_{\mathcal{L}}$ recibe el nombre de *operador de consecuencias inmediata*.

De aquí en adelante, si no existe duda con el álgebra $\mathfrak{L}$ usada, lo denotaremos simplemente como $T_{\mathbb{P}}$.

Nuestro próximo objetivo será el de estudiar la monotonía y continuidad de dicho operador para así poder aplicar los resultados relacionados con el menor punto fijo. Para ello, estudiaremos los resultados presentados en (12).

Teorema 4.2. *Una interpretación $I \in \mathcal{I}_{\mathcal{L}}$ es un modelo de un programa lógico multiadjunto $\mathbb{P}$ si y solo si $T_{\mathbb{P}}(I) \sqsubseteq I$.*

Demostración. Supongamos que tenemos un modelo I para el programa $\mathbb{P}$, entonces para toda regla $A \overset{\vartheta}{\leftarrow}_i \mathcal{B}$ en $\mathbb{P}$ se verifica la siguiente cadena de igualdades

$$\begin{aligned} \vartheta &\leq \hat{I}(A \leftarrow_i B) \\ \vartheta &\leq \hat{I}(A) \dot{\leftarrow}_i \hat{I}(B) \\ \vartheta \dot{\&}_i \hat{I}(\mathcal{B}) &\leq \hat{I}(A) = I(A) \\ \sup\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\} &\leq I(A) \\ T_{\mathbb{P}}(I)(A) &\leq I(A) \end{aligned}$$

Si A no es la cabeza de ninguna regla, entonces tenemos $T_{\mathbb{P}}(I)(A) = \sup \varnothing = \bot \leq I(A)$. La otra implicación es inmediata, ya que la cadena de desigualdades también se verifica en el orden inverso aplicando las propiedades del supremo. Luego también se verifica el recíproco, esto es, si $T_{\mathbb{P}}(I)(A) \leq I(A)$ para cualquier símbolo proposicional A entonces I satisface cualquier regla $A \overset{\vartheta}{\leftarrow}_i \mathcal{B}$. □

La monotonía del operador $T_{\mathbb{P}}$ fue demostrada para el caso de un par adjunto en (6). La prueba para el caso general es similar.

Teorema 4.3. *El operador $T_{\mathbb{P}}$ es monótono.*

Demostración. En primer lugar tomamos dos interpretaciones I, J tales que $I \sqsubseteq J$ con el objetivo de probar que $T_{\mathbb{P}}(I) \sqsubseteq T_{\mathbb{P}}(J)$. Sea A un símbolo proposicional en Π, entonces

$$T_{\mathbb{P}}(I)(A) = \sup\{\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\}$$

Si fuera $\hat{I}(\mathcal{B}) \leq \hat{J}(\mathcal{B})$ para todo $\mathcal{B}$ entonces también tendríamos $\vartheta \dot{\&}_i \hat{I}(\mathcal{B}) \leq \vartheta \dot{\&}_i \hat{J}(\mathcal{B})$, para cada $i \in \{1, \ldots, n\}$, debido a que $\dot{\&}_i$ es creciente. Ahora, aplicando el supremo tendríamos la siguiente desigualdad:

$$T_{\mathbb{P}}(I)(A) \leq T_{\mathbb{P}}(J)(A)$$

para todo $A \in \Pi$. Por tanto, es suficiente probar que $\hat{I}(\mathcal{B}) \leq \hat{J}(\mathcal{B})$ para todo $\mathcal{B}$ para lo que usaremos inducción estructurada:
Si B es una formula atómica, esto es, una fórmula compuesta de un único símbolo proposicional entonces

$$\hat{I}(\mathcal{B}) = I(\mathcal{B}) \leq J(\mathcal{B}) = \hat{J}(\mathcal{B})$$

Para el caso inductivo, consideremos $\mathcal{B} = @\,[\mathcal{B}_1, ..., \mathcal{B}_n]$ y asumimos que $\hat{I}(\mathcal{B}_i) \leq \hat{J}(\mathcal{B}_i)$ para todo $i \in \{1, ..., n\}$. Por la definición de las reglas, sabemos $@$ actúa como un operador de agregación y por lo tanto usando la hipótesis de inducción tenemos:

$$\begin{aligned} \hat{I}(\mathcal{B}) &= \dot{@}[\hat{I}(\mathcal{B}_1), ..., \hat{I}(\mathcal{B}_n)] \\ &\leq \dot{@}[\hat{J}(\mathcal{B}_1), ..., \hat{J}(\mathcal{B}_n)] \\ &= \hat{J}(\mathcal{B}) \end{aligned}$$

□

Una vez probado que el operador de consecuencias inmediata es monótono, podemos aplicar la Proposición 4.1, el Teorema 4.1 y el Teorema 4.3 para asegurar que el modelo mínimo de $\mathbb{P}$ coincide con el punto fijo mínimo de $T_{\mathbb{P}}$. Este hecho nos permite calcular el modelo mínimo de Π mediante iteración del operador de consecuencias inmediata partir de la interpretación mínima $\triangle$, como veremos en un ejemplo práctico más adelante.

Nuestros esfuerzos se centrarán ahora en comprobar que $T_{\mathbb{P}}$ es continuo, lo que nos permitirá aplicar la Proposición 4.3. con el objetivo de asegurarnos que la iteración mencionada anteriormente tendrá un número finito, o al menos numerable, de pasos.

Empezaremos exponiendo la definición de continuidad que será usada.

Definición 4.29. Sea L un retículo completo y sea $f\colon L \to L$ una aplicación. Diremos que f es continua si conserva el supremo de los conjuntos dirigidos, esto es, dado un conjunto dirigido X se tiene

$$f(\sup X) = \sup\{f(x) \mid x \in X\}$$

Una aplicación $g\colon L^n \to L$ será continua siempre que sea continua en cada argumento por separado.

Definición 4.30. Sea $\mathfrak{F}$ un lenguaje interpretado en una Ω-álgebra multiadjunta $\mathfrak{L}$, y sea ω cualquier operador en el lenguaje. Diremos que ω es continuo si su interpretación sobre $\mathfrak{L}$, $\dot{\omega}$, es continua en L.

Enunciaremos ahora un lema que nos permitirá probar la continuidad del operador de consecuencias.

Lema 4.1. *Sea $\mathbb{P}$ un programa lógico multiadjunto y sea $\mathcal{B}$ el cuerpo de una fórmula de $\mathbb{P}$. Supongamos que todos los operadores $@$ en $\mathcal{B}$ son continuos. Sea X un conjunto dirigido de interpretaciones y denotamos $S = \sup X$. Entonces,*

$$S(\mathcal{B}) = \sup\{\hat{J}(\mathcal{B}) \mid J \in X\}.$$

A continuación expondremos dos teoremas sobre la continuidad.

Teorema 4.4. *Si todos los operadores que aparecen en los cuerpos de las reglas de un programa lógico multiadjunto $\mathbb{P}$ son continuos y las conjunciones adjuntas a las implicaciones que aparecen en las reglas son continuas en el segundo argumento, entonces $T_{\mathbb{P}}$ es continuo.*

Demostración. Tenemos que comprobar que para cada subconjunto dirigido de interpretaciones X y para cada fórmula atómica se cumple la siguiente igualdad:

$$T_{\mathbb{P}}(\sup X)(A) = \sup\{T_{\mathbb{P}}(J)(A) \mid J \in X\}$$

Siguiendo la notación del lema anterior escribimos $S = \sup X$, y consideramos la siguiente cadena de igualdades:

$$\begin{array}{rcl} T_{\mathbb{P}}(\sup X)(A) & = & \sup\{\vartheta \dot{\&}_i \hat{S}(\mathcal{B}) \mid A \overset{\vartheta}{\rightarrow}_i \mathcal{B} \in \mathbb{P}\} \\ & \overset{(1)}{=} & \sup\{\vartheta \dot{\&}_i \sup\{\hat{J}(\mathcal{B}) \mid J \in X\} \mid A \overset{\vartheta}{\rightarrow}_i \mathcal{B} \in \mathbb{P}\} \\ & \overset{(2)}{=} & \sup\{\vartheta \dot{\&}_i \hat{J}(\mathcal{B}) \mid J \in X, \text{ y } A \overset{\vartheta}{\rightarrow}_i \mathcal{B} \in \mathbb{P}\} \\ & = & \sup\{\sup\{\vartheta \dot{\&}_i \hat{J}(\mathcal{B}) \mid A \overset{\vartheta}{\rightarrow}_i \mathcal{B} \in \mathbb{P}\} \mid J \in X\} \\ & = & \sup\{T_{\mathbb{P}}(J)(A) \mid J \in X\} \end{array}$$

La igualdad (1) se deduce del Lema 4.1 y la igualdad (2) se obtiene como consecuencia de la continuidad en el segundo argumento de los operadores $\dot{\&}_i$. □

Trataremos de dar ahora un resultado que haga las veces del recíproco del teorema anterior.

Teorema 4.5. *Si $T_{\mathbb{P}}$ es continuo para todo programa lógico multiadjunto $\mathbb{P}$, entonces cada operador que aparezca en el cuerpo de alguna regla es continuo.*

Demostración. Sea $@$ un conectivo n-ario. Supongamos un orden definido por componentes en L^n. Denotaremos la tupla $(y_1, ..., y_n) \in L^n$ como $\bar{y}$, el orden en L^n es: $\bar{y} \leq \bar{z}$ si y solo si $y_i \leq z_i$ para $i = \{1, \ldots, n\}$.
Sea Y un conjunto dirigido en L^n, vamos a demostrar que

$$\dot{@}(\sup Y) = \sup\{\dot{@}(y_1, ..., y_n) \mid (y_1, ..., y_n) \in Y\}$$

La desigualdad

$$\sup\{\dot{@}(y_1, ..., y_n) \mid (y_1, ..., y_n) \in Y\} \leq \dot{@}(\sup Y) \tag{4.1}$$

se tiene directamente gracias a la monotonía del operador $@$ y la definición de supremo.
Para la otra desigualdad, dados n símbolos proposicionales $A_1, ..., A_n \in \Pi$ y una tupla $\bar{y} = (y_1, ..., y_n) \in L^n$, consideremos la interpretación $I_{\bar{y}}$ definida como $I(A_i) = y_i$ para $i = \{1, \ldots, n\}$ y $\bot$ en otro caso. De esta forma tenemos $I_{\bar{y}} \sqsubseteq I_{\bar{z}}$ si y solo si $\bar{y} \leq \bar{z}$.
Consideremos ahora el conjunto X_Y de interpretaciones $I_{\bar{y}}$ para todo $\bar{y} \in Y$, y consideremos también su supremo, $S_Y = \sup X_Y$. Debido al orden de L^n se verifica, para todo $\bar{y} \in Y$

$$(y_1, ..., y_n) = (I_{\bar{y}}(A_1), ..., I_{\bar{y}}(A_n)) \leq (S_Y(A_1), ..., S_Y(A_n))$$

por tanto tenemos

$$\sup Y \leq (S_Y(A_1), ..., S_Y(A_n))$$

y por la monotonía de $\dot{@}$ se cumple entonces

$$\dot{@}(\sup Y) \leq \dot{@}(S_Y(A_1), ..., S_Y(A_n)) = \widehat{S_Y}(@(A_1, ..., A_n)) \tag{4.2}$$

Por otro lado, consideremos el programa $\mathbb{P}$ formado por una sola regla

$$\mathbb{P} = \{A \leftarrow_i @(A_1, ..., A_n)\}$$

asumiendo la monotonía de $T_{\mathbb{P}}$ tenemos la siguiente cadena de igualdades

$$\begin{aligned}
\widehat{S_Y}(@(A_1, ..., A_n)) &= \top \dot{\&}_i \widehat{S_Y}(@(A_1, ..., A_n)) \\
&= \sup\{\vartheta \dot{\&}_i \widehat{S_Y}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P}\} \\
&= T_{\mathbb{P}}(S_Y)(A) \\
&= \sup\{T_{\mathbb{P}}(J_{\bar{y}})(A) \mid J_{\bar{y}} \in X_Y\} \\
&= \sup\{\sup\{\vartheta \dot{\&}_i J_{\bar{y}}(\mathcal{B}) \mid A \overset{\vartheta}{\rightarrow}_i \mathcal{B} \in \mathbb{P}\} \mid J_{\bar{y}} \in X_Y\} \\
&= \sup\{\top \dot{\&}_i \hat{J}_{\bar{y}}(@(A_1, ..., A_n)) \mid J_{\bar{y}} \in X_Y\} \\
&= \sup\{\dot{@}(J_{\bar{y}}(A_1), ..., J_{\bar{y}}(A_n)) \mid J_{\bar{y}} \in X_Y\} \\
&= \sup\{\dot{@}(y_1, ..., y_n) \mid (y_1, ..., y_n) \in Y\}
\end{aligned}$$

Por tanto debido a este resultado y a la Ecuación (4.2) tenemos

$$\dot{@}(\sup Y) \leq \widehat{S_Y}(@(A_1, ..., A_n)) = \sup\{\dot{@}(y_1, ..., y_n) \mid (y_1, \ldots, y_n) \in Y\}$$

□

Con estos resultados hemos visto condiciones necesarias y suficientes para asegurar la continuidad de $T_{\mathbb{P}}$. Por lo tanto con monotonía y continuidad estamos en condiciones de afirmar que el menor punto fijo de $T_{\mathbb{P}}$ se alcanza en un número finito de iteraciones a partir de la interpretación mínima $\triangle$, lo que nos aporta una herramienta muy potente como veremos en el siguiente apartado.

4.4.2 Ejemplo práctico

Veremos por último un ejemplo práctico sobre un programa lógico multiadjunto que podría servir para representar algún conocimiento general sobre el comportamiento de un motor.

Consideremos el retículo multiadjunto

$$([0, 1], \leq, \leftarrow_G, \&_G, \leftarrow_P, \&_P, \wedge_L)$$

donde $\&_G$ y $\&_P$ son los conjuntores de Gödel y producto, respectivamente, y $\leftarrow_G$, $\leftarrow_P$ son sus correspondientes implicaciones adjuntas y, además, usaremos $\wedge_L$ como operador extra, que es el conjuntor de Łukasiewicz.

El conjunto de símbolos proposicionales será:

$$\Pi = \{ \textit{poco aceite, poca agua, mezcla alta en gasolina, sobrecalentamiento, motor ruidoso, alto consumo de carburante} \}$$

Nuestro programa tendrá cinco reglas y tres hechos que presentamos a continuación.

alto consumo de carburante	$\overset{0.8}{\leftarrow}_G$	*mezcla alta en gasolina* $\wedge_L$ *poco aceite*	(4.2)
sobrecalentamiento	$\overset{0.5}{\leftarrow}_P$	*poco aceite*	(4.3)
motor ruidoso	$\overset{0.8}{\leftarrow}_P$	*mezcla alta en gasolina*	(4.4)
sobrecalentamiento	$\overset{0.9}{\leftarrow}_P$	*poca agua*	(4.5)
motor ruidoso	$\overset{1}{\leftarrow}_P$	*poco aceite*	(4.6)
poco aceite	$\overset{0.2}{\leftarrow}_P$	$\top$	(4.7)
poca agua	$\overset{0.2}{\leftarrow}_P$	$\top$	(4.8)
mezcla alta en gasolina	$\overset{0.5}{\leftarrow}_P$	$\top$	(4.9)

Tabla 4.1: Reglas y hechos del programa

Además vemos que el programa será $[0, 1]$-valuado, es decir usará $[0, 1]$ como retículo.

Nuestro primer objetivo será ver qué condiciones debe cumplir una interpretación para ser un modelo del programa lógico multiadjunto.

Los símbolos proposicionales correspondientes a los hechos ((4.7), (4.8) y (4.9)) deben interpretarse de modo que satisfaga su factor de confianza. Si observamos dichos hechos vemos que tendrán que ser mayores o iguales al peso que se indica en sus implicaciones producto, por tanto

$$0.2 \leq I(\textit{poco aceite}) \quad 0.2 \leq I(\textit{poca agua}) \quad 0.5 \leq I(\textit{mezcla alta en gasolina}) \quad (1)$$

Veremos ahora qué podemos deducir de las reglas. De las reglas (4.3) y (4.5), que tienen como cabeza *sobrecalentamiento*, se deduce

$$\begin{aligned} 0.5 \dot{\&}_P I(\textit{poco aceite}) &\leq I(\textit{sobrecalentamiento}), \quad \text{y} \\ 0.9 \dot{\&}_P I(\textit{poca agua}) &\leq I(\textit{sobrecalentamiento}) \end{aligned}$$

por tanto es evidente que I deberá ser mayor que el supremo de ambos:

$$\sup\{0.5\dot{\&}_P I(\textit{poco aceite}), 0.9\dot{\&}_P I(\textit{poca agua})\} \leq I(\textit{sobrecalentamiento})$$

Siguiendo un razonamiento análogo, de las reglas (4.4) y (4.6) deducimos que

$$\sup\{0.8\dot{\&}_P I(\textit{mezcla alta en gasolina}), 1\dot{\&}_P I(\textit{poco aceite})\} \leq I(\textit{motor ruidoso})$$

Por último, de la primera regla se puede deducir la siguiente asignación para *alto consumo de carburante*

$$0.8\dot{\&}_G(I(\textit{mezcla alta en gasolina})\dot{\wedge}_L I(\textit{poco aceite})) \leq I(\textit{alto consumo de carburante})$$

Una vez hemos visto los valores mínimos podemos empezar a pensar en las preguntas del programa lógico multiadjunto.

Planteamos por ejemplo la pregunta ?*sobrecalentamiento* al programa. El menor valor que le puede asignar un modelo I tanto a *poco aceite* como a *poca agua* es 0.2 y conocer que una cota inferior de sobrecalentamiento en función de los valores de *poco aceite* y *poca agua*, se deduce a partir de las desigualdades mostradas en la Ecuación (1), obtenemos que

$$\sup\{0.5\dot{\&}_P 0.2, 0.9\dot{\&}_P 0.2\} = 0.18$$

de donde, podemos afirmar que cualquier respuesta correcta para la pregunta debe ser menor que 0.18.

Análogamente

$$\sup\{0.8\dot{\&}_P 0.5, 1\dot{\&}_P 0.2\} = 0.4$$

por tanto, si hiciéramos la pregunta ?*motor ruidoso*, cualquier respuesta correcta debería ser menor que 0.4.

Observando las condiciones dadas que debe cumplir cualquier interpretación para ser un modelo del programa, podemos afirmar que la siguiente interpretación J es un modelo del programa introducido en el Cuadro 4.1.

$$\begin{aligned}
J(\textit{poco aceite}) &= 0.25 \\
J(\textit{poca agua}) &= 0.35 \\
J(\textit{mezcla alta en gasolina}) &= 0.90 \\
J(\textit{sobrecalentamiento}) &= 0.45 \\
J(\textit{motor ruidoso}) &= 0.75 \\
J(\textit{alto consumo de carburante}) &= 0.55
\end{aligned}$$

A partir de dicho modelo veremos como actúa la interpretación $T_{\mathbb{P}}(J)$.
Para *poco aceite*, *poca agua* y *mezcla alta en gasolina*, que solo aparecen una vez como hechos, hemos de considerar el supremo de un conjunto unitario:

$$\begin{aligned} T_{\mathbb{P}}(J)(\textit{poco aceite}) &= \sup\{0.2\dot{\&}_P J(\top)\} = 0.2 \\ T_{\mathbb{P}}(J)(\textit{poca agua}) &= \sup\{0.2\dot{\&}_P J(\top)\} = 0.2 \\ T_{\mathbb{P}}(J)(\textit{mezcla alta en gasolina}) &= \sup\{0.5\dot{\&}_P J(\top)\} = 0.5 \end{aligned}$$

Para *sobrecalentamiento* existen dos reglas, luego calcularemos el supremo de dos valores

$$\begin{aligned} T_{\mathbb{P}}(J)(\textit{sobrecalentamiento}) &= \sup\{0.5\dot{\&}_P J(\textit{poco aceite}), 0.9\dot{\&}_P J(\textit{poca agua})\} \\ &= \sup\{0.5\dot{\&}_P 0.25, 0.9\dot{\&}_P 0.35\} \\ &= \sup\{0.125, 0.315\} \\ &= 0.315 \end{aligned}$$

de igual manera procedemos para *motor ruidoso*

$$\begin{aligned} T_{\mathbb{P}}(J)(\textit{motor ruidoso}) &= \sup\{0.8\dot{\&}_P J(\textit{mezcla alta en gasolina}), 1\dot{\&}_P J(\textit{poco aceite})\} \\ &= \sup\{0.8\dot{\&}_P 0.9, 1\dot{\&}_P 0.25\} \\ &= \sup\{0.72, 0.315\} \\ &= 0.72 \end{aligned}$$

Para *alto consumo de carburante* solo existe una regla que tiene como cabeza dicho símbolo proposicional, además, esta regla tiene como cuerpo una fórmula no básica ya que aparece el operador extra $\wedge_L$, entonces

$$\begin{aligned} T_{\mathbb{P}}(J)(\textit{alto consumo de carburante }) &= \sup\{0.8\dot{\&}_G \hat{J}(\textit{mezcla alta en gasolina} \wedge_L \textit{poco aceite})\} \\ &= \sup\{0.8\dot{\&}_G J(\textit{mezcla alta en gasolina})\dot{\wedge}_L J(\textit{poco aceite})\} \\ &= \sup\{0.8\dot{\&}_G (0.9\dot{\wedge}_L 0.25)\} \\ &= \sup\{0.8\dot{\&}_G 0.15\} \\ &= 0.15 \end{aligned}$$

Calcularemos ahora un modelo mínimo partiendo del operador $T_{\mathbb{P}}$ e iterando desde la menor interpretación $\triangle$ y de este modo se va obteniendo el punto fijo mínimo que es

el modelo mínimo. Lo representamos en la Cuadro 4.2. Mostraremos a continuación los cálculos para *sobrecalentamiento*, siendo los otros totalmente análogos.

$$\begin{aligned} T_{\mathbb{P}}(\triangle)(\textit{sobrecalentamiento}) &= \sup\{0.5\dot{\&}_P \triangle(\textit{poco aceite}), 0.9\dot{\&}_P \triangle(\textit{poca agua})\} \\ &= \sup\{0.5\dot{\&}_P 0, 0.9\dot{\&}_P 0\} \\ &= \sup\{0, 0\} \\ &= 0. \end{aligned}$$

Recordemos que $T^n_{\mathbb{P}}(\triangle) = T^{n-1}_{\mathbb{P}}(T_{\mathbb{P}}(\triangle))$, luego:

$$\begin{aligned} T^2_{\mathbb{P}}(\triangle)(\textit{sobrecalentamiento}) &= \sup\{0.5\dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{poco aceite}), 0.9\dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{poca agua})\} \\ &= \sup\{0.5\dot{\&}_P 0.2, 0.9\dot{\&}_P 0.2\} \\ &= \sup\{0.1, 0.18\} \\ &= 0.18 \end{aligned}$$

$$\begin{aligned} T^3_{\mathbb{P}}(\triangle)(\textit{sobrecalentamiento}) &= \sup\{0.5\dot{\&}_P T^2_{\mathbb{P}}(\triangle)(\textit{poco aceite}), 0.9\dot{\&}_P T^2_{\mathbb{P}}(\triangle)(\textit{poca agua})\} \\ &= \sup\{0.5\dot{\&}_P 0.2, 0.9\dot{\&}_P 0.2\} \\ &= \sup\{0.1, 0.18\} \\ &= 0.18 \end{aligned}$$

	$\triangle$	$T_{\mathbb{P}}(\triangle)$	$T^2_{\mathbb{P}}(\triangle)$	$T^3_{\mathbb{P}}(\triangle)$
poco aceite	0	0.2	0.2	0.2
poca agua	0	0.2	0.2	0.2
mezcla alta en gasolina	0	0.5	0.5	0.5
sobrecalentamiento	0	0	0.18	0.18
motor ruidoso	0	0	0.4	0.4
alto consumo de carburante	0	0	0	0

Tabla 4.2: Iteraciones sobre el operador de consecuencias

Se termina en la tercera iteración puesto que se repiten los valores de la cuarta y quinta columna, es decir, $T^3_{\mathbb{P}}(\triangle) = T_{\mathbb{P}}(T^2_{\mathbb{P}}(\triangle)) = T^2_{\mathbb{P}}(\triangle)$. Por consiguiente, tenemos que $T^2_{\mathbb{P}}(\triangle)$ es un punto fijo, el punto fijo mínimo.

Tras el estudio del programa podemos sacar varias conclusiones:

1. Lo más importante es la información que nos da el programa.

Condiciones iniciales del motor		Respuesta del motor
0.2 en poca agua		0.18 en sobrecalentamiento
0.2 en poco aceite	$\Rightarrow$	
0.5 en mezcla alta en gasolina		0.4 en motor ruidoso

Tabla 4.3: Conclusiones

Como podemos ver en la Tabla 4.3 bajo las condiciones iniciales dadas, una valor bajo en poca agua y poco aceite (lo que podríamos entender como que el nivel de agua y aceite del motor no es especialmente bajo sino más bien alto) y un valor de 0.5 en el valor de mezcla alta en gasolina (lo que podríamos entender como que la proporción de gasolina en la mezcla es estándar) se obtiene que el motor presenta poco sobrecalentamiento (un valor de 0.18) y un valor de 0.4 en motor ruidoso (se puede entender que el ruido del motor es el normal). En resumen, si al motor no le falta agua ni aceite, y la poporción de gasolina en el combustible es correcta, este no presentará ruido ni se sobrecalentará en exceso, como cabría esperar.

2. En consecuencia, dado el modelo mínimo las respuestas a las preguntas, serían justamente esos valores remarcados.

3. Si cambiamos las condiciones iniciales, el programa dará respuestas diferentes, como hemos visto con la interpretación J, es decir el motor se calentará más o menos y hará más ruido o menos en función de la cantidad de agua, aceite y gasolina en el carburante. Por ejemplo observamos en el cálculo de $T_{\mathbb{P}}$ sobre la interpretación J, que si subimos la proporción de gasolina, el motor será considerablemente más ruidoso.

CAPÍTULO

5

Comparativa entre la programación lógica multiadjunta y el sistema de inferencia de Mamdani

Nuestro objetivo final en este trabajo será hacer una comparativa de las dos teorías estudiadas, en la que trataremos de exponer diferencias y similitudes entre el método de inferencia de Mamdani que vimos en el Capítulo 3 y la programación lógica multiadjunta que desarrollamos en el Capítulo 4.

5.1 Relación entre ambas teorías

La principal similitud entre los sistemas de inferencia difusa de Mamdani y la programación lógica multiadjunta es que ambas teorías siguen la filosofía de las reglas IF-THEN. Teniendo en cuenta este aspecto, un programa lógico multiadjunto puede expresarse como reglas del tipo IF-THEN. A continuación, recurriremos al ejemplo mostrado en la Sección 4.4.2 para ilustrar lo que acabamos de exponer.

Ejemplo 5.1. Recordemos que el conjunto de símbolos proposicionales, considerado en el ejemplo práctico de la Sección 4.4.2, es el siguiente:

$$\Pi = \{ \textit{poco aceite, poca agua, mezcla alta en gasolina, sobrecalentamiento, motor ruidoso, alto consumo de carburante} \}$$

Claramente, estos símbolos proposicionales pueden identificarse como etiquetas lingüísticas. Las variables lingüísticas a las que estas etiquetas estarían asociadas podrían ser los que se muestran a continuación en la Tabla 5.1.

Variables lingüísticas	**Etiquetas lingüísticas**
Variables de entrada	
Nivel de aceite	Bajo
	Medio
	Alto
Nivel de agua	Bajo
	Medio
	Alto
Proporción de gasolina en la mezcla	Baja
	Media
	Alta
Variables de salida	
Temperatura del motor	Normal
	Caliente
	Sobrecalentado
Sonido del motor	Silencioso
	Estándar
	Ruidoso
Consumo de carburante	Bajo
	Estándar
	Alto

Tabla 5.1: Variables y etiquetas lingüísticas

Una vez identificadas las variables y etiquetas lingüísticas, vamos a traducir las reglas mostradas en la Tabla 4.1. Estas reglas equivalen a sus homónimas del tipo IF-THEN en el sistema de inferencia. Aunque la primera diferencia que vemos es que en las reglas IF-THEN no tienen pesos y en cambio estas del programa sí. Una posible equivalencia de estas reglas sería:

Reglas del programa lógico multiadjunto		
alto consumo de carburante	$\overset{0.8}{\leftarrow}_G$	*mezcla alta en gasolina* $\wedge_L$ *poco aceite*
sobrecalentamiento	$\overset{0.5}{\leftarrow}_P$	*poco aceite*
motor ruidoso	$\overset{0.8}{\leftarrow}_P$	*mezcla alta en gasolina*
sobrecalentamiento	$\overset{0.9}{\leftarrow}_P$	*poca agua*
motor ruidoso	$\overset{1}{\leftarrow}_P$	*poco aceite*
poco aceite	$\overset{0.2}{\leftarrow}_P$	$\top$
poca agua	$\overset{0.2}{\leftarrow}_P$	$\top$
mezcla alta en gasolina	$\overset{0.5}{\leftarrow}_P$	$\top$

Reglas IF-THEN			
IF	Proporción en gasolina es Alta AND Nivel de aceite es Bajo	THEN	Consumo de carburante es Alto
IF	Nivel de aceite es Bajo	THEN	Temperatura es Sobrecalentado
IF	Proporción en gasolina es Alta	THEN	Sonido es Ruidoso
IF	Nivel de agua es Bajo	THEN	Temperatura es Sobrecalentado
IF	Nivel de aceite es Bajo	THEN	Sonido es Ruidoso

Tabla 5.2: Equivalencia de reglas

Observemos que en la primera regla, el conjuntor de Łukasiewicz actúa como el operador AND. Además, las implicaciones de Gödel y producto se traducen, en el sistema de inferencia difuso, como THEN. Podemos observar que al traducir las reglas del programa multiadjunto a reglas del tipo IF-THEN, se pierden los pesos de las reglas así como el uso de distintos operadores. En nuestro ejemplo particular, la implicación de Gödel se usa cuando el cuerpo de la regla tiene dos símbolos proposicionales combinados con el conjuntor de Łukasiewicz. Y por otro lado, la implicación producto se usa cuando el cuerpo de la regla está constituido por un solo símbolo proposicional.

Los hechos del programa multiadjunto mostrado en la Tabla 5.2, deben interpretarse como los valores de entrada del sistema de inferencia. Podemos deducir que en un hipotético sistema de inferencia tendríamos que “Nivel de aceite”, “Nivel de agua” y “Proporción de gasolina y aire en la mezcla” serían las variables de entrada, mientras que “Temperatura”, “Sonido del motor” y “Consumo de carburante” actuarían como las variables de salida, tal y como se propone en la Tabla 5.1. Como hemos estudiado anteriormente, el sistema de inferencia de Mamdani solo proporciona una variable de salida, por tanto el programa lógico multiadjunto mostrado en la Tabla 5.2 equivaldría en realidad a tres sistemas de inferencia con las mismas variables de entrada, pero cada uno con una variable de salida diferente.

Observando el programa lógico multiadjunto la siguiente similitud de la que damos cuenta es que, el cálculo deloperador de consecuencias inmediata se puede identificar con la salida del sistema de inferencia difuso. Vamos a ver paso a paso dicha equivalencia en el siguiente ejemplo, mostrando el cálculo de un operador de consecuencia que no se llevó a cabo de manera explícita en la sección anterior.

Ejemplo 5.2. Si nos fijamos en el programa lógico multiadjunto de la Tabla 5.2 y queremos calcular el operador $T_{\mathbb{P}}^2$ para la menor interpretación, $\triangle$:

$$\begin{aligned} T_{\mathbb{P}}^2(\triangle)(\textit{motor ruidoso}) &= \sup\{0.8 \dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{mezcla alta en gasolina}), 1 \dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{poco aceite})\} \\ &= \sup\{0.8 \dot{\&}_P 0.5, 1 \dot{\&}_P 0.2\} \\ &= \sup\{0.4, 0.2\} \\ &= 0.4 \end{aligned}$$

En un hipotético sistema de inferencia equivalente, las variables de entrada serían una proporción de 0.5 de gasolina en la mezcla y un nivel de aceite de 0.2, y la variable de salida que daría el sistema tras los cálculos pertinentes del modelo de Mamdani sería que el ruido del motor es 0.4.

Lo primero que observamos es el uso de la conjunción producto, es en esta conjunción donde toma valor el peso de la regla. Este peso aparece en las reglas del programa lógico multiadjunto, pero no en el sistema de inferencia. La conjunción producto evalúa dicho peso y la interpretación del símbolo proposicional correspondiente. Observamos además, que no hay solo un símbolo proposicional en juego, sino que en el cálculo de $T_{\mathbb{P}}$ influyen todas las reglas del programa que tengan al símbolo proposicional que aparece en la pregunta como cabeza. Es por esto que podemos relacionar el cálculo del supremo que se realiza en el operador de consecuencias inmediata con el operador OR en un sistema de inferencia difuso. Ya que si obviamos el peso, tal y como lo hemos planteado antes, básicamente lo que hacemos es quedarnos con el máximo de los valores de entrada, que es exactamente el papel que juega el operador OR en los sistemas de inferencia difusos.

5.2 Diferencias entre ambas teorías

En esta sección expondremos aquellos procesos, pasos o componentes que diferencia cada método del otro. Trataremos también de dar una explicación a cómo cada método sustituye o trata esas, *a priori*, carencias. De igual manera que si no lo hace veremos qué desventajas o ventajas supone.

Una de las primeras cosas que hemos remarcado durante el capítulo es la existencia de pesos en las reglas de los programas lógicos multiadjuntos. Esto hace que unas reglas

influyan más que otras. Por ejemplo, si queremos inferir sobre el crecimiento de una especie, puede que en ocasiones una epidemia que afecte a dicha especie afecte de manera más directa a su crecimiento que la variable cantidad de alimento.

Ejemplo 5.3. Si observamos el programa lógico que estamos tratando, y queremos preguntar ?*sobrecalentamiento*, tendremos que fijarnos en las siguientes dos reglas de todas las que se dan

sobrecalentamiento	$\overset{0.5}{\leftarrow_P}$	*poco aceite*
sobrecalentamiento	$\overset{0.9}{\leftarrow_P}$	*poca agua*

Por tanto el operador de consecuencias para la interpretación mínima, $\triangle$, será

$$\begin{aligned} T^2_{\mathbb{P}}(\triangle)(\textit{sobrecalentamiento}) &= \sup\{0.5\dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{poco aceite}), 0.9\dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{poca agua})\} \\ &= \sup\{0.5\dot{\&}_P 0.2, 0.9\dot{\&}_P 0.2\} \\ &= \sup\{0.1, 0.18\} \\ &= 0.18 \end{aligned}$$

En conclusión, los pesos contribuyen en este ejemplo a que el nivel de agua influya más en el calentamiento del motor que el nivel de aceite, obsérvese que *poco agua* tiene un peso de 0.9 en la regla frente al 0.5 de *poco aceite*. Esto es así, porque a pesos iguales el resultado de aplicar la implicación producto dependería exclusivamente del valor de $\triangle$(*poco aceite*) y $\triangle$(*poca agua*), pero a mayor peso, mayor será el resultado de la implicación producto.

Ejemplo 5.4. Supongamos que $\triangle$(*poco aceite*)$= 0.3$ y $\triangle$(*poco agua*)$= 0.1$. Si el peso fuera igual, digamos $x \in \mathbb{R}$, tendríamos $x\dot{\&}_P \triangle$(*poco aceite*) $\geq x\dot{\&}_P \triangle$(*poca agua*) simplemente porque $0.3 \geq 0.1$, y consecuentemente sería $T_{\mathbb{P}}(\triangle)(\textit{sobrecalentamiento}) = x\dot{\&}_P \triangle(\textit{poco aceite})$.

Por otro lado, si ponemos pesos de 0.2 y 0.8 respectivamente, resultaría:

$$\begin{aligned} T_{\mathbb{P}}(\triangle)(\textit{sobrecalentamiento}) &= \sup\{0.2\dot{\&}_P \triangle\ (\textit{poco aceite}), 0.8\dot{\&}_P \triangle\ (\textit{poca agua})\} \\ &= \sup\{0.2\dot{\&}_P 0.3, 0.8\dot{\&}_P 0.1\} \\ &= \sup\{0.06, 0.08\} \\ &= 0.08 \end{aligned}$$

De manera que aunque $0.3 \geq 0.1$, la asignación de pesos contribuye a que el nivel de agua sea un factor más determinante.

En los sistemas de inferencia difusa, como sabemos, las reglas no tienen pesos, pero eso no quiere decir que haya ocasiones en los que unos factores no influyan más que otros

en la inferencia. Esto se puede conseguir en la formulación de las reglas. Si observamos el Ejemplo 3.5.5 claramente vemos que el tamaño tumoral es mucho mas influyente que la edad a la hora de valuar el riesgo de cáncer como se comenta al final del problema. Además como vemos en las figuras 3.18 y 3.21 esta proporcionalidad ocurre en ambos métodos, Mamdani y Sugeno. Eso sí, cuanto más variables y etiquetas tenga el problema, dotar de más importancia a unos factores que a otros requerirá un número bastante mayor de reglas, al contrario que en la programación lógica multiadjunta.

Otra de las diferencias que parecen más evidentes al comparar un programa lógico multiadjunto y un sistema de inferencia es que en el primero no aparecen los procesos de fuzzificación ni defuzzificación. En realidad, estos procesos también se llevan a cabo en la programación lógica multiadjunta, lo que ocurre es que no se llevan a cabo de manera explícita. En resumidas cuentas, la asignación de etiquetas lingüísticas puede hacerse de multitud de formas diferentes. Por ejemplo, en un proceso de fuzzificación en un sistema de inferencia de Mamdani, una partición difusa de una variable con cuatro clases puede asociarse a cuatro símbolos proposicionales diferentes (uno por clase) en programación lógica. Nos apoyaremos en el siguiente ejemplo para ilustrar esta comparación.

Ejemplo 5.5. Supongamos que tenemos un controlador difuso que tiene como entrada la variable lingüística *Temperatura* cuyas etiquetas serán "Muy Baja", "Baja", "Alta" y "Muy Alta". Cada vez que demos al controlador un valor de entrada, este deberá pasar un proceso de fuzzificación que nos diga qué grado de pertenencia tiene dicho valor para los conjuntos asociados con las etiquetas mencionadas. En cambio, en un programa lógico multiadjunto dichas etiquetas pueden asociarse directamente a símbolos proposiconales diferentes, no existiendo en dichos programas una variable lingüística per se, sino que se trabaja directamente con las etiquetas:

$$\Pi = \{ \textit{ temperatura muy baja, temperatura baja, temperatura alta, temperatura muy alta } \}$$

Evidentemente, en el programa lógico multiadjunto podrá haber más símbolos proposicionales no relacionados con la temperatura al igual que en el controlador podrá haber más variables de entrada.

Con esta identificación, puede hacerse ahora una comparación en el proceso de inferencia. Con respecto al proceso de inferencia, Mamdani podría considerarse como "un proceso" simplificado de programación lógica en la que los consecuentes solo versan sobre una misma variable. De este modo, programación lógica permite ciclos, mientras que

Mamdani no los permite. Con ciclos nos referimos a que se pueden considerar consecuentes como antecedentes en otras reglas. Esta aparente similitud puede hacer pensar al lector que la inferencia Mamdani es un caso especial de programación lógica, pero esto no es cierto.

Ejemplo 5.6. Supongamos que nuestro objetivo es hacer un programa lógico multiadjunto que podría representar el comportamiento de una casa domótica. El conjunto de símbolos proposicionales será:

$$\Pi = \{\textit{temperatura baja, luminosidad alta, persiana abierta, luz apagada}\}$$

El programa tendrá tres reglas y dos hechos que se presentan en la Tabla 5.3

Luz apagada	$\overset{0.8}{\leftarrow}_G$	*Luminosidad alta* $\wedge_L$ *Persiana abierta*
Persiana abierta	$\overset{0.5}{\leftarrow}_G$	*Luminosidad alta* $\wedge_L$ *Temperatura baja*
Aire ac. apagado	$\overset{0.9}{\leftarrow}_G$	*Temperatura baja*
Temperatura baja	$\overset{0.2}{\leftarrow}_P$	$\top$
Luminosidad alta	$\overset{0.2}{\leftarrow}_P$	$\top$

Tabla 5.3: Reglas y hechos casa domótica

Como se puede observar el ejemplo presenta un ciclo que enlaza con el símbolo *persiana abierta*, ya que actúa como cabeza de en la segunda regla pero además aparece en el cuerpo de la primera. Por tanto, no podríamos hacer un sistema de inferencias de Mamdani equivalente porque los consecuentes no pueden actuar como antecedentes. Deberíamos hacer al menos tres, uno para cada consecuente, para poder interpretar este único programa.

Hablaremos ahora sobre la principal diferencia entre programación lógica y sistemas de inferencia de Mamdani: los grados de verdad tienen interpretaciones completamente diferentes.

- La inferencia de programación lógica determina cotas inferiores de los grados de verdad de los símbolos proposicionales; esto es claramente visible en la definición de modelo mínimo y su cálculo mediante el operador de consecuencias inmediatas.

- Mamdani en cambio, determina cotas superiores a los grados de verdad; esto es visible en el proceso de defuzzificación, en el que se calcula el centro de masa del conjunto difuso construido por la unión de las inferencias. En el método de inferencia de Mamdani, para cada regla IF-THEN, el resultado de la evaluación del antecedente se

relaciona con el consecuente aplicando el procedimiento que se explicó en la Sección 3.5.3, el cual consiste en calcular un alpha-corte para obtener el valor de salida. Como esto se hace para cada regla, entonces se combinan los valores de salida usando el operador máximo. Esto hace que el grado verdad que se obtiene en el consecuente sea una cota superior de la regla IF-THEN.

Esta diferencia que acabamos de remarcar tiene una consecuencia importante: en los sistemas de inferencia de Mamdani no es posible que se de una contradicción en la consecuencia, mientras que por otro lado en programación lógica sí es posible que se de una contradicción. Ilustraremos esta diferencia en el siguiente ejemplo donde vamos a ver un mismo problema resuelto mediante ambas teorías. Nos servirá para ver la diferencia de la contradicción que acabamos de mencionar, pero a su vez como una última comparativa general entre la programación lógica multiadjunta y el sistema de inferencia de Mamdani.

Ejemplo 5.7. El objetivo en este ejemplo será inferir la temperatura de un lugar a partir de la latitud, la altitud, el viento y la humedad del mismo.

Empezaremos tratando de resolver este problema mediante un programa lógico cuyo conjunto de símbolos proposicionales es el siguiente:

$$\Pi = \{ \textit{temperatura alta, temperatura baja, humedad baja, altitud elevada, viento de levante, latitud baja} \}$$

Las reglas del programa son las que damos a continuación en la Tabla 5.4

Reglas del programa lógico multiadjunto		
temperatura alta	$\overset{0.3}{\leftarrow_P}$	*viento de levante*
temperatura alta	$\overset{0.925}{\leftarrow_P}$	*latitud baja*
temperatura alta	$\overset{0.4}{\leftarrow_P}$	*humedad baja*
temperatura alta	$\overset{0.8}{\leftarrow_P}$	*altitud elevada*
viento de levante	$\overset{0.2}{\leftarrow_P}$	$\top$
latitud baja	$\overset{0.8}{\leftarrow_P}$	$\top$
humedad baja	$\overset{0.5}{\leftarrow_P}$	$\top$
altitud elevada	$\overset{0.925}{\leftarrow_P}$	$\top$

Tabla 5.4: Reglas del programa

De acuerdo con la teoría desarrollada en el Capítulo 4, el objetivo es el de calcular el menor punto fijo del operador de consecuencia inmediata partiendo del modelo mínimo. Para ello, iteraremos sobre el operador de consecuencias inmediata. Todos los resultados se

muestran en la Tabla 5.5. A continuación haremos los cálculos para el símbolo *temperatura baja*, siendo los de *temperatura alta* totalmente análogos.

$$\begin{aligned} T_{\mathbb{P}}(\triangle)(\textit{temperatura baja}) &= \sup\{0.4\dot{\&}_P \triangle(\textit{humedad baja}), 0.8\dot{\&}_P \triangle(\textit{altitud elevada})\} \\ &= \sup\{0.4\dot{\&}_P 0, 0.8\dot{\&}_P 0\} \\ &= \sup\{0, 0\} \\ &= 0. \end{aligned}$$

$$\begin{aligned} T^2_{\mathbb{P}}(\triangle)(\textit{temperatura baja}) &= \sup\{0.8\dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{humedad baja}), 1\dot{\&}_P T_{\mathbb{P}}(\triangle)(\textit{altitud elevada})\} \\ &= \sup\{0.4\dot{\&}_P 0.5, 0.8\dot{\&}_P 0.925\} \\ &= \sup\{0.2, 0.74\} \\ &= 0.74. \end{aligned}$$

Y por consiguiente:

$$\begin{aligned} T^3_{\mathbb{P}}(\triangle)(\textit{temperatura baja}) &= \sup\{0.8\dot{\&}_P T^2_{\mathbb{P}}(\triangle)(\textit{humedad baja}), 1\dot{\&}_P T^2_{\mathbb{P}}(\triangle)(\textit{altitud elevada})\} \\ &= \sup\{0.4\dot{\&}_P 0.5, 0.8\dot{\&}_P 0.925\} \\ &= \sup\{0.2, 0.74\} \\ &= 0.74. \end{aligned}$$

	$\triangle$	$T_{\mathbb{P}}(\triangle)$	$T^2_{\mathbb{P}}(\triangle)$	$T^3_{\mathbb{P}}(\triangle)$
viento de levante	0	0.2	0.2	0.2
humedad baja	0	0.8	0.8	0.8
latitud baja	0	0.5	0.5	0.5
altitud elevada	0	0.925	0.925	0.925
temperatura baja	0	0	0.74	0.74
temperatura alta	0	0	0.74	0.74

Tabla 5.5: Iteraciones sobre el operador de consecuencias

Se termina en la tercera iteración puesto que $T^3_{\mathbb{P}}(\triangle) = T^2_{\mathbb{P}}(\triangle)$ y por tanto hemos encontrado el menor punto fijo.

En conclusión, con este resultado interpretamos que el programa lógico multiadjunto devuelve que la temperatura es a la vez alta y baja con un valor alto, 0.74.

Trataremos de abordar ahora el mismo problema mediante el método de inferencia de Mamdani. Para ello en primer lugar definiremos las variables con sus funciones de pertenencia:

1. Viento: Nuestra primera variable de entrada será medida en función a la dirección de la que provenga el viento, considerando levante viento proveniente de noreste o

sureste y poniente viento que viene de noroeste o suroeste. Mediremos el ángulo de incidencia en grados. Esta variable tiene dos etiquetas lingüísticas:

- Levante: función de pertenencia triangular (0, 90, 200).
- Poniente: función de pertenencia triangular (180, 270, 360).

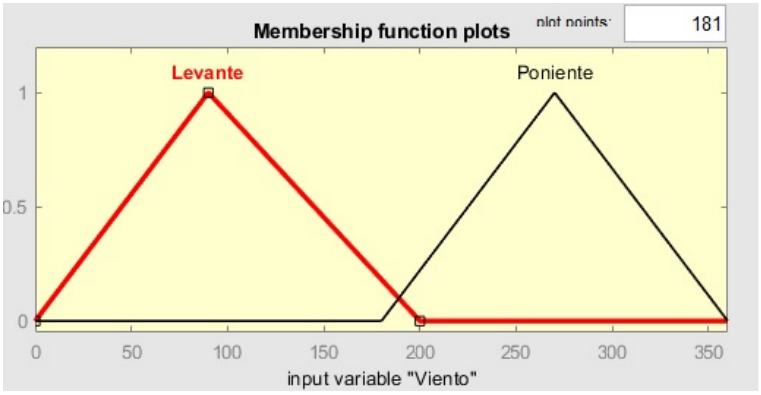


Figura 5.1: Funciones de pertenencia variable Viento

2. Humedad: Medirá el porcentaje de agua en el aire de 0 a 100. Tiene dos etiquetas lingüísticas:

   - Humedad baja: función de pertenencia triangular (0, 10, 40).
   - Humedad alta: función de pertenencia triangular (30, 60, 100).

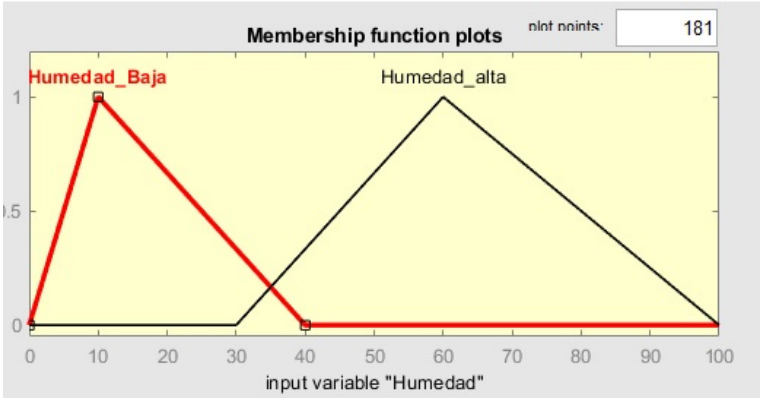


Figura 5.2: Funciones de pertenencia variable Humedad

3. Latitud: Mide la latitud terrestre de 0 a 90, siendo 0 el Ecuador y 90 los polos. De nuevo tenemos dos etiquetas lingüísticas:

   - Latitud baja: función de pertenencia triangular (0, 20, 30).

- Latitud alta: función de pertenencia triangular (25, 40, 90).

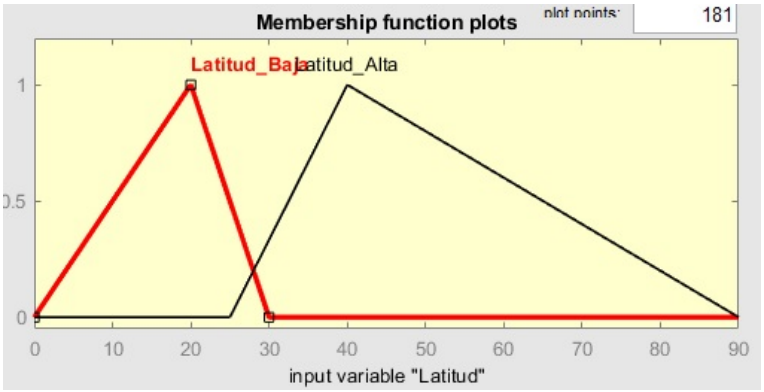


Figura 5.3: Funciones de pertenencia variable Latitud

4. Altitud: Mide la altura con respecto al nivel del mar en metros. Tiene dos etiquetas lingüísticas:

   - Altitud llana: función de pertenencia triangular (0, 100, 300).
   - Altitud elevada: función de pertenencia triangular (200, 500, 1000).

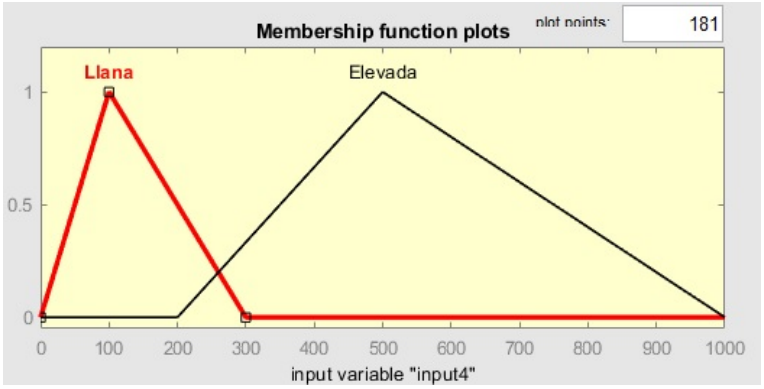


Figura 5.4: Funciones de pertenencia variable Altitud

5. Temperatura: Es nuestra variable de salida, mide los grados centígrados en un rango de -10 a 40, presenta tres etiquetas lingüísticas:

   - Temperatura baja: función de pertenencia triangular (-10, 0, 10).
   - Temperatura media: función de pertenencia triangular (8, 15, 20).
   - Temperatura alta: función de pertenencia triangular (17, 25, 40).

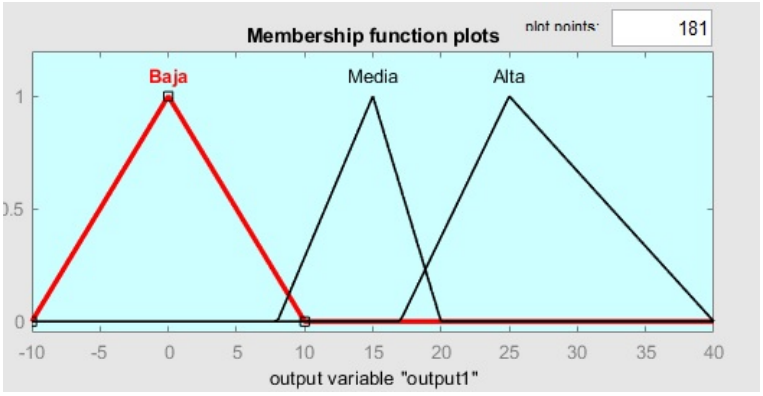


Figura 5.5: Funciones de pertenencia variable Temperatura

Las reglas difusas, en las que se basará el controlador son las equivalentes a las del programa lógico multiadjunto, se muestran a continuación en la Tabla 5.6:

Reglas IF-THEN			
IF	Viento es levante	THEN	Temperatura es alta
IF	Latitud es baja	THEN	Temperatura es alta
IF	Humedad es baja	THEN	Temperatura es baja
IF	Altitud es elevada	THEN	Temperatura es baja

Tabla 5.6: Reglas del controlador difuso

Ahora estamos en disposición de proceder a la inferencia. Daremos al controlador la siguiente entrada:

- Viento: entrando en ángulo de 18^{o}, mirando la primera regla tenemos que su grado de pertenencia a levante es de 0.2.
- Humedad: del 25 %, que pertenece a humedad baja en 0.5.
- Latitud: 16^{o}, que pertenece a latitud baja en un grado de 0.8.
- Altitud: de 447.7 metros, esto tiene un grado de pertenencia a altitud elevada de 0.925.

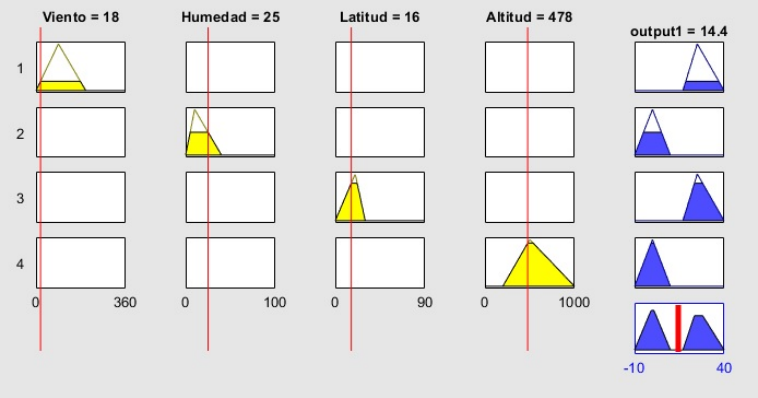


Figura 5.6: Resultados

En este caso, para la inferencia hemos escogido en Matlab, método de recorte para elaborar el conjunto difuso de salida y centroide para defuzzificarlo.

Como podemos ver en la Figura 5.6 con las entradas mencionadas el controlador nos devuelve que la temperatura es de 14.4, que tiene un grado de pertenencia de 0.914 a Temperatura media y con 0 a Temperatura alta y baja.

En conclusión hemos visto en este ejemplo práctico como la programación lógica multiadjunta puede llegar a dar contradicciones en sus salidas mientras que esto no es posible que ocurra en las inferencias de Mamdani.

CAPÍTULO

6

Conclusiones

En el presente trabajo hemos estudiado métodos de inferencia difusa basados en reglas IF-THEN y programación lógica multiadjunta. Hemos presentado los sistemas basados en reglas y más en concreto nos hemos centrado en los sistemas de inferencias de Mamdani y Sugeno. Una vez introducidos de manera teórica, hemos realizado ejemplos prácticos de cada uno de ellos, lo que nos ha permitido establecer una comparativa entre ambos métodos, estudiando así cuales son sus diferencias y similitudes.

Además hemos estudiado la sintaxis y la semántica de la programación lógica multiadjunta. Hemos presentado el operador de consecuencias y estudiado sus principales propiedades. Este marco de trabajo ha sido ilustrado mediante un programa lógico multiadjunto en el que se modeliza como podría comportarse el funcionamiento de un motor.

Para finalizar hemos hecho una comparación entre la programación lógica multiadjunta y el sistema de inferencia de Mamdani de manera práctica. Esta comparación nos ha permitido establecer una similitud entre las reglas de ambas teorías, entre los valores de entrada y los hechos y entre las salidas y el cálculo de $T_{\mathbb{P}}$. Por otro lado, en las diferencias hemos hablado de como se suplen los pesos de las reglas en los sistemas de inferencia de Mamdani, como se realiza en programación lógica multiadjunta el proceso de fuzzificación, la posibilidad de presentar ciclos que tiene la programación lógica y que por el contrario no presenta Mamdani y por último, las salidas contradictorias que pueden darse en un programa lógico pero no en los sistemas de inferencia de Mamdani.

Como trabajo futuro se abren varias vías de investigación. Podría ser interesante también extender la comparativa con otros métodos diferentes de Mamdani, como el de Sugeno, que ya hemos tratado en el Capítulo 3, o el de Tsukamoto. Otra de las ideas que me han surgido finalizando este trabajo es estudiar qué condiciones hay que imponer a un programa lógico multiadjunto para que el menor punto fijo se encuentra en el menor número de iteraciones posibles, ya que en muchas ocasaiones con condiciones específicas he logrado encontrarlo en $T^2_{\mathbb{P}}$. También me gustaría, por el camino que lleva mi futuro, intentar combinar estas teorías con el mundo del *Big Data* y el análisis de datos, ya que pienso que son una gran fuente de información que puede resultar extremadamente útil a la hora de elaborar reglas, puesto que las reglas, en definitiva, vienen basadas en la experiencia.

Bibliografía

[1] Clark, K. Negation as failure. 293–322. 2

[2] Cornejo, M. *Operadores Adjuntos en Entornos Generales y sus Aplicaciones*. PhD thesis, Universidad de Cádiz, 2015. 41

[3] Cornejo, M., Díaz-Moreno, J., y Medina, J. Multi-adjoint relation equations: A decision support system for fuzzy logic. *International Journal of Intelligent Systems 32*, 8 (2017), 778–800. 2

[4] Cornejo, M., Medina, J., y Ramírez-Poussa, E. A comparative study of adjoint triples. *Fuzzy Sets and Systems 211* (2013), 1–14. 2, 48

[5] Cornejo, M., Medina, J., y Ramírez-Poussa, E. Multi-adjoint algebras versus non-commutative residuated structures. *International Journal of Approximate Reasoning 66* (2015), 119–138. 2, 48

[6] Damásio, C., y Pereira, L. M. Monotonic and residuated logic programs. En *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (2001), Springer, pp. 748–759. 2, 49, 53

[7] Deepa, S., Sivanandam, S. N., y Sumathi, S. *Introduction to fuzzy logic using MATLAB*. Springer, 2007. 15

[8] Dubois, D., Prade, H., Bloch, I., Petrosino, A., y Tettamanzi, A. *Fuzzy Logic and Applications: 6th International Workshop, WILF 2005, Crema, Italy, September 15-17, 2005, Revised Selected Papers*. 51

[9] Emden, M. V., y Kowalski, R. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM) 23*, 4 (1976), 733–742. 2, 52

[10] Medina, J. *Retículos Multi-Adjuntos y Teoremas de Continuidad para el Operador de Consecuencias*. PhD thesis, Universidad de Málaga, 2001.

[11] Medina, J., y Medina, E. *Apuntes modelización matemática, Universidad de Cádiz*. 5

[12] Medina, J., M.Ojeda-Aciego, y P.Vojtás. Multi-adjoint logic programming with continous semantics. En *International Conference on Logic Programming and Nonmonotonic Reasoning* (2001), Springer, pp. 351–364. 2, 41, 48, 52

[13] Pavelka, J. On fuzzy logic i many-valued rules of inference. *Mathematical Logic Quarterly 25*, 3-6 (1979), 45–52. 48

[14] Pérez, L. *Fundamentos matemáticos en lógica difusa. TFG en Universidad de Cádiz*. 2013. 41

[15] Ramírez, E. *Multirretículos y reducción de atributos en el análisis de conceptos formales multiadjunto*. PhD thesis, Universidad de Cádiz, 2015. 41

[16] Robinson, J. A machine-oriented logic based on the resolution principle. *Journal of the ACM 12*, 1 (1965), 23–41. 2

[17] Ross, T. *Fuzzy Logic with engineering applications*. Wiley, 2010. 15

[18] Shleeg, A. A., y I.M, E. Comparison of mamdani and sugeno fuzzy interference systems for the breast cancer risk. *International Journal of Computer, Information Science and Engineering 7*, 10 (2013), 387–391. 35

[19] Takagi, T., y Sugeno, M. Fuzzy identification of systems and its applications to modeling and control. *IEEETransactionson System, Man, Cybernetics 15*, 1 (1985), 116r132. 31

[20] Tarski, A., et al. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics 5*, 2 (1955), 285–309. 43

[21] Vojtáš, P. Fuzzy logic programming. *Fuzzy sets and systems 124*, 3 (2001), 361–370. 49

[22] Zadeh, L. Fuzzy sets. *Information and control 8*, 3 (1965), 338–353. 1, 5