LA CONSTRUCCIÓN DE UN SISTEMA DE DETECCIÓN DE INTRUSOS CON XFUZZY[1]

Enrique López González y Cristina Mendaña Cuervo

Departamento de Dirección y Economía de la Empresa, Universidad de León
Facultad de CC. EE. y Empresariales, Campus Vegazana E–24071 León (España)
Phone: +34 987 291742; E–mail: dde{elg,cmc}@unileon.es
URL: http://sicodinet.unileon.es

Contenido

1. INTRODUCCIÓN

En los últimos años, debido al incremento en el uso de los ordenadores y la emergencia del comercio electrónico, la detección de intrusos se ha convertido en una prioridad importante, pues no resulta técnicamente factible construir un sistema invulnerable, pues a pesar de la existencia de numerosas medidas de seguridad para proteger los recursos informáticos de cualquier organización económica y aún cuando se respeten escrupulosamente todas las políticas de seguridad y las recomendaciones de los expertos, no se puede suponer la ausencia de posibles ataques con éxito.

[1] Este trabajo está soportado por el proyecto de investigación DPI 2001–0105 del MCT.

En efecto, nuestra sociedad depende cada vez más del acceso y procesamiento rápido de información, lo que ha supuesto la proliferación del empleo de los ordenadores y de las redes de comunicaciones y con ello la existencia de problemas de acceso desautorizado y manipulación de datos: la cantidad de intentos de accesos no autorizados a la información que existe en Internet ha crecido durante estos últimos años.

Según el Computer Security Institute, más de un 70% de las organizaciones anunciaron al menos un incidente de seguridad durante el último año, frente a un 42% anunciado en 1996. La mayoría de los expertos considera que estos números están muy por debajo de la tasa real, ya que muchas organizaciones evitan dar a conocer sus incidentes y muchas otras ni siquiera los detectan. Además, los intrusos se han vuelto expertos en determinar las debilidades, empleando diversos niveles de engaño antes de irrumpir en un sistema determinado, intentando cubrir sus huellas para que su actividad en el sistema no se descubra fácilmente. De ahí que, la detección de intrusos se haya convertido en una prioridad importante, pues no resulta técnicamente factible construir un sistema totalmente invulnerable, ya que el propio concepto de seguridad es en si mismo "borroso".

La mayoría de sistemas de computación proporcionan un mecanismo de control de acceso como su primera línea de defensa. Sin embargo, esto sólo limita si el acceso a un objeto en el sistema se permite, pero no restringe lo que un sujeto puede hacer con el propio objeto si tiene el acceso para manipularlo. A mayor abundamiento, en sistemas dónde el control de acceso es discrecional, la responsabilidad de la protección de los datos recae sobre el usuario final. Esto requiere a menudo que los usuarios entiendan el mecanismo de protección ofrecido por el sistema y cómo lograr la seguridad deseada usando estos mecanismos.

La cantidad de mensajes publicados en listas de vulnerabilidades como BUGTRAQ ha aumentado de forma exagerada durante los últimos años. Las vulnerabilidades no solo afectan a sistemas tradicionalmente seguros, sino que afectan incluso a sistemas de seguridad: cortafuegos. Por otra parte, aunque muchos escaneos de red y técnicas de ataques son conocidos desde hace varias décadas, no ha sido hasta hace poco tiempo que las herramientas para producir análisis sofisticados a redes han llegado a estar disponibles en el ámbito comercial, estando la mayoría de estas

basadas en algún tipo de Sistema de Detección de Intrusos (SDI), entendido como una herramienta de seguridad encargada de monitorizar los eventos que ocurren en un sistema informático en busca de intentos de intrusión, esto es, cualquier intento de comprometer la confidencialidad, integridad, disponibilidad o evitar los mecanismos de seguridad de una computadora o red.

Las intrusiones se pueden producir de varias formas: atacantes que acceden a los sistemas desde Internet, usuarios autorizados del sistema que intentan ganar privilegios adicionales para los cuales no están autorizados y usuarios autorizados que hacen un mal uso de los privilegios que se les han asignado. También, se puede entender por intrusión a una violación de la política de seguridad del sistema. Pero, en todo caso, conviene poner de manifiesto que cualquier definición de intrusión es necesariamente imprecisa, al igual que los requisitos de política de seguridad no siempre se traducen en un conjunto totalmente definido de acciones. De esta forma, y aún cuando la política define las metas que deben satisfacerse en un sistema, los detectores de brechas de esta política enfocan toda su atención en el conocimiento de pasos o acciones que pueden producir su violación.

La detección de intrusos puede ser dividida en dos categorías principales: la detección de intrusión de anomalías y la detección de intrusión de mal uso (abuso). La primera se refiere a intrusiones que pueden descubrirse basadas en la conducta anómala y uso de recursos de computación. Por ejemplo, si el X usuario sólo usa la computadora de su oficina entre las 9:00 a.m. y las 5 p.m., una actividad en su cuenta fuera de ese horario es anómalo y, por lo tanto, puede ser una intrusión. Posteriormente, considérese a otro usuario Y, que siempre pueda conectarse fuera de las horas de trabajo a través del servidor de la compañía. Una sesión del "login" remota nocturna a su cuenta podría ser considerada extraña, anómala o simplemente "rara". La detección de la anomalía intenta cuantificar la conducta usual o aceptable y señala cualquier conducta irregular como un intruso potencial.

En el contraste, la detección de intrusión de mal uso se refiere a intrusiones que siguen modelos bien definidos de ataque que explotan las debilidades en el sistema y en el software de aplicación. Precisamente, ta-

les modelos pueden escribirse por adelantado, como por ejemplo la explotación de los virus del envío de correo electrónico utilizados en ataques por Internet. Esta técnica representa el conocimiento sobre la conducta mala o inaceptable y busca descubrirlo directamente, en contraposición a la detección de intrusión de anomalías que busca descubrir el complemento de la conducta normal.

En un intento de formalizar este fenómeno, parece evidente que la tendencia natural, basada en la representación en términos de certeza de la información disponible, parece alejarse de lo que la propia realidad demanda. Así, se han empleado técnicas estocásticas, más como una solución suplementaria ante la ausencia de otras, que si bien se muestran como herramientas poderosas para determinados problemas pierden capacidad representativa cuando la información disponible posee altos niveles de subjetividad o vaguedad. De ahí que ante la escasa capacidad de estos modelos para ayudar a la toma de decisiones en entornos cambiantes o turbulentos, cabe plantearse la aplicación de la Teoría de los Subconjuntos Borrosos y la Lógica Borrosa, máxime si se tiene en cuenta que aparejados con ellas se han elaborado un conjunto de técnicas operativas que permiten llevar a cabo los procesos de la toma de decisiones en situaciones caracterizadas por la incertidumbre, en las que cada vez más a menudo pueden encontrarse los responsables e la detección de intrusos. En efecto, con la Lógica Borrosa y la Teoría de los Subconjuntos Borrosos se hace posible el tratamiento tanto de lo subjetivo como de lo incierto, en un intento por representar los fenómenos tal como se muestran en la realidad y de llevar a cabo su tratamiento sin intentar deformarlos o hacerlos precisos o ciertos, de ahí que su aplicación permita realizar un claro cambio de mentalidad, pasando de la utilización de las herramientas disponibles a contar con una "nueva caja de instrumentos" que suministre rigor en el razonamiento secuencial (aproximado), y operatividad práctica ante condiciones de incertidumbre.

De acuerdo con lo anterior, la principal hipótesis de este trabajo es que la Lógica Borrosa es capaz de producir "mejores" reglas que incrementen la flexibilidad y robusted de los sistemas de auditoria informática. De hecho, los Sistemas Basados en Reglas Borrosas (SBRBs) han demostrado ser una herramienta eficaz en problemas de control, clasificación, modelado etc., ante contextos donde la información y/o los datos están afectados de imprecisión no probabilística, lo que justifica el interés por elaborar una

propuesta original de Sistema de Detección de Intrusos aplicando el entorno de desarrollo *Xfuzzy* 3.0, para lo cual se ha contado con la colaboración de D. Eduardo Hernández Gil, continuando y ampliando la línea de trabajo de los autores sobre Auditoria Informática[2].

De forma sintética y general, cabe describir la estructura de este trabajo resumiendo brevemente los contenidos de los diferentes apartados:

En el apartado 2 se introduce la motivación y objetivos principales del esfuerzo investigador.

En el apartado 3 se presenta una visión global de los aspectos teóricos y una metodología general de construcción de los SBRBs.

En el apartado 4 se desarrollan las consideraciones anteriores en su aplicación al problema de la detección de intrusos de sistemas de información, detallando las peculiaridades propias que conlleva la construcción de un SBRB para la auditoria y seguridad informática, para lo cual su implementación práctica será llevada a cabo mediante la aplicación del entorno Xfuzzy3.0.

Finalmente, en el apartado 5 de presenta las principales conclusiones del trabajo realizado, así como se indican alguna línea de investigación y desarrollo futura.

Asimismo, se incluyen tres anexos dedicados el primero a introducir, de una forma necesariamente somera, la Teoría de los Subconjuntos Borrosos, mostrando su capacidad para manejar la incertidumbre que caracteriza a los hechos económicos .En el segundo anexo se presenta una breve introducción al "manual de usuario" del entorno de desarrollo de sistemas borrosos Xfuzzy 3.0. y, por último, en el tercer anexo se incluye el Código en Java del sistema desarrollado.

---

[2] Enrique López González, Angela Díez Díez y Francisco J. Rodríguez Sedano, Cristina Mendaña Cuervo y Jesús Calabozo Morán (2001). "Diseño de un Sistema Borroso para la Detección de Intrusos". Incluido en Actas del VII Congreso del Instituto Internacional de Costos (IIC–ACODI2001).

Enrique López González, Jesús Calabozo Morán, Cristina Mendaña Cuervo, Angela Díez Díez y Francisco J. Rodríguez Sedano (2003). "SIDI v.1. Una propuesta de Sistema inteligente para la Detección de Intrusos". Incluido en Actas del Segundo Congreso Iberoamericano de Seguridad Informática (CIBSI'03).

2. OBJETIVOS DE LA INVESTIGACIÓN

En el presente trabajo se pretende llevar a cabo un enfoque novedoso en la resolución de uno de los principales problemas de seguridad informática que han surgido con el advenimiento de internet y la propia consideración de la información como el principal activo de las organizaciones en la nueva sociedad de la información y el conocimiento: la detección de intrusos.

A este respecto, conviene poner de manifiesto que el entorno actual en el que se desenvuelven la Auditoria y la Seguridad de los Sistemas de Información se caracteriza por una mutabilidad constante, por lo que difícilmente resulta factible considerar que la toma de decisiones basada en el comportamiento pasado pueda llevar a un resultado acertado en el futuro. Además, el sentido de esta tendencia no parece que vaya a disminuir, sino más bien todo lo contrario, se puede aventurar que dicho entorno reflejará situaciones continuamente cambiantes y complejas.

De esta forma, y dado que las posibilidades que los subconjuntos borrosos ofrecen para abordar los problemas de decisión en el campo de la detección de intrusos son amplísimas, con este trabajo, centrado en el desarrollo de una metodología de construcción de un sistema experto borroso de detección de intrusos, se pretende dotar a los informáticos y los economistas de nuevos esquemas que permiten una representación más cercana a la realidad, en un intento de que la toma de decisiones relativa a la detección de intrusos, realizada en ambientes cambiantes y complejos, deje de estar abandonada a su suerte y sea entonces posible seleccionar la solución correcta y acertar en los objetivos esperados de la misma.

En consecuencia, el objetivo principal del presente trabajo va a ser construir un sistema, que esté soportado en un marco teórico (sistemas basados en reglas borrosas)e implementado en un entorno de desarrollo (Xfuzzy 3.0) totalmente operativo, de forma que sea posible la resolución de alguno de los principales problemas de la Detección de Intrusos vinculados al ámbito de la Auditoria y Seguridad de los Sistemas de Información.

3. SISTEMAS BASADOS EN REGLAS BORROSAS

3.1. Introducción

En la década de los años setenta se aceptaba que en los años venideros el hombre llegaría a construir máquinas "pensantes", sin embargo, pasadas tres décadas aún no se ha podido cumplir con esta meta y los ordenadores no logran simular el razonamiento de los humanos, debido principalmente a que están facultados para trabajar con matemáticas precisas, mientras el mundo real está lleno de imprecisión e incertidumbre.

Este dilema puede afrontarse desde dos puntos de vista, a saber: (i) asumir que el problema está en el método de control, y por tanto la solución es aplicar más matemática, y (ii) aceptar que la matemática es el problema, y es aquí donde aparece la lógica borrosa.

En efecto, la lógica borrosa encuentra que en el mundo real son muy escasos los conjuntos no–Fuzzy o convencionales (crisp). Por ejemplo, el conjunto de los mamíferos encuentra un problema al tratar con el ornitorrinco. La lógica borrosa no tiene que tratar con este tipo de excepciones debido a que permite una pertenencia parcial a un conjunto, tal como se muestra en la Figura 1 donde se representa gráficamente un ejemplo de la variable "joven" relativa a la edad de una persona.

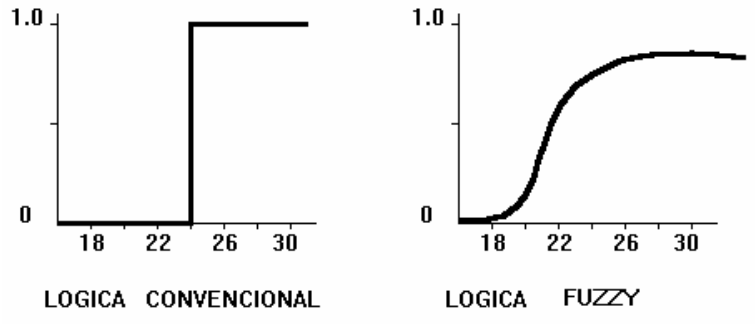


Figura 1

Como es conocido, el propósito de cualquier modelo consiste en capturar el funcionamiento de un sistema cualquiera, pudiendo analizarse su construcción como una colección de objetos denominados variables y parámetros y que están relacionados por medio de otros elementos denominados conectivos u operadores del modelo. Las variables utilizadas en el

modelo corresponden a algunas características del sistema que está siendo modelado. Básicamente se pueden distinguir dos tipos de conectivos que se utilizan en el proceso de modelaje, a saber:

La primera clase de modelos, que utiliza operadores algebraicos, tales como adición, substracción, diferencia, etc., se denominan modelos matemáticos. En estos modelos los parámetros así como los valores de las variables están basados normalmente en valores numéricos.

El segundo tipo de modelos que utilizan conectivos de tipo lógico, tales como "y", "o" y "si–entonces", reciben el nombre de modelos lógicos, incluyendo parámetros de naturaleza lingüística.

Hasta los años 70 la mayoría de los modelos utilizados eran de tipo matemático, pero a partir de 1970 y, especialmente, con la aparición de la inteligencia artificial, se comenzó a considerar estos modelos de tipo lógico como una herramienta para la construcción de sistemas.

Los subconjuntos borrosos, en estas clases de modelizaciones, han sido utilizados primeramente como representación de parámetros, y en algunos casos en los valores de las variables asociadas con el modelo.

De esta forma, los sistemas basados en lógica borrosa pueden controlar más adecuadamente procesos que estén gobernados por reglas intuitivas que difícilmente puede expresarse matemáticamente. Por ejemplo, en el control de un ascensor puede determinarse una desaceleración gradual cuando el ascensor está próximo a su destino.

La gran potencia de esta metodología se debe a la posibilidad de expresar operaciones y controlar las reglas del sistema mediante palabras de uso cotidiano, por ejemplo, volviendo al ejemplo del ascensor, podría programarse como sigue: *"SI está cerca a un piso Y hay orden de parar ENTONCES disminuir la velocidad"*. En este caso, una entrada al sistema de control sería la posición del ascensor, y como "cerca" es un conjunto borroso, el valor de verdad de la premisa y, por tanto, el de la velocidad varía de acuerdo a dicha posición.

La lógica borrosa elimina los altos contenidos de matemática de un proceso y va directo al nivel en que el sistema trabaja, lo que permite aproximarse intuitivamente a la solución de un problema mediante la formulación de reglas. La forma de expresar las reglas de operación mediante palabras permite controlar procesos sencillos con pocas reglas, reduciendo considerablemente la cantidad de código de programación y, por tanto, el

tiempo de diseño, el tiempo de desarrollo de un prototipo, la cantidad de memoria para almacenarlo, etc.

Otra ventaja del control borroso es la fácil modificación que puede llevarse a cabo cambiando algunas premisas y operaciones, o adicionando reglas (el criterio de comportamiento del sistema se encuentra implícito en las reglas), mientras en un sistema convencional, un pequeño cambio requiere de la derivación completa de nuevas ecuaciones, esto es, el control borroso no necesita de la etapa de obtención del modelo matemático del proceso.

La lógica borrosa se trata básicamente de una lógica multievaluada que permite valores intermedios para poder definir evaluaciones convencionales como sí/no, verdadero/falso, negro/blanco, etc. Las nociones como "más bien caliente" o "fresco" pueden formularse matemáticamente y ser procesados por los ordenadores. De esta forma, se ha realizado un intento de aplicar una forma “más humana de pensar” en la programación de los ordenadores.

De acuerdo con lo anterior, cabe plantearse que el empleo del control borroso es recomendable: (i) para procesos muy complejos, cuando no hay un modelo matemático simple.; (ii) en el caso de procesos altamente no lineales; o (iii) si el procesamiento del conocimiento experto (lingüísticamente formulado) puede ser desempeñado. Mientras que, por otro lado, la utilización del control borroso no es una buena idea si: (i) el control convencional teóricamente rinde un resultado satisfactorio; (ii) existe un modelo matemático fácilmente soluble y adecuado; o (iii) el problema no es soluble.

3.2. Estructura y funcionamiento de los Sistemas Basados en Reglas Borrosas: Aspectos teóricos

Una de las áreas de aplicación más importantes de la Teoría de Conjuntos Borrosos y de la Lógica Borrosa la componen los Sistemas Basados en Reglas Borrosas (SBRBs). Este tipo de sistemas constituyen una extensión de los sistemas de reglas basados en Lógica Clásica, pues emplean reglas de tipo “Si–entonces” en las que los antecedentes y consecuentes están compuestos por proposiciones borrosas en lugar de proposiciones de la Lógica Clásica.

En un sentido muy general, un SBRB es un sistema basado en reglas en el que la Lógica Borrosa puede ser empleada tanto como herramienta para representar distintas formas de conocimiento sobre el problema a resolver, como para modelar las interacciones y relaciones existentes entre las variables del mismo.

Las principales aplicaciones de estos sistemas inteligentes son el modelado borroso de sistemas (BARDOSSY y DUCKSTEIN, 1995; PEDRYCZ, 1996) y el control borroso (DRIANKOV, Hellendoorn y REINFRANK, 1993; HIROTA, 1993 y WANG, 1992).

El modelado borroso de sistemas puede ser considerado como una aproximación para modelar un sistema haciendo uso de un lenguaje de descripción basado en Lógica Borrosa con predicados borrosos (Sugeno y YASUKAWA, 1993).

En cambio, los controladores borrosos consisten en un enfoque para la monotorización de procesos donde la estrategia de control aplicada está basada en la experiencia del operador humano representada en forma de reglas lingüísticas de control.

En este apartado se tratará de introducir las nociones básicas de los SBRBs para control borroso, su composición y funcionamiento, para facilitar el posterior estudio de las tareas de construcción que es necesario llevar a cabo para obtenerlos, sin detenerse en la consideración de los principios básicos de la Lógica Borrosa que pueden ser consultados, entre otros, en los trabajos de KLIR y YUAN (1995) o ZIMMERMAN (1996).

El primer modelo de SBRB que trabajó con entradas y salidas reales fue propuesto por MAMDANI (1974), siendo capaz de plasmar las ideas preliminares de ZADEH (1973) en una aplicación de control de una planta de elaboración de cemento. Este tipo de Sistemas Borrosos, los más utilizados desde aquella fecha, se conocen también por el nombre de SBRBs con Fuzzificador y Defuzzificador o Controladores Borrosos (CBs), denominación acuñada por MAMDANI y ASSILIAN (1975) y empleada comúnmente en el ámbito de los sistemas de ingeniería.

Las reglas son del tipo *"Si – entonces"* y en el caso de que el SBRB tenga múltiples entradas y una única salida, presentaría la siguiente estructura: "*Si* X_1 *es* A_1 *y ... y* X_n *es* A_n *entonces Y es B"*, donde X_i e Y son variables lingüísticas de entrada y salida, respectivamente, y los A_i y B son etiquetas lingüísticas asociadas a dichas variables.

Los SBRBs emplean un Sistema de Inferencia que efectúa el Razonamiento Borroso teniendo en cuenta la información contenida en una Base de Conocimiento (BC). Los componentes que dotan al sistema de la capacidad de manejar entradas y salidas reales son los Intefaces de Borrosificación y Clarificación. El primero establece una aplicación entre puntos precisos en el dominio U de las entradas del sistema y conjuntos borrosos definidos sobre el mismo universo de discurso, mientras que el segundo realiza la operación inversa estableciendo una aplicación entre conjuntos borrosos definidos en el dominio V de las salidas y puntos precisos definidos en el mismo universo.

Conviene poner de manifiesto que cualquier SBRB presenta una serie de características particulares, a saber: Por un lado, proporciona un marco natural para incluir conocimiento experto en forma de reglas lingüísticas y permite combinar éste con reglas obtenidas a partir de ejemplos del comportamiento del sistema de un modo muy sencillo. Por otro, presenta una gran libertad a la hora de elegir los Interfaces de Borrosificación y Clarificación, así como el Sistema de Inferencia, de tal forma que permite diseñar el SBRB más adecuado para un problema concreto, aspecto que constituye el principal objeto de atención de este apartado. A este respecto, la Figura 2 muestra la estructura general de los SBRBs.

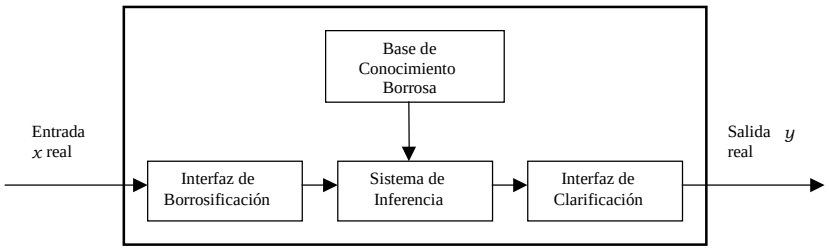


Figura 2

Como se puede observar en la Figura anterior, un SBRB está formado por los siguientes componentes: (i) una Base de Conocimiento, que contiene las reglas lingüísticas que guían el comportamiento del mismo; (ii) un Interfaz de Borrosificación, que se encarga de transformar los datos precisos de entrada en valores manejables en el proceso de razonamiento borroso, es decir, en algún tipo de conjunto borroso; (iii) un Sistema de Inferencia, que emplea estos valores y la información contenida en la Base de Conocimiento para llevar a cabo dicho proceso; y (iv) un Interfaz de Clarificación, que transforma la acción borrosa resultante del proceso de inferencia en una acción precisa que constituye la salida global del SBRB.

3.2.1. La Base de Conocimiento

La Base de Conocimiento (BC) es la parte esencial del control borroso desde el punto de vista de que los restantes tres componentes del sistema se emplean para interpretar las reglas contenidas en ella y hacerlas manejables en problemas concretos.

La BC está formada a su vez por dos componentes distintos: la Base de Reglas lingüísticas y la Base de Datos:

1. La Base de Reglas lingüísticas (BR) está formada por un conjunto de reglas lingüísticas de tipo *"Si – entonces"* que, en el caso de SBRBs con múltiples entradas y una única salida, presentan la siguiente estructura: *Si X_1 es A_1 y ... y X_n es A_n entonces Y es B.*

   La BR está compuesta por una serie de reglas de este tipo unidas por el operador Y (AND), lo que indica que incluso todas ellas pueden dispararse (en paralelo o simultáneamente) ante una entrada concreta.

   La estructura de una regla lingüística puede ser más general si se emplea otro conectivo u operador de conjunción en lugar del Y (AND) para relacionar las variables de entrada en el antecedente, si bien es lo suficientemente general como para incluir otras. Debido a este hecho, y a su simplicidad, no resulta extraño que este tipo de reglas sean las más utilizadas en la literatura especializada.
2. La Base de Datos (BD) contiene la definición de los conjuntos borrosos asociados a los términos lingüísticos empleados en las reglas de la BR, así como los valores de los factores de escala que transforman el universo de discurso en que están definidas las variables de entrada y salida del sistema.

3.2.2. El Interfaz de Borrosificación

Este componente es uno de los que permite al SBRB manejar entradas y salidas reales. Su tarea es la de establecer una aplicación que haga corresponder un conjunto borroso definido en el universo de discurso de la entrada en cuestión a cada valor preciso del espacio de entrada.

Así, el Interfaz de Borrosificación trabaja de la siguiente forma: $A' = F (x_0)$, donde x_0 es un valor preciso de entrada al BC definido en el universo de discurso U, A' es un conjunto borroso definido sobre el mismo dominio U y F es un operador de Borrosificación.

Principalmente, existen dos posibilidades para la elección de F, a saber:

A. Borrosificación no puntual o aproximada: En este caso, A' = F (x_0) = 1 y el grado de pertenencia de los restantes valores de U va disminuyendo según se alejan de x_0. Este segundo tipo de operador de Borrosificación permite el empleo de distintos tipos de funciones de pertenencia. Por ejemplo, en el caso de una función de pertenencia triangular, se puede emplear el siguiente:

$$\mu_{A'}(x) = \begin{cases} 1 - \dfrac{|x - x_0|}{\sigma} & si \quad |x = x_0| \le \sigma \\ 0, \quad en \quad otro \quad caso \end{cases}$$

B. Borrosificación puntual: En este tipo de borrosificación, el más utilizado en la literatura especializada, A' se construye como un conjunto borroso puntual con soporte x_0, es decir, con la siguiente función de pertenencia:

$$\mu_{A'}(x) = \begin{cases} 1, & si \quad x = x_0 \\ 0, & en \quad otro \quad caso \end{cases}$$

En este caso, los valores lingüísticos obtenidos por la Interfaz de Borrosificación son valores numéricos.

Por ejemplo, supóngase un sistema con dos variables de entrada X_1 y X_2. Si la variable X_1 dispone de una partición en tres etiquetas lingüísticas A, B y C, y la variable X_2 dispone de una partición en cuatro etiquetas D, E, F y G, puesto que los nombres de las etiquetas lingüísticas podrían coincidir, aunque para cada variable se trate de etiquetas con particiones lingüísticas diferentes, la acción del Interfaz de Borrosificación para la regla: *"Si X_1 es B y X_2 es G entonces ..."*, consistiría en obtener el valor lingüístico B(x_1), y el valor lingüístico G(x_2), siendo x_1 y x_2 los valores numéricos para las variables de entrada X_1 y X_2 respectivamente en este instante [x_0=(x_1, x_2)].

3.2.3. El Sistema de Inferencia

El Sistema de Inferencia es el componente encargado de llevar a cabo el proceso de inferencia borrosa, para lo cual hace uso de principios de la Lógica Borrosa para establecer una aplicación entre conjuntos borrosos definidos en $U = U_1 \times U_2 \times \ldots \times U_n$ y conjuntos borrosos definidos en V (donde $U_1, \ldots, U_n$ y V son los dominios en los que están definidas las variables de entrada $X_1, \ldots, X_n$ y la de salida Y, respectivamente).

El proceso de inferencia borroso está basado en la aplicación del Modus Ponens Generalizado, extensión del Modus Ponens de la Lógica Clásica propuesta por Zadeh según la siguiente expresión (Zadeh, 1973):

Si X es A entonces Y es B

X es A'

Y es B'

Para llevar a la práctica esta expresión es necesario primero interpretar el tipo de regla que emplea el SBRB. Una regla con la forma *"Si X es A entonces Y es B"* representa una relación borrosa entre A y B definida en U × V. Dicha relación borrosa se expresa mediante un conjunto borroso R cuya función de pertenencia $\mu_R(x,y)$ presenta la forma:

$$\forall x \in U, \quad y \in V: \qquad \mu_R(x,y) = I(\mu_A(x), \mu_B(y))$$

donde $\mu_A(x)$ y $\mu_B(y)$ son las funciones de pertenencia de los conjuntos borrosos A y B, respectivamente, e I es un operador de implicación borroso que modela la relación borrosa existente.

La función de pertenencia del conjunto borroso B', resultante de la aplicación del Modus Ponens Generalizado, se obtiene a partir de la Regla Composicional de Inferencia (RCI), desarrollada en el trabajo de Zadeh (1973) de la siguiente forma: "Si R es una relación borrosa definida de U a V y A' es un conjunto borroso definido en U, entonces el conjunto borroso B', inducido por A', viene dado por la composición R y A'", esto es, B' = A' ° R.

De esta forma, cuando la RCI se aplica sobre reglas cuyo antecedente está formado por n variables de entrada y cuyo consecuente presenta una única variable de salida toma la siguiente expresión:

$$\mu_{B'}(y) = \mathrm{Sup}_{x \in U} \{ T'(\mu_{A'}(x), I(\mu_A(x), \mu_B(y))) \}$$

donde $\mu_{A'}(x) = T(\mu_{A'1}(x), \ldots, \mu_{A'n}(x))$, $\mu_A(x) = T(\mu_{A1}(x), \ldots, \mu_{An}(x))$, T y T' son operadores de conjunción borrosos e I es un operador de implicación.

Como en la mayoría de los casos el Interfaz de Borrosificación transforma la entrada $x_o = (x_1, \ldots, x_n)$ que recibe el sistema en una serie de conjuntos borrosos puntuales $A'_1, \ldots, A'_n$, la expresión de la RCI queda finalmente reducida a la forma:

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

Por tanto, el Sistema de Inferencia de un CB desde un punto de vista operativo deberá realizar las dos tareas siguientes: (i) determinar $\mu_A (x_0)$, mediante los Operadores de Conjunción y (ii) aplicar el Operador de Implicación Borroso, I.

3.2.3.1. Operadores de conjunción borrosos

En relación con los operadores de conjunción, cabe poner de manifiesto que el cálculo de $\mu_A (x_0)$ de la expresión resultante simplificada de la RCI consiste en la aplicación de un operador de conjunción sobre los $\mu_{Ai} (x_i)$:

$$\mu_A (x_0) = T (\mu_{A1} (x_1) , \dots , \mu_{An} (x_n))$$

El valor que resulta de la agregación de las entradas mediante el operador de conjunción recibe el nombre de Grado de Emparejamiento (GE), que se suele representar por h, de esas entradas con la regla. El GE representa, en definitiva, una medida de "coincidencia" de los valores que toman las variables de entrada con los valores lingüísticos que describen el antecedente de esa regla.

La información que proporciona el GE puede ser muy útil, dado que el mismo puede definirse en el sentido de que una regla debe ser considerada (dispararse) cuando los valores de entrada coincidan con la situación que describen sus antecedentes.

Cada regla contenida en la BC del sistema experto tiene asociado un GE con las entradas actuales y dicho valor es un número independientemente del tipo Borrosificación aplicado, ya sea aproximada o puntual.

Por otro lado, conviene resaltar que el operador de conjunción se modela habitualmente con una T–norma, esto es, una función T: $(0,1)\times(0,1) \rightarrow (0,1)$ es una t–norma sii $\forall$ x, y, z $\in (0,1)$ verifica las siguientes propiedades (Mizumoto, 1989; Gupta y Qi, 1991a):

1. Existencia de elemento unidad 1: $T (1,x) = x$
2. Monotonicidad: Si $x \leq y$ entonces $T (x,z) \leq T (y,z)$
3. Conmutatividad: $T (x,y) = T (y,x)$
4. Asociatividad: $T (x,T(y,z)) = T(T (x,y),z)$
5. $T (0,x) = 0$

Las t–normas pueden representar el operador de intersección:

$$\mu_{A\cap B} (x) = T (\mu_A (x) , \mu_B (x))$$

En el Cuadro 1 se presentan alguna de las más frecuentemente utilizadas.

Tipo	**Descripción**
Mínimo	T_Norma definida por la relación $f(x,y) = min(x,y)$
Máximo	T_Norma definida por la relación $f(x,y) = max(x,y)$
Suma Acotada	T_Norma definida por la relación $f(x,y) = min(1, x+y)$
Suma Drástica	T_Norma definida por la relación $f(x,y) = \begin{cases} x, y = 0 \\ y, x = 0 \\ 1, x \neq 1 AND y \neq 1 \end{cases}$
Producto	T_Norma definida por la relación $f(x,y) = xy$
Producto Drástico	T_Norma definida por la relación $f(x,y) = \begin{cases} x, y = 1 \\ y, x = 1 \\ 0, x \neq 1 AND y \neq 1 \end{cases}$
Producto Acotado	T_Norma definida por la relación $f(x,y) = max(0, x+y-1)$
Familia Hamacher	T_Norma definida por la relación $f(x,y) = \frac{xy}{p+(1-p)(x+y-xy)}$ $p \geq 0$
Familia Sugeno	T_Norma definida por la relación $f(x,y) = min(1, x+y+p-xy)$ $p \geq -1$
Familia Yager	T_Norma definida por la relación $f(x,y) = 1 - min(1, ((1-a)^p + (1-b)^p)^{1/p})$ $p \geq 0$
Familia Dubois–Prade	T_Norma definida por la relación $f(x,y) = \frac{xy}{max(x,y,p)}$ $0 \leq p \leq 1$

Cuadro 1

3.2.3.2. Operadores de implicación borrosos

En cuanto a los operadores de Implicación Borrosos, en el Cuadro 2 se ponen de manifiesto alguno de los más empleados en la literatura especializada (entre otros, Trillas y Valverde, 1985; Gupta y Qi, 1991a,b; y Duject y Vincent, 1995).

3.2.4. El Interfaz de Clarificación

Como ya se ha descrito anteriormente, la forma de trabajo del Sistema de Inferencia de un SBRB se aplica al nivel de regla individual. Una vez aplicada la Regla Composicional de Inferencia sobre las m reglas que componen la BR, se obtienen m conjuntos borrosos B'_i que representan las acciones borrosas que ha deducido el SBRB a partir de las entradas que recibió.

Los *Grados de Importancia* de una regla R_i de la BC son los siguientes:

- *Área* de un conjunto borroso B' viene definida por la expresión:

$$s = \int_U \mu_{B'}(u)\, du$$

- *Grado de Emparejamiento* (GE) de una regla R_i, que contiene las variables lingüísticas $A_1,\ldots,A_n$, para los valores numéricos $x_1,\ldots,x_n$ de las variables de entrada:

$$h_i = T(\mu_{A1}(x_1), \ldots \mu_{An}(x_n))$$

  Debe considerarse que los GE han sido calculados por el Sistema de Inferencia y, por tanto, si de nuevo se empleasen en el Interfaz de Clarificación, éstos serían, además de los conjuntos borrosos inferidos provenientes de cada regla de la BC, información proveniente de dicho Sistema de Inferencia.

- *Altura* de un conjunto borroso B', se define como:

$$y = \underset{x \in X}{Sup}\ \mu_{B'}(x)$$

  Estos grados de importancia, se emplean en las siguientes definiciones de los llamados Valores Característicos, los cuales son usados en la definición de la mayoría de los métodos de Clarificación.

- *Máximo Valor* de un conjunto borroso B':

$$G = x \in X \ / \ \mu B'(x) = y$$

Cuando más de un valor de x satisface la condición, el máximo valor se puede obtener por varios métodos (Hellendoorn, Driankov y Reinfrank, 1993; Hellendoorn y Thomas, 1993 y Hellendoorn, 1993), tomando el primero, el último o la media de esos dos.

- *Centro de Gravedad* de un conjunto borroso B' :

$$W_i = \frac{\int_Y y \bullet \mu_{B'_i}(y)\, dy}{\int_Y \mu_{B'_i}(y)\, dy}$$

Dado que el sistema debe devolver una salida precisa, el Interfaz de Clarificación debe asumir la tarea de agregar la información aportada por los conjuntos borrosos individuales y transformarla en un valor preciso, para lo cual, de acuerdo con Emani, Türksen y Goldenberg (1998) existen dos formas de trabajo distintas, a saber: clarificación en la Forma FATI y clarificación en la forma FITA.

Tipo	**Descripción**
Producto	Implicación de Ingeniería definida por la relación $f(x,y) = xy$
Mínimo	Implicación de Ingeniería definida por la relación $f(x,y) = min(x,y)$
Kleene Dienes	Implicación Lógica definida por la relación $f(x,y) = max(1-x,y)$
Lukasiewicz	Implicación Lógica definida por la relación $f(x,y) = min(1,1-x+y)$
Zadeh	Implicación Lógica definida por la relación $f(x,y) = max(min(x,y),1-x)$
Estocástica	Implicación Lógica definida por la relación $f(x,y) = max(1-x,xy)$
Goguen	Implicación Lógica definida por la relación $f(x,y) = min(1,\frac{y}{x})$
Godel	Implicación Lógica definida por la relación $f(x,y) = \begin{cases} 1, x \le y \\ y, x > y \end{cases}$
Aguda	Implicación Lógica definida por la relación $f(x,y) = \begin{cases} 1, x \le y \\ 0, x > y \end{cases}$

Cuadro 2

A. Clarificación en la Forma FATI

El Modo FATI consiste en (i) agregar primero y (ii) clarificar o concretar después *(First Aggregate Then Infer)*, esto es:

- Agregar los conjuntos borrosos individuales inferidos, B'_i, para obtener un conjunto borroso final B', empleando para ello un operador de agregación borroso, G, que modela el operador también que relaciona las reglas de la BC:

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

- Transformar el conjunto borroso B' así obtenido en un valor preciso, y_0, que sería proporcionado como salida del sistema global, mediante un método de clarificación, D:

$$\mu_0 = D(\mu_{B'}(y))$$

En relación con los operadores de Agregación, cabe poner de manifiesto que, como ya se ha indicado, la agregación es un proceso encaminado a combinar el conjunto borroso que proviene de la inferencia de cada regla de la BC. El problema que se plantea radica entonces en la propia definición de dicho operador de agregación o combinación que obtenga un conjunto borroso B' partiendo de los conjuntos borrosos individuales B'_i. Los operadores de agregación más comúnmente utilizados son las funciones t–normas y las t–conormas.

La t–norma más utilizada como operador de agregación es la del mínimo y cuando se emplea se dice que se agrega con el tipo MAR (Razonamiento Aproximado de Mamdani).

Las t–conormas se definen a continuación: Una función S: $(0,1)\times(0,1) \rightarrow (0,1)$ es una t–conorma sii $\forall$ x, y, z $\in$ (0,1) verifica las siguientes propiedades (Mizumoto, 1989; Gupta y Qi, 1991a):

1. Existencia de elemento unidad, 0: $S(0,x) = x$
2. Monotonicidad: Si $x \leq y$ entonces $S(x,z) \leq S(y,z)$
3. Conmutatividad: $S(x,y) = S(y,x)$
4. Asociatividad: $S(x,S(y,z)) = S(S(x,y),z)$
5. Existencia de elemento unidad 1: $S(1,x) = 1$

Las t–conormas pueden representar el operador **unión**:

$$\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x))$$

Las t–conormas más conocidas son:

- Máximo: $S(x, y) = \max(x, y)$
- Suma Acotada: $S(x, y) = \min(0, x + y - 1)$
- Suma Algebraica: $S(x, y) = x + y - x \bullet y$

La t–conorma más empleada como operador de agregación es la del máximo y su uso es el denominado FLR (Formal Logical Reasoning).

En relación a los métodos de clarificación empleados para realizar la anteriormente citada clarificación del conjunto borroso resultante de la agregación B', las posibilidades más utilizadas son las dos siguientes:

- La *Media de los Puntos de Máximo Valor* (denominado en la literatura especializada como MOM) del conjunto borroso B':

  $y_1 = \text{Inf}\{z/\ \mu_{B'}(z) = \text{Inf}\ \mu_{B'}(y)\}$

  $y_2 = \text{Sup}\{z/\ \mu_{B'}(z) = \text{Sup}\ \mu_{B'}(y)\}$

  $$y_0 = \frac{y_1 + y_2}{2}$$

- El *Centro de Gravedad* del conjunto borroso B':

  $$y_0 = \frac{\int_Y y \cdot \mu_{B'}(y)\, dy}{\int_Y \mu_{B'}(y)\, dy}$$

Combinando estas dos formas de concretar el conjunto borroso B' con los operadores de agregación mínimo y máximo para los conjuntos borrosos B'_i, se dispone de los cuatro métodos de clarificación operando en la forma FATI siguientes:

- Media de los Puntos de Máximo Valor del conjunto borroso B' resultado de agregar los conjuntos borrosos individuales B'_i con el conectivo Mínimo, (MAR).
- Centro de Gravedad del conjunto borroso individual B' resultado de agregar los conjuntos borrosos individuales B'_i con el conectivo Mínimo, (MAR).
- Media de los Puntos de Máximo Valor del conjunto borroso B' resultado de agregar los conjuntos borrosos individuales B'_i con el conectivo Máximo, (FLR).
- Centro de Gravedad del conjunto borroso individual B' resultado de agregar los conjuntos borrosos individuales B'_i con el conectivo Máximo, (FLR).

B. Clarificación en la forma FITA

El Modo FITA consiste en (i) clarificar o concretar primero y (ii) agregar después *(First Infer Then Aggregate)*. En esta forma de actuar, se considera individualmente la contribución de cada conjunto borroso inferido y la acción precisa final se obtiene mediante algún tipo de operación (bien una media, una suma ponderada o la selección de uno de ellos, entre otros) efectuada sobre un valor preciso característico obtenido a partir de cada conjunto borroso individual, con lo cual, se evita el calculo del conjunto borroso final B', lo que ahorra una gran cantidad de tiempo de cálculo. Este modo de operación supone una aproximación distinta al concepto representado por el operador *también*.

Como en el caso anterior, cabe resaltar la existencia de una gran variedad de métodos de clarificación que emplean la forma FITA, los cuales pueden clasificarse en familias según como estén formados, a saber:

1. Sumas ponderadas por grados de importancia (Tsukamoto, 1979; Hellendoorn y Thomas, 1993 y Cárdenas, Castillo, Cordón, Herrera y Peregrín, 1994):

- Centro de Gravedad ponderado por el área s_i:

$$y_0 = \frac{\sum_i s_i \cdot W_i}{\sum_i s_i}$$

- Centro de Gravedad ponderado por la altura yi:

$$y_0 = \frac{\sum_i y_i \cdot W_i}{\sum_i y_i}$$

- Centro de Gravedad ponderado por el GE hi:

$$y_0 = \frac{\sum_i h_i \cdot W_i}{\sum_i h_i}$$

- Punto de Máximo Valor ponderado por el área:

$$y_0 = \frac{\sum_i s_i \cdot G_i}{\sum_i s_i}$$

- Punto de Máximo Valor ponderado por la altura:

$$y_0 = \frac{\sum_i y_i \cdot G_i}{\sum_i y_i}$$

- Punto de Máximo Valor ponderado por el GE:

$$y_0 = \frac{\sum_i h_i \cdot G_i}{\sum_i h_i}$$

2. Basados en el conjunto borroso de mayor grado de importancia (Pfluger, Yen y Langari, 1992; Hellendoorn, Driankov y Reinfrank, 1993; Hellendoorn y Thomas, 1993; Hellendoorn, 1993; y Cárdenas, Castillo, Cordón, Herrera Y Peregrín, 1994):

- Centro de Gravedad del conjunto borroso con mayor área:

$$B'_k = \{ B'_i \mid s_i = \text{Max}(s_t), \forall\, t \in \{1,\ldots,m\} \}$$

$$y_0 = W_k$$

- Centro de Gravedad del conjunto borroso con mayor altura:

$$B'k = \{ B'i \mid yi = \text{Max}(yt), \forall\, t \in \{1,\ldots,m\} \}$$

$$y0 = Wk$$

- Centro de Gravedad del conjunto borroso de mayor GE:

$$B'k = \{ B'i \mid hi = \text{Max}(ht), \forall\, t \in \{1,\ldots,m\} \}$$

$$y0 = Wk$$

- Punto de Máximo Valor del conjunto borroso con mayor área:

$$B'k = \{ B'i \mid si = \text{Max}(st), \forall\, t \in \{1,\ldots,m\} \}$$

$$y0 = Gk$$

- Punto de Máximo Valor del conjunto borroso de mayor altura:

$$B'k = \{ B'i \mid yi = \text{Max}(yt), \forall\, t \in \{1,\ldots,m\} \}$$

$$y0 = Gk$$

- Punto de Máximo Valor del conjunto borroso de mayor GE:

$$B'k = \{ B'i \mid hi = \text{Max}(ht), \forall\, t \in \{1,\ldots,m\} \}$$

$$y0 = Gk$$

3. Otros:

- Media de los Máximos Valores:

$$y_0 = \frac{\sum_i G_i}{m}$$

donde m es el número de conjuntos borrosos no nulos obtenidos en el proceso de inferencia.

- Media del Mayor y Menor Valor:

  Gmin = Min Gi , $\forall$ i $\in$ {1,...,m}

  Gmax = Max Gi , $\forall$ i $\in$ {1,...,m}

$$y_0 = \frac{G_{min} + G_{max}}{2}$$

- Centro de Sumas:

$$y_0 = \frac{\sum_i \int_Y y \cdot \mu_{B'i}(y)\, dy}{\sum_i \int_Y \mu_{B'i}(y)\, dy}$$

3.3. Metodología de la construcción de Sistemas Expertos Borrosos (SBRBS)

El buen comportamiento de un SBRB depende directamente de dos factores, la forma en que realiza el proceso de inferencia y la composición de la base de reglas que maneja. De hecho, el proceso de construcción de este tipo de sistemas engloba dos tareas principales, a saber: (i) diseñar el Sistema de Inferencia, es decir la elección de los distintos operadores borrosos que se emplearán para realizar dicho proceso y (ii) obtener una Base de Conocimiento que contenga información adecuada con respecto al problema que se pretende resolver.

3.3.1. Diseño del Sistema de Inferencia y selección de operadores

Respecto al Diseño del Sistema de Inferencia, conviene poner de manifiesto que la complejidad de esta primera tarea depende del tipo de SBRB con el que se esté trabajando. De hecho, cuando se trabaja con sistemas basados en conocimiento, la complejidad en el diseño del mecanismo de inferencia aumenta en gran medida. De ahí que sea preciso definir la com-

posición de tres de los elementos anteriormente introducidos (el Sistema de Inferencia y los Interfaces de Borrosificación y Clarificación), para lo cual, de acuerdo con Takagi y Sugeno (1985), se deben tomar las cinco decisiones siguientes:

1. Definir el conectivo Y, es decir, elegir el operador de conjunción T a usar en el caso de que las reglas de la base presenten más de una variable de entrada. Para esta elección se dispone de distintos operadores de la familia de las t–normas.
2. Definir el operador de implicación en las reglas lingüísticas de tipo "Si–entonces" contenidas en la BC, es decir, elegir el operador borroso I que se empleará para modelar la implicación.

   Existen distintas posibilidades para la elección de este operador. Por ejemplo, cabe recordar que en el primer modelo de SBRB, Mamdani empleó la t–norma del Mínimo (Mamdani, 1974), lo que ha provocado que varios operadores pertenecientes a esta misma familia hayan sido posteriormente empleados para dicha tarea (Gupta y Qi, 1991b).

   Por otro lado, la familia de las funciones de implicación borrosas ofrece una amplia variedad de operadores de implicación clasificados en distintos grupos, dependiendo del modo en que interpretan la implicación borrosa (Trillas y Valverde, 1985).
3. Definir matemáticamente la composición de relaciones borrosas a aplicar en la Regla Composicional de Inferencia. El operador de composición empleado con mayor generalidad es el Sup–T, donde T es una t–norma. De hecho, resulta común el empleo de cuatro tipos de composición basados en otras tantas t–normas: el mínimo, el producto algebraico, el producto acotado y el producto drástico, siendo las más utilizadas las dos primeras (Lee, 1990).
4. Definir el operador de agregación *también*, es decir, elegir el operador GE según el modo de clarificación que emplee el SBRB. Como ya se comentó, en caso de trabajar en Modo FATI, la función del operador de agregación sería la de agregar todos los conjuntos borrosos individuales resultantes de la inferencia en un único conjunto borroso global. Para esa tarea se emplean habitualmente las t–normas y las t–conormas, principalmente el mínimo y el máximo, respectivamente, por su sencillez.

A este respecto, en el trabajo de Bardossy y Duckstein (1995) se propone otra batería de operadores de agregación más sofisticados y se presenta un estudio sobre sus propiedades, así como un estudio comparativo de su comportamiento.

Por otro lado, en caso de trabajar en Modo FITA, los operadores habitualmente empleados son la media, la media ponderada o la selección de algún valor característico de los conjuntos borrosos, en función de algún grado de importancia de la regla que los ha generado en el proceso de inferencia. Como valores característicos se suelen emplear el Centro de Gravedad y el Punto de Máximo Criterio y como grados de importancia de la regla, el área y la altura del conjunto borroso inferido, o el GE de los antecedentes de la misma con la entrada actual al sistema (Hellendoorn y Thomas, 1993 y Sugeno y Yasukawa, 1993).

5. Definir el método de Clarificación a emplear para transformar los conjuntos borrosos individuales o globales resultantes del proceso de inferencia en valores precisos de salida. Los más habituales son el Centro de Gravedad, y la Media de los Máximos (Hellendoorn, Driankov y Reinfrank, 1993), cuando se trabaja en Modo FATI, y el Centro de Gravedad y el Punto de Máximo Criterio, al hacerlo en Modo FITA.

3.3.2. Obtención de la Base de Conocimiento (BC)

La BC es el único componente que depende directamente de la aplicación específica y, al igual que en la construcción del Sistema de Inferencia, su generación conlleva al menos tres tareas principales de diseño, a saber:

1. Selección de las variables relevantes de entrada y salida al sistema, de entre todas las variables disponibles. Ésta puede ser efectuada por un experto o bien empleando métodos estadísticos, basados en analizar la correlación existente entre las variables de las que se dispone, o combinatorios, que analizan la influencia de conjuntos formados por distintas combinaciones de variables (BARDOSSY y DUCKSTEIN, 1995).
2. Descripción de la estructura de la BD que contendrá la semántica de los términos que pueden tomar como valor las variables lingüísticas de entrada y salida, lo cual incluye las siguientes sub–tareas relacionadas: (i) definición de los factores de escala; (ii) elección de los conjuntos de términos posibles para cada variable lingüística, lo que permite determinar la granularidad deseada en el sistema; (iii) elección del tipo

de función de pertenencia a emplear, principalmente triangulares, trapezoidales, gausianas o exponenciales (Hellendoorn, Driankov y Reinfrank, 1993); y (iv) definición de la función de pertenencia del conjunto borroso concreto asociado a cada etiqueta lingüística. Las dos primeras presentan la ventaja de su simplicidad a la hora de efectuar los cálculos computacionales mientras que las dos últimas facilitan tener una transición más suave.

3. Derivación de las reglas lingüísticas que compondrán la base de reglas del sistema. Para ello será necesario determinar el número de éstas así como su composición, mediante la definición del antecedente y el consecuente.

En cuanto a las posibilidades para la definición de la Base de Conocimiento, de acuerdo con los trabajos de Lee (1990) y Berenji (1993), cabe distinguir entre cuatro formas de obtener las reglas borrosas de control, las cuales no son mutuamente exclusivas, a saber:

1. *Experiencia y Conocimientos del Experto*. Es la más ampliamente utilizada, siendo efectiva cuando el operador humano (gerente) es capaz de expresar lingüísticamente las reglas de control que él mismo utiliza para controlar el sistema. Estas reglas son generalmente de tipo Mamdani.
2. *Modelando las Acciones de Control de un Operador*. Las acciones de control se deducen realizando un modelo de las acciones de un operador sin que éste intervenga.
3. *Realizando un Modelado Borroso del Proceso*. Consiste en desarrollar un modelo borroso del sistema y construir las reglas de la BC a partir de él. Esta aproximación es similar a la utilizada tradicionalmente en la Teoría de Control, lo cual ha llevado a distintos autores (Pfluger, Yen y Langari, 1992) a sugerir la necesidad de una identificación de parámetros.
4. *Basándose en Aprendizaje y Auto–Organización*. Esta posibilidad se basa en la habilidad de crear y modificar las reglas de control para mejorar los resultados del controlador utilizando métodos automáticos.

La estructura de datos de la BC envuelve dos tipos de información conceptualmente clasificados anteriormente como Base de Datos (BD) y Base de Reglas (BR). Generalmente, salvo para aplicaciones específicas (Controladores Borrosos adaptativos que modifican sus normas de com-

portamiento en función de los resultados obtenidos por las acciones de control previamente administradas), la BC es una estructura de datos estática con un tamaño predefinido y un contenido fijo.

La estructura de datos de la BC y su correspondiente información debe ser especificada como paso previo o inicial, esto es, antes de que el SBRB comience a funcionar. La información sobre los conjuntos borrosos asociados a los términos lingüísticos empleados en las reglas borrosas constituye la BD.

Para simplificar los cálculos en la implementación práctica de SBRBs se emplean funciones de pertenencia con tramos rectilíneos de modo que cada conjunto borroso se define generalmente por una representación paramétrica con tres o cuatro puntos.

La Figura 3 muestra un ejemplo de representación paramétrica de un conjunto borroso trapezoidal en función de cuatro puntos (x0, x1, x2, x3), donde se mantiene la asunción de que los conjuntos borrosos están normalizados por lo que no es necesario almacenar su altura.

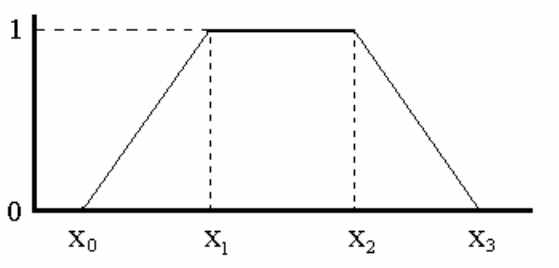


Figura 3

Los valores de estos tres o cuatro puntos de definición del conjunto borroso están definidos en el universo de discurso de la variable correspondiente. La Figura 4 muestra un ejemplo de la partición del universo de una variable en cinco conjuntos borrosos trapezoidales basada en los conjuntos borrosos VS (muy pequeño), S (pequeño), M (medio), L (grande) y VL (muy grande).

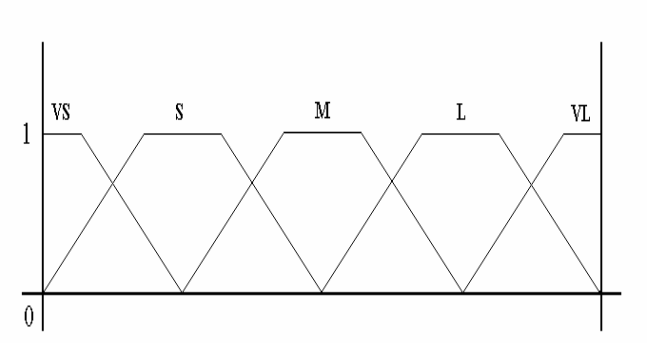


Figura 4

Así pues, cada conjunto borroso de cada una de las particiones borrosas de las variables debe ser descrito y almacenado para constituir la Base de Datos (BD). La información correspondiente a la Base de Reglas consiste en un conjunto de reglas en el que cada una hace referencia a sus correspondientes conjuntos borrosos de dicha BD.

3.3.3. Construir el Interfaz de Borrosificación

La forma más simple y sencilla de codificar el Interfaz de Borrosificación en Control Borroso es utilizando la Borrosificación puntual. La Borrosificación puntual no precisa ninguna operación porque es precisamente en los valores discretos de las variables de entrada donde los conjuntos borrosos valen 1, siendo 0 en el resto. Esto es, los valores de las variables de entrada directamente representan los conjuntos borrosos puntuales.

$$A'(x) = \begin{cases} 1, & si \quad x = x_0 \\ 0, & en\ otro\ caso \end{cases}$$

3.3.4. Aplicar el Sistema de Inferencia

El cálculo del Grado de Emparejamiento (GE) se lleva a cabo recorriendo cada una de las reglas de la BC y calculando el punto de intersección hi entre el conjunto borroso A' (obtenido a partir de la entrada xi) y el conjunto borroso del antecedente de la regla, tal como se muestra en la Figura 25.

Si las reglas tienen más de un antecedente deben calcularse todos los puntos de intersección, el de cada entrada con su correspondiente antecedente de la regla, obteniéndose así tantos valores h_{ij} como antecedentes.

Para calcular el GE deben considerarse las diferentes zonas del conjunto borroso. Por ejemplo, en la Figura 5 cabe considerar las siguientes cinco posibles zonas junto con sus expresiones de altura correspondientes:

$$e_j' \le x_0 : h_{ij} = 0$$

$$x_0 < e_j \le x_1 : h_{ij} = \frac{e_j - x_0}{x_1 - x_0}$$

$$x_1 < e_j \le x_2 : h_{ij} \quad 1$$

$$x_2 < e_j \le x_3 : h_{ij} = \frac{e_j - x_3}{x_2 - x_3}$$

$$x_3 < e_j' : h_{ij} = 0$$

donde e_j es el valor actual de la variable de entrada x_j.

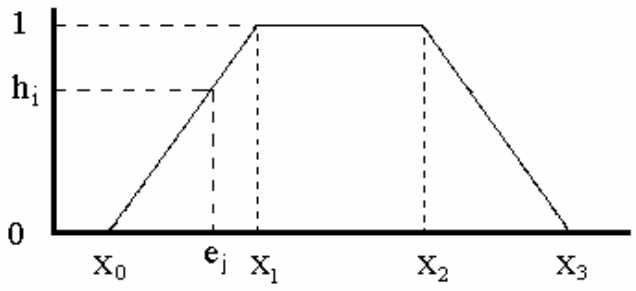


Figura 5

Posteriormente, debe aplicarse el conectivo u operador de conjunción para obtener el i–ésimo valor del GE, h_i, con i=1 hasta N, siendo N el número de reglas de la BC.

Si el operador conectivo de las reglas es la conjunción Y, entonces la función aplicada sera una t–norma (por ejemplo, el mínimo). Pero, si el operador conectivo es la disyunción OR, entonces la función aplicada debería ser una t–conorma (por ejemplo, el máximo).

En todo caso, conviene poner de manifiesto que la elección de la t–norma que implementa el operador conjunción no debe de afectar significativamente en la precisión de la respuesta del SBRB.

En la Figura 6 puede apreciarse gráficamente el mínimo como conectivo *Y*, esto es, la aplicación de la t–norma del mínimo resultando h_i = Min (h_{i1},h_{i2}).

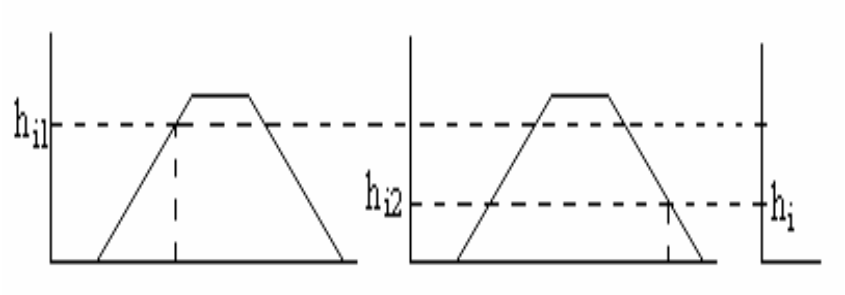


Figura 6

Como ya se ha indicado, los valores h_i deben almacenarse en el Vector de GE implementado en una matriz o lista de números reales de tamaño N.

De esta forma, es posible obtener el conjunto borroso inferido por cada regla utilizando el consecuente y el GE de cada regla. El operador de implicación elegido producirá un conjunto borroso inferido con una forma concreta definida por una expresión paramétrica.

Normalmente, los conjuntos borrosos inferidos desde cada regla deben ser almacenados para llevar a cabo la clarificación. En situaciones concretas no es necesario almacenarlos debido a que el Interfaz de Clarificación puede actuar al mismo tiempo, como será objeto de consideración posteriormente.

Los conjuntos borrosos inferidos podrían describirse con un número variable de puntos, generalmente seis u ocho, de la misma forma que los conjuntos borrosos de los antecedentes y consecuentes se describen en la BC, añadiendo un valor adicional a cada uno de ellos para la altura, ya que no tiene porqué ser 1 como en el caso de los conjuntos borrosos existentes en las reglas borrosas de la Base de Conocimiento.

Así pues, la estructura de datos para estos conjuntos borrosos será una matriz o lista de puntos llamada Vector Consecuente como se mencionó anteriormente. Por ejemplo, si se utiliza la t–norma del mínimo como operador de implicación, los conjuntos borrosos inferidos serán como los presentados en la Figura 7.

Este gráfico es el resultado de aplicar la expresión h del operador de implicación mínimo (I(x,y)=Min(x,y)) sobre el GE y el conjunto borroso consecuente de la regla.

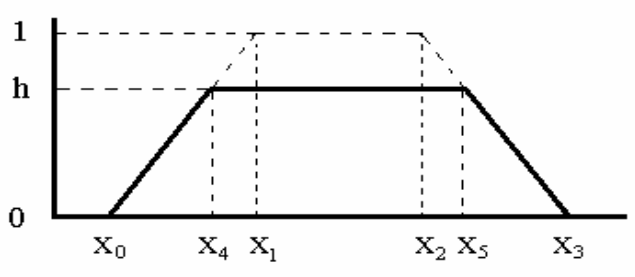


Figura 7

En el caso del operador de implicación del mínimo, como en otros casos, la información sobre la altura de algunos de sus puntos coincide con el GE. Los puntos x_0 y x_3 son los mismos que en el consecuente de la regla. Los otros dos, x_4 y x_5, pueden calcularse como la intersección de las rectas que pasan por x_0 y x_1 y la que pasa por x_2 y x_3 con la horizontal de altura h, respectivamente. Las expresiones son:

$$x_4 = x_0 + (x_1 - x_0) \cdot h_i$$
$$x_5 = x_3 - (x_3 - x_2) \cdot h_i$$

Estas expresiones pueden ser directamente incorporadas al código fuente de un BC con su correspondiente altura, h. Del mismo modo, cualquier otro operador de implicación podría ser usado obteniendo los puntos de definición e implementando la expresión que describa su cálculo.

Por último, para llevar a cabo la implementación del SBRB cabe distinguir entre dos métodos operativos, a saber:

1. *Método Exacto*: Se requiere la representación gráfica del conjunto borroso inferido. A partir de ahí, se obtienen los puntos de definición del conjunto borroso. Estos puntos serán almacenados en la estructura de datos para el Sistema de Inferencia anteriormente citado, es decir, una lista de puntos para cada conjunto borroso inferido procedente de las reglas.

   Por ejemplo, observando la Figura 7 anterior, los puntos de definición que serán almacenados serían los siguientes:

   ($x_0' = x_0$, 0) ($x_3' = x_3$, 0)
   ($x_1' = x_4$, h) ($x_2' = x_5$, h)

donde x_4 y x_5 tienen que ser calculados con la expresión previamente mostrada, y x_0 y x_3 son los puntos de definición para el consecuente de la regla.

La ventaja de este método desde el punto de vista del Sistema de Inferencia es el consumo mínimo de memoria y la eficiencia del proceso de inferencia. La desventaja es que requiere un estudio previo del operador de implicación para determinar la representación gráfica y las expresiones de los puntos. Si se desea añadir a la implementación un grupo importante de operadores de implicación, el proceso de diseño será muy duro y susceptible de cometer errores.

2. *Método Aproximado*: El desarrollo operativo, el universo del consecuente se divide en un conjunto fijo de puntos estableciendo una discretización. La expresión matemática del operador de implicación será directamente incorporada en el código, de modo que el conjunto borroso inferido será calculado con la expresión del operador de implicación, los puntos de la discretización y el GE.

Una posible discretización para el *operador de implicación de Gogüen* se muestra en la Figura 8, donde la expresión del operador de implicación de Gogüen, tal como se mencionó en el Cuadro 2, es la siguiente:

$$I_{Goguen}(x,y) = \begin{cases} 1, & si\ y \le x \\ y, & en\ otro\ caso \end{cases}$$

y, por tanto, la función de discretización f que la aproxima es:

$$f_i = I_{Goguen}(h\ ,\ \mu_B(i))$$

donde B es el consecuente de la regla e i toma un conjunto finito de valores equidistantes entre 0 y el extremo del dominio de la variable.

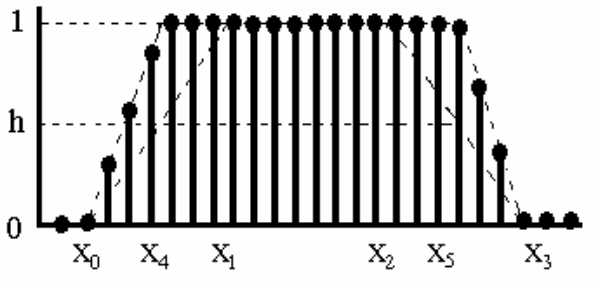


Figura 8

3.3.5. Elaboración del Interfaz de Clarificación

En cuanto a la implementación del desborrosificador o Interfaz de Clarificación, ésta presenta una gran dependencia de la forma de concretización elegida, si bien a los efectos de la presente Memoria tan sólo se concentrará la atención en el Modo FATI.

A este respecto, como se ha mencionado anteriormente, el Interfaz de Clarificación que trabaja en Modo FATI realiza en primer lugar la agregación de los conjuntos borrosos individuales inferidos, B_i', mediante el operador *también*, con objeto de obtener el conjunto borroso final de salida B'. Los operadores de agregación que modelan el conectivo *también* son típicamente el mínimo o el máximo. Las Figuras 9 y 10 muestran gráficamente el comportamiento del conectivo *también* realizado con los operadores máximo y mínimo, respectivamente.

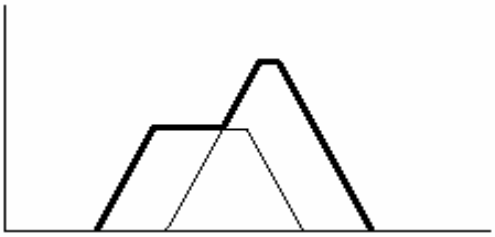


Figura 9

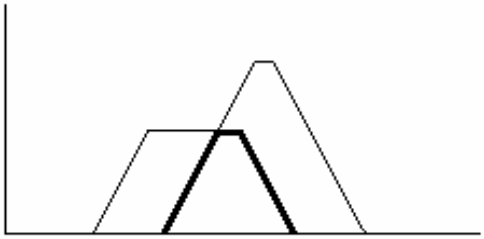


Figura 10

En este caso, la aplicación del Método Exacto conlleva un importante grado de complejidad, pues los operadores de agregación máximo y mínimo deben actuar con los conjuntos borrosos descritos como segmentos rectilíneos. La estructura de datos para manejar el conjunto borroso inferido es una lista de puntos, por lo que debe crearse una nueva lista de puntos auxiliar en el proceso de agregación.

De acuerdo con lo anterior, no resulta extraño que el método comúnmente utilizado consiste en construir una función que agregue los conjuntos borrosos de dos en dos, esto es, obteniendo una nueva lista de puntos que representen el conjunto borroso agregado a partir de otros dos conjuntos iniciales. Ejecutando esta función tantas veces menos uno como conjuntos borrosos individuales hayan sido inferidos, se obtendrá en conjunto borroso agregado final.

El algoritmo de agregación determina las posiciones relativas de los segmentos que integran los conjuntos de entrada y calcula, si existen, los puntos de cruce que serán añadidos a la lista de salida. En muchos casos puede ser interesante diseñar una función de simplificación para estudiar y eliminar los puntos innecesarios añadidos a la lista, lo cual mejorará el rendimiento de la función de agregación.

La agregación de los conjuntos borrosos inferidos en el Método Aproximado cuando se trabaja en Modo FATI es fácil de implementar con un algoritmo que recorra todos los puntos de la discretización de los conjuntos borrosos inferidos calculando el máximo o el mínimo de ellos. La misma operación puede ser utilizada para modelar cualquier otro conectivo también. Las Figuras 11 y 12 presentan gráficamente la agregación con máximo y con mínimo, respectivamente.

Figura 11

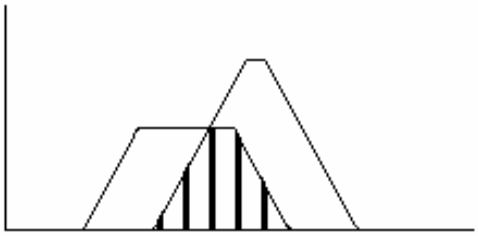


Figura 12

Una vez realizada la agregación, al objeto de poder llevar a cabo propia la concretización o desborrosificación del conjunto borroso final B', se podrían utilizar los dos métodos siguientes:

1. La *Media de los Máximos* (usualmente denominado MOM):

$$y_0 = \frac{y_1 + y_2}{2}$$

donde $y_1 = \text{Min}\ \{z/\ \mu_{B'}(z) = \text{Max}\ \mu_{B'}(y)\}$ y $y_2 = \text{Max}\ \{z/\ \mu_{B'}(z) = \text{Max}\ \mu_{B'}(y)\}$

A efectos ilustrativos, en la Figura 13 se muestra gráficamente dicho método de clarificación del conjunto borroso B', resultado de la agregación de los conjuntos borrosos individuales B'_i mediante el máximo modelando el conectivo *también*.

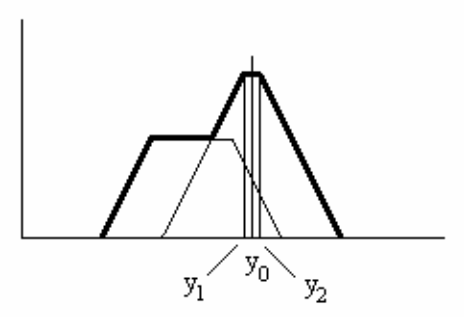


Figura 13

A. En el *Método Exacto*, el punto Media de los Máximos se puede calcular fácilmente, debiéndose tener en cuenta que se pueden presentar las distintas posibilidades:

   A.1. El punto de máximo valor es único: este punto será el extremo de un segmento y puede localizarse comparando simplemente los extremos de los segmentos que componen el conjunto borroso agregado.

   A.2. El máximo valor es un número contable de puntos: de modo similar, este conjunto de puntos serán los extremos de segmentos. El resultado final se calcula como la media del valor más bajo y el más alto.

   A.3. El máximo valor es un conjunto de puntos incontable: es decir, un segmento horizontal. En este caso, si no hay otros puntos individuales, se utilizarán los extremos para calcular el punto medio.

B. En el *Método Aproximado*, la Media de los Máximos se calcula recorriendo la discretización y reteniendo los puntos de máxima altura. Si la máxima altura no es única, entonces se calcula la media de los puntos más bajo y más alto.

2. El *Centro de Gravedad*, cuya expresión exacta es:

$$W = \frac{\int_V y \bullet \mu_{B'}(y)\, dy}{\int_V \mu_{B'}(y)\, dy}$$

La Figura 14 muestra gráficamente el resultado del método de clarificación Centro de Gravedad del conjunto borroso B', resultado de la agre-

gación de los conjuntos borrosos individuales B'$_i$ con el conectivo *también* como mínimo.

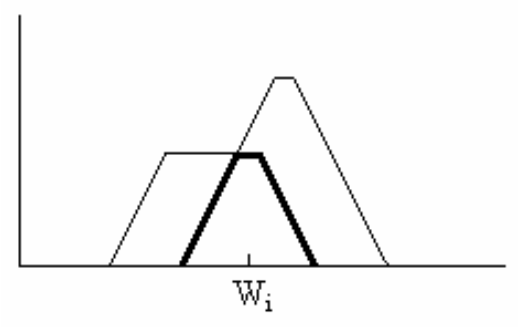


Figura 14

Por su parte, en el Método Aproximado, el Centro de Gravedad se obtendrá fácilmente porque las integrales se calculan como sumas en el caso discreto, de este modo:

$$W = \frac{\sum_{j=1..N} y_j \bullet \mu_{B'}(y_j)\, dy}{\sum_{j=1..N} \mu_{B'}(y_j)\, dy}$$

donde N se corresponde con el número de puntos de la discretización.

4. DISEÑO DE UN SISTEMA EXPERTO HORROROSO DE DETECCIÓN DE INTRUSOS BASADO EN XFUZZY

4.1. Introducción

En este apartado se pretende presentar el diseño de un sistema de detección de intrusos con un SBRB embebido que interpreta esta información de forma adecuada filtrando el exceso de datos que pueden hacer a un administrador disminuir la atención que debe de prestar al sistema. Dicho sistema experto borroso, objeto principal de la atención de este apartado, se basa en la interpretación de la información recibida de gran cantidad de sensores a través de un sistema experto desarrollado con Xfuzzy 3.0 para evaluar a partir de ella la posibilidad de que el usuario en el que se centra el interés suspicaz tenga intenciones nocivas.

Para llevar a cabo tal análisis se ha optado por enfocar este sistema experto borroso en la constatación de ciertos "síntomas" que pueden indi-

car que el presunto atacante se encuentra en alguna de las 5 fases por las que se suele pasar para atacar un sistema, a saber:

- **Investigación externa**: Es la menos peligrosa con amplia diferencia. De hecho, no suele estar abordada como tal en los IDSs comerciales al uso. Consiste en la recopilación de información del sistema a través de procedimientos no intrusivos, que no deberían de despertar la más mínima sospecha y son complicados del detectar al confundirse entre la gran multitud anónima de usuarios. Incluye entre otras cosas la navegación por las páginas del sistema averiguando información al respecto del sitios, sus usuarios y administradores, la investigación de las direcciones de que dispone a través de sus DNSs, etc...
- **Investigación interna**: Son otros métodos más de recopilación de información del sistema antes de intentar acceder a él. La diferencia respecto a la fase anterior, es que en este caso la recopilación de información se hace de forma más intrusiva, a través de peticiones que los usuarios comunes raramente realizan y por lo tanto son más fácilmente detectables. De la misma manera, la peligrosidad de esta "investigación" ya puede calificarse de media, pues puede dejar en las manos del atacante información que desvele importantes problemas de seguridad del sistema objeto de estudio. Incluye procedimientos como los escaneos de puertos, la verificación de existencias de programas vulnerables instalados o las consultas a servicios que pueden proporcionar datos sobre los usuarios del sistema.
- **Intrusión**: Son los intentos de conseguir acceso al sistema de forma ilícita a través de cualquier sistema. Estas actividades implican de forma inequívoca un alto riesgo y ya se incluyen dentro de lo que es el ataque a un sistema propiamente dicho. Incluyen tareas como el intento de ataques de diccionario para acceso a una máquina, el overflow a determinadas aplicaciones o el envío de parámetros con líneas de comando a determinados programas CGI.
- **Borrado de huellas**: Es la eliminación de las pistas que puedan haber quedado en un sistema señalando la intrusión del presunto atacante vigilado. En sí misma no sería peligrosa, pero acompañada de señales de que se ha producido algún intento de los explicados en las fases anteriores indica una seria peligrosidad, ya que corroboraría que el usuario ha conseguido acceder al sistema. Incluye tareas como el bo-

rrado de logs, la creación de nuevas cuentas de usuario o la instalación de troyanos.

- **Aprovechamiento**: Consiste en sacar partido de el acceso que se ha conseguido al sistema para el provecho del intruso. Es muy peligroso, porque en muchas ocasiones se realizan tareas altamente nocivas para el sistema. Incluye tareas como la copia de ficheros, el borrado de ficheros o el uso de recursos del sistema.

Combinando estas 5 etapas, se tratará de obtener un análisis suficientemente concreto y desde un punto de vista "histórico" que normalmente no suele contemplarse en este tipo de software de detección.

4.2. Mapa cognoscitivo de las variables

Al objeto de facilitar una comprensión global del sistema de detección de intrusos, una de las primeras actividades a llevar a cabo consiste en la determinación de las relaciones que pueden existir entre las diferentes variables del modelo, es decir, establecer el conocimiento o expertizaje que revela la opinión o estimación de las vinculaciones entre las variables de entrada y de salida.

En el caso objeto de consideración, se han utilizado las Figuras 15 y 16 para mostrar los dos esquemas asociativo entre variables que proporcionarían los diversos sensores que tendría instalado el DNS y los agrupa en 5 variables intermedias que representan las 5 fases del ataque de un intruso que se han explicado anteriormente. Estas variables son indicativas de las siguientes circunstancias:

- Posibilidad de que se haya realizado una investigación interna
- Posibilidad de que se haya realizado una investigación externa
- Posibilidad de que se haya realizado una intrusión
- Posibilidad de que se hayan borrado huellas que muestren la intrusión
- Posibilidad de que se haya hecho un aprovechamiento del acceso al sistema

Además, estas 5 variables se han agrupado para mayor estructuración, en otras dos variables:

- Posibilidad de investigación, que agrupa a las posibilidades de "investigación externa" e "investigación interna". Por tanto, tal como se

muestra en la Figura 15, indica si el usuario vigilado está haciendo sus indagaciones para entrar al sistema.

- Posibilidad de ataque, que agrupa a las posibilidades de "intrusión", "borrado de huellas" y "aprovechamiento" e indica, tal como se muestra en la Figura 16, si el usuario vigilado ya ha pasado a la acción y ha efectuado un ataque al sistema.

Estas dos variables podrían a su vez ser agrupadas en una única variable fina que definiría si el usuario es realmente malintencionado y se precisa ocuparse por impedirle el acceso al sistema objeto de atención.

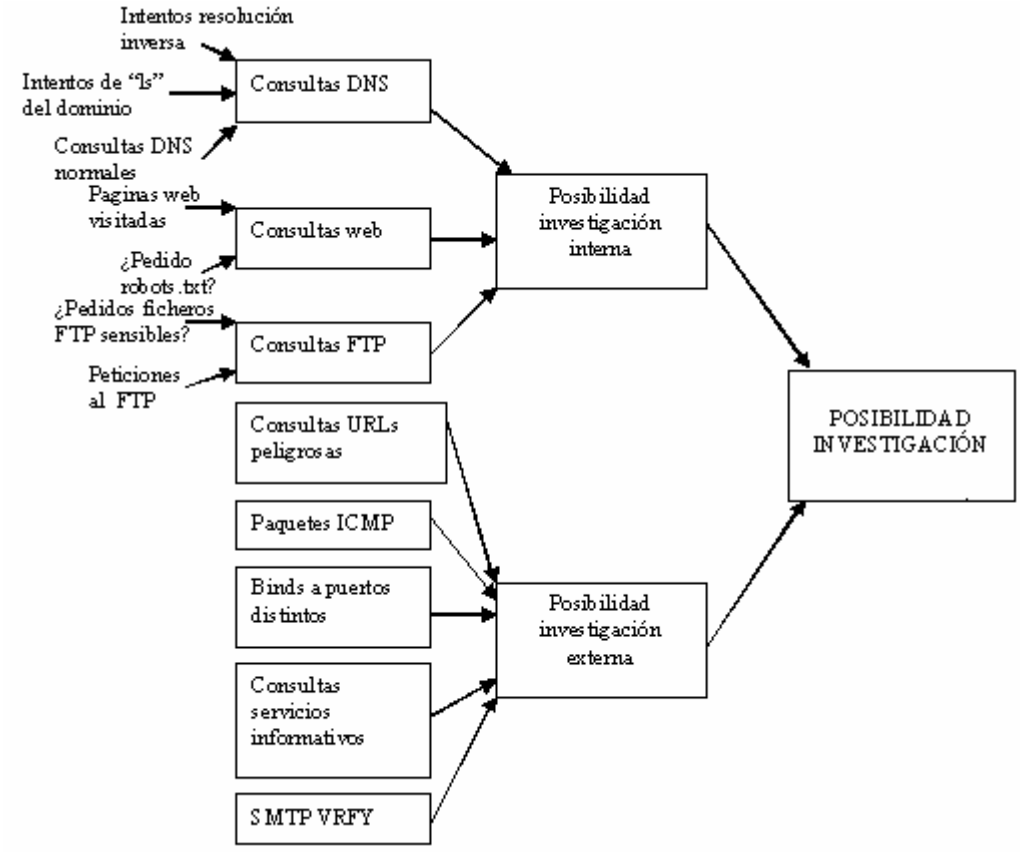


Figura 15

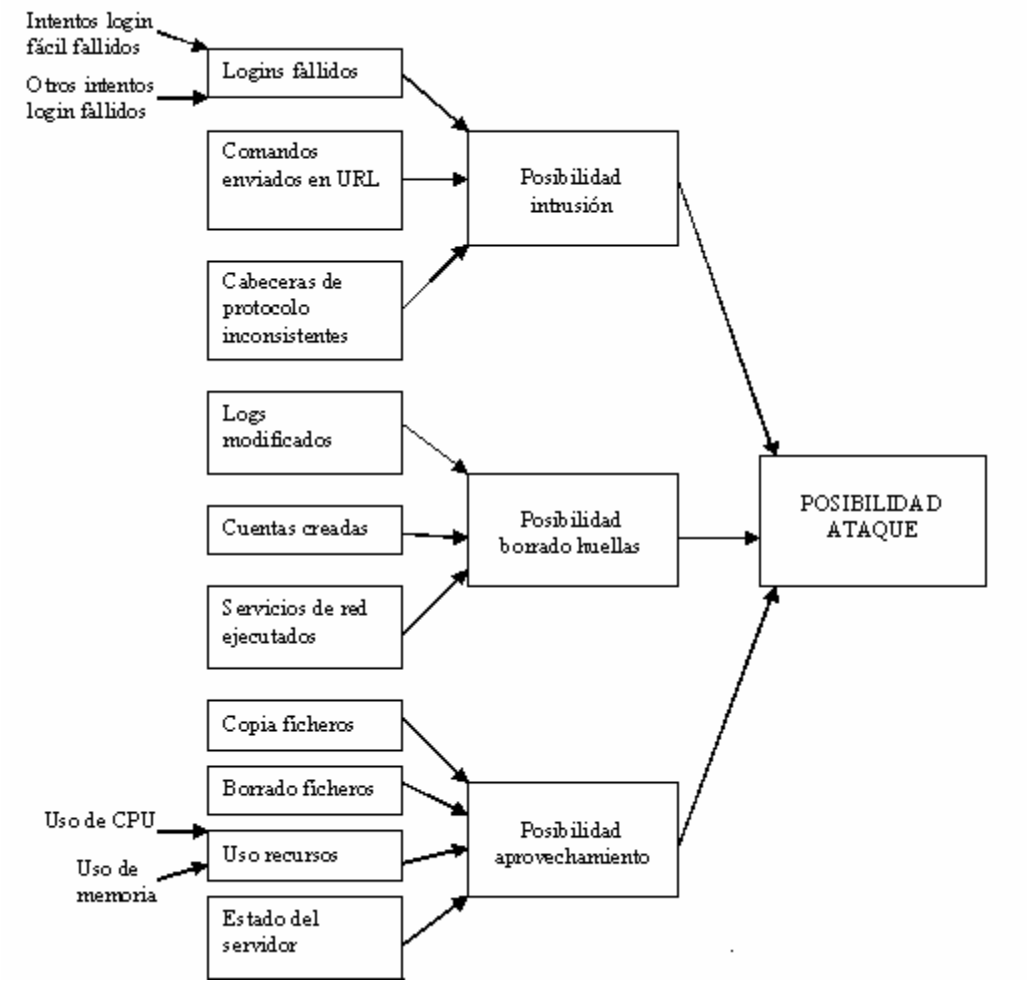


Figura 16

4.3. Etapa de Borrosificación: las variables de entrada.

A los efectos del presente trabajo se han considerado como variables de entrada los principales factores que controlarían los sensores del sistema de detección de intrusos y de los cuales éste debería tomar las medidas correctoras oportunas, que en el caso objeto de estudio se trasladan entonces como entradas al sistema de detección de intrusos propuesto, a saber:

- Intentos de resolución inversa
- Intentos de "ls" del dominio
- Consultas DNS normales
- Páginas web visitadas
- ¿Pedido robots.txt?
- ¿Pedidos ficheros FTP sensibles?
- Ficheros de FTP listados
- Consultas URLs peligrosas
- Paquetes ICMP
- Binds a puertos diferentes
- Consultas servicios informativos
- SMTP VRFY
- Intentos login fácil fallidos
- Otros intentos login fallidos
- Comandos enviados en URL
- Cabeceras de protocolo inconsistentes
- Logs modificados
- Cuentas creadas
- Servicios de red ejecutados
- Copia ficheros
- Borrado ficheros
- Utilización de CPU
- Utilización de Memoria
- Estado del servidor

Todas las medidas se llevan a cabo dentro de un plazo temporal limitad que por defecto ha sido establecido una hora y adicionalmente cabe considerar que gozan de “memoria” o “persistencia”, esto es, una vez alguien ha hecho uno de los intentos desde una dirección concreta queda registrado el valor de esa variable establecido y no se “borra” más que con los valores de utilización posterior o superiores.

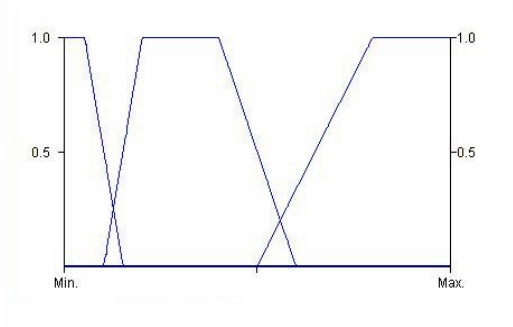
A continuación se describen estas variables, con sus correspondientes distribuciones (valores posibles que pueden tomar).

4.3.1. Intentos de resolución inversa

La resolución inversa DNS es el mecanismo del sistema de nombres de Internet por el cual, a partir de una dirección IP, se proporciona el nombre de ese servidor. Este mecanismo de resolución, si bien puede ser utilizado por cualquier usuario, no suele ser habitual, ya que al navegar los usuarios normales únicamente necesitan la resolución directa (conseguir la dirección IP a partir de un nombre). En numerosas ocasiones esta suele ser una de las primeras maneras por las que un hacker comienza a conocer un sistema, intentando averiguar su nombre a partir de su dirección, pues este proporciona en casi todas las ocasiones una pista infalible del servicio que tiene albergado (www, ftp, chat...).

Esta variable indica el número de peticiones de resolución inversa de direcciones IP diferentes pertenecientes al rango protegido que se han hecho al DNS objeto de estudio (que se supone es el que alberga la resolución inversa) desde la red a la que pertenece la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 3.

IRI	**Normales**	**Altos**	**Excesivos**
No menor que	0	2	10
Igual que	0	4	16
Igual que	1	8	∞
No mayor que	3	12	∞



Cuadro 3

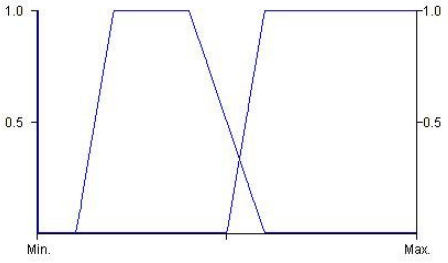
4.3.2. Intentos de "ls" del dominio

El protocolo estándar utilizado por los servidores DNS incluye una gran cantidad de comandos adicionales a los habitualmente utilizados por los usuarios en su acceso. Entre estos comandos, hay uno que es utilizado entre los servidores DNS maestro–esclavo para sincronizar sus contenidos, que es el comando "ls". Este comando devuelve una lista de todos los regis-

tros existentes dentro de un determinado dominio del DNS, lo cual es una información enormemente valiosa para que un hacker comience a investigar las máquinas de las que dispone una corporación. El uso de este comando está habitualmente limitado por los servidores, pero un potencial asaltante podría comprobar si el administrador del sistema DNS ha sido lo suficientemente descuidado como para dejar esta información a la libre disposición de cualquier usuario.

Esta variable indica el número de consultas con el comando "ls" sobre alguno de los dominios a los que pertenezcan las máquinas protegidas que se han hecho al DNS (que se supone que es el que alberga los dominios objeto de la vigilancia) desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 4.

ILD	**Normales**	**Altos**	**Excesivos**
No menor que	0	1	5
Igual que	0	2	6
Igual que	0	4	∞
No mayor que	0	6	∞



Cuadro 4

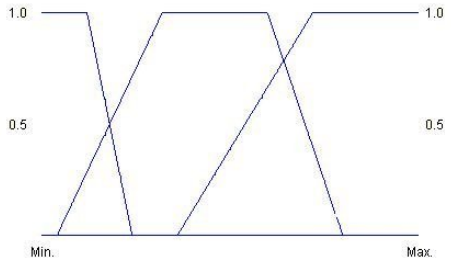
4.3.3. *Consultas DNS normales*

El servicio de nombres de dominio (DNS) es el que proporciona, a partir de un nombre de servidor, la dirección IP que le corresponde. Los usuarios habitualmente utilizan el servidor DNS de su red o proveedor para consultar la dirección IP que le corresponde a los sitios a los que se quieren conectar. Por ello un servidor DNS normalmente sólo recibirá peticiones desde el servidor DNS de un proveedor, aunque las hagan distintos usuarios de él. Por otra parte, para los casos normales, para un mismo dominio no es habitual que los clientes tengan que resolver más de dos o tres direcciones IP (el servidor www, el ftp y quizás otro como chat o un

webmail). Si se observa que se interroga al servidor DNS objeto de la vigilancia por una cantidad apreciable de nombres de máquinas distintos, puede ocurrir que se encuentre ante un usuario que está intentando averiguar si existen máquinas como aquellas por cuyos nombres pregunta en el dominio para averiguar su IP.

Esta variable indica el número de peticiones de resolución de dominios distintos pertenecientes al rango protegido que se han hecho a dicho DNS (que se supone que es el que alberga la resolución inversa) desde la red a la que pertenece la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 5.

CDN	**Normales**	**Altas**	**Excesivas**
No menor que	0	1	9
Igual que	0	8	18
Igual que	3	15	∞
No mayor que	6	20	∞



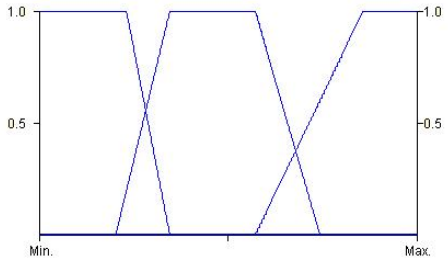
Cuadro 5

4.3.4. Páginas web visitadas

La navegación por las páginas web públicas de un servidor web es probablemente la acción más común y consecuentemente inofensiva que se realiza a través de Internet. Sin embargo, en una fase inicial, cuando un hacker se decide a averiguar información sobre un sistema para intentar abordarle, uno de los sitios donde primero puede conseguir información es la propia web pública, explorando todas sus páginas hasta el mínimo detalle. No es muy común, pero en ocasiones podría obtener detalles sobre la red de la que dispone la empresa o los nombres de los usuarios que pueda tener. Lo que es más probable es que pueda averiguar multitud de detalles sobre la empresa que con un poco de suerte y un administrador descuidado pueden llevarle a la obtención de contraseñas que tengan que ver con nombres mencionados en las páginas.

Esta variable indica el número de páginas web distintas por las que se ha navegado desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 6.

PWV	**Normales**	**Altas**	**Excesivas**
No menor que	0	35	100
Igual que	0	60	150
Igual que	40	100	∞
No mayor que	60	130	∞



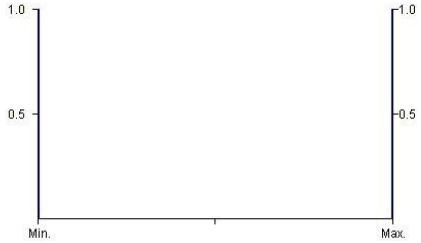
Cuadro 6

4.3.5. ¿Pedido robots.txt?

El fichero robots.txt es un fichero estándar que se encuentra en el directorio raíz de la práctica totalidad de los servidores web y que da información sobre la estructura de directorios para acceso público de la que dispone un servidor web. Esta información normalmente es pedida de forma automática por los robots buscadores cada vez que visitan la página web vigilada para saber que páginas deben de indexar y cuales no, pero un hacker también la podría utilizar para averiguar más información sobre el sistema custodiado. Para diferenciar en este factor lo que pueden ser visitas de auténticos buscadores, sólo se tiene en cuenta cuando ha habido otros factores anteriores que hacen sospechar que quien accede es realmente un usuario malintencionado.

Esta variable indica si se ha hecho una petición del fichero robots.txt al servidor web al que se presta atención desde la dirección IP vigilada. Por tanto, en este caso sólo cabe considerar dos estado posibles de esta variable y su representación gráfica denotará el carácter binario, ya que únicamente puede tener los valores 0 ó 1, tal como se muestra en el Cuadro 7.

PR	SI	NO

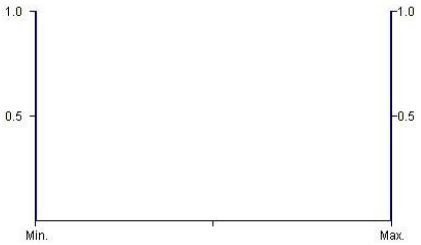


Cuadro 7

4.3.6. ¿Pedidos ficheros FTP sensibles?

En los sistemas basados en unix hay una serie de ficheros de los cuales se puede intentar obtener información altamente valiosa para alguien que quiera penetrar en dicho sistema. Los más sensibles son los que incluyen nombres de usuarios y contraseñas encriptadas, pero también hay otra información sensible aunque lo sea menos como son los servicios que se encuentran abiertos, o su configuración, donde se puede detectar si se está utilizando alguna configuración o versión vulnerable. En ocasiones, un usuario que entre al servidor vigilado a través del servicio FTP, puede intentar conseguir alguno de estos ficheros por si se hubiéran dejado disponibles. Por ello, hay una lista de ficheros, entre los cuales se incluyen los antes mencionados (como son el /etc/passwd, el /etc/shadow, /etc/groups...) que se encuentran en una "lista negra" y que harían levantar serias sospechas sobre los intereses de un usuario que los pide. Esta variable indica si se han pedido mediante FTP alguno de los ficheros que podrían contener información sensible para el acceso al sistema desde la dirección IP vigilada. Por tanto, en este caso sólo cabe considerar dos estado posibles de esta variable y su representación gráfica denotará el carácter binario, ya que únicamente puede tener los valores 0 ó 1, tal como se muestra en el Cuadro 8.

PFS	SI	NO



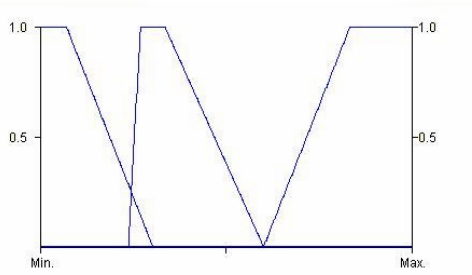
Cuadro 8

4.3.7. Ficheros de FTP listados

El servicio de FTP pone a disposición de los usuarios, muchas veces de forma anónima, un gran número de ficheros para que puedan ser transferidos desde el servidor remoto a los ordenadores locales de cada interesado. El listado y descarga de ficheros mediante FTP es una tarea en principio inofensiva, pero al igual que un usuario malintencionado puede buscar información acerca de una compañía mediante su página web, también puede hacerlo indagando en los ficheros que ofrece al público para su descarga. Es una amenaza no muy grande, pero si se une el hecho de que un usuario este haciendo uso excesivo de este tipo de ficheros a otros factores, se pueden aumentar las sospechas sobre sus malas intenciones.

Esta variable indica la suma del número de peticiones de listados de directorios más el número de peticiones de descarga de ficheros que se han hecho al servidor FTP desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 9.

FFL	**Normales**	**Altos**	**Excesivos**
No menor que	0	70	150
Igual que	0	80	250
Igual que	20	100	∞
No mayor que	90	180	∞



Cuadro 9

4.3.8. Consultas URLs peligrosas

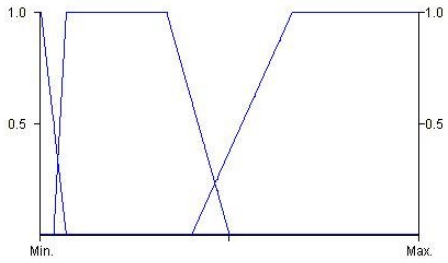
Viene siendo bastante común en los últimos tiempos que se descubran agujeros de seguridad en servidores web o aplicaciones web comunes a través de los cuales se puede conseguir acceso a importante información de servidores. Estos agujeros de seguridad se basan habitualmente en in-

vocar a determinados ejecutables (CGIs u otro tipo de aplicativos) con determinados parámetros.

Uno de los problemas de este tipo más comunes es el que ocurría en versiones antiguas del servidor web IIS a través del que se podía conseguir acceso a la línea de comando pidiendo URLs similares a la "http://destino/scripts/..%../winnt/system32/cmd.exe". Como ejemplo otras URLs que pueden denotar que alguien está intentando aprovecharse de un error de un software para penetrar en el sistema, se dispone la "http://destino:8080/%2e%2e/%2e%2e%5cfichero%00" para servidores de aplicaciones Tomcat o la "http://destino/cgi–bin/webdist.cgi?distloc =;cat%20/etc/passwd" para el script webdist que se distribuye por defecto con algunos servidores web. Para que el sistema funcionase correctamente se debería de mantener una "lista negra" con estas URLs a través de de algún servicio como el CERT.

Esta variable indica el número de URLs de las incluidas en la lista de las que tienen notificados problemas de seguridad han sido consultadas (o intentadas consultar) desde la dirección IP vigilado. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 10.

CUP	**Normales**	**Altas**	**Excesivas**
No menor que	0	1	12
Igual que	0	2	20
Igual que	0	10	∞
No mayor que	2	15	∞



Cuadro 10

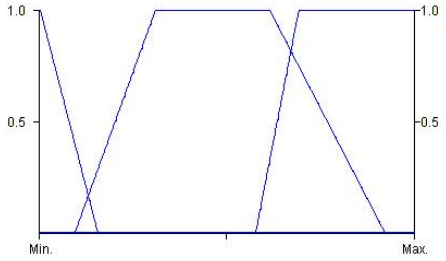
4.3.9. Paquetes ICMP

El ICMP es uno de los protocolos incluidos en la pila IP que se utiliza principalmente para comprobar conectividades entre diferentes equipos de una red. La gran mayoría (o totalidad) de las aplicaciones utilizadas por un

usuario final, no necesitan de este protocolo para nada. Sin embargo, cuando se "cruza" con algún usuario que esté intentando averiguar información sobre sistema custodiado, sí que es bastante habitual que una de las primeras cosas que haga para comprobar si una máquina realmente existe es enviarle un paquete ICMP mediante ping, sabiendo que si obtiene respuesta, es más que posible que detrás de esa dirección haya una máquina a la que poder atacar.

Esta variable indica el número de paquetes ICMP enviados al servidor protegido desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 11.

PI	**Normales**	**Altos**	**Excesivos**
No menor que	0	12	60
Igual que	0	40	100
Igual que	0	80	∞
No mayor que	20	120	∞



Cuadro 11

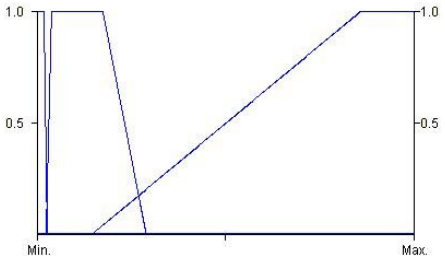
4.3.10. Binds a puertos diferentes

La gran mayoría de las comunicaciones en Internet se realizan a través del protocolo TCP/IP, que es un protocolo orientado a conexión, que establece para cada máquina un total de 65536 puertos diferentes en los cuales pueden albergarse servicios. Este protocolo está orientado a conexión por lo cual, antes de establecer cualquier conexión, lo primero que hace es un intercambio de paquetes sobre el puerto que desee para confirmar que ésta se puede producir. El primero de esos paquetes se llama "BIND" y si obtiene una respuesta a él, aunque no obtenga más información posterior, ya sabe que hay algún servicio funcionando a través de ese puerto. Sabiendo esto, muchos de los usuarios que desean atacar un sistema, antes de probar nada, realizan lo que se denomina un "escaneo de puertos", que consiste en el envío de estos paquetes de "BIND" a diferentes

puertos del servidor para comprobar si hay respuesta y por lo tanto pueden tener algún servicio al que atacar en ese sitio. Los programas más automatizados hacen un auténtico bombardeo de peticiones a todos los puertos de un servidor que son fácilmente detectables, pero otros usuarios más precavidos sólo lo realizan sobre determinados puertos que saben que pueden contener puntos débiles.

Esta variable indica el número puertos diferentes del servidor protegido a los que se han hecho bind TCP o UDP desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 12.

BPD	**Normales**	**Altas**	**Excesivas**
No menor que	0	7	50
Igual que	0	12	300
Igual que	5	60	∞
No mayor que	8	100	∞



Cuadro 12

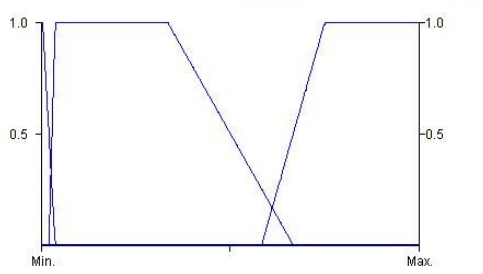
4.3.11. Consultas servicios informativos

De entre los servicios de los que pueden disponer los servidores, hay gran cantidad de ellos que no han sido pensados para su uso habitual por Internet, sino principalmente para compartir información dentro de redes LAN o MAN. No obstante, en muchas ocasiones, ya sea por descuido o por necesidades muy concretas, estos servicios pueden estar disponibles y en ese caso la información que presentan puede ser útil para que un atacante recopile información sobre los usuarios, directorios o las características del servidor protegido. Algunos de estos servicios son: showmount, que muestra que particiones de su disco tiene compartidas mediante NFS un servidor; finger, que muestra información sobre usuarios del sistema; snmpwalk, que es un protocolo para gestionar redes y equipos y puede

mostrargran cantidad de información sobre el hardware y condiciones de funcionamiento de un servidor...

Esta variable indica el número de consultas que se han hecho a los servicios que, como los comentados, pueden prestar información al respecto de usuarios del sistema desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 13.

CSI	**Normales**	**Altas**	**Excesivas**
No menor que	0	1	35
Igual que	0	2	50
Igual que	0	20	∞
No mayor que	2	40	∞



Cuadro 13

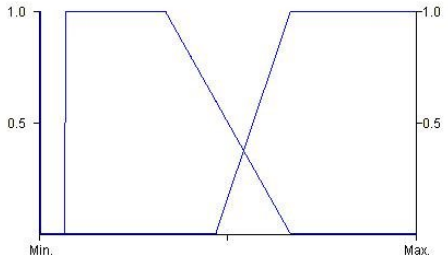
4.3.12. ¿SMTP VRFY?

El protocolo SMTP es el utilizado por los servidores de Internet para enviarse mensajes entre ellos. Entre los comandos incluidos en la especificación SMTP hay uno que si bien es estándar, no suele ser utilizado en el envío por los servidores de correo habituales que es el SMTP VRFY. Sirve para verificar si un usuario de correo existe en el sistema al que se está conectando. Muchos atacantes, pueden utilizar este método para probar de forma bastante desapercibida distintos nombres de usuario para de esta manera hallar usuarios del sistema, con los cuales pueden proceder a posteriores ataques. Como quiera que este comando no se emplea habitualmente, la aparición de este comando puede ser un síntoma interesante de que el usuario vigilado intenta algo.

Esta variable indica el número de veces que se ha utilizado el comando SMTP VRFY accediendo a alguno de los servidores protegidos desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos aso-

ciados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 14.

SV	Normales	Altas	Excesivas
No menor que	0	1	7
Igual que	0	1	10
Igual que	0	5	∞
No mayor que	0	10	∞



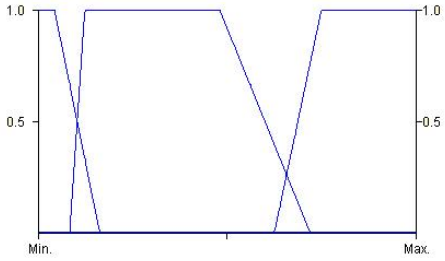
Cuadro 14

4.3.13. Intentos login fácil fallidos

Cuando se hace referencia al login, se cuestiona el intento de un usuario de entrar en el sistema vigilado a través de alguno de los métodos habituales para el sistema operativos, tales como pueden ser un telnet o ssh para unix, o pc–anywhere, VNC o terminal services para un servidor windows. También se puede incluir dentro de esta clasificación los intentos de acceso por FTP con login/password, ya que puede entrañar riesgos comparables a los de los otros métodos de acceso. Con los intentos de login fácil fallidos se trata de contabilizar los fallos que se producen al intentar a un sistema introduciendo como password alguna de las consideradas como fáciles: el mismo nombre de usuario, el nombre de usuario y un número, un simple "enter" o alguna modificación del nombre cambiando el orden de las letras o algunas letras por números.

Esta variable indica el número de intentos de acceso a alguno de los sistemas protegidos que se han hecho intentando introducir alguna de las claves consideradas como fáciles desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 15.

ILF	Normales	Altos	Excesivos
No menor que	0	2	10
Igual que	0	3	20
Igual que	1	12	∞
No mayor que	4	18	∞



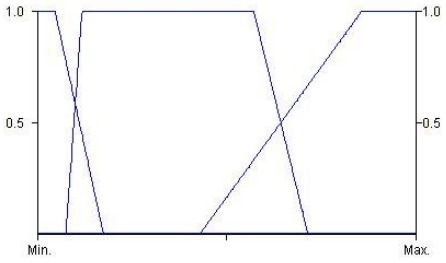
Cuadro 15

4.3.14. Otros intentos login fallidos

Cuando se hace mención de login, cabe referirse al intento de un usuario de entrar en dicho sistema a través de alguno de los métodos habituales para el sistema operativos, tales como pueden ser un telnet o ssh para unix, o pc–anywhere, VNC o terminal services para un servidor windows. También se puede incluir dentro de esta clasificación los intentos de acceso por FTP con login/password, ya que puede entrañar riesgos comparables a los de los otros métodos de acceso. Dentro de "otros intentos de login fallidos" se trata de contabilizar los fallos al entrar al sistema que se han producido introduciendo contraseñas que no son de las consideradas como "fáciles" según el apartado anterior.

Esta variable indica el número de intentos de acceso a alguno de los sistemas protegidos que se han hecho desde la dirección IP vigilada introduciendo claves no incluídas en la lista especificada en el apartado anterior. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 16.

OIL	Normales	Altos	Excesivos
No menor que	0	5	30
Igual que	0	8	60
Igual que	3	40	∞
No mayor que	12	50	∞



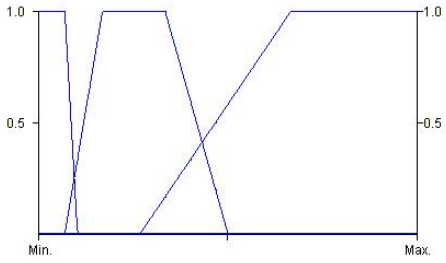
Cuadro 16

4.3.15. Comandos enviados en URL

Según se puede observar estando al tanto de las listas relativas a aspectos de seguridad, buena parte de los agujeros de seguridad que pueden tener aplicaciones web, se refieren a la posibilidad de que, por errores en la programación, se pueda acceder a ficheros o ejecutables localizados fuera del directorio de la aplicación pasando valores inadecuados a algunas de las variables utilizadas. Por ello, un buen método preventivo para detectar ataques de este tipo que se puedan estar dando, es el detectar ciertos comandos o ficheros del sistema especialmente sensibles a los que se puede intentar acceder aprovechándose de errores. Ejemplos de estos ficheros pueden ser "/etc/passwd", "/bin/bash", "system32\cmd.exe" o "root.exe".

Esta variable indica el número de comandos de los considerados como malintencionados enviados al sistema que se enviado en peticiones de URLs al sistema protegido desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 17.

CEU	Normales	Altos	Excesivos
No menor que	0	2	8
Igual que	0	5	20
Igual que	2	10	∞
No mayor que	3	15	∞



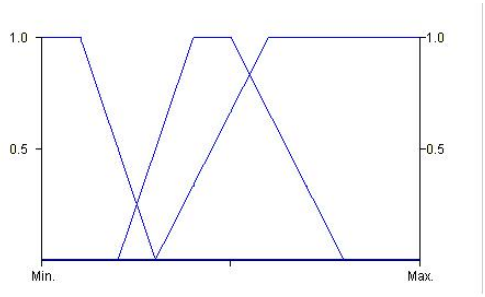
Cuadro 17

4.3.16. Cabeceras de protocolo inconsistentes

Cada protocolo tiene sus propias reglas para construir paquetes e indicar por medio de diferentes campos, el tipo de transmisión, longitud, número de orden, así como gran multitud de datos dependientes de ellos. Una fuente de multitud de ataques (principalmente de denegación de servicio) es precisamente el crear paquetes con cabeceras que se salgan de los estándares, y que provoquen el funcionamiento del sistema de una forma inesperada al no saber como tratarlos. El sensor encargado de controlar las cabeceras de protocolo, debería de tener información sobre todos los protocolos IP, así como los valores válidos para sus cabeceras, y debería de chequear todos los paquetes que pasan a su través para detectar aquellos que no cumplan los estándares. Un número bajo de éstos puede ser debido a problemas en la comunicación a través de la red, pero si se incrementa puede suscitar la sospecha de un ataque.

Esta variable indica el número de cabeceras de protocolo no estándar que se detectan en la red protegida con origen la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altas y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 18.

CPI	**Normales**	**Altas**	**Excesivas**
No menor que	0	2	3
Igual que	0	4	6
Igual que	1	5	∞
No mayor que	3	8	∞



Cuadro 18

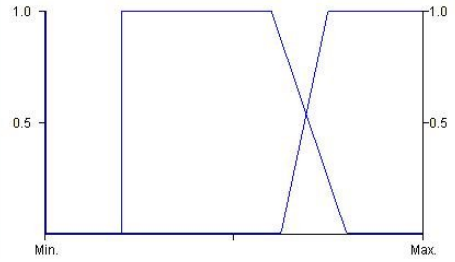
4.3.17. Logs modificados

Todos los sistemas servidores disponen de ficheros de trazas en los que se almacenan multitud de datos de uso y acceso al sistema. Estos son los casos de ficheros como el access_log de Apache, los /var/adm/messages, /var/log/syslog, /var/adm/lastlog o xferlog de unix

o los del visor de eventos de Windows. En caso de que un usuario no autorizado esté entrando en un sistema es posible que sea posible su detección a través de un examen de éstos. Para prevenir esta detección, los intrusos pueden intentar en ocasiones borrar sus huellas modificando los ficheros, cosa que no es en absoluto normal, ya que estos ficheros únicamente varían para añadir nueva información y no para modificar ninguna antigua. En caso de que se detecte algún cambio en estos ficheros hay serias posibilidades de que se esté enfrentando a un intruso.

Esta variable indica el número de ficheros de trazas (logs) que han sufrido modificaciones en su contenido por parte de un usuario que haya entrado desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 19.

LM	**Normales**	**Altos**	**Excesivos**
No menor que	0	1	2
Igual que	0	1	4
Igual que	0	3	∞
No mayor que	0	4	∞



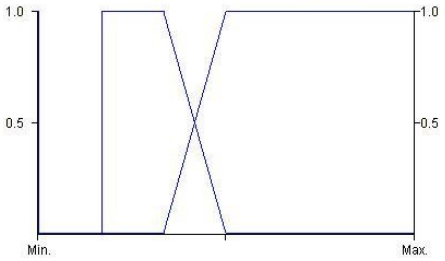
Cuadro 19

4.3.18. Cuentas creadas

Cuando un usuario consigue entrar en un sistema a través de un error o los datos de otro usuario, suele intentar hacerse con otro medio de acceso al sistema para poder seguir entrando en él en el caso de que se corrigiese el error o bien el usuario cambiase de contraseña. El medio más sencillo para esto es la creación de una nueva cuenta de usuario. Normalmente el intruso suele hacerlo de forma que el usuario que se cree pueda parecerse a algún usuario anterior o parezca un administrador o usuario importante del sistema. En cualquiera de los casos, la creación de esta cuenta es fácilmente detectable por un sistema que lleve el control de usuarios y puede dar una alerta a tener en cuenta.

Esta variable indica el número de cuentas que han sido creadas por un usuario que haya entrado desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Alguna y Excesivas. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 20

CC	**Normales**	**Alguna**	**Excesivas**
No menor que	0	1	2
Igual que	0	1	3
Igual que	0	2	∞
No mayor que	0	3	∞



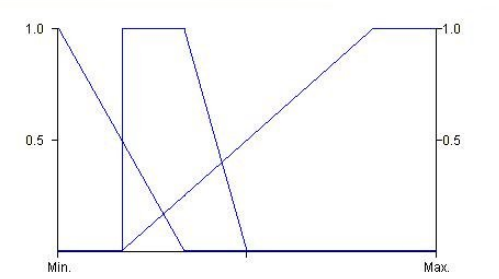
Cuadro 20

4.3.19. Servicios de red ejecutados

Cuando un usuario consigue entrar en un sistema a través de un error o los datos de otro usuario, suele intentar hacerse con otro medio de acceso al sistema para poder seguir entrando en él en el caso de que se corrigiese el error o bien el usuario cambiase de contraseña. Aparte de la creación de un usuario otra forma de hacerlo es instalando un programa que sirva de "puerta trasera" para entrar al sistema de forma discreta sin pasar por los medios habituales de acceso que pueden estar más controlados. Estos programas reciben el nombre de troyanos y se caracterizan por quedarse residentes en el sistema como un servicio de red escuchando en alguno de los puertos TCP, esperando una conexión. En este apartado se tratará de vigilar los procesos no estándar (excluyendo servicios como la web, ftp, telnet, etc.) que están corriendo en el sistema ligados a alguno de los puertos de red y se controlará que el usuario que los ha ejecutado para detectar la posibilidad de que el servicio de red ejecutado y por el que se transmite información pueda ser un troyano.

Esta variable indica el número de programas asociados a un puerto de la red que se han ejecutado por el usuario que haya entrado desde la dirección IP vigilada y se encuentran activos. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 21.

SRE	Normales	Altos	Excesivos
No menor que	0	1	1
Igual que	0	1	5
Igual que	0	2	∞
No mayor que	2	3	∞



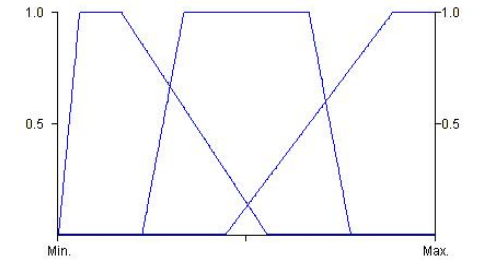
Cuadro 21

4.3.20. Copia ficheros

Una vez que un usuario malintencionado ha accedido al sistema objeto de atención, una de las cosas que puede intentar hacer es extraer de él información que puede ser valiosa. Para detectar estas acciones, este apartado trata del control del número de ficheros que se copian desde el servidor al exterior por alguno de los protocolos disponibles. Esta información se ha de tomar de forma cautelosa si no está acompañada de otros síntomas de ataque, ya que la mera copia de un número alto de ficheros puede deberse a razones ajenas que no tengan que ver con ningún intruso.

Esta variable indica el número de ficheros copiados del servidor a la dirección IP vigilada as través de alguno de los protocolos previstos. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 22.

CF	Normales	Altos	Excesivos
No menor que	0	20	40
Igual que	5	30	80
Igual que	15	60	∞
No mayor que	50	70	∞



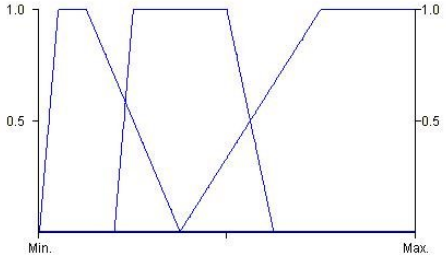
Cuadro 22

4.3.21. Borrado ficheros

Otra de las opciones que puede tomar un usuario malintencionado que entra en sistema custodiado es el sabotear o destruir la información que esté contenida en los sistemas de información allí activos. Para detectar estas acciones, se puede controlar el número de ficheros que el usuario del que se sospecha borra. Dado que aunque también pueda ser realizada por usuarios normales, el borrado es una acción más peligrosa, por lo que será preciso tenerla más en cuenta que la copia de ficheros.

Esta variable indica el número de ficheros borrados del servidor por el usuario que entra desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normales, Altos y Excesivos. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 23.

BF	**Normales**	**Altos**	**Excesivos**
No menor que	0	8	15
Igual que	2	10	30
Igual que	5	20	∞
No mayor que	15	25	∞



Cuadro 23

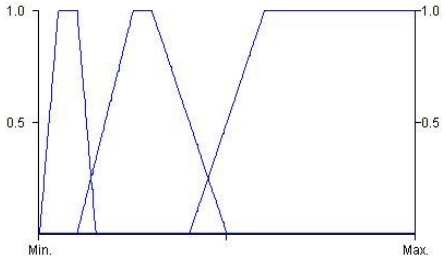
4.3.22. *Utilización de CPU*

Otro de las formas en las que un intruso puede aprovecharse del sistema es utilizando sus recursos para sus propios propósitos: ya sea para procesar información o para saltar a otros sistemas desde él. Una de las formas más sencillas de detectar que alguien está haciendo un abuso de los recursos del sistema protegido es ver el consumo de CPU que están haciendo los procesos que este ha ejecutado. Esto es lo que se encarga de medir esta variable, que de paso sirve para detectar el empleo excesivo de CPU como situación anómala.

Esta variable indica el porcentaje de utilización de CPU consumido por los procesos ejecutados por el usuario que entra desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta varia-

ble: Normal, Mucho y Demasiado. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 24.

UC	**Normal**	**Mucho**	**Demasiado**
No menor que	0	10	40
Igual que	5	25	60
Igual que	10	30	100
No mayor que	15	50	100



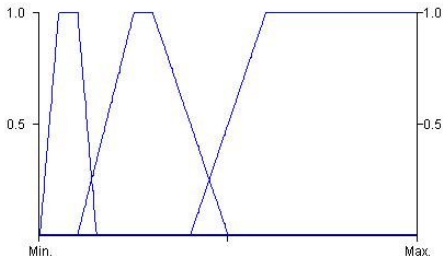
Cuadro 24

4.3.23. Utilización de Memoria

Otro de las formas en las que un intruso puede aprovecharse del sistema es utilizando sus recursos para sus propios propósitos: ya sea para procesar información o para saltar a otros sistemas desde él. Aparte de la CPU, el otro recurso que suele ser más consumido en un computador, es la memoria. Contabilizando el espacio ocupado en memoria por los procesos de un usuario, se puede detectar si esta haciendo algún abuso del sistema y detectar además situaciones anómalas.

Esta variable indica el porcentaje de memoria consumida por los procesos ejecutados por el usuario que entra desde la dirección IP vigilada. En este sentido, cabe considerar tres posibles estados de esta variable: Normal, Mucho y Demasiado. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 25.

UM	**Normal**	**Mucho**	**Demasiado**
No menor que	0	10	40
Igual que	5	25	60
Igual que	10	30	100
No mayor que	15	50	100



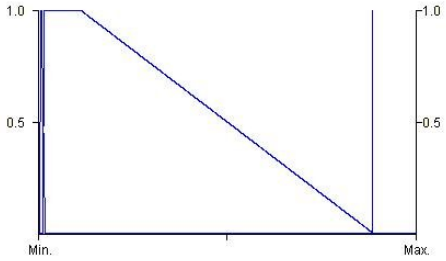
Cuadro 25

4.3.24. Estado del servidor

Algunos de los ataques sobre servidores, lo único que persiguen es provocar que el rendimiento del sistema baje o simplemente deje de funcionar adecuadamente. Estos son los ataques denominados DoS (Denial of Servide o de denegación de servicio). Para detectar si el estado del servidor es el correcto o no se encuentra prestando servicio de red adecuadamente, se pretende medir este factor a través de un "sensor" que haga un ping al sistema vigilado a través de la intranet o red interna y mida el retraso con el que llega, para de esta forma poder conocer su estado de funcionamiento. El ping tiene un timeout de 4000 milisegundos, por lo cual, todo ping que tarde en volver más de este tiempo será descartado y se dará al sistema como totalmente caído.

Esta variable indica el tiempo de respuesta del servidor protegido a un ping desde uno de los sensores situados en su propia red. En este sentido, cabe considerar tres posibles estados de esta variable: Bueno, Malo y Caído. Los números borrosos asociados a estas tres etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 26.

ES	**Bueno**	**Malo**	**Caído**
No menor que	0	30	4000
Igual que	1	50	4000
Igual que	15	500	4000
No mayor que	60	4000	4000



Cuadro 26

4.4. Establecimiento del sistema de inferencia (expertizaje): las variables intermedias

Una vez especificadas las variables de entrada, esto es, las definiciones de cada una de las etiquetas lingüísticas en las que se dividen tales variables y dando por conocidas las relaciones entre las mismas, el siguiente paso en la construcción del sistema de detección de intrusos consiste en el establecimiento de las bases de reglas borrosas del tipo:

SI…ENTONCES que denotarán el grado de influencia de cada variable y permitirán alcanzar la solución final del problema.

En primer término, estas reglas permitirán definir "las variables intermedias" a partir de la combinación de variables de entrada, y a su vez, definir la variable de salida del sistema experto, a partir de la combinación de las intermedias.

A este respecto, las variables intermedias que serán utilizadas en el sistema experto borroso de detección de intrusos son las siguientes:

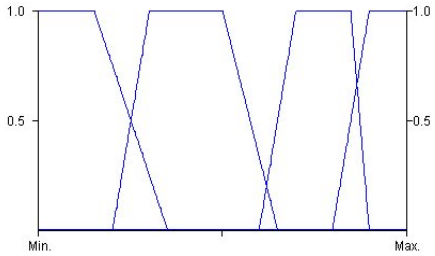
- Consultas DNS
- Consultas web
- Consultas FTP
- Logins fallidos
- Utilización recursos
- Posibilidad investigación interna
- Posibilidad investigación externa
- Posibilidad intrusión
- Posibilidad borrado huellas
- Posibilidad aprovechamiento
- Posibilidad investigación
- Posibilidad ataque

A continuación se describirán estas variables intermedias con sus correspondientes reglas.

4.4.1. Consultas DNS

Esta variable indica la posibilidad de que alguien esté intentando recopilar información sobre el sistema a través de consultas aparentemente inofensivas a los servidores DNS que alojan información sobre el sistema protegido. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 27.

Posibilidad	**Baja**	**Media**	**Alta**	**Excesiva**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 27

Base de Reglas: El valor de "Consultas DNS" se obtiene relacionando las variables que indican los "Intentos de resolución inversa", "Intentos de ls del dominio" y el número de "Consultas DNS normales" tal como se muestra en el Cuadro 28.

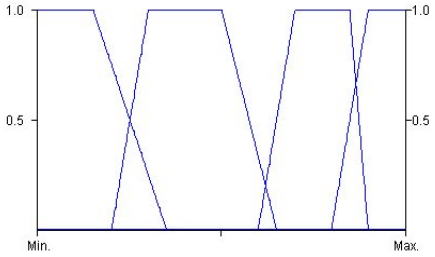
ENTRADAS			**SALIDA**
Intentos de resolución inversa	**Intentos de "ls" del dominio**	**Consultas DNS normales**	***Consultas DNS***
Normales	Normales	Normales	Baja
Normales	Normales	Altas	Baja
Normales	Normales	Excesivas	Media
…….	…….	…….	…….
Normales	Excesivos	Altas	Alta
Normales	Excesivos	Excesivas	Excesiva
Altos	Normales	Normales	Baja
Altos	Normales	Altas	Media
…….	…….	…….	…….
Altos	Excesivos	Excesivas	Excesiva
Excesivos	Normales	Normales	Media
…….	…….	…….	…….
Excesivos	Excesivos	Normales	Alta
Excesivos	Excesivos	Altas	Excesiva
Excesivos	Excesivos	Excesivas	Excesiva

Cuadro 28

4.4.2. Consultas Web

Esta variable indica la posibilidad de que alguien esté intentando recopilar información sobre el sistema a través de consultas aparentemente inofensivas a los servidores web de dicho sistema. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 29.

Posibilidad–4	**Baja**	**Media**	**Alta**	**Excesiva**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 29

Base de Reglas: El valor de "Consultas Web" se obtiene relacionando las variables que indican la cantidad de "páginas web visitadas" y si se ha "¿Pedido robots.txt?" tal como se muestra en el Cuadro 30.

ENTRADAS		**SALIDA**
Páginas web visitadas	**¿Pedido robots.txt?**	***Páginas web***
Normales	Sí	Baja
Normales	No	Baja
Altas	Sí	Media
Altas	No	Media
Excesivas	Sí	Media
Excesivas	No	Alta

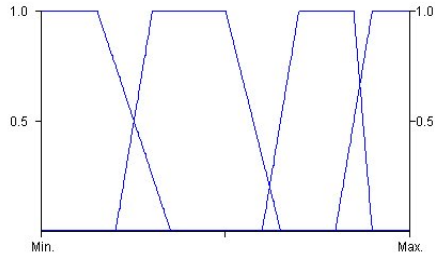
Cuadro 30

4.4.3. Consultas FTP

Esta variable indica la posibilidad de que alguien esté intentando recopilar información sobre el sistema a través de consultas aparentemente inofensivas a los servidores FTP del mismo. En este sentido, cabe conside-

rar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 31.

Posibilidad–4	**Baja**	**Media**	**Alta**	**Excesiva**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 31

Base de Reglas: El valor de "Consultas FTP" se obtiene relacionando las variables que indican la cantidad de "ficheros FTP listados" y si han sido "pedidos ficheros de FTP sensibles" tal como se muestra en el Cuadro 32.

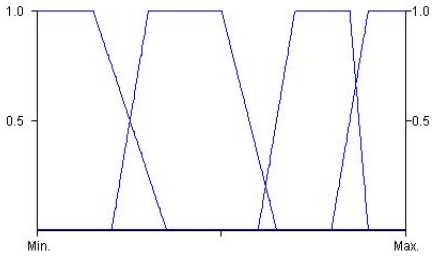
ENTRADA		**SALIDA**
Ficheros de FTP listados	**¿Pedidos ficheros FTP sensibles?**	***Consultas FTP***
Normales	Sí	Media
Normales	No	Baja
Altas	Sí	Alta
Altas	No	Baja
Excesivas	Sí	Alta
Excesivas	No	Media

Cuadro 32

4.4.4. Logins fallidos

Esta variable indica la posibilidad de que alguien esté intentando emplear el método prueba–error para entrar en el sistema vigilado con algún par usuario/password que confía en que puedan ser los adecuados. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 33.

Posibilidad	**Baja**	**Media**	**Alta**	**Excesiva**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 33

Base de Reglas: El valor de "Logins fallidos" se obtiene relacionando las variables que indican la cantidad de "intentos de login fácil fallidos" y la de "otros intentos de login fallidos" tal como se muestra en el Cuadro 34.

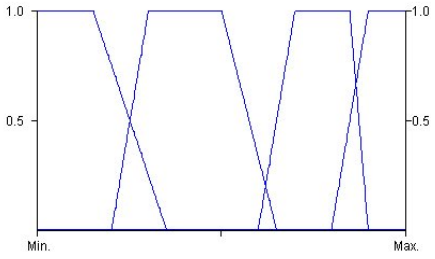
ENTRADAS		**SALIDA**
Intentos *login* fácil fallidos	**Otros intentos *login* fallidos**	***Logins fallidos***
Normales	Normales	Baja
Normales	Altos	Baja
Normales	Excesivos	Media
Altos	Normales	Media
Altos	Altos	Alta
Altos	Excesivos	Alta
Excesivos	Normales	Excesiva
Excesivos	Altos	Excesiva
Excesivos	Excesivos	Excesiva

Cuadro 34

4.4.5. Utilización de recursos

Esta variable indica el consumo de los recursos del sistema que está haciendo el usuario al que estamos vigilando. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 35.

Posibilidad	**Baja**	**Media**	**Alta**	**Excesiva**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 35

Base de Reglas: El valor de "Utilización de recursos" se obtiene relacionando las variables que indican la cantidad de "uso de CPU" y la de "uso de memoria" tal como se muestra en el Cuadro 36.

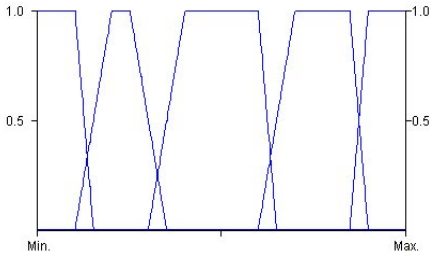
ENTRADAS		**SALIDA**
Utilización de CPU	**Utilización de memoria**	***Utilizació de recursos***
Normal	Normal	Normal
Normal	Mucho	Normal
Normal	Demasiado	Alto
Mucho	Normal	Normal
Mucho	Mucho	Alto
Mucho	Demasiado	Alto
Demasiado	Normal	Alto
Demasiado	Mucho	Excesivo
Demasiado	Demasiado	Excesivo

Cuadro 36

4.4.6. Posibilidad investigación externa

Esta variable indica la posibilidad de que alguien esté intentando conseguir alguna información sobre el sistema de información de la organización objeto de estudio desde fuera a través de métodos aparentemente inocuos, aprovechando la información que se dispone pública y abiertamente a comunidad de Internet. En este sentido, cabe considerar cinco posibles estados de esta variable: Muy Baja (MB), Baja (B), Media (M), Alta (A) y Excesiva (E). Los números borrosos asociados a estas cinco etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 37.

Posibilidad	**M Baja**	**Baja**	**Media**	**Alta**	**Exces.**
No menor que	0	10	30	60	85
Igual que	0	20	40	70	90
Igual que	10	25	60	85	100
No mayor que	15	35	65	90	100



Cuadro 37

Base de Reglas: El valor de "Posibilidad investigación externa" se obtiene relacionando las variables que indican las posibilidad de que alguien esté intentando obtener información del sistema protegido a través de "consultas DNS", "consultas web" y "consultas FTP" tal como se muestra en el Cuadro 38.

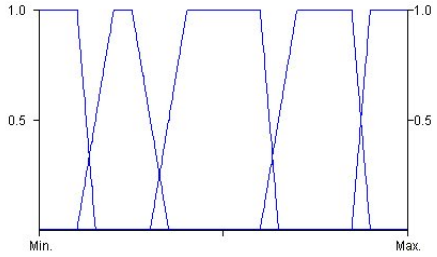
ENTRADAS			**SALIDA**
Consultas DNS	**Consultas web**	**Consultas FTP**	***Posibilidad investigación externa***
Baja	Baja	Baja	Muy baja
Baja	Baja	Media	Muy baja
Baja	Baja	Alta	Media
.....	.....	.....	.....
Baja	Alta	Media	Media
Baja	Alta	Alta	Alta
Media	Baja	Baja	Muy baja
Media	Baja	Media	Baja
.....	.....	.....	.....
Media	Alta	Media	Alta
Media	Alta	Alta	Alta
Alta	Baja	Baja	Baja
Alta	Baja	Media	Media
.....	.....	.....	.....
Excesiva	Alta	Baja	Alta
Excesiva	Alta	Media	Excesiva
Excesiva	Alta	Alta	Excesiva

Cuadro 38

4.4.7. Posibilidad investigación interna

Esta variable indica la posibilidad de que alguien esté intentando conseguir alguna información sobre el sistema de información haciendo intentos de conexión más intrusivos, que no son propios de usuarios en condiciones normales de uso. En este sentido, cabe considerar cinco posibles estados de esta variable: Muy Baja, Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cinco etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 39.

Posibilidad	M Baja	Baja	Media	Alta	Exces.
No menor que	0	10	30	60	85
Igual que	0	20	40	70	90
Igual que	10	25	60	85	100
No mayor que	15	35	65	90	100



Cuadro 39

Base de Reglas: El valor de "Posibilidad investigación interna" se obtiene relacionando las variables que indican las posibilidades de que alguien esté intentando obtener información de dicho sistema a través de "consultas a URLs peligrosas", "paquetes ICMP", "binds a puertos distintos", "consultas a servicios informativos" y el uso comando "SMTP VRFY" tal como se muestra en el Cuadro 40 de la página siguiente.

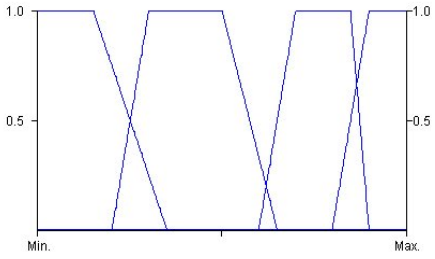
4.4.8. Posibilidad de intrusión

Esta variable indica la posibilidad de que alguien esté haciendo intentos (sean o no acertados) para entrar en el sistema vigilado de una forma no lícita. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 41 que se muestra en la página siguiente.

ENTRADAS					**SALIDA**
Consultas a URLs peligrosas	**Paquetes ICMP**	**Binds a puertos distintos**	**Consultas servicios informativos**	**SMTP VRFY**	***Posibilidad investigación interna***
Normales	Normales	Normales	Normales	Sí	Baja
Normales	Normales	Normales	Normales	No	Muy baja
Normales	Normales	Normales	Altas	Sí	Media
.....	.....	.....	.....	.....	.....
Normales	Normales	Excesivas	Altas	Sí	Excesiva
Normales	Normales	Excesivas	Altas	No	Alta
Normales	Normales	Excesivas	Excesivas	Sí	Excesiva
Normales	Normales	Excesivas	Excesivas	No	Alta
.....	.....	.....	.....	.....	.....
Normales	Altos	Excesivas	Altas	Sí	Excesiva
Normales	Altos	Excesivas	Altas	No	Alta
Normales	Altos	Excesivas	Excesivas	Sí	Excesiva
Normales	Altos	Excesivas	Excesivas	No	Alta
.....	.....	.....	.....	.....	.....
Normales	Excesivos	Excesivas	Excesivas	Sí	Excesiva
Normales	Excesivos	Excesivas	Excesivas	No	Excesiva
Altas	Normales	Normales	Normales	Sí	Media
Altas	Normales	Normales	Normales	No	Baja
.....	.....	.....	.....	.....	.....
Excesivas	Normales	Normales	Normales	Sí	Media
Excesivas	Normales	Normales	Normales	No	Media
Excesivas	Normales	Normales	Altas	Sí	Alta
Excesivas	Normales	Normales	Altas	No	Media
.....	.....	.....	.....	.....	.....
Excesivas	Altos	Excesivas	Altas	Sí	Excesiva
Excesivas	Altos	Excesivas	Altas	No	Alta
Excesivas	Altos	Excesivas	Excesivas	Sí	Excesiva
Excesivas	Altos	Excesivas	Excesivas	No	Excesiva

Cuadro 40

Posibilidad	**Baja**	**Media**	**Alta**	**Exces.**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 41

Base de Reglas: El valor de "Posibilidad de intrusión" se obtiene relacionando las variables que indican la posibilidad de que un usuario esté intentando entrar en el sistema por el método prueba–error ("logins fallidos"), el número de "comandos enviados en URLs" detectados y el número de "paquetes con cabeceras de protocolo inconsistentes" detectados tal como se muestra en el Cuadro 42.

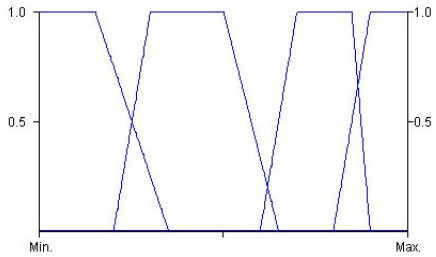
ENTRADAS			**SALIDA**
Logins fallidos	**Comandos enviados en URLs**	**Paquetes con cabeceras inconsistentes**	***Posibilidad de intrusión***
Baja	Normales	Normales	Baja
Baja	Normales	Altas	Baja
Baja	Normales	Excesivas	Baja
Baja	Altos	Normales	Media
.....	.....	.....	.....
Media	Excesivos	Altas	Alta
Media	Excesivos	Excesivas	Excesiva
Alta	Normales	Normales	Media
Alta	Normales	Altas	Media
.....	.....	.....	.....
Excesiva	Altos	Altas	Excesiva
Excesiva	Altos	Excesivas	Excesiva
Excesiva	Excesivos	Normales	Excesiva
Excesiva	Excesivos	Altas	Excesiva
Excesiva	Excesivos	Excesivas	Excesiva

Cuadro 42

4.4.9. Posibilidad de borrado de huellas

Esta variable indica la posibilidad de que alguien esté intentando borrar del sistema que estamos vigilando, huellas que puedan delatar su paso o intrusión o bien esté preparando el sistema para no dejar huellas en sus futuras incursiones. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 43.

Posibilidad–4	**Baja**	**Media**	**Alta**	**Exces.**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 43

Base de Reglas: El valor de "Posibilidad de borrado de huellas" se obtiene relacionando las variables que indican, para el usuario vigilado, el número de ficheros de "logs modificados" por él, las "cuentas creadas", así como los "servicios de red ejecutados" por él; tal como se muestra en el Cuadro 44 de la página siguiente.

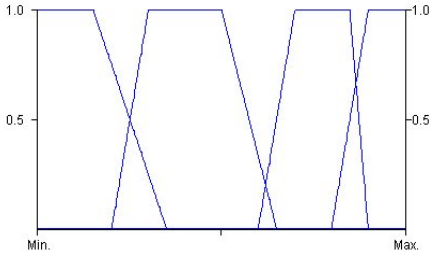
4.4.10. *Posibilidad de aprovechamiento*

Esta variable indica la posibilidad de que el usuario al que vigilamos esté haciendo un aprovechamiento del sistema para fines ilícitos o perniciosos. En este sentido, cabe considerar cuatro posibles estados de esta variable: Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cuatro etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 45, que se muestra en la página siguiente.

ENTRADAS			SALIDA
Logs modifi-cados	**Cuentas creadas**	**Servicios de red ejecutados**	***Posibilidad borrado de huellas***
Normales	Normales	Normales	Baja
Normales	Normales	Altos	Baja
Normales	Normales	Excesivos	Media
…….	…….	…….	…….
Normales	Excesivas	Altos	Alta
Normales	Excesivas	Excesivos	Alta
Altos	Normales	Normales	Baja
Altos	Normales	Altos	Media
…….	…….	…….	…….
Altos	Excesivas	Altos	Alta
Altos	Excesivas	Excesivos	Excesiva
Excesivos	Normales	Normales	Media
Excesivos	Normales	Altos	Alta
…….	…….	…….	…….
Excesivos	Excesivas	Normales	Excesiva
Excesivos	Excesivas	Altos	Excesiva
Excesivos	Excesivas	Excesivos	Excesiva

Cuadro 44

Posibilidad–4	**Baja**	**Media**	**Alta**	**Exces.**
No menor que	0	20	60	80
Igual que	0	30	70	90
Igual que	15	50	85	100
No mayor que	35	65	90	100



Cuadro 45

Base de Reglas: El valor de "Posibilidad de aprovechamiento" se obtiene relacionando las variables que indican, para el usuario vigilado, el número de acciones que ha realizado de "copia de ficheros", de "borrado de ficheros", el "uso de recursos" del sistema que hace y el "estado del servidor" durante su conexión; de la siguiente forma:

El Cuadro 46 muestra la situación de que el Estado del Servidor = Bueno y el Cuadro 47 muestra la situación de que el Estado del Servidor = Malo.

ENTRADAS			**SALIDA**
Copia de ficheros	**Borrado de ficheros**	**Utilización de recursos**	***Posibilidad de aprovechamiento***
Normales	Normales	Normal	Baja
Normales	Normales	Alto	Baja
Normales	Normales	Excesivo	Media
.....	.....	.....	.....
Normales	Excesivos	Excesivo	Alta
Altos	Normales	Normal	Baja
.....	.....	.....	.....
Altos	Excesivos	Excesivo	Excesiva
Excesivos	Normales	Normal	Media
.....	.....	.....	.....
Excesivos	Excesivos	Normal	Excesiva
Excesivos	Excesivos	Alto	Excesiva
Excesivos	Excesivos	Excesivo	Excesiva

Cuadro 46

ENTRADAS			**SALIDA**
Copia de ficheros	**Borrado de ficheros**	**Utilización de recursos**	***Posibilidad de aprovechamiento***
Normales	Normales	Normal	Baja
Normales	Normales	Alto	Baja
Normales	Normales	Excesivo	Alta
.....	.....	.....	.....
Normales	Excesivos	Excesivo	Excesiva
Altos	Normales	Normal	Baja
.....	.....	.....	.....
Altos	Excesivos	Excesivo	Excesiva
Excesivos	Normales	Normal	Media
.....	.....	.....	.....
Excesivos	Excesivos	Normal	Excesiva
Excesivos	Excesivos	Alto	Excesiva
Excesivos	Excesivos	Excesivo	Excesiva

Cuadro 47

El Cuadro 48 muestra la situación de que el Estado del Servidor = Caído

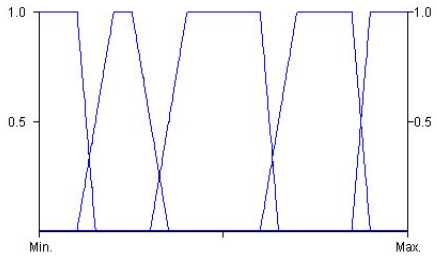
ENTRADAS			**SALIDA**
Copia de ficheros	**Borrado de ficheros**	**Utilización de recursos**	***Posibilidad de aprovechamiento***
Normales	Normales	Normal	Excesiva
Normales	Normales	Alto	Excesiva
Normales	Normales	Excesivo	Excesiva
.....	.....	.....	.....
Normales	Excesivos	Excesivo	Excesiva
Altos	Normales	Normal	Excesiva
.....	.....	.....	.....
Altos	Excesivos	Excesivo	Excesiva
Excesivos	Normales	Normal	Excesiva
.....	.....	.....	.....
Excesivos	Excesivos	Normal	Excesiva
Excesivos	Excesivos	Alto	Excesiva
Excesivos	Excesivos	Excesivo	Excesiva

Cuadro 48

4.4.11. Posibilidad de investigación

Esta variable indica la posibilidad de que el usuario vigilado esté investigando sobre el sistema con la intención de averiguar información que le permita entrar posteriormente en él. En este sentido, cabe considerar cinco posibles estados de esta variable: Muy Baja, Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cinco etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 49.

Posibilidad	**M Baja**	**Baja**	**Media**	**Alta**	**Exces.**
No menor que	0	10	30	60	85
Igual que	0	20	40	70	90
Igual que	10	25	60	85	100
No mayor que	15	35	65	90	100



Cuadro 49

Base de Reglas: El valor de "Posibilidad de investigación" se obtiene relacionando las variables que indican la "posibilidad de investigación interna" y la "posibilidad de investigación externa", tal como se muestra en el Cuadro 50.

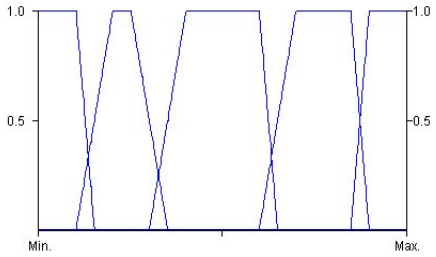
ENTRADA		**SALIDA**
Posibilidad Investigación Externa	**Posibilidad Investigación Interna**	***Posibilidad de investigación***
Muy baja	Muy baja	Muy baja
Muy baja	Baja	Baja
Muy baja	Media	Media
Muy baja	Alta	Alta
.....	**.....**	**.....**
Baja	Excesiva	Excesiva
Media	Muy baja	Baja
.....	**.....**	**.....**
Media	Excesiva	Excesiva
Alta	Muy baja	Media
.....	**.....**	**.....**
Muy alta	Media	Alta
Muy alta	Alta	Excesiva
Muy alta	Excesiva	Excesiva

Cuadro 50

4.4.12. Posibilidad de ataque

Esta variable indica la posibilidad de que el usuario vigilado esté en el interior del sistema realizando alguna acción ilícita o no deseada. En este sentido, cabe considerar cinco posibles estados de esta variable: Muy Baja, Baja, Media, Alta y Excesiva. Los números borrosos asociados a estas cinco etiquetas lingüísticas y la representación gráfica de los mismos se encuentran recogidos en el Cuadro 51.

Posibilidad–5	M Baja	Baja	Media	Alta	Exces.
No menor que	0	10	30	60	85
Igual que	0	20	40	70	90
Igual que	10	25	60	85	100
No mayor que	15	35	65	90	100



Cuadro 51

Base de Reglas: El valor de "Posibilidad de ataque" se obtiene relacionando las variables que indican la "posibilidad de intrusión", la "posibilidad de que esté realzando un borrado de sus huellas" y la "posibilidad de que esté haciendo un aprovechamiento ilícito de su acceso al sistema", tal como se muestra en el Cuadro 52 de la página siguiente.

4.5. Clarificación: la variable de salida

Las variables de salida son las que indican la valoración que hace el sistema de una determinada cuestión en función de todos los factores de entrada que conocemos. En el caso de estudio, se dispone de una única variable de salida que indica la seguridad de que el usuario que está siendo vigilado tenga "malas intenciones" al respecto del sistema o red. Esta variable, ha sido definida con una pregunta: ¿Usuario malintencionado?, la cual indica la posibilidad de que el usuario vigilado esté tenga unas malas intenciones al respecto del sistema o red protegidos. En este sentido, cabe considerar cinco posibles estados de esta variable: Imposible, Improbable, Posible, Probable y Seguro. Los números borrosos asociados a estas cinco etiquetas lingüísticas se encuentran recogidos en el Cuadro 53 y la representación gráfica de los mismos en la Figura 17.

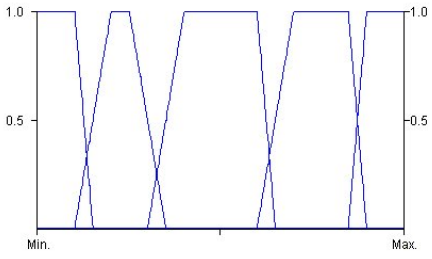


Figura 17

ENTRADAS			**SALIDA**
Posibilidad de intrusión	**Posibilidad de borrado de huellas**	**Posibilidad de aprovechamiento**	***Posibilidad de ataque***
Baja	Baja	Baja	Muy baja
Baja	Baja	Media	Muy baja
Baja	Baja	Alta	Media
Baja	Baja	Excesiva	Alta
.....	.....	.....	.....
Baja	Excesiva	Alta	Excesiva
Baja	Excesiva	Excesiva	Excesiva
Media	Baja	Baja	Baja
Media	Baja	Media	Baja
.....	.....	.....	.....
Media	Excesiva	Alta	Excesiva
Media	Excesiva	Excesiva	Excesiva
Alta	Baja	Baja	Media
Alta	Baja	Media	Media
.....	.....	.....	.....
Alta	Excesiva	Alta	Excesiva
Alta	Excesiva	Excesiva	Excesiva
Excesiva	Baja	Baja	Alta
Excesiva	Baja	Media	Alta
.....	.....	.....	.....
Excesiva	Excesiva	Media	Excesiva
Excesiva	Excesiva	Alta	Excesiva
Excesiva	Excesiva	Excesiva	Excesiva

Cuadro 52

Malintencionado	**Imposible**	**Improbable**	**Posible**	**Probable**	**Seguro**
No menor que	0	10	30	60	85
Igual que	0	20	40	70	90
Igual que	10	25	60	85	100
No mayor que	15	35	65	90	100

Cuadro 53

El valor de "¿Usuario malintencionado?" se obtiene relacionando las variables que indican la "posibilidad de que el usuario esté haciendo una investigación sobre el sistema" y la "posibilidad de que tal usuario esté realizando una ataque", tal como se muestra en el Cuadro 54.

ENTRADAS		**SALIDA**
Posibilidad de Investigación	**Posibilidad de Ataque**	***¿Usuario mal intencionado?***
Muy baja	Muy baja	Imposible
Muy baja	Baja	Improbable
Muy baja	Media	Posible
Muy baja	Alta	Probable
Muy baja	Excesiva	Probable
Baja	Muy baja	Improbable
Baja	Baja	Improbable
Baja	Media	Posible
Baja	Alta	Probable
Baja	Excesiva	Seguro
Media	Muy baja	Improbable
Media	Baja	Posible
Media	Media	Posible
Media	Alta	Probable
Media	Excesiva	Seguro
Alta	Muy baja	Improbable
Alta	Baja	Posible
Alta	Media	Probable
Alta	Alta	Seguro
Alta	Excesiva	Seguro
Excesiva	Muy baja	Posible
Excesiva	Baja	Posible
Excesiva	Media	Probable
Excesiva	Alta	Seguro
Excesiva	Excesiva	Seguro

Cuadro 54

Finalmente, y una vez que se ha obtenido el grado de pertenencia borroso de cada ejemplo a cada subconjunto borroso de la variable final, el último paso en la construcción del sistema experto borrosos de detección de intrusos consiste en determinar, a partir de dicha evaluación borrosa, el valor o calificación concreta o "crisp" del tipo de usuario como malintencionado. Se precisa entonces proceder a la fase de clarificación o desembo-

rronamiento del resultado obtenido, atendiendo al tipo de evaluación de reglas borrosas, esto es, se precisa proceder a la concretización o reducción a un valor específico de toda la información contenida en la figura polinómica extraída de la herramienta de representación gráfica tridimensional *Xf3dplot* incluida en Xfuzzy (veáse Anexo 2) incluida en la Figura 18.

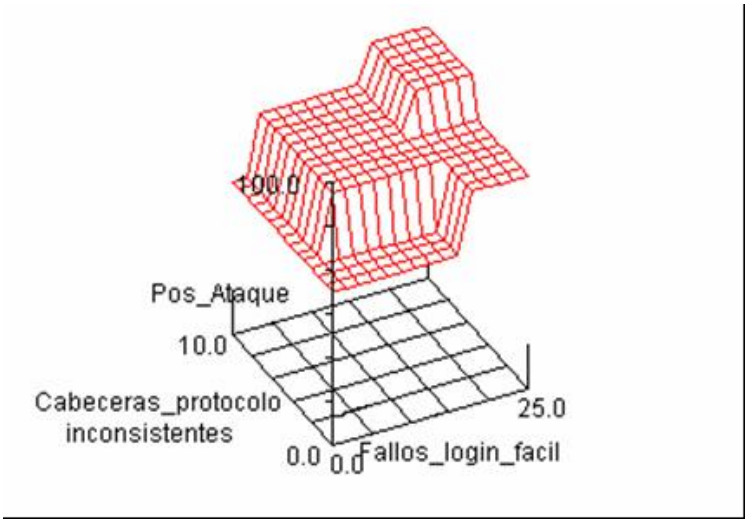


Figura 18

4.6. Detalle del funcionamiento del sistema de detección de intrusos en Xfuzzy 3.0.

Para la implementación de la aplicación, debido a la gran cantidad de variables que impedían la claridad de un esquema completo, se ha decidido que en aras al propósito pedagógico del presente documento desarrollar tan sólo la segunda parte del esquema inicial, esto es, el mapa cognoscitivo o esquema asociativo de la implementación realizada consiste en el que se muestra en la Figura 16 anteriormente citada.

Para llevar a cabo la implementación de dicho sistema experto borroso se ha utilizado XFuzzy, en su versión 3.0 (veáse el Anexo 2), de forma que dicho esquema asociativo, una vez "traducido" al formato de Xfuzzy, ha quedado tal como se muestra en la Figura 19. Así mismo, a los efectos del presente trabajo, el código fuente del sistema de detección de intrusos aplicando Xfuzzy 3.0., en su codificación en Java, se encuentra en el Anexo 3 que acompaña el mismo.

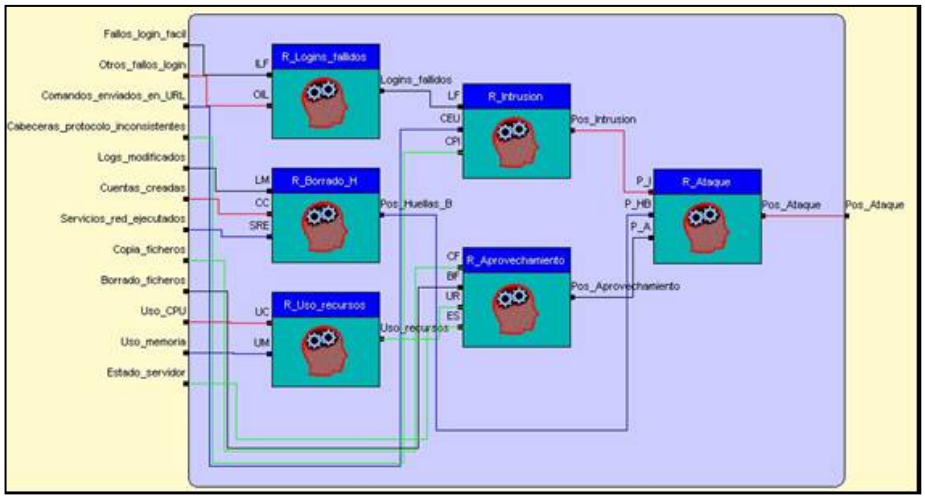


Figura 19

Para la comprobación del funcionamiento del sistema, se han efectuado un conjunto de pruebas con las opciones de verificación que incorpora la propia herramienta Xfuzzy dentro de su menú "Verification".

No obstante, para mostrar su funcionamiento de una forma que podría ser más similar a la que percibiría un usuario que manejase la aplicación final se ha desarrollado un interfaz gráfico HTML.

Al entrar en la aplicación cabe observar en el navegador una pantalla en la que se solicitaría información al respecto de los valores de los parámetros detectados por los diferentes sensores del IDS. Estos valores en general se refieren al número de veces que ha ocurrido un determinado suceso (intentos fallidos de entrar en el sistema, creación de cuentas de usuario, envio de comandos como parámetros a una URL, etc.), pero en el caso de la utilización de la CPU y de la memoria se refieren al porcentaje de este recursos utilizado por el usuario vigilado, mientras que en el caso de estado del servidor hace referencia a los milisegundos que tarda un paquete *ping* en ser respondido por el servidor.

En la Figura 20 se muestra la pantalla en la que se deberá rellenar los datos solicitados en las casillas:

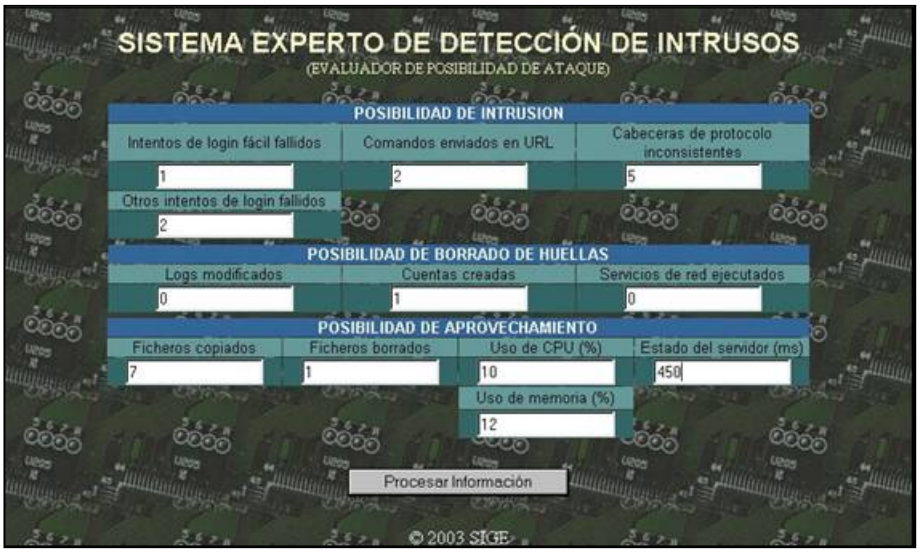


Figura 20

Los datos introducidos y resultados concretos para este ejemplo que han sido introducidos son los que se indican en el Cuadro 55.

ENTRADAS	
Intentos de login fácil fallidos	1
Otros intentos de login fallidos	2
Comandos enviados en URLs	2
Cabeceras de protocolo inconsistentes	5
Logs modificados	0
Cuentas creadas	1
Servicios de red ejecutados	0
Ficheros copiados	7
Ficheros borrados	1
Utiliz. de CPU (%)	10
Utiliz. de Memoria (%)	12
Estado del servidor	450
SALIDA	
Posibilidad de ataque	Baja (22,5%)

Cuadro 55

Una vez introducida dicha información, se podrá pulsar el botón "Procesar Información" y, tras hacer los oportunos cálculos, siguiendo los procedimientos del Xfuzzy, se obtendrá en el navegador una nueva pantalla, donde se muestra la solución al problema planteado, en este caso la información acerca posibilidad de que se esté produciendo un ataque al sistema, así como los valores de las variables intermedias que se han definido para el mismo. Dicha pantalla puede observarse en la Figura 21.

Figura 21

5. CONCLUSIONES Y LÍNEAS DE INVESTIGACIÓN FUTURAS

En este trabajo se ha propugnado un diseño de sistema de detección de intrusos con un sistema experto embebido que interpreta información de forma adecuada, filtrando el exceso de datos que pueden hacer a un administrador disminuir la atención que debe de prestar al sistema, para lo cual en una vez efectuado una descripción del tópico de estudio, se presento el modelo propugnado, analizando las fases de su diseño e implementación práctica, incluyendo la presentación del interface desarrollado junto con un ejemplo de aplicación práctica al respecto donde se evidencia y contrasta la validez del enfoque propuesto en su aplicación en el ámbito de la detección de intrusos.

El esfuerzo investigador no sólo ha permitido obtener resultados de interés, sino que también ha abierto nuevos caminos cuya exploración parece ser prometedora para desarrollos futuros, mencionándose aquí tan sólo dos líneas de investigación y desarrollo al respecto, a saber: en primer término, ampliar la base de expertizaje, tanto en cantidad como en calidad de la información y el conocimiento requeridos y, en segundo lugar, avanzar en el estudio de las técnicas de ajuste de la solución del SBRBs con la aplicación de los Algoritmos Genéticos como herramienta para el aprendizaje de las bases de reglas borrosas.

6. BIBLIOGRAFÍA

BARDOSSY, A. y DUCKSTEIN, L. (1995): *Fuzzy rule–based modeling with application to geophisical biological and engineering system*, CRC Press. New York.

BERENJI, H. (1993): *Fuzzy Logic Controllers*. Incluido en "An Introduction to Fuzzy Logic Applications in Intelligent Systems", YAGER, R. y ZADEH, L. (eds.), Kluwer Academic, Boston, págs. 69–96.

CÁRDENAS, E., CASTILLO, J., CORDÓN. O., HERRERA, F. y PEREGRÍN, A. (1994): *Influence of Fuzzy Implication Functions and Defuzzification Methods in Fuzzy Control*, Busefal, Vol. 57, págs. 69–79.

CASWELL, B. (2003). "SNORT Rules Database". Disponible en http://www.snort.org, 2003

COMPUTER EMERGENCY RESPONSE TEAM (1999). "Results of the distributed systems intruder tool workshop". Disponible en http://www.cert.org/reports/dsit_workshop–final.html",

DRIANKOV, D., Hellendoorn, H. y REINFRANK, M. (1993): *An introduction to fuzzy control*, Springer–Verlag, Berlín.

DUJECT, CH. y VINCENT, N. (1995): *Force Implication: A new approach to human reasoning*, Fuzzy Sets and Systems, Vol. 69, págs. 53–63.

EMAMI, M., TÜRKSEN J. y GOLDENBERG, A. (1998): *Development of A Systematic Methodology of Fuzzy Logic Modelling*, IEEE Transactions on Fuzzy Systems, Vol. 6, págs. 346–361.

GRAHAM, R. (2000). "FAQ about Network Intrusion Detection Systems". Disponible en http://www.robertgraham.com/pubs/network–instrusion–detection.html

GUPTA, M. y QI, J. (1991a): *Theory of T–norms and Fuzzy Inference Methods*, Fuzzy Sets and Systems, Vol. 40, págs. 431–450.

GUPTA, M. y QI, J. (1991b): *Design of Fuzzy Logic Controllers Based on Generalized T–operators*, Fuzzy Sets and Systems, Vol. 40, págs. 473–489.

HELLENDOORN, H. (1993): *Design and Development of Fuzzy Systems at Siemens R&D*, Proceedings of the FUZZ–IEEE'93, San Francisco, págs. 1365–1370.

HELLENDOORN, H. y THOMAS, C. (1993): *Defuzzification in Fuzzy Controllers*, Journal of Intelligent and Fuzzy Systems, Vol. 1, págs. 109–123.

HELLENDOORN, H., DRIANKOV, D. y REINFRANK, M. (1993): *An Introduction to Fuzzy Control*, Springer Verlag, Berlín.

HIROTA, K. (ed.) (1993): *Industrial Applications of Fuzzy Technology*, Springer–Verlag, Berlín.

INTERNET SOCIETY (1998). "RFC 2401: Users' Security Handbook". Disponible en http://www.faqs.org

INTERNET SOCIETY (2000). "RFC 2828: Internet Security Glossary". Disponible en http://www.faqs.org

INTERSECT ALLIANCE (2002). "SNARE: System iNtrusion Analysis & Reporting Environment". Disponible en http://www.intersectalliance.com/projects/Snare/Documentation/

KAUFMANN, A. y GIL ALUJA, J. (1986): *Introducción de la Teoría de los Subconjuntos Borrosos a la Gestión de Empresas*, Milladoiro, Santiago de Compostela.

KAUFMANN, A. y GIL ALUJA, J. (1987): *Técnicas Operativas de Gestión para el Tratamiento de la Incertidumbre*, Hispano Europea, Barcelona.

KAUFMANN, A. y GIL ALUJA, J. (1990): *Las Matemáticas del Azar y de la Incertidumbre. Elementos Básicos para su Aplicación en Economía*, Centro de Estudios Ramón Areces, Madrid.

KAUFMANN, A. y GIL ALUJA, J. (1993): *Técnicas Especiales para la Gestión de Expertos*, Milladoiro, Santiago de Compostela.

KLIR, G. y YUAN, B. (1995): *Fuzzy Sets and Fuzzy Logic*, Prentice–Hall, New York.

LEE, C. (1990): *Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Parts I and II*, IEEE Trans. on Systems, Man and Cybernetics, Vol. 20, nº 2, págs. 404–418 and 419–435.

LÓPEZ GONZÁLEZ, E. (2001): A Methodology for Building Fuzzy Expert Systems (FES) with Spreadsheet to Quality Function Deployment (QFD) of the Target Costing. Incluido en Gil Aluja, J. (ed): "Handbook of Management under Uncertainty". Kluwer Academic Publishers. Dordrecht, págs. 457–535.

MAMDANI, E. (1974): *Applications to fuzzy algorithms for simple dynamic plant*, Proceedings IEEE, Vol. 21, nº 12, págs. 1585–1588.

MAMDANI, E. y ASSILIAN, S. (1975): *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man–Machine Studies, Vol. 7, págs. 1–13.

MIZUMOTO, M. (1989): *Pictorial Representations of Fuzzy Connectives, Part I: cases of t–norms, t–conorms and averaging operators*, Fuzzy Sets and Systems, Vol. 31, págs. 217–242.

MORENO VELO, F. J., BATURONE, I., SÁNCHEZ–SOLANO, S., BARRIGA, A. y SENHADJ, R. (2002): "El entorno Xfuzzy 3.0 de diseño de sistemas borrosos". Proc. XI Congreso Español sobre Tecnologías y Lógica Fuzzy, León, p. 353–358.

NORTCUFF, S. and NOVAK, J. (2001). "Detección de intrusos. Guía avanzada". Prentice Hall (2ª edición)

PEDRYCZ, W. (ed.) (1996): *Fuzzy Modeling: Paradigms and Practice*, Kluwer Academic, Boston.

PFLUGER, N., YEN, J. y LANGARI, R. (1992): *A Defuzzification Strategy for a Fuzzy Logic Controller Employing Prohibitive Information in Command Formulation*, Proceedings of the FUZZY–IEEE 92, San Diego, págs. 712–723.

SUGENO, M. y YASUKAWA, T. (1993): *A fuzzy–logic–based approach to qualitative modelling*, IEEE Transactions on Fuzzy Systems, Vol. 1, nº 1, págs. 7–31.

TAKAGI, T. y SUGENO, M. (1985): *Fuzzy Identification of Systems and Its Applications to Modeling and Control*, IEEE Trans. on Systems, Man, and Cybernetics, Vol. 15, nº 1, págs. 116–132.

TRILLAS, E. y VALVERDE, L. (1985): *On Implication and Indistiguishability in the Setting of Fuzzy Logic*. Incluido en "Management Decision Support Systems Using Fuzzy Sets and Possibility Theory", KACPRYZK, J. y YAGER, R. (eds.), Verlag TÜV, Köln, págs. 198–212.

TSUKAMOTO, T. (1979): *An approach to fuzzy reasoning method*. Incluido en "Advances in Fuzzy Set Theory and Applications" GUPTA, M., RAGADE, R., YAGER, R. (eds.), North–Holland, Amsterdam.

WANG, L. (1992): *Fuzzy systems are universal aproximators*. Incluido en "Proceedings of the First IEEE International Conference on Fuzzy Systems (FUZZ–IEEE'92)", San Diego, págs. 1163–1170.

ZADEH, L. (1973): *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Trans. on Systems, Man and Cybernetics, Vol. 3**,** nº 1, págs. 28–44.

ZIMMERMAN, H. (1996): *Fuzzy Sets Theory and Its Applications*, Kluwer Academic.

ANEXO 1. TRATAMIENTO DE LA INCERTIDUMBRE: LOS SUBCONJUNTOS BORROSOS

AI.1. LA INCERTIDUMBRE Y SUS TIPOS

La incertidumbre, en el ámbito de la gestión, se puede definir por el desconocimiento acerca del estado que va a tomar una determinada variable que afecta a una decisión económica. Por ello, se han de considerar diferentes situaciones en las cuales es posible obtener distintos resultados para cada curso de acción. No es necesariamente que exista ignorancia, sino que resulta imposible establecer la probabilidad de cada uno de los estados que pueden tomar las variables.

Tradicionalmente la presencia de incertidumbre ha sido manejada por medio de herramientas estándar como la teoría de la probabilidad, en particular la metodología Bayesiana y la Teoría de la Utilidad. Únicamente en los últimos años los científicos se han dado cuenta de las limitaciones que tales herramientas presentan.

Muchas disciplinas matemáticas trabajan con la descripción de incertidumbre, por ejemplo, la teoría de la probabilidad, teoría de la información y la Teoría de los Subconjuntos Borrosos. Resulta conveniente clasificar estas disciplinas en función del tipo de incertidumbre que manejan. Para tal fin consideraremos dos tipos de incertidumbre: probabilística y lingüística.

AI.1.1. Incertidumbre probabilística

La incertidumbre probabilística maneja la incertidumbre de que ocurra un determinado evento. El evento en sí mismo está bien definido, encontrándose la incertidumbre en grado de probabilidad de que el mismo tome un estado u otro. Por ejemplo, cuando se dice que se va conseguir beneficios con un 80% de probabilidad, estamos manejando una incertidumbre mensurable.

Además, este tipo de afirmaciones pueden manejarse utilizando métodos estocásticos, tales como el cálculo Bayesiano que permite medir la información de que se dispone, realizar operaciones y llegar a conclusiones sobre la misma.

Los modelos Bayesianos constituyen un cuerpo de herramientas que han sido aplicadas para manejar la incertidumbre. Los más sencillos de ellos no consideran diferentes grados de verdad. Sin embargo, modelos más complejos incorporan estructuras jerarquizadas de análisis de la evidencia que combinan las diversas piezas de la misma utilizando la probabilidad para representar el riesgo.

Es en este contexto donde se encuentra el azar, y su medición a través de probabilidades, el cual es una medida sobre hechos observables y constituye una evaluación que se desearía fuera lo más objetiva posible.

Sin embargo, cuando la posibilidad de ocurrencia de un determinado evento no se puede medir e influye de manera determinante la subjetividad de las apreciaciones realizadas por los seres humanos, no pueden utilizarse probabilidades, por tanto, se precisa de otra herramienta que sea capaz de trabajar con este tipo de información.

En este sentido, aunque con demasiada frecuencia se afirma que un evento es probable cuando no existe posibilidad de medición, se está tratando de utilizar la objetividad intentando mantener la fuerza de la teoría de probabilidades, pero de acuerdo con Kaufmann y Gil aluja ([3]) si se pretende ser honesto se ha de manejar por medio de valuaciones que aún siendo más débiles resultan más realistas.

AI.1.2. Incertidumbre lingüística

Un tipo diferente de incertidumbre es la que se encuentra en el lenguaje natural. Para poder efectuar su manejo, hay que tener en cuenta la imprecisión inherente a la mayoría de las expresiones humanas, con el fin de evaluar los conceptos y derivar conclusiones. Así, conceptos tales como "precio alto", "demanda baja" o "fuerte competencia" que no se corresponden con definiciones exactas se pueden considerar como dentro de este tipo de incertidumbre. Por otro lado, la consideración del precio de un artículo por un directivo como "alto" puede diferenciarse sustancialmente de la opinión que emita otro, debido a que las personas utilizan su conocimiento para elevar los juicios y por tanto este influirá de manera determinante en ellos, siendo poco posible que dos individuos coincidan.

---

[3] A. Kaufmann y J. Gil Aluja (1986). Introducción de la teoría de los subconjuntos borrosos en la Gestión de Empresas, Santiago de Compostela.

La ciencia que trabaja con la forma en que las personas evalúan conceptos es la Psicolingüística, estando demostrado por ella que los humanos utilizan palabras como categorías subjetivas para clasificar conceptos tales como "precio", "demanda" e "inflación". Por medio de la utilización de estas categorías subjetivas, hechos del mundo real pueden ser evaluados por el grado en que satisfacen determinados criterios.

En este sentido, aún cuando la mayoría de estos conceptos no están definidos de forma precisa, los humanos pueden utilizarlos para evaluar y tomar decisiones complejas que se sustentan en una variedad de factores. Gracias a ello, utilizando esta abstracción y pensando en analogías, unas pocas expresiones pueden describir contextos complejos que resultarían difíciles de modelar con precisión matemática.

Si se plantean consideraciones del tipo: "es posible alcanzar el objetivo", puede parecer que la sentencia es igual que la de la incertidumbre probabilística, sin embargo existen diferencias sustanciales como las siguientes, a saber:

- El evento en sí mismo no está definido claramente
- El significado de la expresión posible en este segundo caso no lleva aparejado un sentido matemático.

AI.1.3. Incertidumbre en la toma de decisiones de gestión

Las afirmaciones, términos y reglas utilizadas en la gestión de las organizaciones económicas son normalmente ambiguas. El encargado de la toma de decisiones no suele disponer de recoger todos los datos relevantes para la misma, por lo que no es capaz de convertir las decisiones en precisas. El directivo debe tomar decisiones que conciernen al futuro de su organización, basándose en su criterio personal y subjetivo. Así, la posibilidad de que entre un conjunto de opciones exista alguna con mayor grado de ocurrencia que las otras da lugar al comienzo del proceso de toma de decisiones.

En el entorno del mundo real, él responsable de la toma las decisiones en la mayoría de las circunstancias difícilmente puede observar un experimento repetidamente para eliminar la aleatoriedad del mismo. Consecuentemente, mientras el criterio probabilístico es aplicable a tomas de decisiones con resultados observables, no puede utilizarse para cualquier tipo de toma de decisiones. Por ello, se precisa de métodos eficientes no sólo para estimar el grado de verdad sino también para construir expresiones a partir de datos que sean ambiguos por sí mismos.

Tradicionalmente existe una tendencia en la gestión a utilizar información precisa, aunque esa información es difícil de obtener en muchos casos y por ello puede tener un impacto adverso en la aplicación de modelos de gestión. Esta tendencia limita la aplicación de dichos modelos a problemas del mundo real. Además, otro problema relacionado con la imposibilidad de disponer de información precisa lo constituye el enriquecimiento de datos que fuerza a que el modelo utilice información irreal([4]).

Los directivos toman decisiones sin tener un amplio número de informaciones precisas. Por ejemplo, la decisión "mejorar la calidad", puede llevar aparejada la toma de otras decisiones por parte de diferentes empleados de la empresa, muchas de ellas precisas. La habilidad de los directivos para manejar ese tipo de situaciones les aporta una gran flexibilidad en la toma de decisiones, mayor de la que obtendrían con métodos tradicionales.

AI.2. TRATAMIENTO DE LA INCERTIDUMBRE

AI.2.1. Introducción

Algunas veces es necesario estimar la probabilidad de ocurrencia de un determinado evento en ausencia de conocimiento sobre sus frecuencias relativas. Realizar estimaciones en tales situaciones suele ser impreciso, quedando la utilidad de los métodos Bayesianos limitada por la divergencia entre la intuición y la interpretación del análisis de riesgo de los mismos.

Kaufmann y Gil Aluja introdujeron en los primeros años de la década de los ochenta la aplicación de la Teoría de los Subconjuntos Borrosos como una herramienta de representación de la ambigüedad y vaguedad de los sistemas económicos. Aunque en la gestión de las empresas se han propuesto numerosos modelos de decisión, estos generalmente no incorporan la ambigüedad inherente a la información disponible, siendo reducida en muchos casos de manera errónea a la aleatoriedad. Sin embargo, para los citados autores, conviene tener en cuenta una notable diferencia entre aleatoriedad y ambigüedad: mientras la aleatoriedad se relaciona con la incertidumbre acerca de la ocurrencia de un evento, la ambigüedad lo hace con la incertidumbre del grado de ocurrencia de un evento.

---

[4] R. Ackoff (1986). *Management in Small Doses*. John Wiley. New York.

En la teoría tradicional de conjuntos, la pertenencia se definía en términos binarios, un elemento pertenecía o no al conjunto. Sin embargo la Teoría de los Subconjuntos Borrosos permite que un elemento pertenezca a varios conjuntos con un grado de verdad. Por ello, la ausencia de límites estrictos entre los conjuntos permite añadir flexibilidad en la toma de decisiones.

La intención de la Teoría de los Subconjuntos Borrosos no es la de reemplazar a la teoría de la probabilidad en la medición de la aleatoriedad probabilística, sino proporcionar una manera natural de trabajar con problemas en los que la fuente de la imprecisión radica en la ausencia de criterios estrictos más que en presencia de variables aleatorias.

AI.2.2. Subconjuntos Borrosos: Conceptos básicos

Un conjunto ordinario está definido siempre con respecto a algún universo de discurso *X*, que por sí mismo es un conjunto clásico. En particular se define un subconjunto con la ayuda de una función característica que describe la pertenencia al subconjunto. Sea *X* un universo de discurso y sea *S* un subconjunto de *X*. La función característica asociada con *S* es

$$\mu_S : X \to \{0,1\}$$

tal que para cualquier elemento *x* del universo, $\mu_S(x) = 1$ si *x* es un miembro de *S* y $\mu_S(x) = 0$ si *x* no es un miembro de *S*. La Figura AI.1 muestra la idea de una función característica, siendo el eje *X* el conjunto de los números reales y *S* el subconjunto de los números reales que están entre 0 y 20, mientras que en la Figura AI.2 se muestra un subconjunto clásico en un plano.

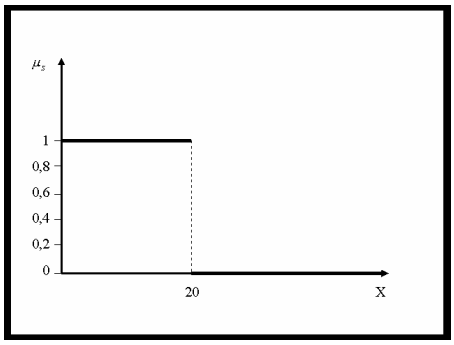


Figura AI.1

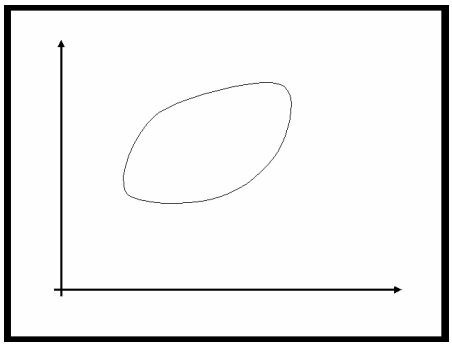


Figura AI.2

Al igual que ocurre con un subconjunto clásico, un subconjunto borroso esta definido también con respecto a un conjunto clásico, denominado universo o dominio. El subconjunto borroso generaliza la idea de un subconjunto clásico extendiendo el rango de la función característica del par *{0,1}* al intervalo *I=[0,1].* La siguiente definición establece el concepto de un subconjunto borroso.

Por tanto, a los efectos del presente anexo cabe considerar como definición de Subconjunto Borroso la siguiente: Sea *X* un conjunto que sirve de universo. Un subconjunto borrosos asociado con la función característica: $\mu_S : X \to [0,1]$.

En el marco de trabajo de la Teoría de los Subconjuntos Borrosos se denomina generalmente a la función característica del mismo como función de pertenencia asociada al mismo. Esta terminología conlleva la idea de que para cada *x*, $\mu_S(x)$ indica el grado en el que *x* es miembro del conjunto A. En la Figura AI.3 se muestra un ejemplo de subconjunto borroso con respecto a un eje, mientras que en la Figura AI.4 se representa en un plano, diferenciando el grado de pertenencia mediante una escala de grises.

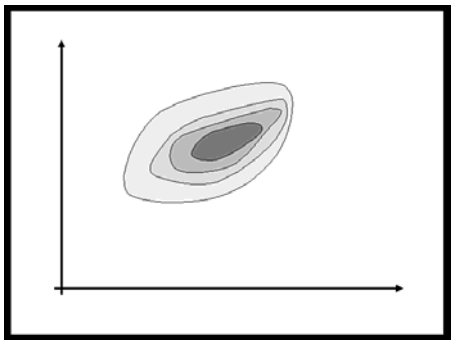


Figura AI.3

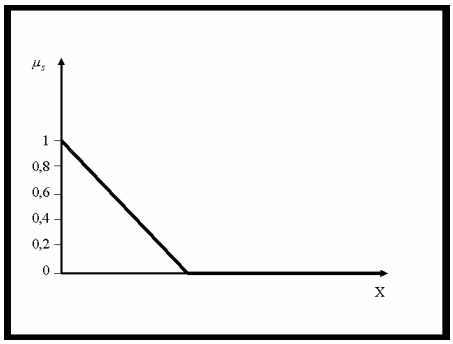


Figura AI.4

Así, un elemento puede no ser miembro del subconjunto *X*, siendo $\mu(x) = 0;$ puede pertenecer en un nivel bajo, siendo $\mu(x) = 0{,}1;$ puede pertenecer más o menos al subconjunto, siendo $\mu(x) = 0{,}5;$ puede ser casi un miembro del subconjunto, siendo $\mu(x) = 0{,}8;$ y por último, puede ser un miembro estricto, siendo $\mu(x) = 1;$ Las Figuras I.5 y I.6 muestran dos clases importantes de funciones de pertenencia: la triangular y la trapezoidal.

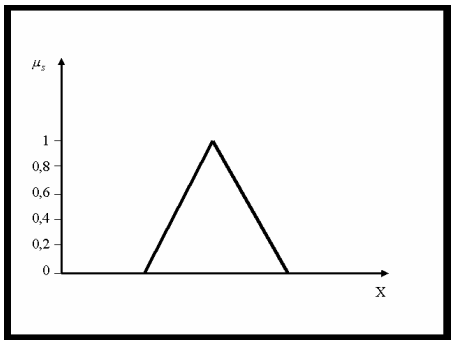


Figura AI.5

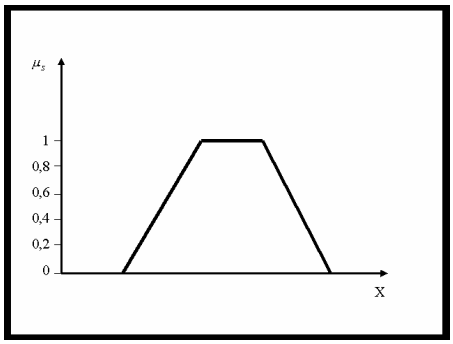


Figura AI.6

Los subconjuntos borrosos pueden utilizarse tanto para describir conceptos vagos, que no tienen límites estrictos, como conceptos ambiguos o conceptos que describen subconceptos distinguibles. La ausencia de límites estrictos es adecuada para la asignación de grados de pertenencia para diferentes elementos, permitiendo que más de uno de ellos sea miembro del conjunto.

Los grados de pertenencia de un subconjunto borroso representan el nivel de compatibilidad de un elemento con el conjunto y no debe ser interpretado como una probabilidad. Además la función de pertenencia de un subconjunto borroso puede representarse de una manera analítica como una fórmula mejor que como un número finito de elementos.

En situaciones en las que el universo de discurso no son los números reales, el grado de pertenencia debe estar expresado explícitamente. Así, por ejemplo, si $X = \{x_1, x_2, x_3, x_4\}$, un subconjunto podría ser $A = \{0{,}7 / x_1, 0{,}3 / x_2, 1 / x_3, 0 / x_4\}$, representando un subconjunto borroso de *X*. El significado de los términos de *A* en la forma a_i / x_i indican el grado de pertenencia al subconjunto de cada elemento x_i del conjunto. De acuerdo con esta interpretación, en el ejemplo anterior el elemento x_1 pertenece al subconjunto *A* con un grado 0,7. En la práctica es común suprimir aquellos elementos cuyo grado de pertenencia es 0.

De esta forma, el subconjunto quedaría: $A = \{0{,}7 / x_1, 0{,}3 / x_2, 1 / x_3\}$.

Tradicionalmente, se ha utilizado el término μ_A para indicar la función de pertenencia de un subconjunto *A*. En el presente trabajo se ha optado por una notación ligeramente diferente, utilizando $A(x)$ para indicar la función de pertenencia del subconjunto *A*, y por tanto, los términos quedarían como $A(x_i) / x_i$. Aplicando esta representación a conjuntos clásicos se obtendría entonces que, si $X = \{x_1, x_2, x_3, x_4\}$, el subconjunto clásico $B = \{x_1, x_3\}$ puede expresarse como $B = \{1/x_1, 0/x_2, 1/x_3, 0/x_4\}$.

La utilización de esta notación permite dejar patente que los subconjuntos clásicos son casos excepcionales de los subconjuntos borrosos en los cuales los grados de pertenencia sólo pueden ser cero o uno.

Los subconjuntos borrosos son especialmente útiles a la hora de representar conceptos con límites imprecisos. Así, por ejemplo, si el universo de discurso lo constituyen determinados artículos sustitutivos, *X*, y estamos interesados en representar el subconjunto de ellos que son caros, la utilización de los subconjuntos borrosos en esta situación libera de la restricción de incluir o no los artículos como miembros del subconjunto, pudiendo especificar para cada uno un grado de pertenencia entre 0 y 1. Esta habilidad para representar de una manera más natural conceptos imprecisos es particularmente importante en el diseño de sistemas inteligentes, ya que permiten representar de una manera más acertada los tipos de conceptos del razonamiento humano y más concretamente en la economía y la empresa. Por ello, términos tales como calidad alta, precio barato, competencia fuerte, etc. son representados de una manera más natural mediante subconjuntos borrosos que mediante subconjuntos clásicos. Además en muchos casos el valor del grado de la función de pertenencia para un subconjunto borroso depende de un juicio subjetivo, como la creencia en una idea. En este sentido, en diversas ocasiones tiene más valor la forma de la función de pertenencia que los valores de la misma.

Muchas de las definiciones y operaciones asociadas con los subconjuntos borrosos son extensiones directas de las correspondientes definiciones de los subconjuntos clásicos. Además, esas definiciones y operaciones que son extensión de los subconjuntos borrosos ordinarios engloban a éstos cuando se restringen a valores de pertenencia 0 o 1, apareciendo también nuevas definiciones y operaciones que no se encontraban o que no tenían sentido para los subconjuntos clásicos.

AI.2.2.2. Operaciones con los subconjuntos borrosos

La descripción de las operaciones se hace normalmente extendiéndolas desde los subconjuntos clásicos, existiendo varias posibles alternativas. Como se dijo anteriormente muchas de estas operaciones de los subconjuntos borrosos incluyen a las respectivas en los subconjuntos clásicos cuando los valores de pertenencia se restringen a 0 o 1. Por ello, la simbología utilizada es la misma que para el enfoque tradicional. Sin embargo, para aquellas operaciones específicas de los subconjuntos borrosos se utilizará una representación específica.

1. Unión

Sean *A* y *B* dos subconjuntos borrosos de *X*, su unión será otro subconjunto borroso *C* de *X*, denotado, $C(x)$, tal que para cada $x \in X$

$$C(x) = Max\left[A(x), B(x)\right] = A(x) \vee B(x)$$

Es una práctica común en la literatura relacionada con los subconjuntos borrosos utilizar el símbolo $\vee$ como el operador Máximo.

Así, por ejemplo si $X = [a, b, c, d, e]$ y *A* y *B* son dos subconjuntos borrosos de *X* donde

$$A = [1/a,\ 0{,}7/b,\ 0{,}3/c,\ 0/d,\ 0{,}9/e] \quad \text{y}$$

$$B = [0{,}2/a,\ 0{,}9/b,\ 0{,}4/c,\ 1/d,\ 0{,}4/e]$$

Entonces $C = [1/a,\ 0{,}9/b,\ 0{,}4/c,\ 1/d,\ 0{,}9/e]$ y gráficamente podemos representar la unión tal como se muestra en la Figura AI.7.

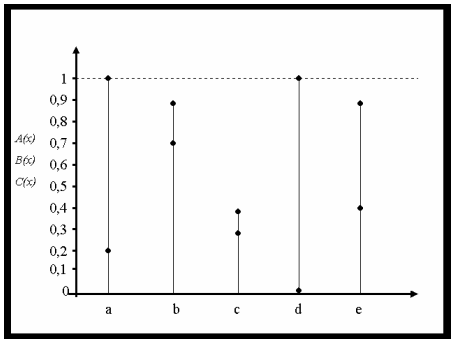


Figura AI.7

2. Intersección

Sean A y B dos subconjuntos borrosos de X, su intersección será otro subconjunto borroso D de X, denotado, tal que para cada $x \in X$

$$D(x) = Min\left[A(x), B(x)\right] = A(x) \wedge B(x)$$

A su vez, es práctica común en la literatura relacionada con los subconjuntos borrosos utilizar el símbolo $\wedge$ como el operador Mínimo.

Así, de acuerdo con los subconjuntos A y B del ejemplo anterior, su intersección será

$$D = \left[0{,}2 / a, 0{,}7 / b, 0{,}3 / c, 0 / d, 0{,}4 / e\right]$$

y gráficamente su representación se muestra en la Figura AI.8.

Las operaciones de Máximo y Mínimo, que juegan un papel de alta trascendencia en la Teoría de los Subconjuntos Borrosos, pueden escribirse en términos algebraicos como:

$$Max(A(x), B(x)) = \frac{A(x) + B(x) + \left|A(x) - B(x)\right|}{2}$$

$$Min(A(x), B(x)) = \frac{A(x) + B(x) - \left|A(x) - A(x)\right|}{2}$$

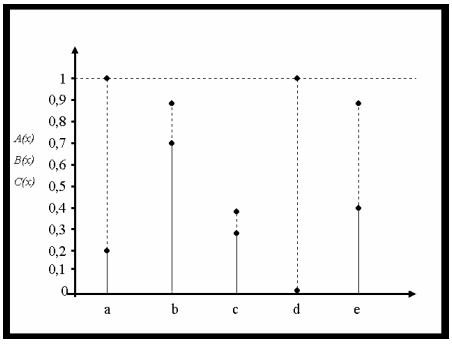


Figura AI.8

Además, cumplen diferentes propiedades y coinciden exactamente con las correspondientes para los conjuntos clásicos.

Si A y B son dos subconjuntos borrosos de X, las siguientes son las propiedades que se cumplen para su unión y su intersección.

Unión:

- Conmutativa: $A \cup B = B \cup A$
- Idempotencia: $A \cup A = A$
- Asociativa: $A \cup (B \cup C) = (A \cup B) \cup C = A \cup B \cup C$
- Distributiva: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- Elemento neutro: $A \cup \phi = A$
- Elemento absorbente: $A \cup X = X$
- Inclusión: $A \subset A \cup B$
- Sí $A \subset B$
  $B = A \cup B$
- Sí $A \subset B$ y $B \subset C$, entonces $A \subset C$

Intersección:

- Conmutativa: $A \cap B = B \cap A$
- Idempotencia: $A \cap A = A$
- Asociativa: $A \cap (B \cap C) = (A \cap B) \cap C = A \cap B \cap C$
- Distributiva: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- Elemento neutro: $A \cap X = A$
- Elemento absorbente: $A \cap \phi = X$
- Inclusión: $A \cap B \subset A$
- Sí $A \subset B$
  $A = A \cap B$

Otras operaciones a realizar con los subconjuntos borrosos son la complementación y la suma disyuntiva.

3. Complementación

En cuanto a la complementación, si tenemos que *A* es un subconjunto borroso de *X*, el complementario o negación de *A*, se define como el subconjunto borroso, $\overline{A}$, tal que para cada $x \in A$.

$$\overline{A}(x) = 1 - A(x)$$

En el ejemplo anterior, el complementario del subconjunto borroso *A,* sería:

$$A = \left[0/a, 0{,}3/b, 0{,}7/c, 1/d, 0{,}1/e\right]$$

y gráficamente, tal como se muestra en la Figura AI.9.

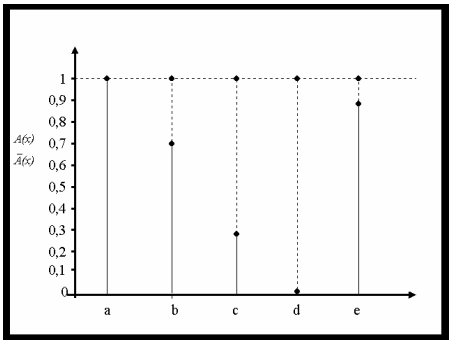


Figura AI.9

De este modo, se puede decir que la negación es el complemento de un subconjunto borroso con respecto al espacio completo y además cumple las siguientes propiedades:

- Doble negación o involución

$$\left(\overline{A}\right) = A$$

- Ley de Morgan

$$\left(\overline{A \cup B}\right) = \overline{A} \cap \overline{B}$$

$$\left(\overline{A \cap B}\right) = \overline{A} \cup \overline{B}$$

- $\overline{\phi} = X$
- $\overline{X} = \phi$

4. Suma Limitada

Por su parte la suma limitada, denotada por $D = A \oplus B$, se define de la siguiente forma:

$$D = Min\left[1, A(x) + B(x)\right]$$

Siguiendo con el ejemplo, la suma limitada de los subconjuntos borrosos A y B quedaría como:

$$D = \left[1/a, 1/b, 0{,}7/c, 1/d, 1/e\right]$$

y su representación gráfica tal como se muestra en la Figura AI.10.

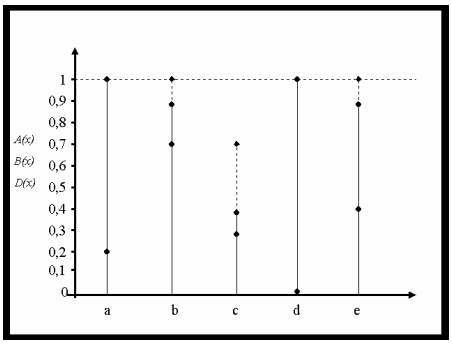


Figura AI.10

Esta operación está estrechamente relacionada con la unión de subconjuntos borrosos. De hecho, si A y B son dos subconjuntos clásicos, entonces, $D = A \oplus B = A \cup B$.

La descripción de las operaciones anteriores se ha realizado para referenciales finitos, resultando sencilla su generalización a referenciales infinitos. Así, gráficamente la representación de la unión, intersección, negación y suma limitada, se pueden observar en las Figuras I.11, I.12, I.13 y I.14 respectivamente.

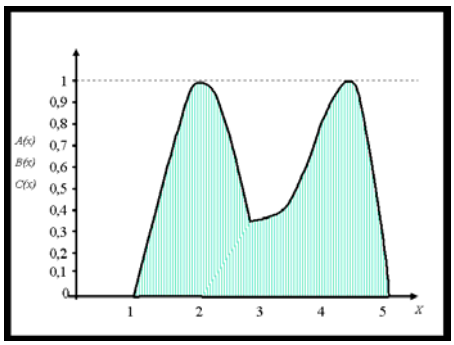


Figura AI.11

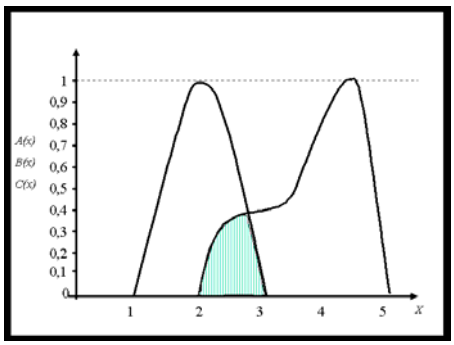


Figura AI.12

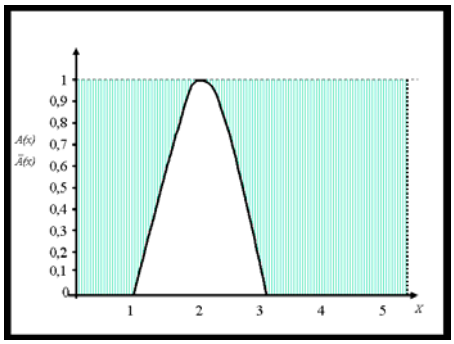


Figura AI.13

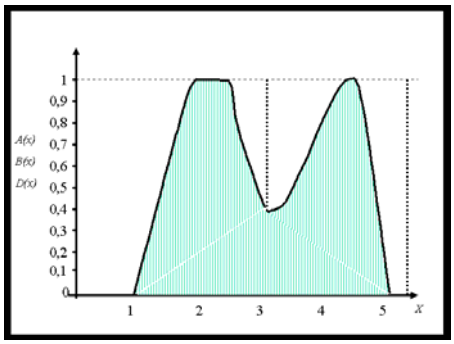


Figura AI.14

AI.2.3. Números Borrosos

En muchas situaciones, las personas son capaces únicamente de caracterizar la información numérica de forma imprecisa. Por ejemplo, se utilizan términos tales como alrededor de 5, cerca de 0, más o menos 8, etc. Éstos son ejemplos de lo que se denominan números borrosos.

Utilizando la Teoría de los Subconjuntos Borrosos se pueden representar estos números borrosos como subconjuntos borrosos del conjunto de los números reales. De cualquier modo, para que sea factible utilizar estos números borrosos se debe conseguir realizar operaciones con tales números, estando entre ellas la suma, sustracción, multiplicación y división. Todo el proceso de realización de estas operaciones recibe el nombre de Aritmética Borrosa.

Se define un número borroso como un subconjunto borroso del referencial de los números reales cuya función de pertenencia es *normal*, es decir, existe un valor del subconjunto para el cual su grado de pertenencia es 1, y *convexa*, la función de pertenencia a la izquierda y derecha de ese valor es monótona creciente.

AI.2.3.1. Números Borrosos Triangulares y Trapezoidales

De entre todos los números borrosos destacan por su facilidad de utilización, el *número borroso triangular* y el *número borroso trapezoidal*. El primero, Figura AI.15, posee la particularidad de venir determinado por tres cantidades: una por debajo de la cual no va a descenderse, otra a la que por encima no será posible llegar y, finalmente, aquella que representa

el máximo nivel de *presunción*. El segundo, Figura AI.16, viene determinado por cuatro cantidades; la primera y la cuarta tienen la misma interpretación que en el caso de los triangulares, mientras que la segunda y tercera representan un intervalo de confianza para el cual se tiene el máximo nivel de presunción, siendo el número borroso triangular un caso particular del trapezoidal cuando los valores segundo y tercero de este último coinciden. Ambos tipos de números borrosos permiten formalizar de manera muy fidedigna gran cantidad de situaciones de la empresa en la que se estiman magnitudes localizadas en el futuro.

Así, por ejemplo, la estimación del precio de un artículo en el futuro podría realizarse afirmando que no va a costar menos de 40 unidades monetarias, lo más posible es que cueste 45 y como mucho costará 55 unidades monetarias, definiéndose en el campo de incertidumbre un número borroso triangular, tal como se muestra en la Figura AI.15. Por otro lado. si lo más posible fuera que el precio pudiera encontrarse entre 43 y 45 con el resto de presunciones iguales, se habría definido un número borroso trapezoidal, como se muestra en la Figura AI.16.

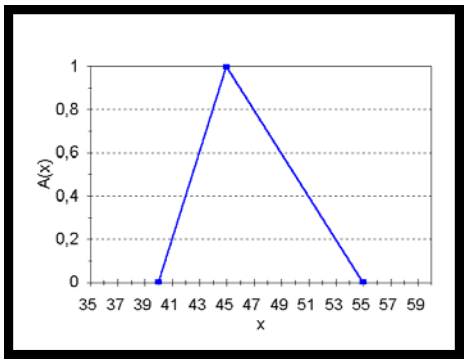


Figura AI.15

En este sentido, en el ámbito de la economía y la gestión de las empresas se estudian problemas cuyas magnitudes se proyectan hacia el futuro, no exigiendo frecuentemente una precisión absoluta, sino un reflejo lo más cercano de la realidad posible. Por este motivo, resulta de gran interés la utilización, en este contexto, de los números borrosos triangulares y trapezoidales.

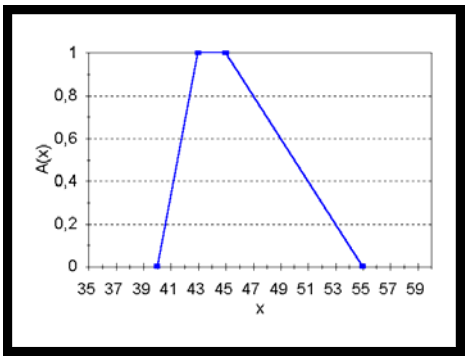


Figura AI.16

AI.2.3.2. Funciones de pertenencia

Los números borrosos triangulares poseen funciones de pertenencia lineales, distinguiéndose tres puntos característicos en el mismo, por lo que de manera ternaria podría representarse como: $A=(a_1,a_2,a_3)$

Y gráficamente como se muestra en la Figura AI.17.

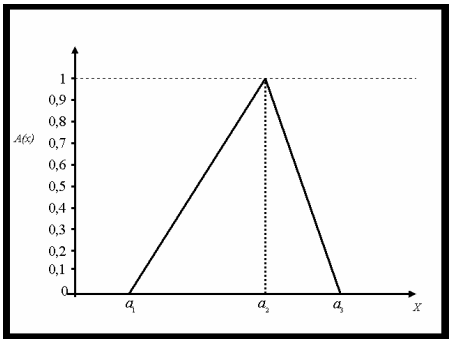


Figura AI.17

Su función de pertenencia viene dada por la siguiente expresión:

$$A(x)=\begin{cases}0 & x\le a_1\\ \dfrac{x-a_1}{a_2-a_1} & a_1<x\le a_2\\ \dfrac{a_3-x}{a_3-a_2} & a_2\le x<a_3\\ 0 & a_3\le x\end{cases}$$

Por otro lado los números borrosos trapezoidales poseen cuatro puntos característicos aunque sus funciones de pertenencia tienen carácter lineal, representándose en forma de cuádrupla como:

$$A = (a_1, a_2, a_3, a_4)$$

y gráficamente como se muestra en la Figura AI.18.

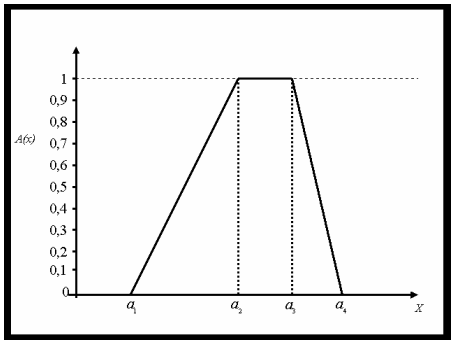


Figura AI.18

Su función de pertenencia viene dada por la siguiente expresión:

$$A(x) = \begin{cases} 0 & x \le a_1 \\ \dfrac{x - a_1}{a_2 - a_1} & a_1 < x < a_2 \\ 1 & a_2 \le x \le a_3 \\ \dfrac{a_4 - x}{a_4 - a_3} & a_3 < x < a_4 \\ 0 & a_4 \le x \end{cases}$$

AI.2.3.3. Operaciones con los números borrosos triangulares y trapezoidales

Dados dos números borrosos triangulares $A = (a_1, a_2, a_3)$ y $B = (b_1, b_2, b_3)$ se define:

- Suma: $A + B = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$

y de acuerdo con la función de pertenencia sería:

$$A(x)+B(x)=\begin{cases}0 & x\le a_1+b_1\\ \dfrac{x-(a_1+b_1)}{(a_2+b_2)-(a_1+b_1)} & a_1+b_1<x\le a_2+b_2\\ \dfrac{(a_3+b_3)-x}{(a_3+b_3)-(a_2+b_2)} & a_2+b_2\le x<a_3+b_3\\ 0 & a_3+b_3\le x\end{cases}$$

- Resta: $A-B=(a_1-b_3,a_2-b_2,a_3-b_1)$

  y de acuerdo con la función de pertenencia sería:

$$A(x)-B(x)=\begin{cases}0 & x\le a_1-b_3\\ \dfrac{x-(a_1-b_3)}{(a_2-b_2)-(a_1-b_3)} & a_1-b_3<x\le a_2-b_2\\ \dfrac{(a_3-b_1)-x}{(a_3-b_1)-(a_2-b_2)} & a_2-b_2\le x<a_3-b_1\\ 0 & a_3-b_1\le x\end{cases}$$

- Multiplicación por un número real:
  $\forall\, k\in\Re$, $k\cdot A=(min(ka_1,ka_3),ka_2,max(ka_1,ka_3))$

- Multiplicación en $\Re^+\cup 0$: $A\cdot B=(a_1\cdot b_1,a_2\cdot b_2,a_3\cdot b_3)$

  y de acuerdo con la función de pertenencia sería:

$$A(x)\cdot B(x)=\begin{cases}0 & x\le a_1\cdot b_1\\ \dfrac{x-(a_1\cdot b_1)}{(a_2\cdot b_2)-(a_1\cdot b_1)} & a_1\cdot b_1<x\le a_2\cdot b_2\\ \dfrac{(a_3\cdot b_3)-x}{(a_3\cdot b_3)-(a_2\cdot b_2)} & a_2\cdot b_2\le x<a_3\cdot b_3\\ 0 & a_3\cdot b_3\le x\end{cases}$$

Por otro, en el caso de números borrosos trapezoidales, si tenemos dos tales que $A=(a_1,a_2,a_3,a_4)$ y $B=(b_1,b_2,b_3,b_4)$ se define:

- Suma: $A+B=(a_1+b_1,a_2+b_2,a_3+b_3,a_4+b_4)$

  y de acuerdo con la función de pertenencia sería:

$$A(x)+B(x)=\begin{cases}0 & x\le a_1+b_1\\ \dfrac{x-(a_1+b_1)}{(a_2+b_2)-(a_1+b_1)} & a_1+b_1<x<a_2+b_2\\ 1 & a_2+b_2\le x\le a_3+b_3\\ \dfrac{(a_4+b_4)-x}{(a_4+b_4)-(a_3+b_3)} & a_3+b_3<x<a_4+b_4\\ 0 & a_4+b_4\le x\end{cases}$$

- Resta: $A-B=(a_1-b_4,a_2-b_3,a_3-b_2,a_4-b_1)$

  y de acuerdo con la función de pertenencia sería:

$$A(x)-B(x)=\begin{cases}0 & x\le a_1-b_4\\ \dfrac{x-(a_1-b_4)}{(a_2-b_3)-(a_1-b_4)} & a_1-b_4<x<a_2-b_3\\ 1 & a_2-b_3\le x\le a_3-b_2\\ \dfrac{(a_4-b_1)-x}{(a_4-b_1)-(a_3-b_2)} & a_3-b_2<x<a_4-b_1\\ 0 & a_4-b_1\le x\end{cases}$$

- Multiplicación por un real:

  $\forall\, k\in\Re$, $k\cdot A=(min(ka_1,ka_4),min(ka_2,ka_3),max(ka_2,ka_3),max(ka_1,ka_4))$

- Multiplicación en $\Re^+\cup 0$: $A\cdot B=(a_1\cdot b_1,a_2\cdot b_2,a_3\cdot b_3,a_4\cdot b_4)$

  y de acuerdo con la función de pertenencia sería:

$$A(x)\cdot B(x)=\begin{cases}0 & x\le a_1\cdot b_1\\ \dfrac{x-(a_1\cdot b_1)}{(a_2\cdot b_2)-(a_1\cdot b_1)} & a_1\cdot b_1<x<a_2\cdot b_2\\ 1 & a_2\cdot b_2\le x\le a_3\cdot b_3\\ \dfrac{(a_4\cdot b_4)-x}{(a_4\cdot b_4)-(a_3\cdot b_3)} & a_3\cdot b_3<x<a_4\cdot b_4\\ 0 & a_4\cdot b_4\le x\end{cases}$$

Las operaciones anteriores para números borrosos triangulares y trapezoidales son las únicas cuyos resultados son también números borrosos triangulares o trapezoidales respectivamente.

AI.2.3.4. Distancia entre números borrosos

Para aquellos problemas en los que se realizan estimaciones inciertas, es importante conocer las distancias que las separan para poder tomar decisiones y establecer prioridades o jerarquías.

En este sentido, para dos números borrosos *A* y *B* con respecto al referencial *E* se define la Distancia de Hamming que los separa como la suma de las distancias a izquierda y derecha a cada nivel de presunción α.

Las distancias a izquierda y derecha se definen como siguen:

$$\forall \alpha \in [0,1] \quad d_I(A,B) = \int_{\alpha=0}^{1} \left| A_{\alpha}^{1} - B_{\alpha}^{1} \right| d\alpha$$

donde A_{α}^{1} es el primer elemento de *E* del que $A(A_{\alpha}^{1}) = \alpha$ y B_{α}^{1} es el respectivo para ese subconjunto borroso *B,* mostrándose su representación gráfica en la Figura AI.19.

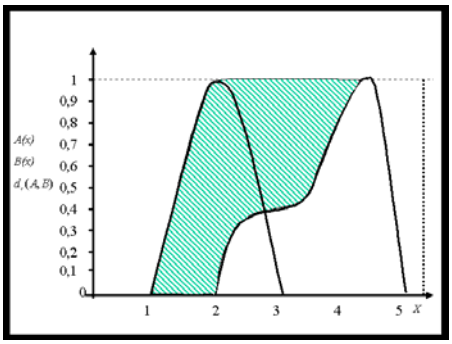


Figura AI.19

Por su parte la distancia a derecha puede representarse como:

$$\forall \alpha \in [0,1] \quad d_D(A,B) = \int_{\alpha=0}^{1} \left| A_{\alpha}^{2} - B_{\alpha}^{2} \right| d\alpha$$

donde A_{α}^{2} es el último elemento de *E* del que $A(A_{\alpha}^{2}) = \alpha$ y B_{α}^{2} es el respectivo para ese subconjunto borroso *B.*

Gráficamente, ambas distancias se representan tal como se muestra en la Figura AI.20.

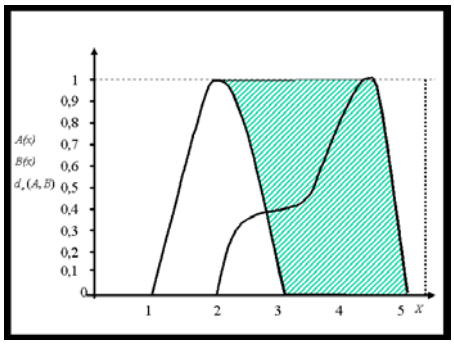


Figura AI.20

De acuerdo con las dos distancias anteriores la distancia total que separa a dos números borrosos se define por la suma de la distancia a izquierda y a derecha de los mismos.

$$d(A,B) = d_I(A,B) + d_D(A,B) =$$
$$= \int_{\alpha=0}^{1} \left(\left| A_\alpha^1 - B_\alpha^1 \right| + \left| A_\alpha^2 - B_\alpha^2 \right| \right) d\alpha$$

AI.2.4. Etiquetas lingüísticas

En muchas aplicaciones se utilizan los subconjuntos borrosos para representar el significado de un concepto determinado. Por ejemplo, se puede definir la expresión "precio alto" como un subconjunto borroso del conjunto de precios. El concepto "competencia fuerte" se puede representar como un subconjunto borroso del conjunto de número de competidores, etc. Un empleo de esta capacidad de representación de los subconjuntos borrosos consiste en ayudar a definir valores lingüísticos.

Si, por ejemplo, *P* es una variable que toma sus valores en el conjunto *X*, la manera normal a través de la cual se puede presentar la información de esta variable es por medio de expresiones como *P es x*, donde *x* es algún valor en el conjunto *X*. En este sentido, Zadeh([5]) sugiere además que se puede extender la idea de que la variable *P* tome algún valor lingüístico.

---

[5] L.A. Zadeh (1975). "The Concept of a Linguistic Variable and itys Aplications to Aproximate reasoning". *Information Sciences* Vol. 8, p. 199-249 and 301-357 and Vol. 9, p. 43-80.

Así si *P* define la variable precio del artículo *M* se puede expresar el mismo de la forma:

P es alto

De este modo, la expresión anterior es un ejemplo de un valor lingüístico, siendo necesario cuando se utilizan las mismas establecer la relación de un subconjunto borroso con el valor de la variable.

Así, si por ejemplo los valores que el precio del artículo *M* puede tomar se encuentran entre 0 y 100, una representación de los subconjuntos borrosos *precio bajo, precio medio* y *precio alto*, podría ser tal como se muestra en la Figura AI.21.

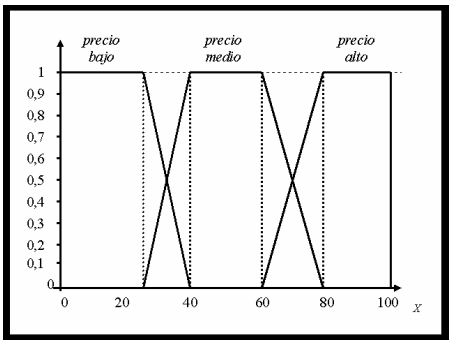


Figura AI.21

Con la utilización de etiquetas lingüísticas aparece un tipo de incertidumbre, denominada imprecisión, que surge de la inexactitud inherente al uso de las etiquetas que sin embargo pueden representar fielmente el razonamiento humano.

En este sentido, es normal que los individuos tengan serios inconvenientes para expresar con valores numéricos exactos sus apreciaciones sobre el comportamiento de determinadas variables. Bajo estas circunstancias, parece más adecuado expresar sus opiniones por medio de valores lingüísticos en vez de valores numéricos exactos, es decir, suponer que el dominio de las variables que intervienen en un problema es un conjunto de términos lingüísticos. Así, en situaciones donde intervienen individuos, los cuales usan más bien expresiones lingüísticas que numéricas para dar sus opiniones, se consigue modelar de manera más directa gran cantidad de problemas reales con la utilización de etiquetas lingüísticas, ya que permite representar la información (casi siempre poco precisa) de manera muy aproximada a como se expresa inicialmente.

En este sentido, el principal problema en el uso de variables lingüística se haya en la determinación del conjunto de etiquetas a utilizar para expresar las opiniones de los individuos. Para ello, se ha de determinar el nivel de distinción al que se quiere expresar la incertidumbre, o lo que es lo mismo la *granularidad* de la incertidumbre del conjunto de etiquetas, y la semántica de las etiquetas, o lo que es lo mismo, qué tipo de funciones de pertenencia utilizar para caracterizar los valores lingüísticos.

El número de etiquetas determinará la granularidad del conocimiento incierto que se pueda expresar. Diferentes estudios han llegado a conclusiones tanto sobre la *cardinalidad* es decir, el número de etiquetas que se ha de entender que sea impar ([6]), así como sobre el límite de granularidad que se entiende no debe ser mayor de 11 o 13 etiquetas ([7]).

Normalmente, la semántica de las etiquetas se da mediante números difusos definidos sobre el intervalo unidad [0,1], los cuales se describen utilizando funciones de pertenencia. Dado que las etiquetas son aproximaciones de expresiones lingüísticas propias de los individuos, se considerará que las funciones de pertenencia trapezoidales lineales son suficientemente buenas para recoger la imprecisión de las expresiones humanas, ya que conseguir valores más exactos puede ser una tarea imposible e innecesaria. Esta representación establece una cuádrupla $(a_i, b_i, \alpha_i, \beta_i)$, siendo los dos primeros parámetros el intervalo en el cual la función de pertenencia toma el valor 1, y el tercero y el cuarto la amplitud a la izquierda y la derecha, respectivamente. Así, en el ejemplo anterior de los precios las cuádruplas que definen las tres etiquetas lingüísticas serían:

$$precio\ bajo = (0,28,0,6);\ \ precio\ medio = (40,60,6,10);\ \ precio\ alto = (80,100,10,0)$$

6 R. Beyth-Marom (1982). "How probable is probable?. A Numerical Taxonomy Traslation of Verbal Probability Expressions"*, Journal of Forecasting*. Vol. 1, p. 257-269.

7 P.P. Bonissone y K.S. Decker (1986). "Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity", in: L.H. Kanal y J.F. Lemmer, Eds., *Uncertainty in Artificial Intelligence* (North-Holland), p. 217-247.

ANEXO 2. EL ENTORNO XFUZZY 3.0 DE DISEÑO DE SISTEMAS BORROSOS

A2.1. INTRODUCCIÓN

A medida que la aplicación de los sistemas borrosos se ha ido generalizando, la necesidad de disponer de herramientas informáticas que faciliten su desarrollo se ha incrementado notablemente. En respuesta a esta creciente necesidad se han desarrollado numerosas herramientas comerciales específicas([8]) y se han incorporado módulos borrosos a herramientas comerciales de propósito general([9]). Respecto al software de libre distribución, algunas universidades y centros de investigación han desarrollado herramientas específicas([10]), aunque para estos centros resulta complicado el mantenimiento y actualización de estas herramientas.

En este apartado se presentará el entorno de desarrollo de sistemas borrosos Xfuzzy 3.0. creado por miembros del Instituto de Microelectrónica de Sevilla ([11]) disponible bajo licencia pública GNU en la dirección http://www.imse.cnm.es/Xfuzzy/.

---

8 Página principal de FuzzyTech: http://www.fuzzytech.corn/
Página principal de FIDE: http://www.aptronix.corn/fide/
Página principal de TILShell: http://www.ortech-engr.corn/fuzzy/ TilShell.htrnl

9 Página principal de Fuzzy Logic Toolbox de MatLab:
http://www.rnathworks.corn/products/ fuzzylogic/

10 R. Hartwig, C. Labinsky, S. Nordhoff, B. Landorff, P. Jensch, J. Schwanke (1996) "Free Fuzzy Logic System Design Tool: FOOL", Proc. 4th European Congress on Intelligent Techniques and Soft Computing (EUFIT'96), p. 2274-2277, Aachen.
O.G. Duarte, G. Pérez (1999). "UNFUZZY: Fuzzy Logic System Analysis, Design, Simulation and Implementation Software", Proc. 1999 Eusftat-Estylf Joint Com., p. 251-254, Mallorca.
B. Charnomordic, P. Glorennec y S. Guillaume (2002). “An open source portable sofware for fuzzy inference systems”. Proc. XI Congreso Español sobre Tecnologías y Lógica Fuzzy, pp. 349-351, León.

11 D.R. López, F.J. Moreno Velo, A. Barriga, S. Sánchez Solano (1997). "XFL: A Language for the Definition of Fuzzy Systems", Proc. 6th IEEE Int. Como on Fuzzy Systems (FUZZIEEE'97), p. 1585-1591, Barcelona.
D.R. López, C.J. Jiménez, l. Baturone, A. Barriga, S. Sánchez Solano (1998). "Xfuzzy: A Design Environment for Fuzzy Systems", Proc. 7th IEEE Int. Conf. on Fuzzy Systerns (FUZZIEEE'98), p. 1060-1065, Anchorage.

Dicho entorno de desarrollo de sistemas borrosos proporciona un conjunto de herramientas que cubren en su mayor parte las diferentes etapas y aspectos relacionados con el diseño de sistemas de inferencia difusa. Estas herramientas en su conjunto desarrollan una interfaz gráfica homogénea que permite diseñar sistemas borrosos complejos por medio de un lenguaje de descripción formal que admite bases de reglas jerárquicas y modificadores lingüísticos, conectivas difusas, funciones de pertenencia y métodos de clarificación extensibles por el usuario.

Xfuzzy 3.0 es un entorno de desarrollo para sistemas de inferencia basados en lógica borrosa. Está formado por varias herramientas que cubren las diferentes etapas del proceso de diseño de sistemas borrosos, desde su descripción inicial hasta la implementación final. Sus principales características son la capacidad para desarrollar sistemas complejos y la flexibilidad para permitir al usuario extender el conjunto de funciones disponibles. El entorno ha sido completamente programado en Java, de forma que puede ser ejecutado sobre cualquier plataforma que tenga instalado el JRE (*Java Runtime Environment*). La Figura A2.1 muestra el flujo de diseño de Xfuzzy 3.0.

La etapa de descripción incluye herramientas gráficas para la definición del sistema borroso. La etapa de verificación está compuesta por herramientas de simulación, monitorización y representación gráfica del comportamiento del sistema. La etapa de ajuste facilita la aplicación de algoritmos de aprendizaje. Finalmente, la etapa de síntesis incluye herramientas para generar descripciones en lenguajes de alto nivel para implementaciones software o hardware.

El nexo entre todas las herramientas es el uso de un lenguaje de especificación común, XFL3, que permite expresar relaciones muy complejas entre variables borrosas por medio de bases de reglas jerárquicas y conectivas, modificadores lingüísticos, funciones de pertenencia y métodos de clarificación definidos por el usuario.

F.J. Moreno Velo, S. Sánchez Solano, A. Barriga, l. Baturone, D.R. López (2001). "XFL3: An Specification Language for Fuzzy Systems", Mathware & Soft Computing, Vol. VIII, n. 3, p. 239-253.

F. J. Moreno Velo, I. Baturone, S. Sánchez-Solano, A. Barriga, R. Senhadj (2002).”El entorno Xfuzzy 3.0 de diseño de sistemas borrosos”. Proc. XI Congreso Español sobre Tecnologías y Lógica Fuzzy, p. 353-358, León.

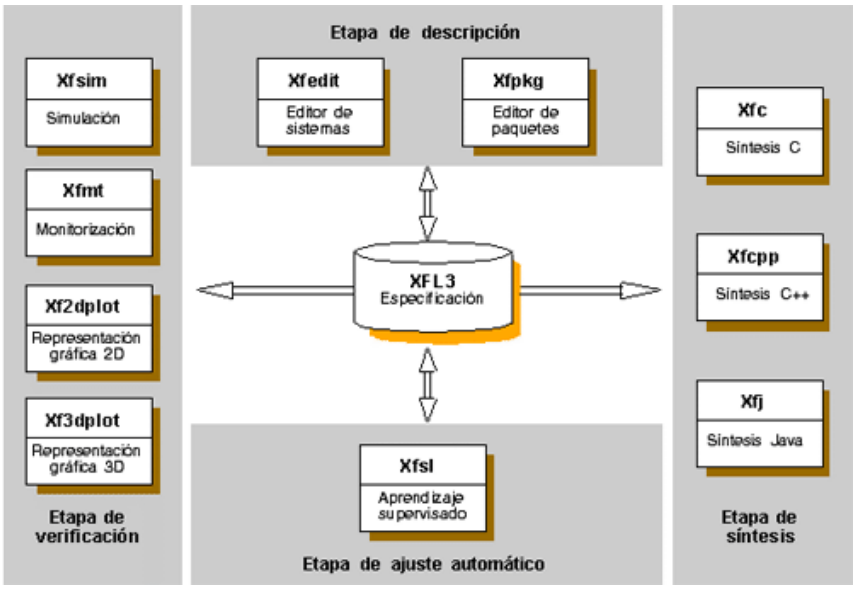


Figura A2.1.

Las diferentes herramientas pueden ser ejecutadas de forma independiente o desde la ventana principal del entorno (veáse la Figura A2.2). Esta ventana permite trabajar con varios sistemas simultáneamente, contiene enlaces a las diferentes herramientas y permite acceder a la ayuda del entorno, esto es, el entorno integra a todas ellas bajo una interfaz gráfica de usuario que facilita el proceso de diseño.

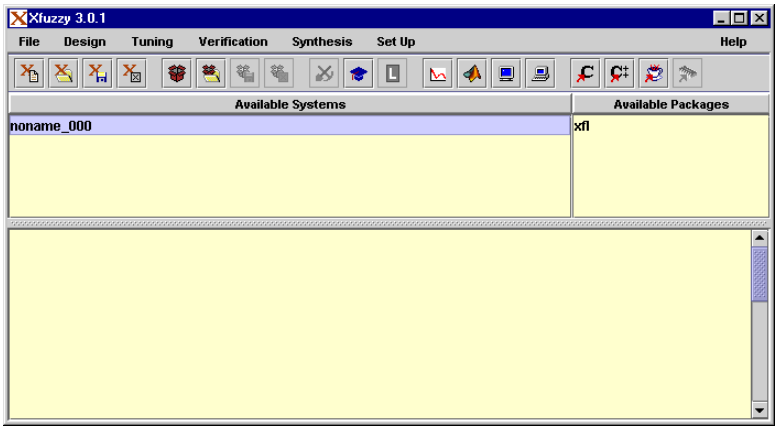


Figura A2.2

La barra de menús en la ventana principal contiene los enlaces a las diferentes herramientas. Bajo la barra de menús se sitúa una barra de botones con las opciones más utilizadas. La zona central de la ventana muestra dos listas. La primera es la lista de sistemas cargados (el entorno puede trabajar con varios sistemas simultáneamente). La segunda lista contiene los paquetes cargados. El resto de la ventana principal está ocupado por un área de mensajes.

La barra de menús está dividida en las diferentes etapas del desarrollo de un sistema. El menú *File* permite crear (*create*), cargar (*load*), salvar (*save*) y cerrar (*close*) un sistema borroso. Este menú contiene también las opciones para crear, cargar, salvar y cerrar un paquete de funciones. El menú termina con la opción para salir del entorno. El menú *Design* se utiliza para editar el sistema borroso seleccionado (*xfedit*) o el paquete de funciones (*package*) seleccionado (*xfpkg*). El menú *Tuning* contiene los enlaces a la herramienta de adquisición de conocimiento (bajo desarrollo), la herramienta de aprendizaje supervisado (*xfsl*) y la herramienta de aprendizaje por refuerzo (bajo desarrollo). El menú *Verification* permite representar el comportamiento del sistema mediante una gráfica bidimensional (xf2dplot) o tridimensional (*xf3dplot)*, monitorizar el sistema (*xfmt*) y simularlo (*xfsim*). El menú *Synthesis* está dividido en dos partes: la síntesis software, que genera descripciones del sistema en C (*xfc*), C++ (*xfcpp*), y Java (*xfj*); y la síntesis hardware que implementa la descripción de un sistema mediante un circuito borroso (bajo desarrollo). El menú *Set Up* se utiliza para modificar el directorio de trabajo del entorno, salvar los mensajes del entorno en un fichero de log externo, cerrar el fichero de log, limpiar el área de mensajes de la ventana principal y cambiar la apariencia (*look & feel*) del entorno.

Muchas opciones de la barra de menús sólo están activas cuando se selecciona un sistema borroso. Para seleccionar un sistema borroso basta con pulsar sobre su nombre en la lista de sistemas. Una doble pulsación sobre el nombre abrirá la herramienta de edición. El mismo resultado se obtiene presionando la tecla *Enter* una vez que el sistema ha sido seleccionado. La tecla *Insert* creará un nuevo sistema y la tecla *Delete* se utiliza para cerrar el sistema. Estos aceleradores son comunes a todas las listas del entorno: *Insert* se utiliza para insertar un nuevo elemento a la lista; *Enter* o una doble pulsación editará el elemento seleccionado; y *Delete* quitará el elemento de la lista.

Los siguientes apartados describen brevemente las características de Xfuzzy 3.0.

A2.2. *XFL3*: EL LENGUAJE DE ESPECIFICACIÓN DE XFUZZY 3.0

La definición de lenguajes formales para la especificación de sistema borrosos presenta varias ventajas. Sin embargo, pueden plantearse dos objetivos contradictorios. Por una parte es deseable disponer de un lenguaje genérico y altamente expresivo, capaz de aplicar todos los formalismos basados en lógica borrosa, pero, al mismo tiempo, las (posibles) restricciones impuestas por la implementación final del sistema deben ser consideradas. En este sentido, algunos lenguajes están enfocados hacia la expresividad, mientras otros están enfocados hacia las implementaciones software o hardware.

XFL3 es un lenguaje de especificación formal que permite describir sistemas borrosos [9]. Sus principales características son la separación de la definición de la estructura del sistema respecto a la descripción matemática de los operadores empleados en cada sistema, la capacidad de definir sistemas con bases de reglas jerárquicas, la capacidad de expresar relaciones complejas entre las variables por medio de diversos modificadores lingüísticos y conectivas difusas, y la posibilidad de ampliar el conjunto de funciones disponibles como conectivas difusas, modificadores lingüísticos, funciones de pertenencia y métodos de corrección. En concreto, XFL3 permite al usuario definir funciones de pertenencia y operadores paramétricos y admite el uso de modificadores lingüísticos que permiten describir relaciones más complejas entre las variables. Además, el lenguaje XFL3, así como las herramientas basadas en él, emplean Java como lenguaje de programación. Esto significa el uso de una ventajosa metodología orientada a objetos y flexibilidad para ejecutar la nueva versión de Xfuzzy en cualquier plataforma que tenga instalado JRE (*Java Runtime Environment*).

XFL3 divide la descripción de un sistema borroso en dos partes: la definición lógica de la estructura del sistema, que es incluida en ficheros con extensión "*.xfl*", y la definición matemática de las funciones borrosas, que son incluidas en ficheros con extensión "*.pkg*" (*packages*).

El lenguaje permite la definición de sistemas complejos. XFL3 no limita el número de variables lingüísticas, funciones de pertenencia, reglas borrosas, etc. Los sistemas pueden ser definidos mediante bases de reglas

jerárquicas y las bases de reglas pueden expresar relaciones complejas entre las variables lingüísticas usando las conectivas AND y OR y modificadores lingüísticos como mayor que, más pequeño que, distinto a, etc. XFL3 permite al usuario definir sus propias funciones borrosas por medio de paquetes (*packages*). Estas nuevas funciones pueden ser usadas como funciones de pertenencia, conectivas borrosas, modificadores lingüísticos y métodos de clarificación. El paquete estándar *xfl* contiene las funciones más habituales.

La descripción de la estructura de un sistema borroso, incluida en ficheros "*.xfl*", emplea una sintaxis formal basada en 8 palabras reservadas: *import*, *operatorset*, *type*, *extends*, *rulebase*, *using*, *if* y *system*. Una especificación XFL3 contiene varios objetos que definen conjuntos de operadores, tipos de variables, bases de reglas y la descripción del comportamiento global del sistema. Un conjunto de operadores (*operator set*) describe la selección de las funciones asignadas a los diferentes operadores borrosos. Un tipo de variable contiene la definición del universo de discurso, las etiquetas lingüísticas y las funciones de pertenencia relacionadas con una variable lingüística. Una base de reglas define las relaciones lógicas entre las variables lingüísticas. Por último, el comportamiento global del sistema incluye la descripción de la jerarquía de bases de reglas.

A2.2.1. Conjuntos de operadores

Un conjunto de operadores (*operator set*) en XFL3 es un objeto que contiene las funciones matemáticas asignadas a cada operador borroso. Los operadores borrosos pueden ser binarios (como las T-normas y S-normas empleadas para representar conectivas entre variables lingüísticas, implicaciones o agregaciones de reglas), unarios (como las C-normas y los operadores relacionados con los modificadores lingüísticos), o pueden estar asociados con métodos de clarificación.

XFL3 define los conjuntos de operadores mediante el siguiente formato:

```
operatorset identifier {
  operator assigned_function(parameter_list);
  operator assigned_function(parameter_list);
  ..... }
```

No es necesario especificar todos los operadores. Cuando uno de ellos no está definido, su valor por defecto es asumido. La siguiente tabla muestra los operadores (y sus funciones por defecto) actualmente usados en XFL3.

Operador	**Tipo**	**Función por defecto**
and	binary	min(a,b)
or	binary	max(a,b)
implication, imp	binary	min(a,b)
also	binary	max(a,b)
not	unary	(1-a)
very, strongly	unary	a^2
moreorless	unary	(a)^(1/2)
slightly	unary	4*a*(1-a)
defuzzification, defuz	defuzzification	center of area

Las funciones asignadas son definidas en ficheros externos a los que se denominan paquetes (*packages*). El formato para identificar una función es "*package.function*". El nombre del paquete, "*xfl*" en el siguiente ejemplo, puede ser eliminado si el paquete ha sido importado previamente (usando el comando "*import package;*").

```
operatorset systemop {
 and xfl.min();
 or xfl.max();
 imp xfl.min();
 strongly xfl.pow(3);
 moreorless xfl.pow(0.4);
}
```

A2.2.2. Tipos de variables lingüísticas

Un tipo XFL3 es un objeto que describe un tipo de variable lingüística. Esto significa definir su universo de discurso, dar nombre a las etiquetas lingüísticas que cubren dicho universo y especificar las funciones de pertenencia asociadas a cada etiqueta. El formato de definición de un tipo es el siguiente:

```
type identifier [min, max; card] {
  label membership_function(parameter_list);
  label membership_function(parameter_list);
  ..... }
```

donde *min* y *max* son los límites del universo de discurso y *card* (cardinalidad) es su número de elementos discretos. Si la cardinalidad no es especificada se asume su valor por defecto (actualmente, 256). Cuando no se definen explícitamente los límites, el universo de discurso es considerado entre 0 a 1.

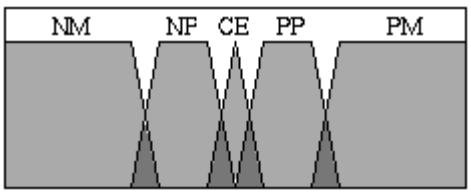
El formato del identificador ("*identifier*") de una etiqueta lingüística es similar al del identificador de un operador, es decir, "*package.function*" o simplemente "*function*" si el paquete donde el usuario ha definido las funciones de pertenencia ha sido importado previamente.

XFL3 soporta mecanismos de herencia en las definiciones de tipos (como su precursor XFL). La cabecera de la definición utilizada para expresar herencia es como sigue:

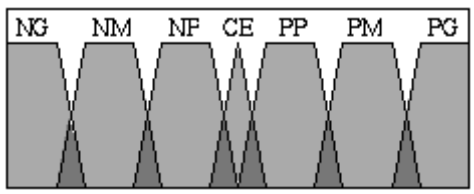
type identifier extends identifier {

Los tipos definidos de esta manera heredan automáticamente el universo de discurso y las etiquetas de sus padres. Las etiquetas definidas en el cuerpo de la definición del tipo son añadidas a las etiquetas de sus padres o sobrescriben a éstas si tienen los mismos nombres.

```
type Tinput1 [-90,90] {
 NM xfl.trapezoid(-100,-90,-40,-30);
 NP xfl.trapezoid(-40,-30,-10,0);
 CE xfl.triangle(-10,0,10);
 PP xfl.trapezoid(0,10,30,40);
 PM xfl.trapezoid(30,40,90,100);
}
```




```
type Tinput2 extends Tinput1 {
 NG xfl.trapezoid(-100,-90,-70,-60);
 NM xfl.trapezoid(-70,-60,-40,-30);
 PM xfl.trapezoid(30,40,60,70);
 PG xfl.trapezoid(60,70,90,100);
}
```



A2.2.3. Bases de reglas

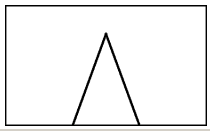
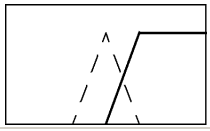
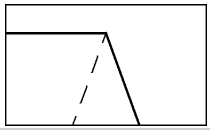
Una base de reglas en XFL3 es un objeto que contiene las reglas que definen las relaciones lógicas entre las variables lingüísticas. Su formato de definición es el siguiente:

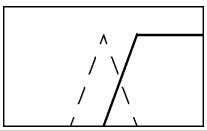
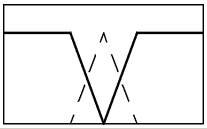
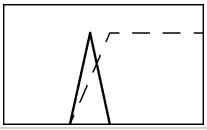
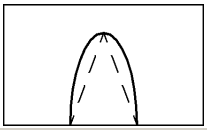
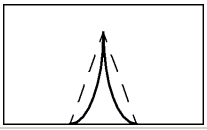
```
rulebase identifier (input_list : output_list) using operatorset {
  [factor] if (antecedent) -> consecuent_list;
  [factor] if (antecedent) -> consecuent_list;
  ..... }
```

El formato de definición de las variables de entrada y salida es "*type identifier*", donde *type* hace referencia a uno de los tipos de variables lingüísticas previamente definidas. La selección del conjunto de operadores es opcional, de forma que cuando no es definido explícitamente se emplean los operadores por defecto. Es posible aplicar a las reglas pesos o factores de confidencia (con valor por defecto de 1).

El antecedente de una regla describe la relación entre las variables de entrada. XFL3 permite expresar antecedentes complejos combinando proposiciones básicas mediante conectivas y modificadores lingüísticos. Por otra parte, cada consecuente de una regla describe la asignación de un valor lingüístico a una variable de salida como "*variable = label*".

Una proposición básica relaciona una variable de entrada con una de sus etiquetas lingüísticas. XFL3 admite diferentes relaciones como igualdad, desigualdad y varios modificadores lingüísticos. La siguiente tabla muestra las diferentes relaciones ofrecidas por XFL3.

Proposiciones básicas	Descripción	Representación
variable == label	equal to	
variable >= label	equal or greater than	
variable <= label	equal or smaller than	

variable > label	greater than	
variable < label	smaller than	
variable != label	not equal to	
variable %= label	slightly equal to	
variable ~= label	moreorless equal to	
variable += label	strongly equal to	

En general, el antecedente de una regla está formado por una proposición compleja. Las proposiciones complejas están compuestas de varias proposiciones básicas conectadas mediante conectivas borrosas y modificadores lingüísticos. La siguiente tabla muestra cómo generar proposiciones complejas en XFL3.

Proposiciones complejas	**Descripción**
proposition & proposition	and operator
proposition | proposition	or operator
!proposition	not operator
%proposition	slightly operator
~proposition	moreorless operator
+proposition	strongly operator

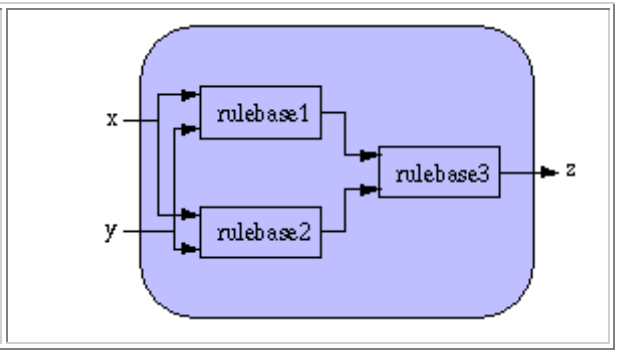
A2.2.4. Comportamiento global del sistema

La descripción del comportamiento global del sistema requiere definir las variables globales de entrada y salida del sistema, así como la jerarquía de bases de reglas. Esta descripción en XFL3 es como sigue:

```
system (input_list : output_list) {
 rule_base_identifier(inputs : outputs);
 rule_base_identifier(inputs : outputs);
..... }
```

El formato de definición de las variables de entrada y salida globales es el mismo que el empleado en la definición de las bases de reglas. Las variables internas que pueden aparecer establecen interconexiones en serie o en paralelo entre las bases de reglas. Las variables internas deben aparecer como variables de salida de una base de reglas antes de ser empleadas como variables de entrada de otras bases de reglas.

```
system (Type1 x, Type2 y :
Type3 z) {
rulebase1( x, y : inner1);
rulebase2( x, y : inner2);
rulebase3(inner1, inner2 : z);
}
```



A2.2.5. Paquetes de funciones

Una de las grandes ventajas de XFL3 es que las funciones asignadas a los operadores borrosos pueden ser definidas libremente por el usuario en ficheros externos (denominados paquetes o "*packages*"), lo que proporciona una enorme flexibilidad al entorno. Cada package puede incluir un número ilimitado de definiciones.

En XFL3 pueden definirse cuatro tipos de funciones: funciones binarias que pueden ser usadas como T-normas, S-normas y funciones de implicación; funciones unarias que están relacionadas con los modificadores lingüísticos; funciones de pertenencia que son usadas para describir etiquetas lingüísticas; y métodos de clarificación.

Una definición de función incluye su nombre (y posibles alias), los parámetros que definen su comportamiento junto con las restricciones de estos parámetros, la descripción de su comportamiento en los diferentes lenguajes en los que puede ser compilado (C, C++ y Java) e, incluso, la descripción de las derivadas de la función (si va a ser utilizada con mecanismos de aprendizaje basados en gradiente). Esta información es la base para generar automáticamente una clase Java que incorpora todas las capacidades de la función y puede ser empleada por cualquier especificación XF3.

A2.2.5.1. Definición de funciones binarias

Las funciones binarias pueden ser asignadas al operador de conjunción (and), al operador de disyunción (or), a la función de implicación (imp) y al operador de agregación de reglas (also). La estructura de una definición de función binaria en un paquete de funciones es como sigue:

binary identifier { blocks }

Los bloques que pueden aparecer en la definición de una función binaria son *alias*, *parameter*, *requires*, *java*, *ansi_c*, *cplusplus*, *derivative* y *source*.

El bloque alias se utiliza para definir nombres alternativos para identificar a la función. Cualquiera de esos identificadores puede ser usado para hacer referencia a la función. La sintaxis del bloque alias es:

alias identifier, identifier, ... ;

El bloque *parameter* permite la definición de los parámetros de los que depende la función. Su formato es:

parameter identifier, identifier, ... ;

El bloque *requires* expresa las restricciones sobre los valores de los parámetros por medio de una expresión Booleana en Java que valida los valores de los parámetros. La estructura de este bloque es:

requires { expression }

Los bloques *java*, *ansi_c* and *cplusplus* describen el comportamiento de la función por medio de su descripción como el cuerpo de una función

en los lenguajes de programación Java, C y C++, respectivamente. Las variables de entrada para estas funciones son '*a*' y '*b*'. El formato de estos bloques es el siguiente:

java { Java_function_body }
ansi_c { C_function_body }
cplusplus { C++_function_body }

El bloque *derivative* describe la derivada de la función con respecto a las variables de entrada '*a*' y '*b*'. Esta descripción consiste en una expresión Java de asignamiento a la variable '*deriv*[]'. La derivada de la función con respecto a la variable de entrada '*a*' debe ser asignada a '*deriv*[0]', mientras que la derivada de la función con respecto a la variable de entrada '*b*' debe ser asignada a '*deriv*[1]'. La descripción de la derivada de la función permite propagar la derivada de la función de error del sistema utilizada por los algoritmos de aprendizaje supervisado basados en gradiente descendente. El formato es:

derivative { Java_expressions }

El bloque *source* es utilizado para definir código Java que es directamente incluido en el código de la clase generada para la definición de la función. Este código permite definir métodos locales que pueden ser empleados dentro de otros bloques. La estructura es:

source { Java_code }

El siguiente ejemplo muestra la definición de la T-norma mínimo, también usada como función de implicación de Mamdani.

```
binary min {
 alias mamdani;
 java { return (a<b? a : b); }
 ansi_c { return (a<b? a : b); }
 cplusplus { return (a<b? a : b); }
 derivative {
  deriv[0] = (a<b? 1: (a==b? 0.5 : 0));
  deriv[1] = (a>b? 1: (a==b? 0.5 : 0));
 }
}
```

A2.2.5.2. Definición de funciones unarias

Las funciones unarias son usadas para describir modificadores lingüísticos. Estas funciones pueden ser asignadas a los modificadores no (*not*), fuertemente (*strongly*), más o menos (*more-or-less*) y ligeramente (*slightly*). La estructura de la definición de una función unaria es como sigue:

unary identifier { blocks }

Los bloques que pueden aparecer en la definición de una función unaria son *alias*, *parameter*, *requires*, *java*, *ansi_c*, *cplusplus*, *derivative* y *source*.

El bloque *alias* se utiliza para definir nombres alternativos para identificar a la función. Cualquiera de esos identificadores puede ser usado para hacer referencia a la función. La sintaxis del bloque *alias* es:

alias identifier, identifier, ... ;

El bloque *parameter* permite la definición de los parámetros de los que depende la función. Su formato es:

parameter identifier, identifier, ... ;

El bloque *requires* expresa las restricciones sobre los valores de los parámetros por medio de una expresión Booleana en Java que valida los valores de los parámetros. La estructura de este bloque es:

requires { expression }

Los bloques *java*, *ansi_c* and *cplusplus* describen el comportamiento de la función por medio de su descripción como el cuerpo de una función en los lenguajes de programación Java, C y C++, respectivamente. La variable de entrada para estas funciones es '*a*'. El formato de estos bloques es el siguiente:

java { Java_function_body }
ansi_c { C_function_body }
cplusplus { C++_function_body }

El bloque *derivative* describe la derivada de la función con respecto a la variable de entrada '*a*'. Esta descripción consiste en una expresión Java de asignamiento a la variable '*deriv*'. La descripción de la derivada de la

función permite propagar la derivada de la función de error del sistema utilizada por los algoritmos de aprendizaje supervisado basados en gradiente descendente. El formato es:

derivative { Java_expressions }

El bloque *source* es utilizado para definir código Java que es directamente incluido en el código de la clase generada para la definición de la función. Este código permite definir métodos locales que pueden ser empleados dentro de otros bloques. La estructura es:

source { Java_code }

El siguiente ejemplo muestra la definición de la C-norma de Yager, que depende del parámetro w.

```
unary yager {
 parameter w;
 requires { w>0 }
 java { return Math.pow( ( 1 - Math.pow(a,w) ) , 1/w ); }
 ansi_c { return pow( ( 1 - pow(a,w) ) , 1/w ); }
 cplusplus { return pow( ( 1 - pow(a,w) ) , 1/w ); }
 derivative { deriv = - Math.pow( Math.pow(a,-w) -1, (1-w)/w ); }
}
```

A2.2.5.3. Definición de funciones de pertenencia

Las funciones de pertenencia son asignadas a las etiquetas lingüísticas que forman un tipo de variable lingüística. La estructura de una definición de función de pertenencia en un paquete de funciones es como sigue:

mf identifier { blocks }

Los bloques que pueden aparecer en la definición de una función de pertenencia son *alias*, *parameter*, *requires*, *java*, *ansi_c*, *cplusplus*, *derivative* y *source*.

El bloque *alias* se utiliza para definir nombres alternativos para identificar a la función. Cualquiera de esos identificadores puede ser usado para hacer referencia a la función. La sintaxis del bloque *alias* es:

alias identifier, identifier, ... ;

El bloque *parameter* permite la definición de los parámetros de los que depende la función. Su formato es:

parameter identifier, identifier, ... ;

El bloque *requires* expresa las restricciones sobre los valores de los parámetros por medio de una expresión Booleana en Java que valida los valores de los parámetros. Esta expresión puede usar también los valores de las variables '*min*' y '*max*', que representan los valores mínimo y máximo del universo de discurso de la variable lingüística considerada. La estructura de este bloque es:

requires { expression }

Los bloques *java*, *ansi_c* y *cplusplus* describen el comportamiento de la función por medio de su descripción como el cuerpo de una función en los lenguajes de programación Java, C y C++, respectivamente. El formato de estos bloques es el siguiente:

java { Java_function_body }
ansi_c { C_function_body }
cplusplus { C++_function_body }

La definición de una función de pertenencia incluye no sólo la descripción del comportamiento de la función en sí misma, sino también del comportamiento de la función bajo la acción de los modificadores *greater-or-equal* y *smaller-or-equal*, así como el cálculo de los valores del centro y la base de la función de pertenencia. Como consecuencia, los bloques *java*, *ansi_c* y *cplusplus* se dividen en los siguientes subbloques:

equal { code }
greatereq { code }
smallereq { code }
center { code }
basis { code }

El subbloque *equal* describe el comportamiento de la función. Los subbloques *greatereq* y *smallereq* describen la acción de los modificadores greater-or-equal y smaller-or-equal respectivamente. La variable de entrada en estos subbloques se denomina '*x*'. El código puede usar los valores de los parámetros de la función, así como las variables '*min*' y '*max*', que

representan los valores mínimo y máximo del universo de discurso de la función. Los subbloques *greatereq* y *smallereq* pueden ser omitidos. En ese caso las transformaciones correspondientes son calculadas recorriendo todos los valores del universo de discurso. Sin embargo, resulta mucho más eficiente usar la función analítica, por lo que la definición de estos subbloques está fuertemente recomendada.

Los subbloques *center* y *basis* describen el centro y la base de la función de pertenencia. El código de estos subbloques puede usar los valores de los parámetros de la función y las variables '*min*' y '*max*'. Esta información es usada por varios métodos de clarificación simplificados. Estos subbloques son opcionales y su función por defecto devuelve un valor nulo.

El bloque *derivative* describe la derivada de la función con respecto a cada parámetro. Este bloque es también dividido en los subbloques *equal*, *greatereq*, *smallereq*, *center* y *basis*. El código de estos subbloques consiste en expresiones Java que asignan valores a la variable '*deriv*[]'. El valor de '*deriv*[i]' representa la derivada de la función con respecto al i-ésimo parámetro de la función de pertenencia. La descripción de la derivada de la función permite calcular la derivada de la función de error del sistema utilizada por los algoritmos de aprendizaje basados en gradiente descendente. El formato es:

derivative { subblocks }

El bloque *source* es utilizado para definir código Java que es directamente incluido en el código de la clase generada para la definición de la función. Este código permite definir métodos locales que pueden ser empleados dentro de otros bloques. La estructura es:

source { Java_code }

El siguiente ejemplo muestra la definición de una función de pertenencia en forma de campana.

```
mf bell {
 parameter a, b;
 requires { a>=min && a<=max && b>0 }
 java {
  equal { return Math.exp( -(a-x)*(a-x)/(b*b) ); }
  greatereq { if(x>a) return 1; return Math.exp( - (x-a)*(x-a)/(b*b) ); }
  smallereq { if(x<a) return 1; return Math.exp( - (x-a)*(x-a)/(b*b) ); }
  center { return a; }
  basis { return b; }
 }
 ansi_c {
  equal { return exp( -(a-x)*(a-x)/(b*b) ); }
  greatereq { if(x>a) return 1; return exp( - (x-a)*(x-a)/(b*b) ); }
  smallereq { if(x<a) return 1; return exp( - (x-a)*(x-a)/(b*b) ); }
  center { return a; }
  basis { return b; }
 }
 cplusplus {
  equal { return exp( -(a-x)*(a-x)/(b*b) ); }
  greatereq { if(x>a) return 1; return exp( - (x-a)*(x-a)/(b*b) ); }
  smallereq { if(x<a) return 1; return exp( - (x-a)*(x-a)/(b*b) ); }
  center { return a; }
  basis { return b; }
 }
 derivative {
  equal {
  double aux = (x-a)/b;
   deriv[0] = 2*aux*Math.exp(-aux*aux)/b;
   deriv[1] = 2*aux*aux*Math.exp(-aux*aux)/b;
  }
  greatereq {
   if(x>a) { deriv[0] = 0; deriv[1] = 0; }
   else {
    double aux = (x-a)/b;
    deriv[0] = 2*aux*Math.exp(-aux*aux)/b;
    deriv[1] = 2*aux*aux*Math.exp(-aux*aux)/b;
   }
  }
  smallereq {
   if(x<a) { deriv[0] = 0; deriv[1] = 0; }
   else {
    double aux = (x-a)/b;
    deriv[0] = 2*aux*Math.exp(-aux*aux)/b;
    deriv[1] = 2*aux*aux*Math.exp(-aux*aux)/b;
   }
  }
  center { deriv[0] = 1; deriv[1] = 0; }
  basis { deriv[0] = 0; deriv[1] = 1; }
 }
}
```

A2.2.5.4. Definición de métodos de clarificación

Los métodos de clarificación obtienen el valor representativo de un conjunto borroso. Estos métodos son utilizados en la etapa final del proceso de inferencia borroso cuando no es posible trabajar con conclusiones borrosas. La estructura de una definición de método de clarificación en un paquete de funciones es como sigue:

defuz identifier { blocks }

Los bloques que pueden aparecer en una definición de método de clarificación son *alias*, *parameter*, *requires*, *definedfor*, *java*, *ansi_c*, *cplusplus* y *source*.

El bloque *alias* se utiliza para definir nombres alternativos para identificar al método. Cualquiera de esos identificadores puede ser usado para hacer referencia al método. La sintaxis del bloque *alias* es:

alias identifier, identifier, ... ;

El bloque *parameter* permite la definición de los parámetros de los que depende el método. Su formato es:

parameter identifier, identifier, ... ;

El bloque *requires* expresa las restricciones sobre los valores de los parámetros por medio de una expresión Booleana en Java que valida los valores de los parámetros. La estructura de este bloque es:

requires { expression }

El bloque *definedfor* se utiliza para enumerar los tipos de funciones de pertenencia que el método puede usar como conclusiones parciales. Este bloque ha sido incluido porque algunos métodos de clarificación simplificados solamente trabajan con ciertas funciones de pertenencia. Este bloque es opcional. Por defecto, se asume que el método puede trabajar con todas las funciones de pertenencia. La estructura del bloque es:

definedfor identificador, identificador, ... ;

El bloque *source* es utilizado para definir código Java que es directamente incluido en el código de la clase generada para la definición del método. Este código permite definir funciones locales que pueden ser empleadas dentro de otros bloques. La estructura es:

source { Java_code }

Los bloques *java*, *ansi_c* y *cplusplus* describen el comportamiento del método por medio de su descripción como el cuerpo de una función en los lenguajes de programación Java, C y C++, respectivamente. El formato de estos bloques es el siguiente:

java { Java_function_body }
ansi_c { C_function_body }
cplusplus { C++_function_body }

La variable de entrada para estas funciones es el objeto '*mf*', que encapsula al conjunto borroso obtenido como conclusión del proceso de inferencia. El código puede usar el valor de las variables '*min*', '*max*' y '*step*', que representan respectivamente el mínimo, el máximo y la división del universo de discurso del conjunto borroso. Los métodos de clarificación convencionales se basan en recorrer todos los valores del universo de discurso calculando el grado de pertenencia para cada valor del universo. Por otra parte, los métodos de clarificación simplificados suelen recorrer las conclusiones parciales calculando el valor representativo en términos de los grados de activación, centros, bases y parámetros de estas conclusiones parciales. Como se muestra en la siguiente tabla, el modo en que dicha información es accedida por el objeto *mf* depende del lenguaje de programación utilizado.

Descripción	**java**	**ansi_c**	**cplusplus**
membership degree	mf.compute(x)	mf.compute(x)	mf.compute(x)
partial conclusions	mf.conc[]	mf.conc[]	mf.conc[]
number of partial conclusions	mf.conc.length	mf.length	mf.length
activation degree of the i-th conclusion	mf.conc[i].degree()	mf.degree[i]	mf.conc[i]->degree()
center of the i-th conclusion	mf.conc[i].center()	center(mf.conc[i])	mf.conc[i]->center()
basis of the i-th conclusion	mf.conc[i].basis()	basis(mf.conc[i])	mf.conc[i]->basis()
j-th parameter of the i-th conclusion	mf.conc[i].param(j)	param(mf.conc[i],j)	mf.conc[i]->param(j)
number of the input variables in the rule base	mf.input.length	mf.inputlength	mf.inputlength
values of the input variables in the rule base	mf.input[]	mf.input[]	mf.input[]

El siguiente ejemplo muestra la definición del método clásico de clarificación del centro de área.

```
defuz CenterOfArea {
 alias CenterOfGravity, Centroid;
 java {
  double num=0, denom=0;
  for(double x=min; x<=max; x+=step) {
   double m = mf.compute(x);
   num += x*m;
   denom += m;
  }
  if(denom==0) return (min+max)/2;
  return num/denom;
 }
 ansi_c {
  double x, m, num=0, denom=0;
  for(x=min; x<=max; x+=step) {
   m = compute(mf,x);
   num += x*m;
   denom += m;
  }
  if(denom==0) return (min+max)/2;
  return num/denom;
 }
cplusplus {
  double num=0, denom=0;
  for(double x=min; x<=max; x+=step) {
   double m = mf.compute(x);
   num += x*m;
   denom += m;
  }
  if(denom==0) return (min+max)/2;
  return num/denom;
 }
}
```

El siguiente ejemplo muestra la definición de un método de clarificación simplificado, la media borrosa ponderada (*Weighted Fuzzy Mean*).

```
defuz WeightedFuzzyMean {
 definedfor triangle, isosceles, trapezoid, bell, rectangle;
 java {
  double num=0, denom=0;
  for(int i=0; i<mf.conc.length; i++) {
   num += mf.conc[i].degree()*mf.conc[i].basis()*mf.conc[i].center();
   denom += mf.conc[i].degree()*mf.conc[i].basis();
  }
  if(denom==0) return (min+max)/2;
  return num/denom;
 }
 ansi_c {
  double num=0, denom=0;
  int i;
  for(i=0; i<mf.length; i++) {
   num += mf.degree[i]*basis(mf.conc[i])*center(mf.conc[i]);
   denom += mf.degree[i]*basis(mf.conc[i]);
  }
  if(denom==0) return (min+max)/2;
  return num/denom;
 }
cplusplus {
  double num=0, denom=0;
  for(int i=0; i<mf.length; i++) {
   num += mf.conc[i]->degree()*mf.conc[i]->basis()*mf.conc[i]->center();
   denom += mf.conc[i]->degree()*mf.conc[i]->basis();
  }
  if(denom==0) return (min+max)/2;
  return num/denom;
 }
}
```

A2.2.5.5. El paquete estándar xfl

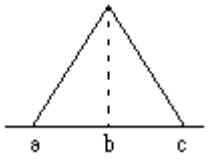
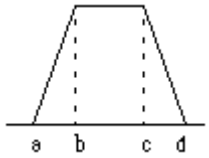
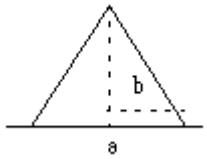
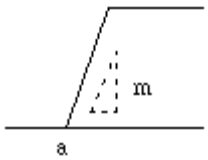
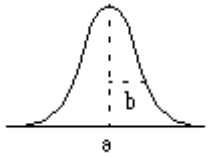
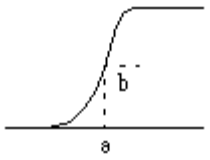
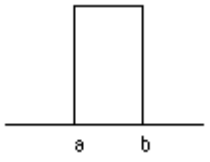
El lenguaje de especificación XFL3 permite al usuario definir sus propias funciones de pertenencia, métodos de clarificación y funciones relacionadas con las conectivas borrosas y los modificadores lingüísticos. Con objeto de facilitar el uso de XFL3, las funciones más conocidas han sido incluidas en un paquete estándar llamado *xfl*. Las funciones binarias incluidas son las siguientes:

Nombre	**Tipo**	**Descripción Java**
min	T-norm	(a<b? a : b)
prod	T-norm	(a*b)
bounded_prod	T-norm	(a+b-1>0? a+b-1: 0)
drastic_prod	T-norm	(a==1? b: (b==1? a : 0))
max	S-norm	(a>b? a : b)
sum	S-norm	(a+b-a*b)
bounded_sum	S-norm	(a+b<1? a+b: 1)
drastic_sum	S-norm	(a==0? b : (b==0? a : 0))
dienes_resher	Implication	(b>1-a? b : 1-a)
mizumoto	Implication	(1-a+a*b)
lukasiewicz	Implication	(b<a? 1-a+b : 1)
dubois_prade	Implication	(b==0? 1-a : (a==1? b : 1))
zadeh	Implication	(a<0.5 || 1-a>b? 1-a : (a<b? a : b))
goguen	Implication	(a<b? 1 : b/a)
godel	Implication	(a<=b? 1 : b)
sharp	Implication	(a<=b? 1 : 0)

Las funciones unarias incluidas en el paquete *xfl* son:

Nombre	**Parámetro**	**Descripción Java**
not	-	(1-a)
sugeno	l	(1-a)/(1+a*l)
yager	w	Math.pow((1 - Math.pow(a,w)) , 1/w)
pow	w	Math.pow(a,w)
parabola	-	4*a*(1-a)

Las funciones de pertenencia definidas en el paquete *xfl* son las siguientes:

Nombre	Parámetros	Descripción
triangle	a,b,c	
trapezoid	a,b,c,d	
isosceles	a,b	
slope	a,m	
bell	a,b	
sigma	a,b	
rectangle	a,b	

singleton	a	
parametric	unlimited	-

Los métodos de clarificación definidos en el paquete estándar son:

Nombre	**Tipo**	**Definido para**
CenterOfArea	Conventional	any function
FirstOfMaxima	Conventional	any function
LastOfMaxima	Conventional	any function
MeanOfMaxima	Conventional	any function
FuzzyMean	Simplified	triangle, isosceles, trapezoid, bell, rectangle, singleton
WeightedFuzzyMean	Simplified	triangle, isosceles, trapezoid, bell, rectangle
Quality	Simplified	triangle, isosceles, trapezoid, bell, rectangle
GammaQuality	Simplified	triangle, isosceles, trapezoid, bell, rectangle
MaxLabel	Simplified	singleton
TakagiSugeno	Simplified	parametric

A2.3. ENTORNO DE DESARROLLO *XFUZZY 3.0.*.

A2.3.1. Fase de descripción

El primer paso en el desarrollo de un sistema borroso consiste en seleccionar una descripción preliminar del sistema. Esta descripción será posteriormente refinada como resultado de las etapas de ajuste y verificación. Xfuzzy 3.0 contiene dos herramientas que facilitan la descripción de sistemas borrosos: *xfedit* y *xfpkg*. La primera está dedicada a la definición lógica del sistema, es decir, la definición de sus variables lingüísticas y las relaciones lógicas entre ellas. Por otra parte, la herramienta *xfpkg* facilita la descripción de las funciones matemáticas asignadas a los operadores borrosos, los modificadores lingüísticos, las funciones de pertenencia y los métodos de clarificación.

A2.3.1.1. Herramienta de edición de sistemas – Xfedit

La herramienta *xfedit* proporciona una interfaz gráfica para facilitar la descripción de sistemas borrosos, evitando al usuario la necesidad de conocer en profundidad el lenguaje XFL3. La herramienta está formada por un conjunto de ventanas que permiten al usuario crear y editar los conjuntos de operadores, los tipos de variables lingüísticas y las bases de reglas incluidas en el sistema borroso, así como describir la estructura jerárquica del sistema bajo desarrollo. La herramienta puede ser ejecutada directamente desde la línea de comandos con la expresión "*xfedit file.xfl*", o desde la ventana principal del entorno usando la opción *System Edition* del menú *Design*.

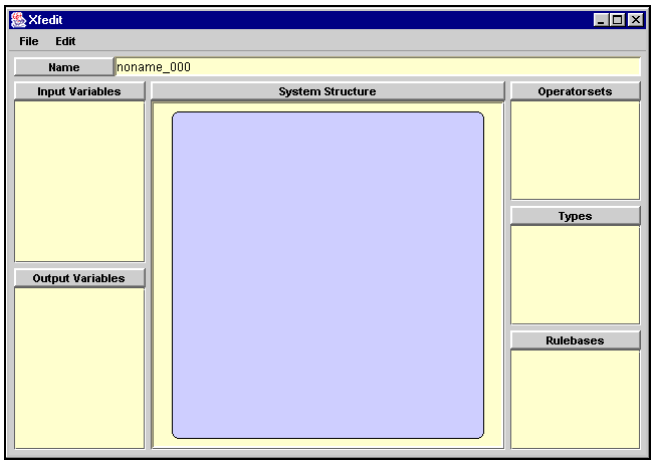


Figura A2.3

La Figura A2.3 muestra la ventana principal de *xfedit*. El menú *File* contiene las siguientes opciones: "*Save*", "*Save As*", "*Load Package*", "*Edit XFL3 File*" y "*Close Edition*". Las opciones "*Save*" y "*Save As*" se utilizan para salvar el estado actual de la definición del sistema. La opción "*Load Package*" permite importar nuevas funciones que puedan ser asignadas a los operadores borrosos. La opción "*Edit XFL3 File*" abre una ventana de texto para editar la descripción XFL3 del sistema. La última opción del menú se emplea para cerrar la herramienta. El campo *Name* bajo la barra de menús no es editable. El nombre del sistema bajo desarrollo puede cambiarse mediante la opción *Save As*. El cuerpo de la ventana está dividido en tres partes: la parte de la izquierda contiene las listas de las varia-

bles de entrada y salida globales; la parte de la derecha incluye las listas de los conjuntos de operadores, tipos de variables lingüísticas y bases de reglas; por último, la zona central muestra la estructura jerárquica del sistema.

Los aceleradores para las diferentes listas son los habituales en el entorno: la tecla *Insert* crea un nuevo elemento para cada lista; la tecla *Delete* se utiliza para eliminar el elemento (cuando no ha sido usado); la tecla *Enter* o una doble pulsación permite la edición del elemento.

La creación de un sistema borroso en Xfuzzy usualmente comienza con la definición de conjuntos de operadores (*operator sets*). La Figura A2.4 muestra la ventana usada para editar conjuntos de operadores en *xfedit*. Tiene un comportamiento simple. El primer campo contiene el identificador del conjunto de operadores. Los restantes campos contienen listas desplegables para asignar funciones a los diferentes operadores borrosos. Si la función seleccionada necesita la introducción de parámetros, se abrirá una nueva ventana para introducirlos. Las funciones disponibles en cada lista son las definidas en el paquete cargado. No es necesario seleccionar todos los campos. Una barra de comandos en la parte inferior de la ventana presenta cuatro opciones: "*Ok*", "*Apply*", "*Reload*" y "*Cancel*". La primera opción salva el conjunto de operadores y cierra la ventana. La segunda sólo salva los últimos cambios. La tercera opción recupera los últimos valores salvados para cada campo. La última cierra la ventana desechando los cambios realizados.

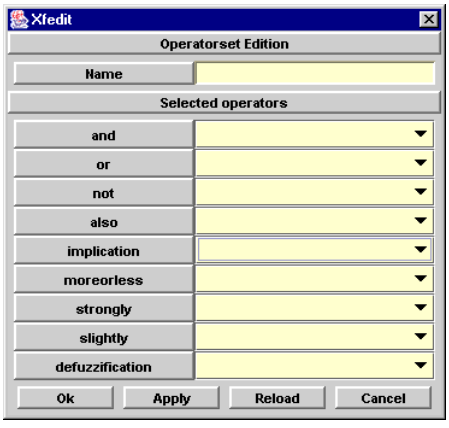


Figura A2.4

El siguiente paso en la descripción del sistema borroso es crear los tipos de las variables lingüísticas (*linguistic variable types*) mediante la ventana de *Creación de tipos* mostrada abajo. Un tipo nuevo necesita la introducción de su identificador y su universo de discurso (mínimo, máximo y cardinalidad). Como puede observarse en la Figura A2.5, la ventana incluye varios tipos predefinidos correspondientes a las particiones más habituales del universo de discurso. Estos tipos predefinidos contienen distribuciones homogéneas de funciones triangulares, trapezoidales, en forma de campana y singularidades borrosas. Otros tipos predefinidos son singularidades y campanas iguales que son habitualmente usadas como opción inicial para tipos de variables de salida. Cuando se selecciona uno de los tipos predefinidos es preciso introducir el número de funciones de pertenencia de la partición. Los tipos predefinidos también incluyen una opción vacía, que genera un tipo sin ninguna función de pertenencia, y la extensión de un tipo ya existente (seleccionado en el campo *Parent*), que implementa el mecanismo de herencia de XFL3.

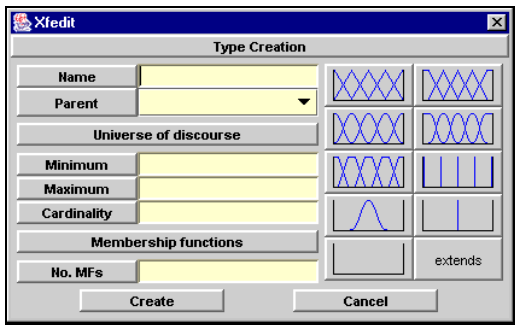


Figura A2.5

Una vez que se ha creado un tipo, éste puede ser editado usando la ventana de *Edición de tipos.* Esta ventana permite la modificación del nombre del tipo y del universo de discurso, así como añadir, editar o borrar las funciones de pertenencia del tipo editado. En la Figura A2.6, la ventana muestra una representación gráfica de las funciones de pertenencia donde la función seleccionada se representa con un color diferente. La parte inferior de la ventana contiene una barra de comandos con los botones habituales para salvar o descartar los cambios y para cerrar la ventana. Se precisa tener en consideración que las modificaciones en la definición del universo de discurso pueden afectar a las funciones de pertenencia. Por ello, se realiza una validación de los parámetros de las funciones

de pertenencia antes de salvar las modificaciones, apareciendo un mensaje de error cuando la definición de una función de pertenencia se convierte en inválida.

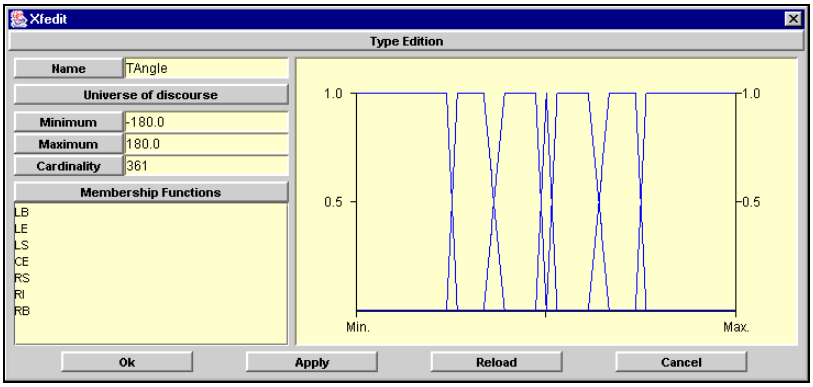


Figura A2.6

Una función de pertenencia puede ser creada o editada a partir de la lista de funciones de pertenencia con los aceleradores habituales (tecla *Insert* y tecla *Enter* o doble pulsación). La Figura A2.7 se muestra la ventana de edición de funciones de pertenencia. La ventana dispone de campos para introducir el nombre de la etiqueta lingüística, para seleccionar la clase de función de pertenencia y para incluir los valores de los parámetros. La parte de la derecha de la ventana muestra una representación gráfica de todas las funciones de pertenencia donde la función que está siendo editada aparece con un color diferente. La parte inferior de la ventana muestra una barra de comandos con tres opciones: *Set*, para cerrar la ventana y salvar los cambios; *Refresh*, para redibujar la representación gráfica; y *Cancel*, para cerrar la ventana sin salvar las modificaciones.

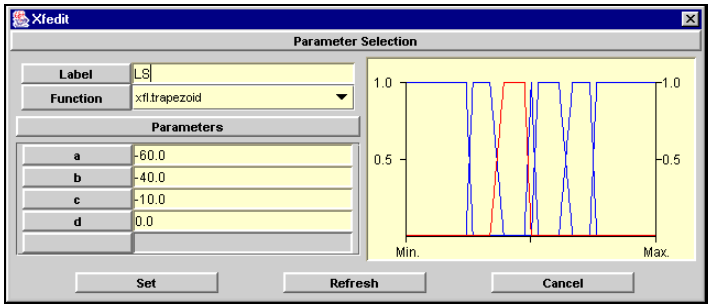


Figura A2.7

El tercer paso en la definición de un sistema borroso consiste en describir las bases de reglas que expresan las relaciones entre las variables del sistema. Las bases de reglas pueden ser creadas, editadas y eliminadas de la lista correspondiente mediante los aceleradores habituales (*Insert*, *Enter* o doble click y *Delete*). La Figura A2.8 describe la ventana que facilita la edición de bases de reglas.

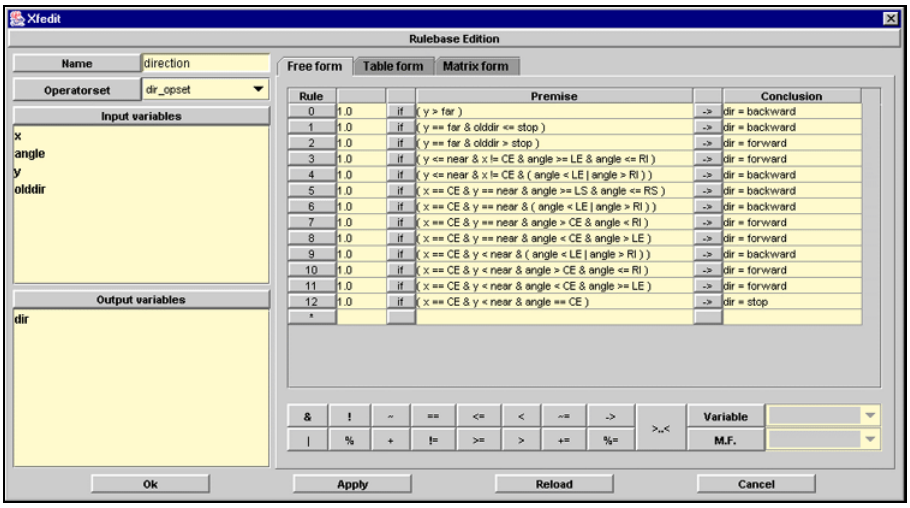


Figura A2.8

La ventana de edición de bases de reglas está dividida en tres zonas: la parte de la izquierda contiene los campos para introducir los nombres de la base de reglas y el conjunto de operadores usado, y para introducir la lista de variables de entrada y salida. La zona de la derecha se utiliza para mostrar el contenido de las reglas incluidas en la base de reglas. La parte inferior de la ventana contiene la barra de comandos con los botones habituales para salvar o descartar las modificaciones y para cerrar la ventana.

Las variables de entrada y salida pueden ser creadas, editadas o eliminadas con las secuencias de teclas habituales. La información necesaria para definir una variable es el nombre y el tipo de la variable.

Los contenidos de las reglas pueden mostrarse en tres formatos: libre, tabular y matricial. El formato libre usa tres campos por cada regla. El primero contiene el peso o factor de confidencia de la regla. El segundo campo muestra el antecedente de la regla. Se trata de un campo autoeditable, en el que los cambios se llevan a cabo seleccionando el término a

modificar (el símbolo "?" indica un término vacío) y usando los botones de la ventana. El tercer campo de cada regla contiene la descripción del consecuente. Es también un campo auto-editable que puede ser modificado pulsando el botón "->". Es posible generar reglas nuevas introduciendo valores en la última fila (marcada con el símbolo "*").

La barra de botones localizada en la parte inferior de la representación de la base de reglas en formato libre permite crear términos unidos por conjunciones (botón "&") y disyunciones (botón "|"), términos modificados por los modificadores lingüísticos *not* (botón "!"), *more or less* (botón "~"), *slightly* (botón "%") y *strongly* (botón "+"), y términos simples que relacionan una variable y una etiqueta con las cláusulas *equal to* ("=="), *not equal to* ("!="), *greater than* (">"), *smaller than* ("<"), *greater or equal to* (">="), *smaller or equal to* ("<="), *approximately equal to* ("~="), *strongly equal to* ("+=") y *slightly equal to* ("%="). El botón "->" se utiliza para añadir la conclusión de una regla. El botón ">..<" se emplea para eliminar un término conjuntivo o disyuntivo (p.e. el término "v == 1 & ?" se transforma en "v == 1"). El formato libre (Figura A2.9) permite describir relaciones más complejas entre las variables que los otros formatos.

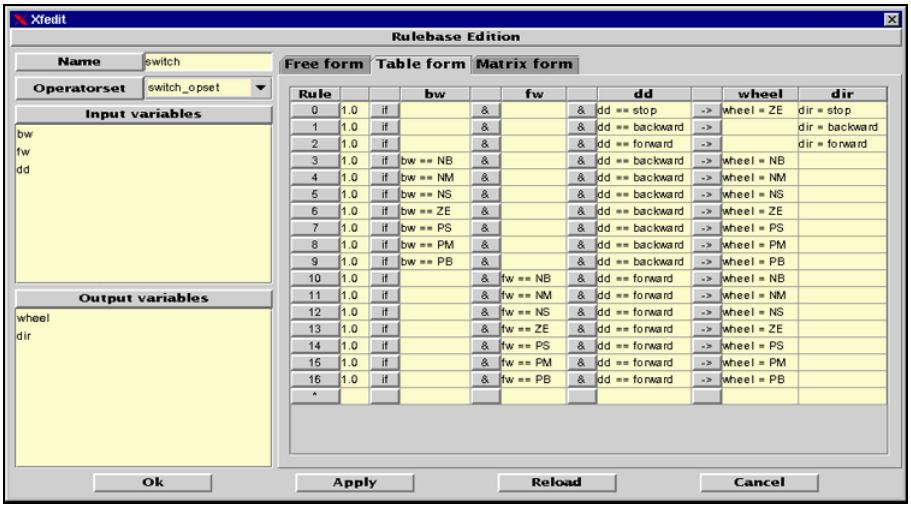
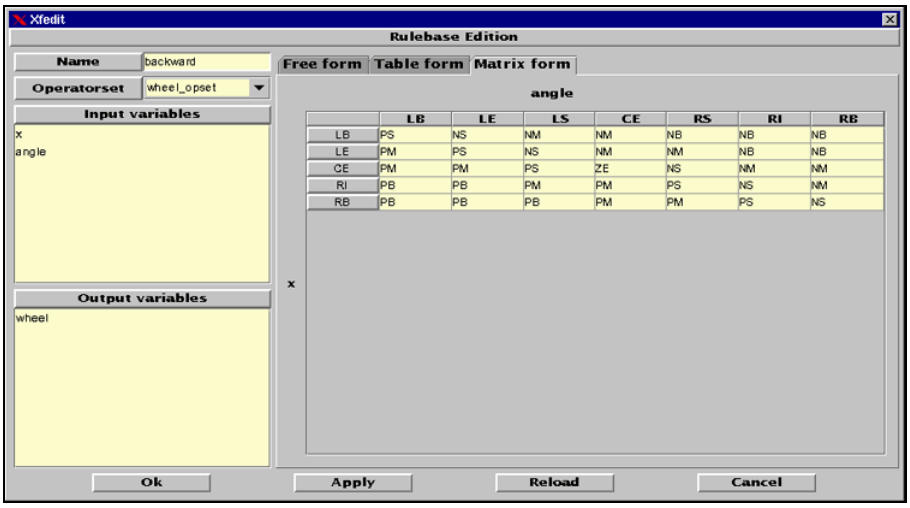


Figura A2.9

El formato tabular (Figura A2.10) resulta útil para definir reglas cuyos antecedentes usan sólo los operadores *and* y *equal*. Cada regla dispone de un campo para introducir el factor de confidencia y una lista desplegable

por cada variable de entrada y de salida. No es necesario seleccionar todos los campos de variables, pero al menos una variable de entrada y otra de salida deben ser seleccionadas siempre. Si una base de reglas contiene una regla que no puede ser expresada en formato tabular, la tabla no puede ser abierta y se genera un mensaje de error.



	LB	LE	LS	CE	RS	RI	RB
LB	PS	NS	NM	NM	NB	NB	NB
LE	PM	PS	NS	NM	NM	NB	NB
CE	PM	PM	PS	ZE	NS	NM	NM
RI	PB	PB	PM	PM	PS	NS	NM
RB	PB	PB	PB	PM	PM	PS	NS

Figura A2.10

El formato matricial está diseñado específicamente para describir bases de reglas con dos entradas y una salida. Este formato muestra el contenido de la base de reglas en un formato claro y compacto. El formato matricial genera reglas como *"if(x==X & y==Y) -> z=Z"*, es decir, reglas con factor de confidencia 1.0 y formadas por la conjunción de dos igualdades. Aquellas bases de reglas que no tienen el número de variables adecuado o que contienen reglas con un formato diferente no pueden ser mostradas en formato matricial.

Una vez que los conjuntos de operadores, los tipos de variables y las bases de reglas han sido definidos, el siguiente paso en la definición de un sistema borroso es definir las variables de entrada y salida globales utilizando la ventana de *Propiedades de las variables* (Figura A2.11). La información necesaria para crear una variable es el nombre y el tipo de la variable.

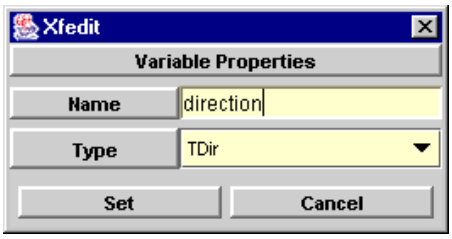


Figura A2.11

El paso final en la definición de un sistema borroso es la descripción de su estructura (veáse a modo de ejemplo la Figura A2.12). La tecla utilizada para introducir un nuevo módulo (una llamada a una base de reglas) en una jerarquía es la tecla *Insert*. Para establecer enlaces entre módulos, el usuario debe presionar el botón izquierdo del ratón situando el puntero sobre el nodo que representa a la variable de origen y soltar el botón con el puntero situado sobre el nodo de la variable de destino. Para eliminar un enlace, el usuario debe seleccionarlo pulsando sobre el nodo de la variable de destino y presionar la tecla *Delete*. La herramienta no permite crear lazos entre módulos.

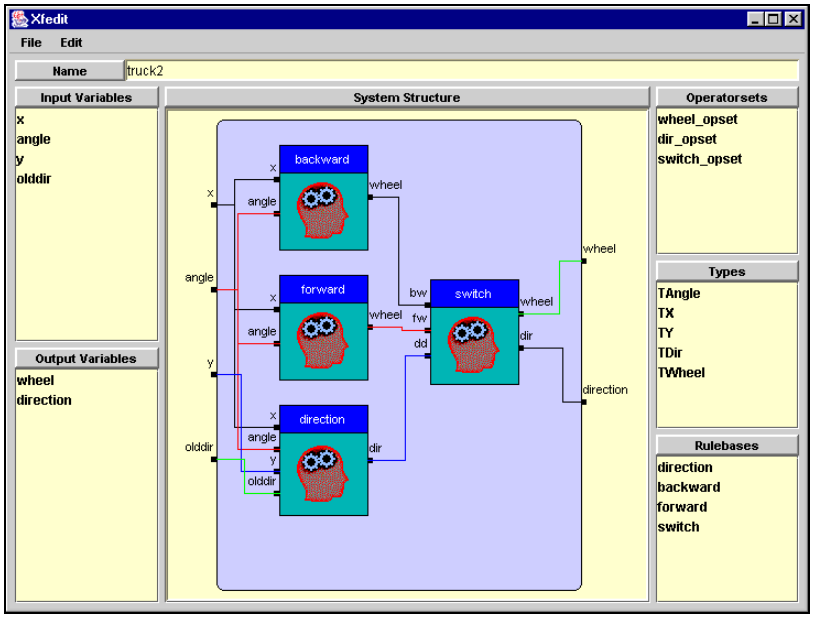


Figura A2.12

A2.3.1.2. Herramienta de edición de paquetes – Xfpkg

La descripción de un sistema borroso en el entorno Xfuzzy 3.0 se divide en dos partes. La estructura lógica del sistema (incluyendo las definiciones de conjuntos de operadores, tipos de variables, bases de reglas y estructura de comportamiento jerárquica) se especifica en ficheros con extensión ".xfl" y puede ser editada de forma gráfica con xfedit. Por otra parte, la descripción matemática de las funciones usadas como conectivas borrosas, modificadores lingüísticos, funciones de pertenencia y métodos de clarificación se especifican en paquetes (*packages*).

La herramienta *xfpkg* está dedicada a facilitar la edición de paquetes. La herramienta implementa una interfaz gráfica de usuario (Figura A2.13) que muestra las listas de las diferentes funciones incluidas en el paquete y los contenidos de los diferentes campos de una definición de función. La mayoría de estos campos contienen código que describe la función en diferentes lenguajes de programación. Este código debe ser introducido manualmente. La herramienta puede ser ejecutada desde la línea de comandos o desde la ventana principal del entorno, usando la opción *Edit package* en el menú *Design*.

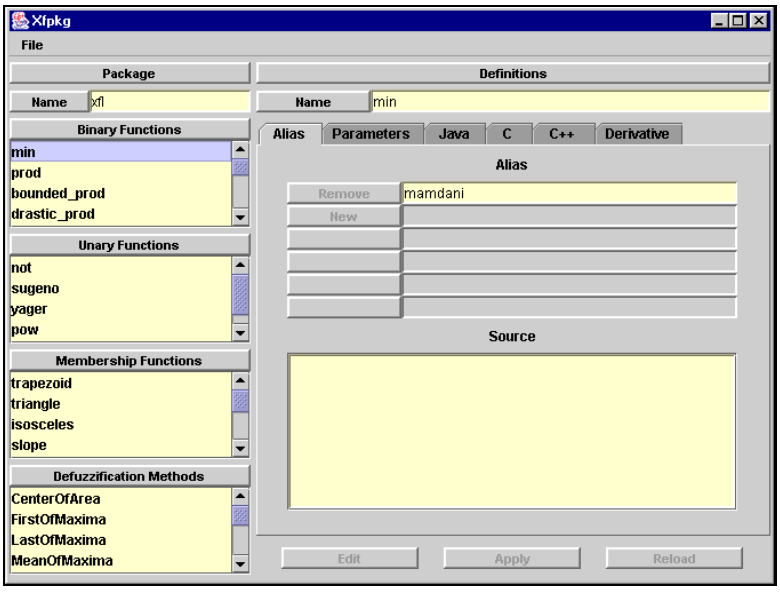


Figura A2.13

La Figura anterior muestra la ventana principal de *xfpkg*. El menú *File* contiene las opciones "*Save*", "*Save as*", "*Compile*", "*Delete*" y "*Close edition*". Las primeras dos opciones de utilizan para salvar el paquete en un fichero. La opción "*Compile*" lleva a cabo el proceso de compilación que genera los ficheros ".java" y ".class" correspondientes a cada función definida en el paquete. La opción "*Delete*" se utiliza para eliminar el fichero que contiene el paquete y todos los ficheros ".java" y ".class" generados por el proceso de compilación. La última opción se emplea para cerrar la herramienta.

La ventana principal está dividida en dos partes. La zona de la izquierda contiene cuatro listas que muestran las diferentes clases de funciones incluidas en el paquete: funciones binarias (relacionadas con los operadores de conjunción, disyunción, agregación e implicación), funciones unarias (asociadas a los modificadores lingüísticos), funciones de pertenencia (relacionadas con las etiquetas lingüísticas) y métodos de clarificación (usados para obtener valores representativos de las conclusiones borrosas). La parte derecha de la ventana principal muestra el contenido de los diferentes campos de una definición de función. La parte inferior de esta zona contiene un grupo de tres botones: "*Edit*", "*Apply*" y "*Reload*". Al seleccionar una función de una de las listas sus campos no pueden ser editados hasta que el usuario ejecuta el comando *Edit*. El comando *Apply* salva los cambios de la definición. Esto incluye la generación de los ficheros ".java" y ".class". El comando *Reload* descarta las modificaciones realizadas y actualiza los campos con los valores previamente salvados (veáse la Figura A2.14).

Los campos de una definición de función se distribuyen entre seis paneles tabulados. El panel *Alias* contiene la lista de identificadores alternativos y los bloques fuente con el código Java de métodos locales que pueden ser usados en otros campos y que son directamente incorporados en el fichero ".java".

El panel *Parameters* contiene la enumeración de los parámetros usados por la función que está siendo editada. El panel incluye también el campo *requires*, donde se describen las restricciones sobre los valores de los parámetros.

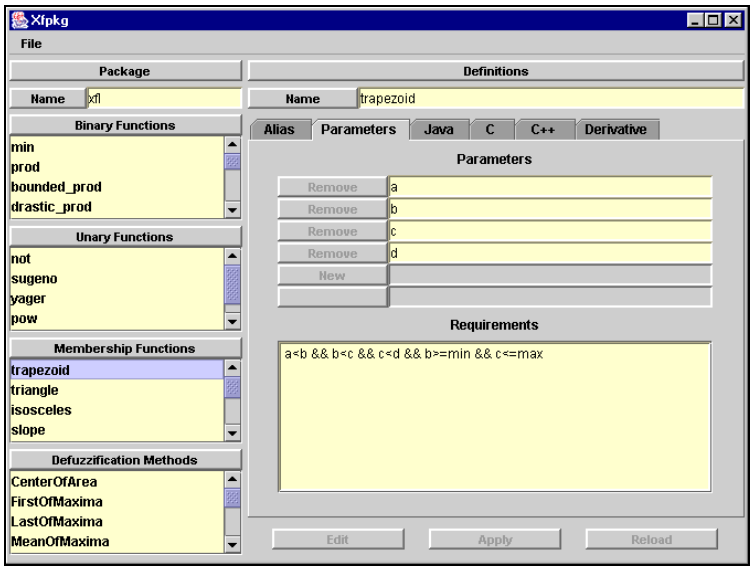


Figura A2.14

Los paneles *Java*, *C* y *C++* contienen la descripción del comportamiento de la función en estos lenguajes de programación (Figura A2.15).

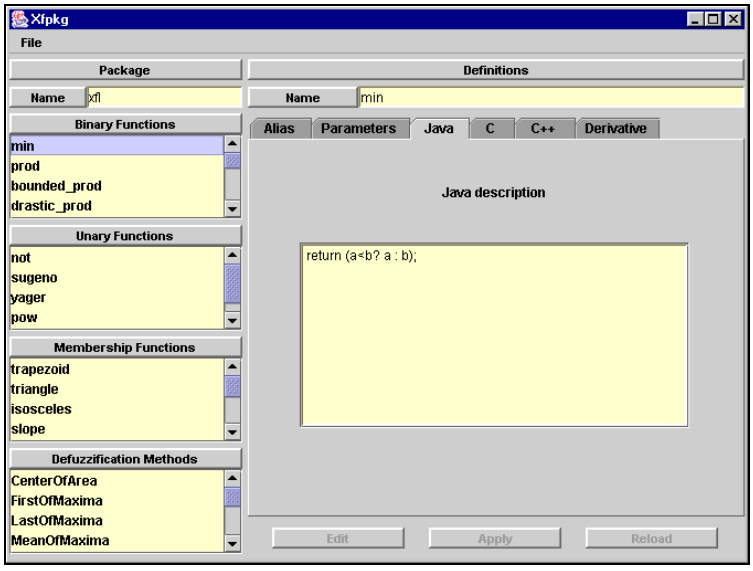


Figura A2.15

La definición de funciones de pertenencia necesita información adicional para describir el comportamiento de la función en los diferentes lenguajes de programación (Figura A2.16). En estos casos, los paneles *Java*, *C*, *C++* y *Derivative* contienen cuatro campos para mostrar el contenido de los subbloques *equal*, *greatereq*, *smallereq*, *center*, y *basis*.

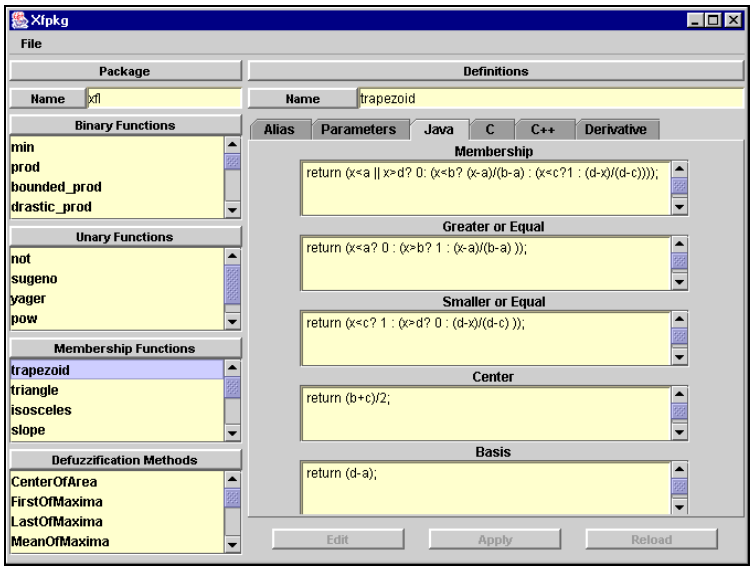


Figura A2.16

Los métodos de clarificación pueden incluir la enumeración de las funciones de pertenencia que pueden ser usadas por cada método. Dicha enumeración aparece en el panel *Parameters* (Figura A2.17).

La herramienta *xfpkg* implementa una interfaz gráfica que permite al usuario visualizar y editar la definición de las funciones incluidas en un paquete. Esta herramienta se usa para describir de un modo gráfico el comportamiento matemático de las funciones definidas. En este sentido la herramienta es el complemento de *xfedit*, que describe la estructura lógica del sistema en la etapa de descripción de un sistema borroso.

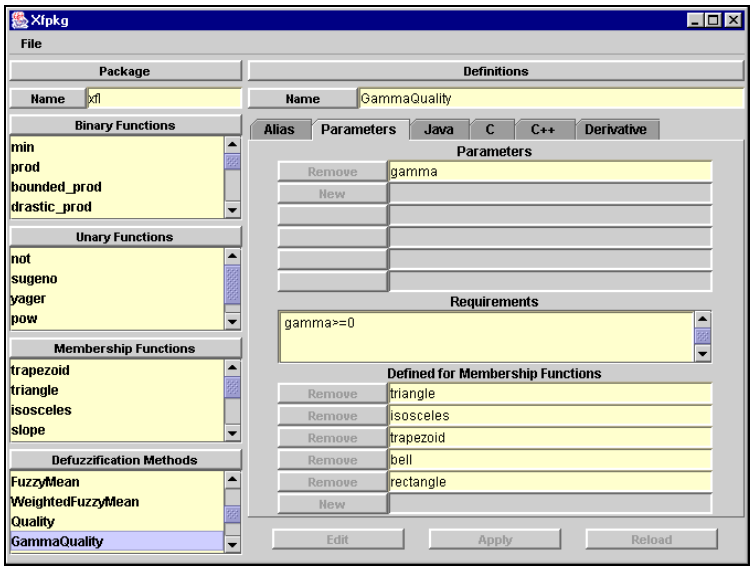


Figura A2.17

A2.3.2. Fase de verificación

La etapa de verificación en el proceso de diseño de sistemas borrosos consiste en estudiar el comportamiento del sistema borroso bajo desarrollo. El objetivo de dicho estudio es detectar las posibles desviaciones frente al comportamiento esperado e identificar las causas de estas desviaciones.

El entorno Xfuzzy 3.0 cubre la etapa de verificación con cuatro herramientas. La primera de ellas es xf2dplot, que muestra el comportamiento del sistema mediante una gráfica bidimensional. La segunda herramienta, xf3dplot, genera una representación gráfica tridimensional del comportamiento del sistema. La herramienta de monitorización, xfmt, muestra los grados de activación de las distintas reglas y variables lingüísticas, así como los valores de las diferentes variables internas, para un conjunto dado de entradas. Por último, la herramienta xfsim está dirigida hacia la simulación del sistema dentro de su entorno de operación (real o modelado), permitiendo ilustrar la evolución del sistema mediante representaciones gráficas de las variables seleccionadas por el usuario.

A2.3.2.1. Herramienta de representación gráfica bidimensional - Xf2dplot

La herramienta *xf2dplot* permite estudiar el comportamiento de una variable de salida del sistema borroso en función de una variable de entrada. La herramienta genera una representación gráfica bidimensional (Figura A2.18) que muestra la variación de la variable de salida seleccionada con respecto a la variable de entrada seleccionada. Cuando el sistema tiene más de una variable de entrada, el usuario debe introducir un valor para cada una de las variables de entrada no seleccionadas. La herramienta puede ser ejecutada desde la línea de comandos con la expresión "*xf2dplot file.xfl*", o desde la ventana principal del entorno usando la opción "*2D Plot*" del menú *Verification*.

La ventana principal de la herramienta está dividida en dos partes: La parte de la izquierda está dedicada a conFigurar la representación gráfica, mientras que la parte de la derecha es ocupada por la gráfica. La zona de conFiguración contiene una serie de campos donde se introducen los valores a los que se fijan las variables de entrada no seleccionadas. Dos listas desplegables permiten seleccionar las variables de entrada y salida que serán representadas. Finalmente, los dos botones de la parte inferior se utilizan para actualizar la representación gráfica (*Plot*) y salir de la herramienta (*Close*).

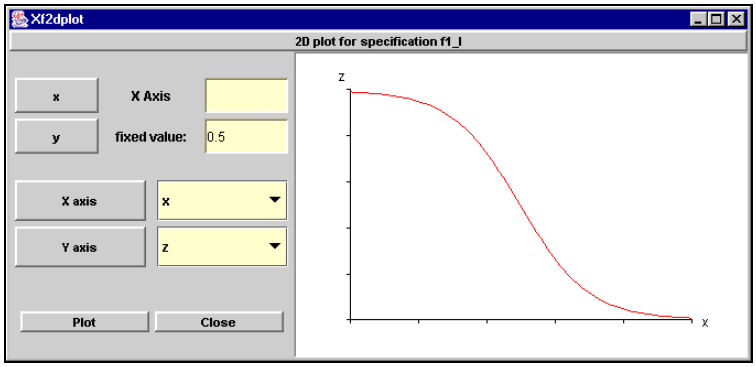


Figura A2.18

A2.3.2.2. Herramienta de representación gráfica tridimensional - Xf3dplot

La herramienta *xf3dplot* ilustra el comportamiento de un sistema borroso mediante una representación tridimensional, es decir, una superficie que muestra una variable de salida como una función de dos variables de

entrada. Por tanto, el sistema que va a ser representado debe tener al menos dos variables de entrada (Figura A2.19). Si el sistema tiene más de dos variables de entrada, el usuario debe fijar el valor de las variables no seleccionadas. La herramienta puede ser ejecutada desde la línea de comandos con la expresión "*xf3dplot file.xfl*", o desde la ventana principal del entorno usando la opción "*Surface 3D Plot*" del menú *Verification*.

La ventana principal de la herramienta está dividida en dos partes: La parte de la izquierda está dedicada a conFigurar la representación gráfica, mientras que la parte de la derecha es ocupada por la gráfica. La zona de conFiguración contiene una serie de campos donde se introducen los valores a los que se fijan las variables de entrada no seleccionadas. Tres listas desplegables permiten seleccionar las variables asignadas a cada eje. El último campo contiene el número de puntos usados en la partición de los ejes X e Y. La elección de este parámetro es importante porque determina la resolución de la representación. Un valor bajo del parámetro puede hacer que se excluyan detalles importantes del comportamiento del sistema. Por otra parte, un valor alto hará que la superficie representada sea difícil de entender al usar un grid excesivamente denso. El valor por defecto de este parámetro es 40. La zona de conFiguración termina con un par de botones. El primero de ello (*Plot*) se utiliza para actualizar la representación gráfica de acuerdo a la conFiguración actual. El segundo (*Close*) permite salir de la herramienta.

La representación gráfica incluye la posibilidad de rotar la superficie usando los dos botones deslizantes situados en la parte derecha e inferior de la gráfica. Esta capacidad de rotación facilita la interpretación de la superficie representada.

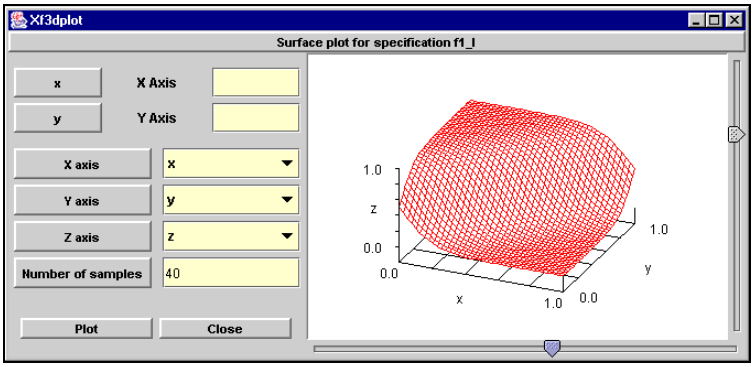


Figura A2.19

A2.3.2.3. Herramienta de monitorización de inferencias – Xfmt

El propósito de la herramienta *xfmt* es monitorizar el proceso de inferencia del sistema, esto es, mostrar gráficamente los valores de las diferentes variables internas y los grados de activación de las reglas y las etiquetas lingüísticas para un conjunto de valores de entrada determinado (Figura A2.20). La herramienta puede ser ejecutada desde la línea de comandos con la expresión "*xfmt file.xfl*", o desde la ventana principal del entorno usando la opción "*Monitor*" del menú *Verification*.

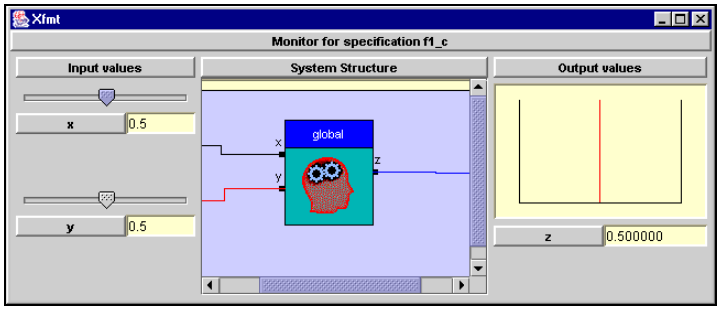


Figura A2.20

La ventana principal de *xfmt* está dividida en tres partes. La zona de la izquierda se utiliza para introducir los valores de las variables de entrada globales. Asociado con cada variable, existe un campo para introducir manualmente el valor y un botón deslizante para introducir el valor como una posición en el rango de la variable. La parte de la derecha de la ventana muestra los conjuntos borrosos asociados con los valores de las variables de salida globales, así como los valores "*crisp*" (clarificados) para esas variables. Estos valores son mostrados como singularidades borrosas (*singletones*) en las gráficas de los conjuntos borrosos (si un conjunto borroso es ya un singletone, la gráfica sólo muestra este singletone). El centro de la ventana ilustra la estructura jerárquica del sistema.

La herramienta también incluye una ventana para monitorizar los valores internos del proceso de inferencia de cada base de reglas (Figura A2.21). A esta ventana se accede pulsando sobre la base de reglas en la representación de la estructura jerárquica del sistema.

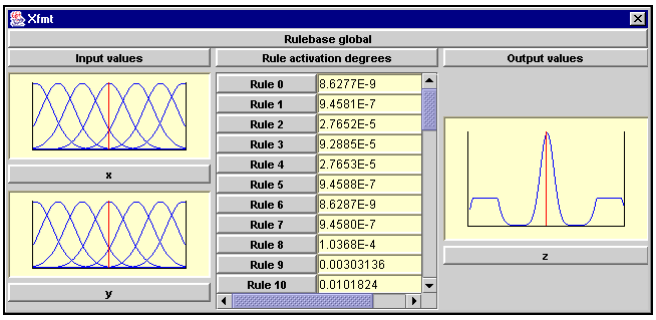


Figura A2.21

La ventana de monitorización de reglas está dividida en tres partes. Los valores de las variables de entrada se muestran en la parte izquierda como singularidades borrosas sobre las funciones de pertenencia asignadas a las diferentes etiquetas lingüísticas. La parte central de la ventana contiene un conjunto de campos con los grados de activación de cada regla. En la parte de la derecha se muestran los valores de las variables de salida obtenidas en el proceso de inferencia. Si el conjunto de operadores usado en la base de reglas especifica un método de clarificación, el valor de salida es clarificado y la gráfica muestra tanto el valor borroso como el valor *crisp* finalmente asignado a la variable de salida.

A2.3.2.4. Herramienta de simulación – Xfsim

La herramienta *xfsim* está dirigida a estudiar sistemas realimentados. Para ello la herramienta realiza la simulación del comportamiento del sistema borroso conectado a una planta o proceso externo. La herramienta puede ser ejecutada desde la línea de comandos mediante la expresión "*xfsim file.xfl*", o desde la ventana principal del entorno con la opción "*Simulation*" del menú *Verification*.

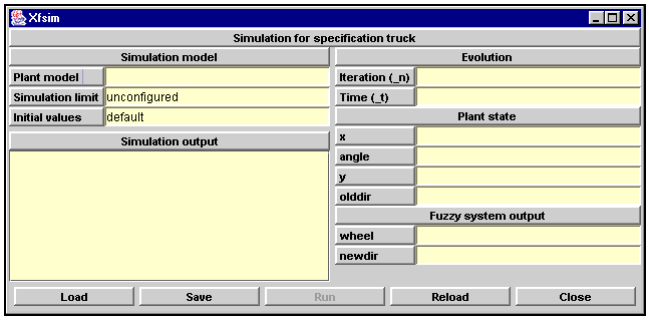


Figura A2.22

La ventana principal de *xfsim* se muestra en la Figura A2.22. La conFiguración del proceso de simulación se realiza en la parte izquierda de la ventana, mientras que la parte de la derecha muestra el estado del sistema realimentado. La parte inferior de la ventana contiene una barra de botones con las opciones "*Load*", "*Save*", "*Run/Stop*", "*Reload*" y "*Close*". La primera opción se utiliza para cargar una conFiguración para el proceso de simulación. La segunda salva la conFiguración actual en un fichero externo. La opción *Run/Stop* permite iniciar y parar el proceso de simulación. La opción *Reload* descarta la conFiguración actual y reinicia la herramienta. La última opción se emplea para salir de la herramienta.

La conFiguración del proceso de simulación se realiza seleccionando el modelo de la planta conectada al sistema borroso y sus valores iniciales, las condiciones de fin de simulación y la lista de las salidas que se desean obtener del proceso de simulación. Estas salidas pueden consistir en ficheros de log, para almacenar los valores de las variables seleccionadas, y representaciones gráficas de estas variables. La descripción del estado de la simulación contiene el número de iteraciones, el tiempo trascurrido desde el inicio de la simulación, los valores de las variables de entrada del sistema borroso (que representan el estado de la planta) y los valores de las variables de salida del sistema borroso (que representan la acción del sistema borroso sobre la planta).

A2.3.3. Fase de ajuste

La etapa de ajuste constituye habitualmente una de las tareas más complejas en el diseño de sistemas borrosos. El comportamiento del sistema depende de la estructura lógica de su base de reglas y de las funciones de pertenencia de sus variables lingüísticas. El proceso de ajuste suele dirigirse normalmente a modificar los diferentes parámetros de las funciones de pertenencia que aparecen en la definición del sistema. Ya que el número de parámetros que deben ser modificados simultáneamente es elevado, un proceso de ajuste manual resultaría claramente incómodo por lo que se requiere el uso de técnicas automáticas. Los dos tipos de mecanismos de aprendizaje más ampliamente utilizados son los denominados "aprendizaje supervisado" y "aprendizaje por refuerzo". En las técnicas de aprendizaje supervisado el comportamiento deseado del sistema es descrito mediante un conjunto de patrones de entrenamiento (y de test), mientras que en el aprendizaje por refuerzo lo que se conoce no es la salida

exacta del sistema sino el efecto que el sistema debe producir sobre su entorno, haciendo necesario por tanto la monitorización de su comportamiento en línea.

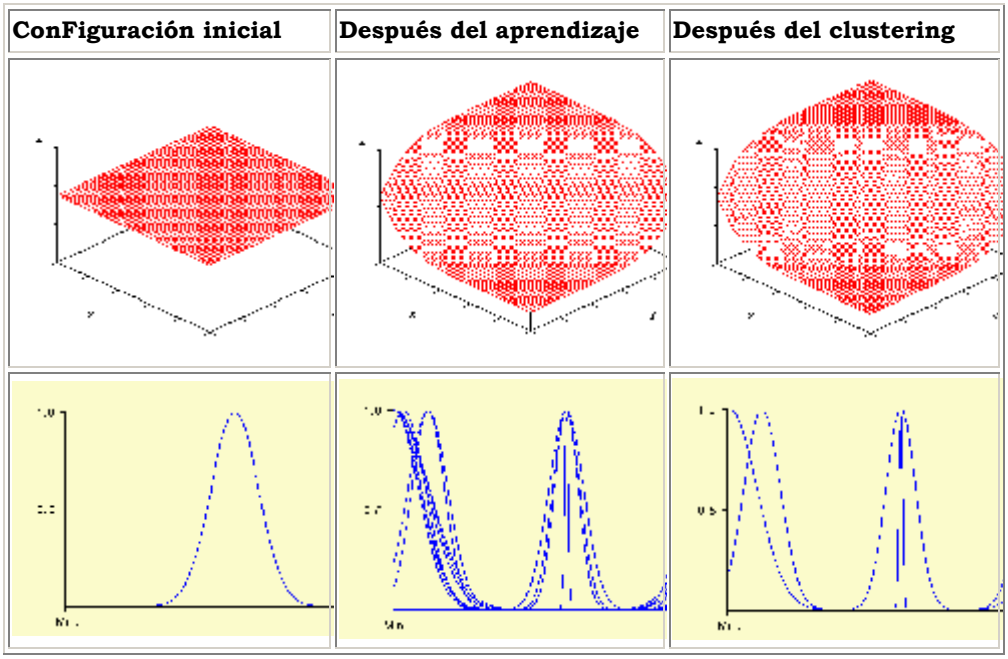
El entorno Xfuzzy 3.0 incluye una herramienta dedicada a la etapa de ajuste, xfsl, que está basada en el uso de algoritmos de aprendizaje supervisado. El aprendizaje supervisado intenta minimizar una función de error que evalúa la diferencia entre el comportamiento actual del sistema y el comportamiento deseado definido mediante un conjunto de patrones de entrada/salida.

En este sentido, cabe poner de manifiesto que en el directorio "examples/approxim/" contiene algunos ejemplos de procesos de ajuste. La conFiguración inicial del sistema se especifica en el fichero *plain.xfl*, que define un sistema borroso con dos variables de entrada y una de salida. El sistema incluye una definición de tipo para cada variable. Los tipos de las dos variables de entrada contienen siete funciones de pertenencia con forma de campana uniformemente distribuidas a lo largo del universo de discurso. El tipo de la variable de salida contiene 49 funciones de pertenencia idénticas con forma de campana y situadas en el centro del universo de discurso, de manera que el comportamiento entrada/salida de esta conFiguración inicial corresponde a una superficie plana.

El fichero *f1.trn* contiene 441 patrones que describen la superficie dada por la expresión:

$$z = 1 \; / \; (\; 1 + \exp(10*(x-y)) \;)$$

La siguiente tabla muestra los resultados del proceso de aprendizaje utilizando este fichero de entrenamiento. El algoritmo de aprendizaje utilizado fue el *Marquardt-Levenberg* con valores de control 0.1, 1.5 y 0.7. Como función de error se utilizó el error cuadrático medio (*Mean Square Error*). Se aplicó el aprendizaje a todos los parámetros del sistema. El proceso de ajuste afectó principalmente a las funciones de pertenencia de la variable de salida, haciendo que el comportamiento entrada/salida del sistema se aproximara a la superficie deseada. Como resultado, estas funciones de pertenencia forman varios grupos. Aplicando técnicas de clustering como postprocesado, el número de funciones puede ser reducido a seis.



A2.3.4. Fase de síntesis

La etapa de síntesis es el último paso en el flujo de diseño de un sistema. Su objetivo es generar una implementación del sistema que pueda ser usada externamente. Existen dos tipos diferentes de implementaciones finales para sistemas borrosos: implementaciones software e implementaciones hardware. La síntesis software genera la representación del sistema en un lenguaje de programación de alto nivel. La síntesis hardware genera un circuito microelectrónico que implementa el proceso de inferencia descrito por el sistema borroso.

Las implementaciones software resultan útiles cuando no existen fuertes restricciones sobre la velocidad de inferencia, el tamaño del sistema o el consumo de potencia. Este tipo de implementación puede ser generada a partir de cualquier sistema borroso desarrollado en Xfuzzy. Por otra parte, las implementaciones hardware son más adecuadas cuando se requiere alta velocidad o bajo consumo de área y potencia, pero, para que esta solución sea eficiente, es necesario imponer ciertas restricciones sobre el sistema borroso, de forma que la síntesis hardware no es tan genérica como la alternativa software.

Xfuzzy proporciona al usuario tres herramientas para síntesis software: *xfc*, que genera una descripción del sistema en ANSI-C; *xfcpp*, para generar una descripción C++; y *xfj*, que describe el sistema borroso mediante una clase Java. Precisamente, un ejemplo de éste último, en su aplicación al caso objeto de estudio propuesto, la detección de intrusos, puede observarse en el Anexo 3 del presente trabajo que sigue a continuación.

ANEXO 3. CÓDIGO EN JAVA DEL SISTEMA DE DETECCCIÓN DE INTRUSOS

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
//     Fuzzy Inference Engine ids      //
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
package IDS;
public class ids implements FuzzyInferenceEngine {
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //      Membership function of an input variable       //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private abstract class InnerMembershipFunction {
  double min, max, step;
  abstract double param(int i);
  double center() { return 0; }
  double basis() { return 0; }
  abstract double isEqual(double x);
  double isSmallerOrEqual(double x) {
   double degree=0, mu;
   for(double y=max; y>=x ; y-=step) if((mu = isEqual(y))>degree) degree=mu;
   return degree;
  }

  double isGreaterOrEqual(double x) {
   double degree=0, mu;
   for(double y=min; y<=x ; y+=step) if((mu = isEqual(y))>degree) degree=mu;
   return degree;
  }
  double isEqual(MembershipFunction mf) {
   if(mf instanceof FuzzySingleton)
    { return isEqual( ((FuzzySingleton) mf).getValue()); }
   if((mf instanceof OutputMembershipFunction) &&
      ((OutputMembershipFunction) mf).isDiscrete() ) {
    double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
    double deg = 0;
    for(int i=0; i<val.length; i++){
     double mu = isEqual(val[i][0]);
     double minmu = (mu<val[i][1] ? mu : val[i][1]);
     if( deg<minmu ) deg = minmu;
    }
    return deg;
   }
   double mu1,mu2,minmu,degree=0;
   for(double x=min; x<=max; x+=step){
    mu1 = mf.compute(x);
    mu2 = isEqual(x);
    minmu = (mu1<mu2 ? mu1 : mu2);
    if( degree<minmu ) degree = minmu;
   }
   return degree;
  }
```

```
double isGreaterOrEqual(MembershipFunction mf) {
 if(mf instanceof FuzzySingleton)
  { return isGreaterOrEqual( ((FuzzySingleton) mf).getValue()); }
 if((mf instanceof OutputMembershipFunction) &&
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = isGreaterOrEqual(val[i][0]);
   double minmu = (mu<val[i][1] ? mu : val[i][1]);
   if( deg<minmu ) deg = minmu;
  }
  return deg;
 }
 double mu1,mu2,minmu,degree=0,greq=0;
 for(double x=min; x<=max; x+=step){
  mu1 = mf.compute(x);
  mu2 = isEqual(x);
  if( mu2>greq ) greq = mu2;
  if( mu1<greq ) minmu = mu1; else minmu = greq;
  if( degree<minmu ) degree = minmu;
 }
 return degree;
}
double isSmallerOrEqual(MembershipFunction mf) {
 if(mf instanceof FuzzySingleton)
  { return isSmallerOrEqual( ((FuzzySingleton) mf).getValue()); }
 if((mf instanceof OutputMembershipFunction) &&
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = isSmallerOrEqual(val[i][0]);
   double minmu = (mu<val[i][1] ? mu : val[i][1]);
   if( deg<minmu ) deg = minmu;
  }
  return deg;
 }
 double mu1,mu2,minmu,degree=0,smeq=0;
 for(double x=max; x>=min; x-=step){
  mu1 = mf.compute(x);
  mu2 = isEqual(x);
  if( mu2>smeq ) smeq = mu2;
  if( mu1<smeq ) minmu = mu1; else minmu = smeq;
  if( degree<minmu ) degree = minmu;
 }
 return degree;
}
double isGreater(MembershipFunction mf, InnerOperatorset op) {
 if(mf instanceof FuzzySingleton)
  { return op.not(isSmallerOrEqual( ((FuzzySingleton) mf).getValue())); }
```

```
 if((mf instanceof OutputMembershipFunction) &&
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = op.not(isSmallerOrEqual(val[i][0]));
   double minmu = (mu<val[i][1] ? mu : val[i][1]);
   if( deg<minmu ) deg = minmu;
  }
  return deg;
 }
 double mu1,mu2,minmu,gr,degree=0,smeq=0;
 for(double x=max; x>=min; x-=step){
  mu1 = mf.compute(x);
  mu2 = isEqual(x);
  if( mu2>smeq ) smeq = mu2;
  gr = op.not(smeq);
  minmu = ( mu1<gr ? mu1 : gr);
  if( degree<minmu ) degree = minmu;
 }
 return degree;
}
double isSmaller(MembershipFunction mf, InnerOperatorset op) {
 if(mf instanceof FuzzySingleton)
  { return op.not(isGreaterOrEqual( ((FuzzySingleton) mf).getValue())); }
 if((mf instanceof OutputMembershipFunction) &&
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = op.not(isGreaterOrEqual(val[i][0]));
   double minmu = (mu<val[i][1] ? mu : val[i][1]);
   if( deg<minmu ) deg = minmu;
  }
  return deg;
 }
 double mu1,mu2,minmu,sm,degree=0,greq=0;
 for(double x=min; x<=max; x+=step){
  mu1 = mf.compute(x);
  mu2 = isEqual(x);
  if( mu2>greq ) greq = mu2;
  sm = op.not(greq);
  minmu = ( mu1<sm ? mu1 : sm);
  if( degree<minmu ) degree = minmu;
 }
 return degree;
}
double isNotEqual(MembershipFunction mf, InnerOperatorset op) {
 if(mf instanceof FuzzySingleton)
  { return op.not(isEqual( ((FuzzySingleton) mf).getValue())); }
 if((mf instanceof OutputMembershipFunction) &&
```

```
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = op.not(isEqual(val[i][0]));
   double minmu = (mu<val[i][1] ? mu : val[i][1]);
   if( deg<minmu ) deg = minmu;
  }
  return deg;
 }
 double mu1,mu2,minmu,degree=0;
 for(double x=min; x<=max; x+=step){
  mu1 = mf.compute(x);
  mu2 = op.not(isEqual(x));
  minmu = (mu1<mu2 ? mu1 : mu2);
  if( degree<minmu ) degree = minmu;
 }
 return degree;
}
double isApproxEqual(MembershipFunction mf, InnerOperatorset op) {
 if(mf instanceof FuzzySingleton)
  { return op.moreorless(isEqual( ((FuzzySingleton) mf).getValue())); }
 if((mf instanceof OutputMembershipFunction) &&
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = op.moreorless(isEqual(val[i][0]));
   double minmu = (mu<val[i][1] ? mu : val[i][1]);
   if( deg<minmu ) deg = minmu;
  }
  return deg;
 }
 double mu1,mu2,minmu,degree=0;
 for(double x=min; x<=max; x+=step){
  mu1 = mf.compute(x);
  mu2 = op.moreorless(isEqual(x));
  minmu = (mu1<mu2 ? mu1 : mu2);
  if( degree<minmu ) degree = minmu;
 }
 return degree;
}
double isVeryEqual(MembershipFunction mf, InnerOperatorset op) {
 if(mf instanceof FuzzySingleton)
  { return op.very(isEqual( ((FuzzySingleton) mf).getValue())); }
 if((mf instanceof OutputMembershipFunction) &&
    ((OutputMembershipFunction) mf).isDiscrete() ) {
  double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
  double deg = 0;
  for(int i=0; i<val.length; i++){
   double mu = op.very(isEqual(val[i][0]));
```

```
    double minmu = (mu<val[i][1] ? mu : val[i][1]);
    if( deg<minmu ) deg = minmu;
   }
   return deg;
  }
  double mu1,mu2,minmu,degree=0;
  for(double x=min; x<=max; x+=step){
   mu1 = mf.compute(x);
   mu2 = op.very(isEqual(x));
   minmu = (mu1<mu2 ? mu1 : mu2);
   if( degree<minmu ) degree = minmu;
  }
  return degree;
 }
 double isSlightlyEqual(MembershipFunction mf, InnerOperatorset op) {
  if(mf instanceof FuzzySingleton)
   { return op.slightly(isEqual( ((FuzzySingleton) mf).getValue())); }
  if((mf instanceof OutputMembershipFunction) &&
     ((OutputMembershipFunction) mf).isDiscrete() ) {
   double[][] val = ((OutputMembershipFunction) mf).getDiscreteValues();
   double deg = 0;
   for(int i=0; i<val.length; i++){
    double mu = op.slightly(isEqual(val[i][0]));
    double minmu = (mu<val[i][1] ? mu : val[i][1]);
    if( deg<minmu ) deg = minmu;
   }
   return deg;
  }
  double mu1,mu2,minmu,degree=0;
  for(double x=min; x<=max; x+=step){
   mu1 = mf.compute(x);
   mu2 = op.slightly(isEqual(x));
   minmu = (mu1<mu2 ? mu1 : mu2);
   if( degree<minmu ) degree = minmu;
  }
  return degree;
 }
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
//          Abstract class of an operator set          //
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
private abstract class InnerOperatorset {
 abstract double and(double a, double b);
 abstract double or(double a, double b);
 abstract double also(double a, double b);
 abstract double imp(double a, double b);
 abstract double not(double a);
 abstract double very(double a);
 abstract double moreorless(double a);
 abstract double slightly(double a);
 abstract double defuz(OutputMembershipFunction mf);
```

```
 }
//+------------------------------------------------------+//
//      Class for the conclusion of a fuzzy rule        //
//+------------------------------------------------------+//
private class InnerConclusion {
 private double degree;
 private InnerMembershipFunction mf;
 private InnerOperatorset op;
 InnerConclusion(double degree, InnerMembershipFunction mf, InnerOperatorset op) {
  this.op = op;
  this.degree = degree;
  this.mf = mf;
 }
 public double degree() { return degree; }
 public double compute(double x) { return op.imp(degree,mf.isEqual(x)); }
 public double center() { return mf.center(); }
 public double basis() { return mf.basis(); }
 public double param(int i) { return mf.param(i); }
 public double min() { return mf.min; }
 public double max() { return mf.max; }
 public double step() { return mf.step; }
 public boolean isSingleton() {
  return mf.getClass().getName().endsWith("MF_xfl_singleton");
 }
}
//+------------------------------------------------------+//
//      Membership function of an output variable       //
//+------------------------------------------------------+//
private class OutputMembershipFunction implements MembershipFunction {
 public InnerConclusion[] conc;
 public double[] input;
 private InnerOperatorset op;
 OutputMembershipFunction() {
  this.conc = new InnerConclusion[0];
 }
 public void set(int size, InnerOperatorset op, double[] input) {
  this.input = input;
  this.op = op;
  this.conc = new InnerConclusion[size];
 }
 public void set(int pos, double dg, InnerMembershipFunction imf) {
  this.conc[pos] = new InnerConclusion(dg,imf,op);
 }
 public double compute(double x) {
  double dom = conc[0].compute(x);
  for(int i=1; i<conc.length; i++) dom = op.also(dom,conc[i].compute(x));
  return dom;
 }
 public double defuzzify() {
  return op.defuz(this);
 }
```

```
 public double min() {
  return conc[0].min();
 }
 public double max() {
  return conc[0].max();
 }
 public double step() {
  return conc[0].step();
 }
 public boolean isDiscrete() {
  for(int i=0; i<conc.length; i++) if(!conc[i].isSingleton()) return false;
  return true;
 }
  public double[][] getDiscreteValues() {
  double[][] value = new double[conc.length][2];
  for(int i=0; i<conc.length; i++) {
   value[i][0] = conc[i].param(0);
   value[i][1] = conc[i].degree();
  }
  return value;
 }
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
//      Membership function MF_xfl_trapezoid      //
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private class MF_xfl_trapezoid extends InnerMembershipFunction {
  double a;
  double b;
  double c;
  double d;
  MF_xfl_trapezoid( double min, double max, double step, double param[]){
   super.min = min;
   super.max = max;
   super.step = step;
   this.a = param[0];
   this.b = param[1];
   this.c = param[2];
   this.d = param[3];
  }
  double param(int _i) {
   switch(_i) {
    case 0: return a;
    case 1: return b;
    case 2: return c;
    case 3: return d;
    default: return 0;
   }
  }
  double isEqual(double x) {
   return (x<a || x>d? 0: (x<b? (x-a)/(b-a) : (x<c?1 : (d-x)/(d-c))));
  }
```

```
  double isGreaterOrEqual(double x) {
   return (x<a? 0 : (x>b? 1 : (x-a)/(b-a) ));
  }
  double isSmallerOrEqual(double x) {
   return (x<c? 1 : (x>d? 0 : (d-x)/(d-c) ));
  }
  double center() {
   return (b+c)/2;
  }
  double basis() {
   return (d-a);
  }
 }
//+------------------------------------------------------+//
//      Membership function MF_xfl_singleton      //
//+------------------------------------------------------+//
 private class MF_xfl_singleton extends InnerMembershipFunction {
  double a;
  MF_xfl_singleton( double min, double max, double step, double param[]){
   super.min = min;
   super.max = max;
   super.step = step;
   this.a = param[0];
  }
  double param(int _i) {
   switch(_i) {
    case 0: return a;
    default: return 0;
   }
  }
  double isEqual(double x) {
   return (x==a? 1 : 0);
  }
  double isGreaterOrEqual(double x) {
   return (x>=a? 1 : 0);
  }
  double isSmallerOrEqual(double x) {
   return (x<=a? 1 : 0);
  }
  double center() {
   return a;
  }
 }
//+------------------------------------------------------+//
//      Membership function MF_xfl_triangle      //
//+------------------------------------------------------+//
 private class MF_xfl_triangle extends InnerMembershipFunction {
  double a;
  double b;
  double c;
  MF_xfl_triangle( double min, double max, double step, double param[]){
```

```
   super.min = min;
   super.max = max;
   super.step = step;
   this.a = param[0];
   this.b = param[1];
   this.c = param[2];
  }
  double param(int _i) {
   switch(_i) {
    case 0: return a;
    case 1: return b;
    case 2: return c;
    default: return 0;
   }
  }
  double isEqual(double x) {
   return (a<x && x<=b? (x-a)/(b-a) : (b<x && x<c? (c-x)/(c-b) : 0));
  }
  double isGreaterOrEqual(double x) {
   return (x<a? 0 : (x>b? 1 : (x-a)/(b-a) ));
  }
  double isSmallerOrEqual(double x) {
   return (x<b? 1 : (x>c? 0 : (c-x)/(c-b) ));
  }
  double center() {
   return b;
  }
  double basis() {
   return (c-a);
  }
 }
//+------------------------------------------------------+//
//     Operator set OP_opset      //
//+------------------------------------------------------+//
private class OP_opset extends InnerOperatorset {
 double and(double a, double b) {
  return (a<b? a : b);
 }
 double or(double a, double b) {
  return (a>b? a : b);
 }
 double also(double a, double b) {
  return (a>b? a : b);
 }
 double imp(double a, double b) {
  return (a<b? a : b);
 }
 double not(double a) {
  return 1-a;
 }
 double very(double a) {
```

```
  double w = 2.0;
   return Math.pow(a,w);
 }
 double moreorless(double a) {
  double w = 0.5;
   return Math.pow(a,w);
 }
 double slightly(double a) {
   return 4*a*(1-a);
 }
double defuz(OutputMembershipFunction mf) {
  double max = mf.max();
 double maxdegree=0, center=0;
 for(int i=0; i<mf.conc.length; i++)
  if(mf.conc[i].degree() >= maxdegree)
   { center = mf.conc[i].center(); maxdegree = mf.conc[i].degree(); }
 return center;
 }
}
//+------------------------------------------------------+//
//  Type TP_T_ILF  //
//+------------------------------------------------------+//
private class TP_T_ILF {
 private double min = 0.0;
 private double max = 25.0;
 private double step = 1.0;
 double _p_Normales[] = { -1.0,0.0,1.0,4.0 };
 double _p_Altos[] = { 2.0,3.0,12.0,18.0 };
 double _p_Excesivos[] = { 15.625,18.75,25.0,28.125 };
 MF_xfl_trapezoid Normales = new MF_xfl_trapezoid(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altos = new MF_xfl_trapezoid(min,max,step,_p_Altos);
 MF_xfl_trapezoid Excesivos = new MF_xfl_trapezoid(min,max,step,_p_Excesivos);
}
//+------------------------------------------------------+//
//  Type TP_T_OIL  //
//+------------------------------------------------------+//
private class TP_T_OIL {
 private double min = 0.0;
 private double max = 70.0;
 private double step = 1.0;
 double _p_Normales[] = { -1.0,0.0,3.0,12.0 };
 double _p_Altos[] = { 5.0,8.0,40.0,50.0 };
 double _p_Excesivos[] = { 30.0,60.0,70.0,78.75 };
 MF_xfl_trapezoid Normales = new MF_xfl_trapezoid(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altos = new MF_xfl_trapezoid(min,max,step,_p_Altos);
 MF_xfl_trapezoid Excesivos = new MF_xfl_trapezoid(min,max,step,_p_Excesivos);
}
//+------------------------------------------------------+//
//  Type TP_T_CEU  //
//+------------------------------------------------------+//
private class TP_T_CEU {
```

```
 private double min = 0.0;
 private double max = 30.0;
 private double step = 1.0;
 double _p_Normales[] = { -3.75,0.0,2.0,3.0 };
 double _p_Altas[] = { 2.0,5.0,10.0,15.0 };
 double _p_Excesivas[] = { 8.0,20.0,30.0,33.75 };
 MF_xfl_trapezoid Normales = new MF_xfl_trapezoid(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altas = new MF_xfl_trapezoid(min,max,step,_p_Altas);
 MF_xfl_trapezoid Excesivas = new MF_xfl_trapezoid(min,max,step,_p_Excesivas);
}
//+------------------------------------------------------+//
//  Type TP_T_CPI  //
//+------------------------------------------------------+//
private class TP_T_CPI {
 private double min = 0.0;
 private double max = 10.0;
 private double step = 1.0;
 double _p_Normales[] = { -1.25,0.0,1.0,3.0 };
 double _p_Altas[] = { 2.0,4.0,5.0,8.0 };
 double _p_Excesivas[] = { 3.0,6.0,10.0,11.25 };
 MF_xfl_trapezoid Normales = new MF_xfl_trapezoid(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altas = new MF_xfl_trapezoid(min,max,step,_p_Altas);
 MF_xfl_trapezoid Excesivas = new MF_xfl_trapezoid(min,max,step,_p_Excesivas);
}
//+------------------------------------------------------+//
//  Type TP_T_LM  //
//+------------------------------------------------------+//
private class TP_T_LM {
 private double min = 0.0;
 private double max = 5.0;
 private double step = 1.0;
 double _p_Normales[] = { 0.0 };
 double _p_Altos[] = { 1.0,1.005,3.0,4.0 };
 double _p_Excesivos[] = { 3.125,3.75,5.0,5.625 };
 MF_xfl_singleton Normales = new MF_xfl_singleton(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altos = new MF_xfl_trapezoid(min,max,step,_p_Altos);
 MF_xfl_trapezoid Excesivos = new MF_xfl_trapezoid(min,max,step,_p_Excesivos);
}
//+------------------------------------------------------+//
//  Type TP_T_CC  //
//+------------------------------------------------------+//
private class TP_T_CC {
 private double min = 0.0;
 private double max = 6.0;
 private double step = 1.0;
 double _p_Normales[] = { 0.0 };
 double _p_Alguna[] = { 1.0,1.005,2.0,3.0 };
 double _p_Excesivas[] = { 2.0,3.0,6.0,6.75 };
 MF_xfl_singleton Normales = new MF_xfl_singleton(min,max,step,_p_Normales);
 MF_xfl_trapezoid Alguna = new MF_xfl_trapezoid(min,max,step,_p_Alguna);
 MF_xfl_trapezoid Excesivas = new MF_xfl_trapezoid(min,max,step,_p_Excesivas);
```

```
}
//+------------------------------------------------------+//
//  Type TP_T_SRE  //
//+------------------------------------------------------+//
private class TP_T_SRE {
 private double min = 0.0;
 private double max = 6.0;
 private double step = 1.0;
 double _p_Normales[] = { -0.05,0.0,2.0 };
 double _p_Altos[] = { 1.0,1.005,2.0,3.0 };
 double _p_Excesivos[] = { 1.0,5.0,6.0,6.75 };
 MF_xfl_triangle Normales = new MF_xfl_triangle(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altos = new MF_xfl_trapezoid(min,max,step,_p_Altos);
 MF_xfl_trapezoid Excesivos = new MF_xfl_trapezoid(min,max,step,_p_Excesivos);
}
//+------------------------------------------------------+//
//  Type TP_T_CF  //
//+------------------------------------------------------+//
private class TP_T_CF {
 private double min = 0.0;
 private double max = 90.0;
 private double step = 1.0;
 double _p_Normales[] = { 0.0,5.0,15.0,50.0 };
 double _p_Altos[] = { 20.0,30.0,60.0,70.0 };
 double _p_Excesivos[] = { 40.0,80.0,90.0,101.25 };
 MF_xfl_trapezoid Normales = new MF_xfl_trapezoid(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altos = new MF_xfl_trapezoid(min,max,step,_p_Altos);
 MF_xfl_trapezoid Excesivos = new MF_xfl_trapezoid(min,max,step,_p_Excesivos);
}
//+------------------------------------------------------+//
//  Type TP_T_BF  //
//+------------------------------------------------------+//
private class TP_T_BF {
 private double min = 0.0;
 private double max = 40.0;
 private double step = 1.0;
 double _p_Normales[] = { 0.0,2.0,5.0,15.0 };
 double _p_Altos[] = { 8.0,10.0,20.0,25.0 };
 double _p_Excesivos[] = { 15.0,30.0,40.0,45.0 };
 MF_xfl_trapezoid Normales = new MF_xfl_trapezoid(min,max,step,_p_Normales);
 MF_xfl_trapezoid Altos = new MF_xfl_trapezoid(min,max,step,_p_Altos);
 MF_xfl_trapezoid Excesivos = new MF_xfl_trapezoid(min,max,step,_p_Excesivos);
}
//+------------------------------------------------------+//
//  Type TP_T_UC  //
//+------------------------------------------------------+//
private class TP_T_UC {
 private double min = 0.0;
 private double max = 100.0;
 private double step = 1.0;
 double _p_Normal[] = { 0.0,5.0,10.0,15.0 };
```

```
 double _p_Mucho[] = { 10.0,25.0,30.0,50.0 };
 double _p_Demasiado[] = { 40.0,60.0,100.0,112.5 };
 MF_xfl_trapezoid Normal = new MF_xfl_trapezoid(min,max,step,_p_Normal);
 MF_xfl_trapezoid Mucho = new MF_xfl_trapezoid(min,max,step,_p_Mucho);
 MF_xfl_trapezoid Demasiado = new MF_xfl_trapezoid(min,max,step,_p_Demasiado);
}
//+------------------------------------------------------+//
//  Type TP_T_UM  //
//+------------------------------------------------------+//
private class TP_T_UM {
 private double min = 0.0;
 private double max = 100.0;
 private double step = 1.0;
 double _p_Normal[] = { 0.0,5.0,10.0,15.0 };
 double _p_Mucho[] = { 10.0,25.0,30.0,50.0 };
 double _p_Demasiado[] = { 40.0,60.0,100.0,112.5 };
 MF_xfl_trapezoid Normal = new MF_xfl_trapezoid(min,max,step,_p_Normal);
 MF_xfl_trapezoid Mucho = new MF_xfl_trapezoid(min,max,step,_p_Mucho);
 MF_xfl_trapezoid Demasiado = new MF_xfl_trapezoid(min,max,step,_p_Demasiado);
}
//+------------------------------------------------------+//
//  Type TP_T_ES  //
//+------------------------------------------------------+//
private class TP_T_ES {
 private double min = 0.0;
 private double max = 4500.0;
 private double step = 1.0;
 double _p_Bueno[] = { 0.0,1.0,15.0,60.0 };
 double _p_Malo[] = { 30.0,50.0,500.0,4000.0 };
 double _p_Caido[] = { 4000.0 };
 MF_xfl_trapezoid Bueno = new MF_xfl_trapezoid(min,max,step,_p_Bueno);
 MF_xfl_trapezoid Malo = new MF_xfl_trapezoid(min,max,step,_p_Malo);
 MF_xfl_singleton Caido = new MF_xfl_singleton(min,max,step,_p_Caido);
}
//+------------------------------------------------------+//
//  Type TP_T_UR  //
//+------------------------------------------------------+//
private class TP_T_UR {
 private double min = 0.0;
 private double max = 100.0;
 private double step = 1.0;
 double _p_Normal[] = { -1.0,0.0,35.0,50.0 };
 double _p_Alto[] = { 40.0,60.0,80.0,90.0 };
 double _p_Excesivo[] = { 80.0,90.0,100.0,101.0 };
 MF_xfl_trapezoid Normal = new MF_xfl_trapezoid(min,max,step,_p_Normal);
 MF_xfl_trapezoid Alto = new MF_xfl_trapezoid(min,max,step,_p_Alto);
 MF_xfl_trapezoid Excesivo = new MF_xfl_trapezoid(min,max,step,_p_Excesivo);
}
//+------------------------------------------------------+//
//  Type TP_T_Posibilidad4  //
//+------------------------------------------------------+//
```

```
 private class TP_T_Posibilidad4 {
  private double min = 0.0;
  private double max = 100.0;
  private double step = 1.0;
  double _p_Baja[] = { -1.0,0.0,15.0,35.0 };
  double _p_Media[] = { 20.0,30.0,50.0,65.0 };
  double _p_Alta[] = { 60.0,70.0,85.0,90.0 };
  double _p_Excesiva[] = { 80.0,90.0,100.0,101.0 };
  MF_xfl_trapezoid Baja = new MF_xfl_trapezoid(min,max,step,_p_Baja);
  MF_xfl_trapezoid Media = new MF_xfl_trapezoid(min,max,step,_p_Media);
  MF_xfl_trapezoid Alta = new MF_xfl_trapezoid(min,max,step,_p_Alta);
  MF_xfl_trapezoid Excesiva = new MF_xfl_trapezoid(min,max,step,_p_Excesiva);
  }
 //+------------------------------------------------------+//
 //  Type TP_T_Posibilidad5  //
 //+------------------------------------------------------+//
 private class TP_T_Posibilidad5 {
  private double min = 0.0;
  private double max = 100.0;
  private double step = 1.0;
  double _p_Muy_Baja[] = { -1.0,0.0,10.0,15.0 };
  double _p_Baja[] = { 10.0,20.0,25.0,35.0 };
  double _p_Media[] = { 30.0,40.0,60.0,65.0 };
  double _p_Alta[] = { 60.0,70.0,85.0,90.0 };
  double _p_Excesiva[] = { 85.0,90.0,100.0,101.0 };
  MF_xfl_trapezoid Muy_Baja = new MF_xfl_trapezoid(min,max,step,_p_Muy_Baja);
  MF_xfl_trapezoid Baja = new MF_xfl_trapezoid(min,max,step,_p_Baja);
  MF_xfl_trapezoid Media = new MF_xfl_trapezoid(min,max,step,_p_Media);
  MF_xfl_trapezoid Alta = new MF_xfl_trapezoid(min,max,step,_p_Alta);
  MF_xfl_trapezoid Excesiva = new MF_xfl_trapezoid(min,max,step,_p_Excesiva);
  }
 //+------------------------------------------------------+//
 //  Rulebase RL_R_Logins_fallidos  //
 //+------------------------------------------------------+//
 private MembershipFunction[] RL_R_Logins_fallidos(MembershipFunction ILF, MembershipFunc-
tion OIL) {
  double _rl;
  double _input[] = new double[2];
  if(ILF instanceof FuzzySingleton)
    _input[0] = ((FuzzySingleton) ILF).getValue();
  if(OIL instanceof FuzzySingleton)
    _input[1] = ((FuzzySingleton) OIL).getValue();
  OP_opset _op = new OP_opset();
  OutputMembershipFunction Logins_fallidos = new OutputMembershipFunction();
  Logins_fallidos.set(9,_op,_input);
  TP_T_ILF _t_ILF = new TP_T_ILF();
  TP_T_OIL _t_OIL = new TP_T_OIL();
  TP_T_Posibilidad4 _t_Logins_fallidos = new TP_T_Posibilidad4();
  int _i_Logins_fallidos=0;
  _rl = _op.and(_t_ILF.Normales.isEqual(ILF),_t_OIL.Normales.isEqual(OIL));
  Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Baja); _i_Logins_fallidos++;
```

```
 _rl = _op.and(_t_ILF.Normales.isEqual(ILF),_t_OIL.Altos.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Baja); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Normales.isEqual(ILF),_t_OIL.Excesivos.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Media); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Altos.isEqual(ILF),_t_OIL.Normales.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Media); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Altos.isEqual(ILF),_t_OIL.Altos.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Alta); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Altos.isEqual(ILF),_t_OIL.Excesivos.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Alta); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Excesivos.isEqual(ILF),_t_OIL.Normales.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Excesiva); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Excesivos.isEqual(ILF),_t_OIL.Altos.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Excesiva); _i_Logins_fallidos++;
 _rl = _op.and(_t_ILF.Excesivos.isEqual(ILF),_t_OIL.Excesivos.isEqual(OIL));
 Logins_fallidos.set(_i_Logins_fallidos,_rl, _t_Logins_fallidos.Excesiva); _i_Logins_fallidos++;
 MembershipFunction[] _output = new MembershipFunction[1];
 _output[0] = new FuzzySingleton(Logins_fallidos.defuzzify());
 return _output;
 }
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //  Rulebase RL_R_Uso_recursos  //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private MembershipFunction[] RL_R_Uso_recursos(MembershipFunction UC, MembershipFunction
UM) {
 double _rl;
 double _input[] = new double[2];
 if(UC instanceof FuzzySingleton)
  _input[0] = ((FuzzySingleton) UC).getValue();
 if(UM instanceof FuzzySingleton)
  _input[1] = ((FuzzySingleton) UM).getValue();
 OP_opset _op = new OP_opset();
 OutputMembershipFunction Uso_recursos = new OutputMembershipFunction();
 Uso_recursos.set(9,_op,_input);
 TP_T_UC _t_UC = new TP_T_UC();
 TP_T_UM _t_UM = new TP_T_UM();
 TP_T_UR _t_Uso_recursos = new TP_T_UR();
 int _i_Uso_recursos=0;
 _rl = _op.and(_t_UC.Normal.isEqual(UC),_t_UM.Normal.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Normal); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Normal.isEqual(UC),_t_UM.Mucho.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Normal); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Normal.isEqual(UC),_t_UM.Demasiado.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Alto); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Mucho.isEqual(UC),_t_UM.Normal.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Normal); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Mucho.isEqual(UC),_t_UM.Mucho.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Alto); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Mucho.isEqual(UC),_t_UM.Demasiado.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Alto); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Demasiado.isEqual(UC),_t_UM.Normal.isEqual(UM));
```

```
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Alto); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Demasiado.isEqual(UC),_t_UM.Mucho.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Excesivo); _i_Uso_recursos++;
 _rl = _op.and(_t_UC.Demasiado.isEqual(UC),_t_UM.Demasiado.isEqual(UM));
 Uso_recursos.set(_i_Uso_recursos,_rl, _t_Uso_recursos.Excesivo); _i_Uso_recursos++;
 MembershipFunction[] _output = new MembershipFunction[1];
 _output[0] = new FuzzySingleton(Uso_recursos.defuzzify());
 return _output;
 }
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //  Rulebase RL_R_Intrusion  //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private MembershipFunction[] RL_R_Intrusion(MembershipFunction LF, MembershipFunction
CEU, MembershipFunction CPI) {
 double _rl;
 double _input[] = new double[3];
 if(LF instanceof FuzzySingleton)
  _input[0] = ((FuzzySingleton) LF).getValue();
 if(CEU instanceof FuzzySingleton)
  _input[1] = ((FuzzySingleton) CEU).getValue();
 if(CPI instanceof FuzzySingleton)
  _input[2] = ((FuzzySingleton) CPI).getValue();
 OP_opset _op = new OP_opset();
 OutputMembershipFunction Pos_Intrusion = new OutputMembershipFunction();
 Pos_Intrusion.set(36,_op,_input);
 TP_T_Posibilidad4 _t_LF = new TP_T_Posibilidad4();
 TP_T_CEU _t_CEU = new TP_T_CEU();
 TP_T_CPI _t_CPI = new TP_T_CPI();
 TP_T_Posibilidad4 _t_Pos_Intrusion = new TP_T_Posibilidad4();
 int _i_Pos_Intrusion=0;
 _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Normales.isEqua
l(CPI));
 Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Baja); _i_Pos_Intrusion++;
 _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Altas.isEqual(CP
I));
 Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Baja); _i_Pos_Intrusion++;
 _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Excesivas.isEqua
l(CPI));
 Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Baja); _i_Pos_Intrusion++;
 _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Normales.isEqual(CP
I));
 Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
 _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Altas.isEqual(CPI));
 Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
```

```
  _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Excesivas.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Normales.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Altas.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Baja.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Excesivas.isEqu
al(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Normales.isEq
ual(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Baja); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Altas.isEqual(
CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Excesivas.isEq
ual(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Normales.isEqual(
CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Altas.isEqual(CPI))
;
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Excesivas.isEqual(
CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Normales.isEq
ual(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Altas.isEqual(
CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Media.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Excesivas.isEq
ual(CPI));
```

```
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Normales.isEqual
(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Altas.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Media); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Excesivas.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Normales.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Altas.isEqual(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Excesivas.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Normales.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Altas.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Alta.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Excesivas.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Normales.is
Equal(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Altas.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Normales.isEqual(CEU)),_t_CPI.Excesivas.is
Equal(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Normales.isEqua
l(CPI));
```

```
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Alta); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Altas.isEqual(CP
I));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Altas.isEqual(CEU)),_t_CPI.Excesivas.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Normales.is
Equal(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Altas.isEqua
l(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  _rl =
_op.and(_op.and(_t_LF.Excesiva.isEqual(LF),_t_CEU.Excesivas.isEqual(CEU)),_t_CPI.Excesivas.is
Equal(CPI));
  Pos_Intrusion.set(_i_Pos_Intrusion,_rl, _t_Pos_Intrusion.Excesiva); _i_Pos_Intrusion++;
  MembershipFunction[] _output = new MembershipFunction[1];
  _output[0] = new FuzzySingleton(Pos_Intrusion.defuzzify());
  return _output;
 }
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //  Rulebase RL_R_Borrado_H  //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private MembershipFunction[] RL_R_Borrado_H(MembershipFunction LM, MembershipFunction
CC, MembershipFunction SRE) {
  double _rl;
  double _input[] = new double[3];
  if(LM instanceof FuzzySingleton)
   _input[0] = ((FuzzySingleton) LM).getValue();
  if(CC instanceof FuzzySingleton)
   _input[1] = ((FuzzySingleton) CC).getValue();
  if(SRE instanceof FuzzySingleton)
   _input[2] = ((FuzzySingleton) SRE).getValue();
  OP_opset _op = new OP_opset();
  OutputMembershipFunction Pos_Huellas_B = new OutputMembershipFunction();
  Pos_Huellas_B.set(27,_op,_input);
  TP_T_LM _t_LM = new TP_T_LM();
  TP_T_CC _t_CC = new TP_T_CC();
  TP_T_SRE _t_SRE = new TP_T_SRE();
  TP_T_Posibilidad4 _t_Pos_Huellas_B = new TP_T_Posibilidad4();
  int _i_Pos_Huellas_B=0;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Normales.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Baja); _i_Pos_Huellas_B++;
```

```
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Altos.isEqua
l(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Baja); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Excesivos.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Normales.isEq
ual(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Altos.isEqual(S
RE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Excesivos.isEq
ual(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Normales.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Altos.isEqua
l(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Normales.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Excesivos.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Normales.isEqua
l(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Baja); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Altos.isEqual(SR
E));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Excesivos.isEqua
l(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Normales.isEqual(S
RE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Altos.isEqual(SRE)
);
```

```
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Excesivos.isEqual(
SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Normales.isEqua
l(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Altos.isEqual(SR
E));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Altos.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Excesivos.isEqu
al(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Normales.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Media); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Altos.isEqua
l(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Normales.isEqual(CC)),_t_SRE.Excesivos.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Normales.isEq
ual(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Alta); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Altos.isEqual(
SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Alguna.isEqual(CC)),_t_SRE.Excesivos.isEq
ual(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Normales.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Altos.isEqu
al(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
```

```
  _rl =
_op.and(_op.and(_t_LM.Excesivos.isEqual(LM),_t_CC.Excesivas.isEqual(CC)),_t_SRE.Excesivos.is
Equal(SRE));
  Pos_Huellas_B.set(_i_Pos_Huellas_B,_rl, _t_Pos_Huellas_B.Excesiva); _i_Pos_Huellas_B++;
  MembershipFunction[] _output = new MembershipFunction[1];
  _output[0] = new FuzzySingleton(Pos_Huellas_B.defuzzify());
  return _output;
 }
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //  Rulebase RL_R_Aprovechamiento  //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private MembershipFunction[] RL_R_Aprovechamiento(MembershipFunction CF, Membership-
Function BF, MembershipFunction UR, MembershipFunction ES) {
  double _rl;
  double _input[] = new double[4];
  if(CF instanceof FuzzySingleton)
   _input[0] = ((FuzzySingleton) CF).getValue();
  if(BF instanceof FuzzySingleton)
   _input[1] = ((FuzzySingleton) BF).getValue();
  if(UR instanceof FuzzySingleton)
   _input[2] = ((FuzzySingleton) UR).getValue();
  if(ES instanceof FuzzySingleton)
   _input[3] = ((FuzzySingleton) ES).getValue();
  OP_opset _op = new OP_opset();
  OutputMembershipFunction Pos_Aprovechamiento = new OutputMembershipFunction();
  Pos_Aprovechamiento.set(81,_op,_input);
  TP_T_CF _t_CF = new TP_T_CF();
  TP_T_BF _t_BF = new TP_T_BF();
  TP_T_UR _t_UR = new TP_T_UR();
  TP_T_ES _t_ES = new TP_T_ES();
  TP_T_Posibilidad4 _t_Pos_Aprovechamiento = new TP_T_Posibilidad4();
  int _i_Pos_Aprovechamiento=0;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Normal
.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Bueno.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqua
l(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Norma
l.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.isEqua
l(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isEqual
(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqual(U
R)),_t_ES.Bueno.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.isEqu
al(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.isEqu
al(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Norma
l.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqu
al(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Norm
al.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Bueno.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Normal
.isEqual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqua
l(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Norma
l.isEqual(UR)),_t_ES.Malo.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.isEqua
l(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Baja);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isEqual
(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqual(U
R)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.isEqu
al(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.isEqu
al(UR)),_t_ES.Malo.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Norma
l.isEqual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Media);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Alta);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqu
al(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Norm
al.isEqual(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Malo.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Malo.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Normal
.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqua
l(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Norma
l.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Normales.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Caido.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.isEqua
l(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isEqual
(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqual(U
R)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.isEqu
al(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.isEqu
al(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Altos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Norma
l.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
```

```
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Normales.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Normal.isE
qual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Alto.isEqu
al(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Altos.isEqual(BF)),_t_UR.Excesivo.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Norm
al.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Alto.is
Equal(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  _rl =
_op.and(_op.and(_op.and(_t_CF.Excesivos.isEqual(CF),_t_BF.Excesivos.isEqual(BF)),_t_UR.Excesi
vo.isEqual(UR)),_t_ES.Caido.isEqual(ES));
  Pos_Aprovechamiento.set(_i_Pos_Aprovechamiento,_rl, _t_Pos_Aprovechamiento.Excesiva);
_i_Pos_Aprovechamiento++;
  MembershipFunction[] _output = new MembershipFunction[1];
  _output[0] = new FuzzySingleton(Pos_Aprovechamiento.defuzzify());
  return _output;
 }
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //  Rulebase RL_R_Ataque  //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 private MembershipFunction[] RL_R_Ataque(MembershipFunction P_I, MembershipFunction
P_HB, MembershipFunction P_A) {
  double _rl;
  double _input[] = new double[3];
  if(P_I instanceof FuzzySingleton)
   _input[0] = ((FuzzySingleton) P_I).getValue();
  if(P_HB instanceof FuzzySingleton)
```

```
  _input[1] = ((FuzzySingleton) P_HB).getValue();
 if(P_A instanceof FuzzySingleton)
  _input[2] = ((FuzzySingleton) P_A).getValue();
 OP_opset _op = new OP_opset();
 OutputMembershipFunction Pos_Ataque = new OutputMembershipFunction();
 Pos_Ataque.set(64,_op,_input);
 TP_T_Posibilidad4 _t_P_I = new TP_T_Posibilidad4();
 TP_T_Posibilidad4 _t_P_HB = new TP_T_Posibilidad4();
 TP_T_Posibilidad4 _t_P_A = new TP_T_Posibilidad4();
 TP_T_Posibilidad5 _t_Pos_Ataque = new TP_T_Posibilidad5();
 int _i_Pos_Ataque=0;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_A)
);
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Muy_Baja); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Media.isEqual(P_
A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Muy_Baja); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_A)
);
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Excesiva.isEqual(
P_A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_
A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Baja); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Media.isEqual(P
_A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_
A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Excesiva.isEqua
l(P_A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_A)
);
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Baja); _i_Pos_Ataque++;
 _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Media.isEqual(P_
A));
 Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
```

```
  _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_A)
);
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Excesiva.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Baja.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Media.isEqua
l(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Alta.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Baja.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Excesiva.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Baja); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Media.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Baja); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Excesiva.isEqua
l(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Baja.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Media.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Alta.isEqual(P
_A));
```

```
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Excesiva.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Media.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Excesiva.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Baja.isEqua
l(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Media.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Alta.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Media.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Excesiva.is
Equal(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_A)
);
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Media.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_A)
);
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
```

```
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Excesiva.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Media); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Media.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Excesiva.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Baja.isEqual(P_A)
);
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Media.isEqual(P_
A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Alta.isEqual(P_A))
;
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Excesiva.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Baja.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Media.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Alta.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Alta.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Excesiva.isEq
ual(P_A));
```

```
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Baja.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Media.isEqua
l(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Alta.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Baja.isEqual(P_HB)),_t_P_A.Excesiva.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Baja.isEqua
l(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Media.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Alta.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Media.isEqual(P_HB)),_t_P_A.Excesiva.is
Equal(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Baja.isEqual(
P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Media.isEqual
(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Alta); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Alta.isEqual(P
_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Alta.isEqual(P_HB)),_t_P_A.Excesiva.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
```

```
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Baja.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Media.is
Equal(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Alta.isEq
ual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  _rl =
_op.and(_op.and(_t_P_I.Excesiva.isEqual(P_I),_t_P_HB.Excesiva.isEqual(P_HB)),_t_P_A.Excesiva.
isEqual(P_A));
  Pos_Ataque.set(_i_Pos_Ataque,_rl, _t_Pos_Ataque.Excesiva); _i_Pos_Ataque++;
  MembershipFunction[] _output = new MembershipFunction[1];
  _output[0] = new FuzzySingleton(Pos_Ataque.defuzzify());
  return _output;
 }
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 //              Fuzzy Inference Engine                //
 //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
 public double[] crispInference(double[] _input) {
  MembershipFunction Fallos_login_facil = new FuzzySingleton(_input[0]);
  MembershipFunction Otros_fallos_login = new FuzzySingleton(_input[1]);
  MembershipFunction Comandos_enviados_en_URL = new FuzzySingleton(_input[2]);
  MembershipFunction Cabeceras_protocolo_inconsistentes = new FuzzySingleton(_input[3]);
  MembershipFunction Logs_modificados = new FuzzySingleton(_input[4]);
  MembershipFunction Cuentas_creadas = new FuzzySingleton(_input[5]);
  MembershipFunction Servicios_red_ejecutados = new FuzzySingleton(_input[6]);
  MembershipFunction Copia_ficheros = new FuzzySingleton(_input[7]);
  MembershipFunction Borrado_ficheros = new FuzzySingleton(_input[8]);
  MembershipFunction Uso_CPU = new FuzzySingleton(_input[9]);
  MembershipFunction Uso_memoria = new FuzzySingleton(_input[10]);
  MembershipFunction Estado_servidor = new FuzzySingleton(_input[11]);
  MembershipFunction Pos_Ataque;
  MembershipFunction i0;
  MembershipFunction i1;
  MembershipFunction i2;
  MembershipFunction i3;
  MembershipFunction i4;
  MembershipFunction[] _call;
  _call = RL_R_Logins_fallidos(Fallos_login_facil,Otros_fallos_login); i0=_call[0];
  _call = RL_R_Intrusion(i0,Comandos_enviados_en_URL,Cabeceras_protocolo_inconsistentes);
i1=_call[0];
  _call = RL_R_Borrado_H(Logs_modificados,Cuentas_creadas,Servicios_red_ejecutados);
i2=_call[0];
  _call = RL_R_Uso_recursos(Uso_CPU,Uso_memoria); i3=_call[0];
  _call = RL_R_Aprovechamiento(Copia_ficheros,Borrado_ficheros,i3,Estado_servidor); i4=_call[0];
  _call = RL_R_Ataque(i1,i2,i4); Pos_Ataque=_call[0];
```

```
 double _output[] = new double[1];
 if(Pos_Ataque instanceof FuzzySingleton)
  _output[0] = ((FuzzySingleton) Pos_Ataque).getValue();
 else _output[0] = ((OutputMembershipFunction) Pos_Ataque).defuzzify();
 return _output;
 }
 public double[] crispInference(MembershipFunction[] _input) {
 MembershipFunction Fallos_login_facil = _input[0];
 MembershipFunction Otros_fallos_login = _input[1];
 MembershipFunction Comandos_enviados_en_URL = _input[2];
 MembershipFunction Cabeceras_protocolo_inconsistentes = _input[3];
 MembershipFunction Logs_modificados = _input[4];
 MembershipFunction Cuentas_creadas = _input[5];
 MembershipFunction Servicios_red_ejecutados = _input[6];
 MembershipFunction Copia_ficheros = _input[7];
 MembershipFunction Borrado_ficheros = _input[8];
 MembershipFunction Uso_CPU = _input[9];
 MembershipFunction Uso_memoria = _input[10];
 MembershipFunction Estado_servidor = _input[11];
 MembershipFunction Pos_Ataque;
 MembershipFunction i0;
 MembershipFunction i1;
 MembershipFunction i2;
 MembershipFunction i3;
 MembershipFunction i4;
 MembershipFunction[] _call;
  _call = RL_R_Logins_fallidos(Fallos_login_facil,Otros_fallos_login); i0=_call[0];
  _call = RL_R_Intrusion(i0,Comandos_enviados_en_URL,Cabeceras_protocolo_inconsistentes);
i1=_call[0];
  _call = RL_R_Borrado_H(Logs_modificados,Cuentas_creadas,Servicios_red_ejecutados);
i2=_call[0];
  _call = RL_R_Uso_recursos(Uso_CPU,Uso_memoria); i3=_call[0];
  _call = RL_R_Aprovechamiento(Copia_ficheros,Borrado_ficheros,i3,Estado_servidor); i4=_call[0];
  _call = RL_R_Ataque(i1,i2,i4); Pos_Ataque=_call[0];
 double _output[] = new double[1];
 if(Pos_Ataque instanceof FuzzySingleton)
  _output[0] = ((FuzzySingleton) Pos_Ataque).getValue();
 else _output[0] = ((OutputMembershipFunction) Pos_Ataque).defuzzify();
 return _output;
 }
 public MembershipFunction[] fuzzyInference(double[] _input) {
 MembershipFunction Fallos_login_facil = new FuzzySingleton(_input[0]);
 MembershipFunction Otros_fallos_login = new FuzzySingleton(_input[1]);
 MembershipFunction Comandos_enviados_en_URL = new FuzzySingleton(_input[2]);
 MembershipFunction Cabeceras_protocolo_inconsistentes = new FuzzySingleton(_input[3]);
 MembershipFunction Logs_modificados = new FuzzySingleton(_input[4]);
 MembershipFunction Cuentas_creadas = new FuzzySingleton(_input[5]);
 MembershipFunction Servicios_red_ejecutados = new FuzzySingleton(_input[6]);
 MembershipFunction Copia_ficheros = new FuzzySingleton(_input[7]);
 MembershipFunction Borrado_ficheros = new FuzzySingleton(_input[8]);
 MembershipFunction Uso_CPU = new FuzzySingleton(_input[9]);
```

```
  MembershipFunction Uso_memoria = new FuzzySingleton(_input[10]);
  MembershipFunction Estado_servidor = new FuzzySingleton(_input[11]);
  MembershipFunction Pos_Ataque;
  MembershipFunction i0;
  MembershipFunction i1;
  MembershipFunction i2;
  MembershipFunction i3;
  MembershipFunction i4;
  MembershipFunction[] _call;
  _call = RL_R_Logins_fallidos(Fallos_login_facil,Otros_fallos_login); i0=_call[0];
  _call = RL_R_Intrusion(i0,Comandos_enviados_en_URL,Cabeceras_protocolo_inconsistentes);
i1=_call[0];
  _call = RL_R_Borrado_H(Logs_modificados,Cuentas_creadas,Servicios_red_ejecutados);
i2=_call[0];
  _call = RL_R_Uso_recursos(Uso_CPU,Uso_memoria); i3=_call[0];
  _call = RL_R_Aprovechamiento(Copia_ficheros,Borrado_ficheros,i3,Estado_servidor); i4=_call[0];
  _call = RL_R_Ataque(i1,i2,i4); Pos_Ataque=_call[0];
  MembershipFunction _output[] = new MembershipFunction[1];
  _output[0] = Pos_Ataque;
  return _output;
 }
 public MembershipFunction[] fuzzyInference(MembershipFunction[] _input) {
  MembershipFunction Fallos_login_facil = _input[0];
  MembershipFunction Otros_fallos_login = _input[1];
  MembershipFunction Comandos_enviados_en_URL = _input[2];
  MembershipFunction Cabeceras_protocolo_inconsistentes = _input[3];
  MembershipFunction Logs_modificados = _input[4];
  MembershipFunction Cuentas_creadas = _input[5];
  MembershipFunction Servicios_red_ejecutados = _input[6];
  MembershipFunction Copia_ficheros = _input[7];
  MembershipFunction Borrado_ficheros = _input[8];
  MembershipFunction Uso_CPU = _input[9];
  MembershipFunction Uso_memoria = _input[10];
  MembershipFunction Estado_servidor = _input[11];
  MembershipFunction Pos_Ataque;
  MembershipFunction i0;
  MembershipFunction i1;
  MembershipFunction i2;
  MembershipFunction i3;
  MembershipFunction i4;
  MembershipFunction[] _call;
  _call = RL_R_Logins_fallidos(Fallos_login_facil,Otros_fallos_login); i0=_call[0];
  _call = RL_R_Intrusion(i0,Comandos_enviados_en_URL,Cabeceras_protocolo_inconsistentes);
i1=_call[0];
  _call = RL_R_Borrado_H(Logs_modificados,Cuentas_creadas,Servicios_red_ejecutados);
i2=_call[0];
  _call = RL_R_Uso_recursos(Uso_CPU,Uso_memoria); i3=_call[0];
  _call = RL_R_Aprovechamiento(Copia_ficheros,Borrado_ficheros,i3,Estado_servidor); i4=_call[0];
  _call = RL_R_Ataque(i1,i2,i4); Pos_Ataque=_call[0];
  MembershipFunction _output[] = new MembershipFunction[1];
  _output[0] = Pos_Ataque;
```

```
  return _output;
 }
}
```