Tesis de licenciatura en Ciencias de la Computación

Sistema experto basado en lógica difusa para la Central Nuclear Embalse

-- Director
Lic. Osvaldo Gonzalez

-- Tesistas
Sergio Amitrano
Rodolfo Di Chiazza
Jorge Lorenzo

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales

ÍNDICE GLOBAL

GUÍA

El trabajo realizado consta del desarrollo de una aplicación basado en un estudio e investigación de un problema de monitoreo y diagnóstico. De esta manera, la documentación presentada se encuentra dividida en dos partes: la documentación de estudio e investigación y la documentación de desarrollo.

La **documentación de desarrollo** constituye la documentación del sistema. En este documento se describen, utilizando la metodología de desarrollo estructurado, las fases de relevamiento, análisis y diseño. A su vez se cuenta con los agregados de las minutas de las reuniones realizadas con los usuarios más la información correspondiente al sistema manual actualmente en uso.

A través de esta documentación se explican los aspectos convencionales del desarrollo sin profundizar en cada una de las etapas, dado que los aportes fundamentales del trabajo se encuentran descriptos en la documentación de estudio e investigación.

La **documentación de estudio e investigación** cubre, a través de sus distintos capítulos, los puntos principales del trabajo. En dicha documentación se encuentra la siguiente clasificación.

En los **capítulos I** y **II** se realiza una introducción general al trabajo presentado, se hace una descripción de las motivaciones que llevaron a su desarrollo y se presenta un breve resumen del problema tratado.

En el **capítulo III** se introducen los puntos fundamentales de tres temas relacionados con el trabajo: **Lógica Difusa, Sistemas Expertos y Sistemas Expertos Difusos**. En este capítulo se repasan los conceptos más importantes que es necesario conocer para comprender los capítulos siguientes.

En el **capítulo IV** se decriben los primeros pasos en el desarrollo del sistema de ayuda, el marco donde se ubicó inicialmente el trabajo y una definición preliminar del problema planteado. Adicionalmente se incluye un análisis de las razones que llevan a justificar la decisión del desarrollo de un sistema experto para atacar el problema planteado por **N**ucleoenergética **A**rgentina **S**. **A**. (N.A.S.A. en lo sucesivo).

En el **capítulo V** se desarrolla uno de los puntos principales del trabajo presentado. En este capítulo se describe, entre otras cosas, el desarrollo formal de un lenguaje difuso de especificación de las reglas del sistema experto.

En el **capítulo VI** se describen las características principales de la implementación del sistema experto.

Finalmente, en el **capítulo VII**, se finaliza el trabajo describiendo las conclusiones principales a las que se ha arribado.

I. RESUMEN

El trabajo aquí presentado cubre la investigación de un problema de supervisión y diagnóstico en la Central Nuclear Embalse. Los operadores de esta central manejan procedimientos de operación ante posibles eventos anormales. El problema consiste en detectar la presencia de dichos eventos anormales, para activar el procedimiento de operación correspondiente. La solución propuesta incluye la definición de un lenguaje que se adapta a las necesidades observadas. Los conceptos que se manejan en el contexto del problema tienen características difusas, y esto es contemplado para darle la expresividad necesaria al lenguaje. El trabajo también propone un sistema experto para deducir nueva información a partir de las reglas expresadas por los expertos y del estado actual de la planta. La implementación final, que consiste del sistema experto basado en un lenguaje con características difusas, fue entregada a los expertos para su puesta en funcionamiento. Los mismos, tras quedar conformes con los resultados obtenidos, proyectan la futura puesta en producción.

Palabras clave: Lógica Difusa, Sistemas Expertos, Programación Lógica.

Abstract

The report we here enclose covers research on a supervision and diagnosis problem in Embalse Nuclear Station. The operators in this station handle operating procedures in case of possible abnormal events. The problem consists in spotting the presence of such abnormal events to activate the corresponding operating procedure. The proposed solution includes the definition of a language that can be adapted to these needs. The concepts used in the context of the problem have vague characteristics in order to provide the language with the necessary expressivity. The report also proposes an expert system to infer new information from the rules designed by experts and from the current state of the station. The final implementation, which consists in an expert system based on a language with vague characteristics, was handed in to the experts to be carried out. The experts, after approving the achievements, are planning the future implementation.

Keywords: Fuzzy Logic, Expert Systems, Logic Programming.

II. INTRODUCCIÓN AL TRABAJO REALIZADO

II.A. Motivación

II.A.1. Comienzo - Presentación

Los antecedentes de la presente tesis se originan en un pedido de **N**ucleoenergética **A**rgentina **S. A**. (N.A.S.A.). El requerimiento original fue diseñar e implementar un subsistema que supervise y detecte eventos anormales en la **C**entral **N**uclear **E**mbalse (C.N.E.). Dicho subsistema forma parte de un sistema integral. Mientras que el subsistema que es motivo de este trabajo tiene como objetivo el detectar los eventos anormales producidos, otro subsistema actúa una vez detectados los mismos presentando a los operadores el curso de acción - o procedimiento de operación - correspondiente para efectuar la mitigación. La integración final de ambos subsistemas quedará a cargo de los responsables de N.A.S.A.

II.A.2. Descripción y Antecedentes en la C.N.E.

La Central Nuclear Embalse es una planta nuclear de tipo CANDU 600 que genera energía eléctrica. Está ubicada en Embalse Río Tercero, provincia de Córdoba, y tiene un reactor en su interior cuya potencia es de 648 MWe. Fue construida en 1983, y pertenece a la Comisión Nacional de Energía Atómica (CNEA).

Las instalaciones de la C.N.E. cuentan con una Sala de Control Principal (SCP) y una Sala de Control Secundaria (SCS). En la SCP, un equipo de operadores cuenta con la información y controles necesarios para la operación segura de la planta. La SCS cuenta con la instrumentación y el aparataje necesario para que, en caso de que los operadores deban abandonar la SCP por alguna razón, el reactor pueda ser detenido en forma segura.

El control automático de planta en la C.N.E. es realizado por dos computadoras funcionalmente idénticas, estando una de ellas en operación (computadora DCCX) y la otra en reserva (computadora DCCY). Las operaciones fundamentales de estas computadoras son el control de diferentes sistemas y dispositivos de la planta, como ser el reactor, la turbina, la presión del sistema principal de transporte de calor y vapor, los niveles de los generadores de vapor, etc..

Las operaciones de mitigación de los eventos anormales en la C.N.E. estaban inicialmente documentadas en un manual, denominado Manual de Eventos Anormales. Su volumen, formato y filosofía de diagnóstico hacían dificultoso su seguimiento teniendo en cuenta el estado emocional del operador ante una emergencia. Posteriormente se reemplazó el Manual de Eventos Anormales por una nueva y mejorada documentación, denominada **P**rocedimientos de **O**peración para **E**ventos **A**normales (POEAs). Estos nuevos documentos operativos, diseñados en forma de diagramas lógicos, incluyen el diagnóstico específico de cada evento anormal.

La C.N.E. definió la necesidad de incrementar la confiabilidad y seguridad de la operatoria de la planta mediante la incorporación de un sistema computarizado de ayuda al operador que esté basado en los Procedimientos de Operación para Eventos Anormales. Para ver la especificación original del sistema pedido por N.A.S.A., ver [BAT/2].

II.B. Descripción del problema: Detección y mitigación de eventos anormales

II.B.1. Resumen

Las tareas de detección y mitigación de eventos anormales en la C.N.E. son llevadas a cabo por un grupo de operadores presentes en la Sala de Control Principal (S.C.P.) de la planta. Estos últimos observan habitualmente los instrumentos presentes en la S.C.P. que indican el estado actual del reactor y de la planta en general. La C.N.E. cuenta con documentación escrita que es utilizada por dichos operadores. Cuando se detecta una condición anormal en el reactor, se consulta la documentación para determinar cuál es el procedimiento adecuado para mitigar la situación presentada. Esta documentación indica además los pasos a seguir por el operador dependiendo del evento anormal detectado.

II.B.2. Especificación

El proyecto abarca la automatización en la detección de los eventos anormales que se presentan en la C.N.E.. La manera propuesta para lograr esto es mediante un sistema experto que incluya dentro de sus reglas y su base de conocimientos los procedimientos actualmente utilizados por los operadores de la planta, logrando de esta manera ayudar a éstos últimos en su tarea habitual de monitoreo y control del reactor. Dichos procedimientos se encuentran especificados en la documentación denominada POEAs (**P**rocedimientos de **O**peración ante **E**ventos **A**normales), el cuál se utiliza actualmente en la C.N.E.

La documentación del sistema de POEAs contiene diagramas que indican gráficamente las condiciones que se deben cumplir para determinar la existencia de un evento anormal conocido, y las acciones a seguir en cada caso en formato de diagrama de flujo. Los POEAs se encuentran numerados y actualmente existen catorce (14) POEAs vigentes para la cobertura de otros tantos eventos anormales. La estructura principal de cada POEA es un diagrama lógico, donde la primera parte corresponde al diagnóstico del evento anormal, y la segunda parte corresponde a las acciones a tomar parar mitigar el evento correspondiente.

Para que se lleven a cabo las acciones de mitigación correspondientes a un determinado POEA, es necesario que se cumplan las condiciones que están documentadas en forma gráfica en la primer parte de la especificación del procedimiento (conocidas como *condiciones de entrada*). Estas condiciones de entrada se encuentran representadas en forma de íconos para su rápida verificación, junto con información complementaria.

Debido a la revisión y actualización a la cual son sometidos los POEAs, el sistema a desarrollar debía ser fácilmente extensible. Para ello, se debería poder expresar las nuevas reglas que involucren un nuevo procedimiento de manera tal que sea lo más parecido al lenguaje propio del dominio. De esta manera se logra que el experto pueda agregar nuevas reglas para ampliar la base de conocimientos del sistema, en la medida que se generen nuevos procedimientos de operación.

La función principal del sistema final es ayudar al grupo de operadores a detectar eventos anormales en la Central, y, más generalmente, colaborar en el reconocimiento constante del estado físico real de la planta y en la toma de decisión sobre el procedimiento a seguir ante cualquier anormalidad. Dichas tareas involucran monitorear periódicamente una gran cantidad de señales, dispositivos, medidores, mensajes de computadora, alarmas, gráficos de tendencias y demás controles implementados y centralizados en la S.C.P. de la Central. La tarea del grupo de operadores es muy difícil, debido a la gran cantidad de controles que deben realizarse para asegurar que la planta se encuentra ante un evento anormal catalogado dentro del sistema de POEAs. Es por esta razón que el sistema pedido debe aliviar la carga de trabajo del grupo de operadores, brindandoles más tiempo a los mismos para supervisar globalmente una situación dada. En los casos reales en los que se presenta un evento anormal, los operadores deben estar atentos a todas las señales que se presentan referentes al evento detectado, y esta tarea puede afectar la detección de otro posible evento anormal que se esté gestando en la planta, siendo este un ejemplo concreto de situaciones que han sucedido en la planta.

Durante las acciones operativas de mitigación, como se ha dicho antes, el operador debe monitorear una determinada cantidad de parámetros relacionados con la seguridad en forma continua y confiable. Esta importante tarea puede no ser suficientemente confiable dependiendo de la complejidad del evento o de la simultaneidad de diferentes operaciones. Debido a esto, el sistema de ayuda al operador realizará una supervisión continua de esos parámetros que podrán ser críticos, dependiendo del tipo de evento que se trate.

Los eventos anormales que se consideran como posibles dentro de la C.N.E. incluyen también incidentes de operación, para los cuales es necesario recuperar el estado de funcionamiento normal o, en última instancia, llevar a la planta a una *parada segura*. También se consideran posibles estados "optimizables" de la planta, para los cuales existen procedimientos a seguir y acciones para lograr optimizar la productividad de la misma.

Al ser una tarea específicamente de ayuda, el sistema no deberá tomar decisiones de ningún tipo. Se limitará a efectuar un diagnóstico del estado de la planta e informar cualquier detección al grupo de operadores, quienes serán responsables de confirmar dicho diagnóstico y tomar las decisiones necesarias. Para que el sistema cumpla realmente la función de ayuda, el mismo deberá evitar recargar al operador con tareas extras. El sistema debe ser autosuficiente para realizar todas sus deducciones, y no deberá solicitar información a los operadores aún en el caso en que no se cuente con la misma; de lo contrario se estaría contradiciendo el objetivo a cumplir, que es lograr la mayor liberación de tareas de supervisión por parte de los operadores.

En la próxima página se detalla el POEA número 6.

BAJA PRESION AIRE INSTR.

ó C/I N° 1697 7512 P1
AIRE INSTRUMENTACION P <

PL 20-V 15

MAS UNA O MAS DE LAS SIGUIENTES

C/I N°			MENSAJE		
155	7512	PS1	AIRE INSTR TQ#1	P	<
156	7512	PS2	AIRE INSTR TQ#2	P	<
157	7512	PS3	AIRE INSTR TQ#3	P	<
968	7513	PS2	AIRE RESP. ED SCIO	P	<
969	7512	PS503	AIRE INSTR TQ#7	P	<
970	7512	PS504	AIRE INSTR TQ#6	P	<
971	7512	PS506	AIRE INSTR TQ#8	P	<
972	7512	PS507	AIRE INSTR TQ#9	P	<
973	7512	PS508	AIRE INSTR TQ#10	P	<
974	7512	PS509	AIRE INSTR TQ#11	P	<
975	7512	PS510	AIRE INSTR TQ#12	P	<
832	7512	P21	AIRE INSTR TQ#21	P	<
833	7512	P22	AIRE INSTR TQ#22	P	<
1708	7512	Z1#1	AIRE SERVICIO BOP FALTA		
1693	7511	P8	AIRE COMPRIMIDO	P	<

POSTERIORMENTE

REDUCCION DE POTENCIA ESCALONADA INICIADA	TEMPERATURA MODERADOR ALTA / BAJA	NIVEL MODERADOR ALTO / BAJO	PRESION GAS DE COBERTURA DEL MODERA-DOR ANORMAL	TANQUE DE AIRE AUXILIAR PRESION BAJA
PL6-V1	PL5-V1	PL6-V2	PL6-V4	PL1-V2

BLOQUEO TURBINA	GENERADOR DE VAPOR NIVEL ALTO	VALVULAS DE SEGURIDAD DEL SPV ABIERTA	PRESION AIRE INSTRUMENTOS VALV. SEG. SPV BAJA	NIVEL CONDENSADOR ANORMAL
PL13-V1	PL10-V1	PL3-V31	PL5-V35	PL12-V5

C/I 147 : 3231 P 3 GAS DE COBERTURA MDOR P >
C/I 346 : 3432 REF EMC1A VALV. PPAL ABIERTA
C/I 352 : 3732 RL 27 BAR AB STEP BACK INIC
C/I 683 : 3210 T 11 C TEMP SAL D2O CDRIA CNL C T >
C/I 684 : 3210 T 11 B TEMP SAL D2O CDRIA CNL B T >
C/I 685 : 3210 T 11 A TEMP SAL D2O CDRIA CNL A T >
C/I 686 : 3210 T 11 A TEMP D2O CDRIA CNL A T <
C/I 687 : 3210 T 11 B TEMP D2O CDRIA CNL B T <
C/I 688 : 3210 T 11 C TEMP D2O CDRIA CNL C T <
C/I 668 : 3210 L 13 A L D2O CDRIA CNL A ALTO L >
C/I 669 : 3210 L 13 B L D2O CDRIA CNL B ALTO L >
C/I 670 : 3210 L 13 C L D2O CDRIA CNL C ALTO L >
C/I 671 3210 L 13 A D2O CDRIA CNL A L <
C/I 672 3210 L 13 B D2O CDRIA CNL B L <
C/I 673 3210 L 13 C D2O CDRIA CNL C L <
C/I 641 3620 L 1 C GDOR VAP 1 L RGO ALTO L >
C/I 642 3620 L 2 C GDOR VAP 2 L RGO ALTO L >
C/I 643 3620 L 3 C GDOR VAP 3 L RGO ALTO L >
C/I 644 3620 L 4 C GDOR VAP 4 L RGO ALTO L >
C/I 1256 4100 PL 654 TURBINA BLOQUEO

Hasta aquí tenemos la primera parte del POEA número 6. Esta primer parte es la denominada "condiciones de entrada" del POEA. La segunda parte, correspondiente a las acciones a tomar para mitigar el evento anormal una vez detectado el mismo, no se transcribe dado que es irrelevante para este trabajo.

Para confirmar la detección de este evento anormal y llevar a cabo el procedimiento de operación correspondiente, deben darse las condiciones de entrada enumeradas anteriormente, las cuales representan condiciones sobre señales del reactor. Existen diversos "tipos" de señales, siendo en el ejemplo anterior alarmas, entradas de contacto y mensajes de computadora. Sin embargo, y como fue largamente comentado y analizado, existen estados del reactor "similares" a los que cumplen todas las condiciones, en los cuales se puede afirmar con "cierta veracidad" que hay una tendencia a que el evento anormal se produzca en un lapso de tiempo determinado. Estas heurísticas son las que posee el experto de la Central, y mediante las cuales los mismos operadores se guían cuando no hay un "matching" exacto entre la situación actual de la planta y las condiciones necesarias para detectar un evento anormal catalogado en el sistema de POEAs.

En este mismo ejemplo, tenemos que la activación de la alarma de baja presión de aire de instrumentación o la presencia del mensaje correspondiente a la entrada de contacto n°1697 provocan que fallen otros sistemas o dispositivos (lo cual es indicado por los mensajes que siguen a continuación de la frase "más una o más de las siguientes"). Como consecuencia de esto, en un lapso de tiempo determinado, aparecerá por lo menos algún mensaje de los que siguen a continuación de la frase "más una o más de las siguientes", proveniente de la computadora de la S.C.P. Ahora, dado que este efecto en cadena toma una cantidad de tiempo determinado, es posible que un estado dado de la planta indique la detección de la alarma o el mensaje que aparecen en la parte superior, y sin embargo todavía no se detecte ningún mensaje de los inferiores. Este caso indicaría una *tendencia* muy probable a que se desate el evento anormal en cuestión, y el sistema debería advertir dicha situación.

La teoría de conjuntos difusos y lógica difusa será utilizada para modelizar el lenguaje de definición de POEAs. Posteriormente, el sistema experto incluirá en su base de conocimientos información difusa, y el método de deducción utilizado también usará dicha teoría. En el próximo capítulo se hará una introducción a los conceptos básicos de la misma.

II.C. Objetivo

El objetivo principal de la presente tesis es definir un lenguaje propio para el dominio específico del problema, y a partir de éste construir un sistema experto que cumpla la función de detección de eventos anormales en la C.N.E.

III. INTRODUCCIÓN A CONCEPTOS TEÓRICOS BÁSICOS

Dos temas fundamentales que están presentes en el desarrollo del presente trabajo, han sido los de Lógica Difusa y Sistemas Expertos. En el presente capítulo se describen los conceptos principales - a un nivel introductorio - relacionados con estos temas. La comprensión de estos conceptos es necesaria para continuar la lectura de este documento en los capítulos siguientes.

III.A. Teoría de Conjuntos difusos y Lógica difusa

La introducción de las teorías de conjuntos difusos y lógica difusa amplía las nociones de grados de pertenencia y de veracidad correspondientes a las teorías clásicas. Estos cambios surgen a partir de la necesidad de modelar las incertezas propias de los sistemas del mundo real. En los puntos siguientes se explicarán los conceptos básicos y las definiciones principales relacionados con estas teorías.

III.A.1. Conjuntos difusos

III.A.1.a) Conceptos básicos

En la teoría clásica de conjuntos, la relación de pertenencia de un elemento a un conjunto está definida a partir de una función de pertenencia cuya imagen es un conjunto de dos elementos. Esta función se denomina función característica.

En la teoría de conjuntos difusos, la función característica de un conjunto clásico puede ser generalizada de manera tal que los valores asignados por dicha función a los elementos del conjunto caigan dentro de un rango numérico especificado. De esta manera se indica el grado de pertenencia de estos elementos al conjunto. Valores grandes de la función denotan mayor grado de pertenencia al conjunto. Tal función es llamada *función de pertenencia*. El rango más frecuentemente usado para la función de pertenencia es el intervalo [0,1].

El conjunto definido por una función de pertenencia cuya imagen es el intervalo [0,1] se denomina *conjunto difuso*. Si el conjunto clásico es X, entonces la función de pertenencia f, definida sobre los elementos de X, define el conjunto difuso A sobre X. Simbólicamente,

$$f_A: X \rightarrow [0,1]$$

f_A define el *conjunto difuso A*.
Dependiendo si X es finito o infinito, se usa la siguiente notación, respectivamente

$$A = \sum_{x \in X} f_A(x) / x \qquad\qquad A = \int_{x \in X} f_A(x) / x$$

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

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

Ejemplo: Sea X los números reales y A el conjunto difuso de los números reales cercanos a cero. Existen muchas maneras diferentes de definir este conjunto, depende el grado de la magnitud que se está midiendo para decir si un elemento esta cerca o lejos. Una posible forma sería mediante la siguiente función de pertenencia:

$$f_A(x) = 1 / (1 + 10 x^2)^2$$

Definiciones

1. Conjunto vacío: Un conjunto difuso A es *vacío* sii $f_A(x) = 0 \ \forall \ x \in X$.

2. Igualdad: Dos conjuntos difusos A y B son *iguales* sii $f_A(x) = f_B(x)\ \forall\ x \in X$.

3. Corte α : Es posible definir conjuntos clásicos en base a conjuntos difusos en función del grado de pertenencia de sus elementos. Este concepto se denomina *corte* α. Dado un conjunto difuso A definido sobre X y cualquier número $\alpha \in [0,1]$, el *corte* α, ${}^{\alpha}A$, y el *corte* α *fuerte*, ${}^{\alpha+}A$, son respectivamente los siguientes conjuntos clásicos:

$${}^{\alpha}A = \{x \mid f_A(x) \geq \alpha\}$$
$${}^{\alpha+}A = \{x \mid f_A(x) > \alpha\}$$

4. Altura: La *altura (height)* de un conjunto difuso A, se define como $h(A) = \text{supremo}\{f_A(x) / x \in X\}$.

5. Conjunto Normal: Un conjunto difuso A se dice *normal* sii $h(A) = 1$.

6. Soporte: El *soporte* de un conjunto difuso A es un conjunto en el sentido clásico, que está formado por los elementos cuyo valor de pertenencia en A no es cero, es decir, *soporte*$(A) = \{x \mid f_A(x) > 0\}$

7. Convexidad: Otro concepto importante de los conjuntos difusos definidos sobre R^n es su *convexidad*, que requiere que los conjuntos clásicos definidos por un *corte* α sean convexos (en el sentido clásico) para cada $\alpha \in (0,1]$. Un conjunto clásico A definido sobre R^n se dice convexo sii $\forall\ x_1,x_2 \in A\ \ \forall\ \lambda \in [0,1]\ (\lambda x_1 + (1-\lambda)x_2) \in A$. Se puede demostrar que un conjunto difuso A es convexo sii $f_A(\lambda x_1 + (1-\lambda)x_2) \geq \text{mín}\ [f_A(x_1), f_A(x_2)]\ \forall\ \lambda \in [0,1],\ \forall\ x_1,x_2$. Para ver la demostración, remitirse a [KLI/95].

III.A.1.b) Operaciones básicas

Las tres operaciones básicas sobre conjuntos clásicos - el complemento, la intersección y la unión - pueden ser generalizadas para los conjuntos difusos de varias maneras. Sin embargo, una generalización particular, la que resulta en operaciones que son usualmente llamadas *operaciones standard de conjuntos difusos*, tiene un significado especial en la teoría de conjuntos difusos. Si A y B son conjuntos difusos sobre X,

El complemento standard de A *es* A', siendo $f_{A'}(x) = 1 - f_A(x)$

La intersección standard de A y B *es* $A \cap B$, siendo $f_{A \cap B}(x) = \textbf{mín}\ (f_A(x), f_B(x))$

La unión standard de A y B *es* $A \cup B$, siendo $f_{A \cup B}(x) = \textbf{máx}\ (f_A(x), f_B(x))$

La *intersección standard* difusa (operador mín) produce, para cualquier par de conjuntos difusos, el conjunto difuso más grande de aquellos producidos por todas las intersecciones difusas posibles. Por el contrario, la *unión standard* difusa (operador máx) produce el conjunto difuso más pequeño de aquellos producidos por todas las uniones difusas posibles. Es decir, las *operaciones standard* ocupan una posición específica dentro de todo el espectro de operaciones difusas básicas: la intersección standard difusa es la intersección difusa *más débil*, mientras que la unión standard difusa es la unión difusa *más fuerte*.

Se puede ver que estas definiciones se comportan en los casos extremos (es decir, los casos 0 y 1) igual que en la teoría clásica. Como se ha dicho, existen muchas otras definiciones posibles para las operaciones básicas. Sin embargo, siempre se exigen estas condiciones denominadas condiciones de frontera o límite, dado que la teoría difusa es una extensión de la teoría clásica.

Propiedades
Sean A, B y C tres conjuntos difusos.

Leyes de De Morgan

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

Leyes distributivas

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

Por otro lado, también se puede extender el concepto de inclusión de la teoría clásica de conjuntos de la siguiente manera. Siendo A y B dos conjuntos difusos, A está *incluido* en B ó A está *contenido* en B si y sólo si la función de pertenencia de A es menor o igual que la de B para todo elemento. Simbólicamente,

$A(x) \subseteq B(x)$ sii $f_A(x) \leq f_B(x)$ $\forall$ x $\in$ X.

Un estudio más detallado de las posibles definiciones para las operaciones básicas entre los conjuntos difusos puede encontrarse en [KLI/95].

III.A.1.c) Relaciones difusas

Dado $X = X_1 \times X_2 \times \ldots \times X_n$, decimos que R es una *relación difusa* en X sii R es un subconjunto difuso de X.
Es decir, debe existir una función de pertenencia f_R definida así:

$f_R: X \rightarrow [0,1]$

donde, para cada n-upla $(x_1, x_2, \ldots, x_n) \in X$, habrá un valor asociado $f_R(x_1, x_2, \ldots, x_n)$.

III.A.1.d) Variables lingüísticas

El concepto de número difuso juega un papel fundamental en la formulación de *variables difusas cuantitativas*. Estas son variables cuyos estados son números difusos. Primero se verá qué significa un número difuso.

Un número difuso es un conjunto difuso definido sobre el conjunto de números reales R. La función de pertenencia de un número difuso A tiene la forma

$f_A : R \rightarrow [0,1]$

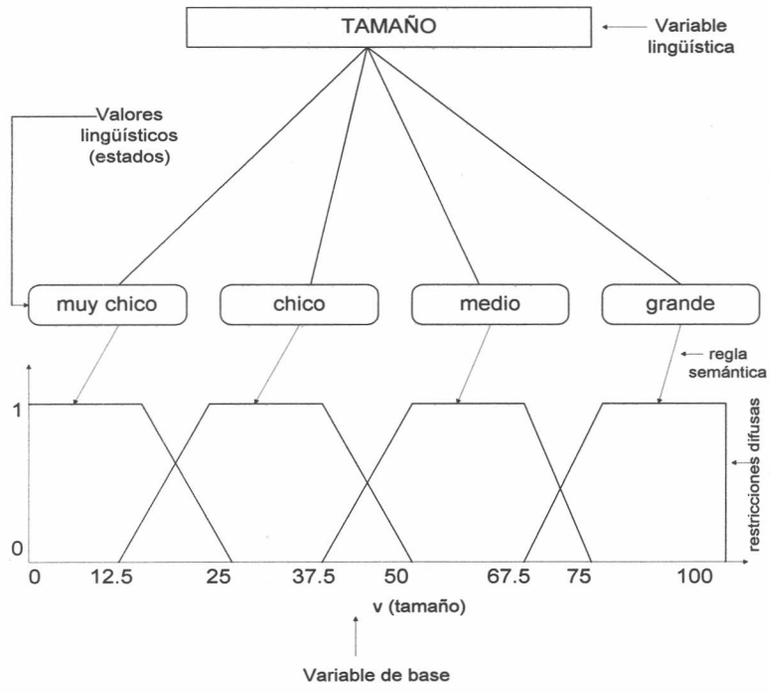
Además, el conjunto A sobre R debe poseer al menos las tres propiedades siguientes para ser considerado un número difuso:

1. A debe ser un conjunto difuso *normal*, es decir, debe existir $x \in X / f_A(x)=1$
2. A debe ser convexo, es decir, ${}^{\alpha}A$ debe ser un intervalo cerrado $\forall \alpha \in (0,1]$
3. El soporte de A, ${}^{0+}A$, debe ser acotado.

Cuando los números difusos representan conceptos lingüísticos, tales como *muy chico, chico, mediano, grande*, etc., en un determinado contexto, las construcciones resultantes son llamadas usualmente *términos o valores lingüísticos*.

Cada variable lingüística está definida en términos de una variable *base*, la cual toma valores en un rango especificado de los números reales. Una variable base es una variable en el sentido clásico, como podría ser cualquier variable física (temperatura, presión, etc) así como también cualquier otra variable numérica (edad, salario, altura, etc). En una variable lingüística, los términos lingüísticos representando valores aproximados de una variable base son mapeados a números difusos apropiados.

El siguiente ejemplo ilustra claramente el concepto:



III.A.2. Lógica difusa

III.A.2.a) Lógicas Multivaluadas

La presunción básica sobre la cual se basa la lógica clásica (o lógica bivaluada) - que cada proposición es o bien verdadera o bien falsa - ha sido cuestionada por Aristóteles, quien ya hacía referencia al problemático estado de verdad de los asuntos que son concernientes al futuro. Las proposiciones que son acerca de eventos futuros, mantiene, no son realmente ni verdaderas ni falsas, sino potencialmente cualquiera de ambas; así, su valor de verdad es indeterminado, al menos antes del evento.

Ahora es ampliamente aceptado que las proposiciones cuyo valor de verdad es problemático no están restringidas a eventos futuros. Como una consecuencia del *principio de incertidumbre* de Heisenberg, por ejemplo, es sabido que los valores de verdad de ciertas proposiciones en mecánica cuántica son inherentemente indeterminados debido a limitaciones fundamentales de medición. A fin de poder tratar con dichas proposiciones, debemos relajar la dicotomía verdadero/falso de la lógica clásica bivaluada.

Una forma de hacerlo es extender la lógica clásica a una lógica de *n-valores*, para un valor arbitrario n (n>2). Muchas lógicas de este tipo fueron desarrolladas aproximadamente durante 1930. Para un n dado, los valores de verdad son números racionales en el intervalo [0,1]. Estos valores se obtienen dividiendo el intervalo entre 0 y 1 exclusivamente. El conjunto T_n de valores de verdad de una lógica *n-valuada* se define como

$$T_n = \{ 0 = \frac{0}{n-1} , \frac{1}{n-1} , \frac{2}{n-1} , ... , \frac{n-2}{n-1} , \frac{n-1}{n-1} = 1 \}$$

Estos valores pueden ser interpretados como *grados de verdad.*

La primer serie de lógicas *n-valuada* para n>2 fue propuesta por Lukasiewicz a principios de 1930 [KLI/95] como una generalización de su lógica tri-valuada (n=3). Usa valores de verdad en T_n y define las primitivas por las siguientes ecuaciones:

$$\begin{aligned} \bar{a} &= 1 - a \\ a \wedge b &= \text{mín}(a , b) \\ a \vee b &= \text{máx}(a , b) \\ a \Rightarrow b &= \text{mín}(1 , 1 + b - a) \end{aligned} \qquad (3.1)$$

$$a \Leftrightarrow b = 1 - |a - b|$$

Para cada n≥2, la lógica n-valuada de Lukasiewicz se denota usualmente en la literatura por L_n. Los valores de verdad de L_n son tomados de T_n y sus primitivas se definen por (3.1). La secuencia (L_2 , L_3 , ..., L_∞) de estas lógicas contiene dos casos extremos - las lógicas L_2 y L_∞ . La lógica L_2 es claramente la lógica clásica bivaluada. L_∞ es una lógica *infinitamente-valuada* cuyos valores de verdad son tomados del conjunto infinito numerable T_∞ de todos los números racionales en el intervalo [0,1].

Cuando no tomamos los valores de verdad sólo del conjunto T_∞ , sino que además aceptamos como valores de verdad cualquier número *real* en el intervalo [0,1], obtenemos una lógica *infinitamente-valuada* (no numerable) alternativa.

El término *lógica infinitamente-valuada* es generalmente usado en la literatura para indicar la lógica cuyos valores de verdad están representados por todos los números reales en el intervalo [0,1]. Podemos ver que ésta lógica es isomórfica a la teoría de conjuntos difusos basada en los operadores difusos standard, de la misma forma que la lógica bivaluada es isomórfica a la teoría clásica de conjuntos. Para obtener más información, remitirse por ejemplo a [KLI/95].

III.A.2.b) Proposiciones difusas

La diferencia fundamental entre proposiciones clásicas y proposiciones difusas radica en el rango de sus valores de verdad. Mientras que cada proposición clásica requiere ser verdadera o falsa, la verdad o falsedad de las proposiciones difusas es una cuestión de aproximación. Asumiendo que verdad y falsedad son expresados por los valores 1 y 0 respectivamente, el grado de verdad de cada proposición difusa es expresado por un número en el intervalo [0,1].

III.A.2.b)1) Proposiciones difusas sin condicional

La forma canónica de las proposiciones difusas p de este tipo es expresada por la sentencia:

p: V is *F* (3.2)

donde V es una variable que toma valores v de algún conjunto universal V , y *F* es un conjunto difuso sobre V que representa un predicado difuso, tal como alto, caro, bajo, normal, etc. Dado un valor particular de V (p.ej. v), este valor pertenece a *F* con grado de pertenencia *F*(v). Este grado de pertenencia es interpretado como un valor de verdad, T(p), de la proposición p. Es decir,

$$T(p) = F(v)$$

para cada valor determinado v de la variable V en la proposición p. Esto significa que T es en efecto un conjunto difuso sobre [0,1], el cual asigna el grado de pertenencia *F*(v) a cada valor v de la variable V.

Se puede ver que el rol de la función T es proveer un "puente" entre conjuntos difusos y proposiciones difusas. Aunque esta conexión entre grados de pertenencia en *F* y grados de verdad de las proposiciones difusas asociadas p es numéricamente trivial, tiene importancia conceptual.

En algunas proposiciones difusas, los valores de una variable V en (3.2) son asignados a individuos en un conjunto dado I. Es decir, la variable V se transforma en una función V: I → V , donde V(i) es el valor de V para el individuo i en V. La forma canónica (3.2) debe ser entonces modificada a la forma:

p: V(i) es *F* (3.3)

donde $i \in I$.

Consideremos, por ejemplo, que I es un conjunto de personas, donde cada persona está caracterizada por su *Edad*, y el conjunto difuso expresando el predicado *Joven* viene dado. Denotando nuestra variable por *Edad* y nuestro conjunto difuso por *Joven*, podemos ejemplificar la forma general (3.3) por la proposición difusa específica

p: *Edad*(i) is *Joven*

El grado de verdad de esta proposición, T(p) , es determinado para cada persona i en I a través de la ecuación:

$$T(p)=f_{Joven}(Edad(i))$$

III.A.2.b)2) Proposiciones difusas condicionales

Las proposiciones p de este tipo son expresadas por la forma canónica

p: Si X es A, entonces Y es B (3.4)

donde X, Y son variables cuyos valores están en los conjuntos universales X e Y respectivamente, y A, B son conjuntos difusos sobre X e Y respectivamente. Estas proposiciones pueden ser vistas también como proposiciones de la forma

$$< X , Y > \text{ es } R$$

donde R es un conjunto difuso sobre X x Y que está determinado para cada $x \in X$ y cada $y \in Y$ por la fórmula

$$R(x,y) = \delta [A(x) , B(y)]$$

donde δ denota una operación binaria sobre [0,1] representando una *implicación difusa* apropiada (en el capítulo "**III.A.2.e.** Implicaciones difusas" se explica detalladamente este concepto).

Si tomamos, por ejemplo y a modo de ilustración, una implicación difusa particular como la de Lukasiewicz , tendríamos que

$$\delta(a,b)=\text{mín}(1,1-a+b)$$

y entonces

$$T(p) = R(x,y) = \text{mín}(1,1-x+y)$$

para cada valor determinado (x,y) del conjunto X x Y en la proposición p.

III.A.2.c) Modificadores Lingüísticos

Los modificadores lingüísticos son términos lingüísticos especiales por los cuales otros términos lingüísticos son modificados. Términos lingüísticos como *muy, más o menos, bastante, extremadamente* son ejemplos de modificadores, que pueden ser usados para modificar predicados difusos. Por ejemplo, la proposición "x es joven" puede ser modificada por el modificador *muy* de la siguiente manera: "x es *muy* joven".

En general, dada una proposición difusa

p: x es F

y un modificador lingüístico H, podemos construir una proposición modificada

Hp: x es HF

donde HF denota el predicado difuso obtenido al aplicar el modificador H al predicado dado F.

Es importante recalcar que estos modificadores lingüísticos no son aplicables a predicados bivalentes. Por ejemplo, los términos lingüísticos *muy horizontal, muy embarazada, muy rectangular* no son significativos. Por lo tanto, los modificadores lingüísticos no existen en la lógica clásica.

Cualquier modificador lingüístico H puede ser interpretado como una operación unaria h, sobre el intervalo [0,1]. Por ejemplo, el modificador *muy* es usualmente interpretado como la operación unaria $h(a)=a^2$, mientras que el predicado *algo* como $h(a)=\sqrt{a}$ ($a \in [0,1]$). A las operaciones unarias que representan modificadores lingüísticos las llamaremos sólo *modificadores*.

Dado un predicado difuso *F* sobre X y un modificador h que representa un modificador lingüístico H, el predicado difuso modificado H*F* es determinado para cada $x \in X$ por la ecuación

$$HF(x) = h(F(x))$$

Esto significa que las propiedades de los modificadores lingüísticos pueden ser estudiadas mediante el estudio de las propiedades de los modificadores (operaciones) asociados.

Cualquier modificador h es una biyección creciente. Si $h(a) < a \ \forall \ a \in [0,1]$, el modificador es llamado *fuerte*, si $h(a) > a \ \forall \ a \in [0,1]$, el modificador es llamado *débil*. El caso especial del modificador que cumple $h(a)=a \ \forall \ a \in [0,1]$ es llamado *modificador identidad.*

Un modificador fuerte refuerza un predicado difuso al cual es aplicado, y , consecuentemente, reduce el valor de verdad de la proposición asociada. Un modificador débil, por el contrario, debilita el predicado y, por lo tanto, el valor de verdad de la proposición aumenta.

Es fácil ver que cada modificador h satisface las condiciones siguientes:

1. $h(0)=0$ y $h(1)=1$
2. h es una función continua
3. si h es fuerte, entonces h^{-1} es débil y viceversa.
4. Dado otro modificador g, la composición de g con h y h con g son también modificadores, y además si ambas h y g son fuertes (débiles), asi lo son las composiciones.

Una clase conveniente de funciones que satisfacen estas condiciones es la clase

$$h_\alpha(a)=a^\alpha$$

donde $\alpha \in R^+$ es un parámetro por el cual los modificadores individuales en esta clase son distinguidos y $a \in [0,1]$. Cuando $\alpha<1$, h_α es un modificador débil, cuando $\alpha>1$, h_α es un modificador fuerte ; h_1 es el modificador identidad. Esta clase de modificadores (como cualquier otra clase aceptable) nos permite capturar el significado de cada modificador lingüístico relevante en un contexto determinado mediante la elección de un valor apropiado para el parámetro α.

III.A.2.d) Inferencia de proposiciones difusas condicionales

Las reglas de inferencia en lógica clásica están basadas en varias tautologías. Estas reglas de inferencia pueden ser generalizadas dentro del contexto de la lógica difusa para facilitar el razonamiento aproximado. De las tres reglas de inferencia clásicas, *modus ponens, modus tollens y silogismo hipotético,* veremos la primera de estas. Primero veremos el caso bivalente.

Consideremos las variables X e Y , que toman valores de los conjuntos X e Y respectivamente, y asumamos que las variables están relacionadas por una función $\forall x \in X \ \forall y \in Y \ \exists \ f / y=f(x)$. Entonces, dada $X=x$, podemos inferir que $Y=f(x)$. Similarmente, sabiendo que el valor de X está en un conjunto A dado, podemos inferir que el valor de Y está en el conjunto $B=\{y \in Y \mid y=f(x) , x \in A\}$

Asumamos ahora que las variables están relacionadas por una relación arbitraria XxY, no necesariamente una función. Entonces, dada $X=u$ y una relación R, podemos inferir que $Y \in B$, donde $B=\{y \in Y \mid \langle x,y \rangle \in R\}$. Similarmente, sabiendo que $X \in A$, podemos inferir que $X \in B$, donde $B=\{ y \in Y \mid \langle x,y \rangle \in R, x \in A\}$. Observar que esta inferencia puede ser igualmente expresada en términos de funciones características χ_A , χ_B , χ_R de conjuntos A, B, R respectivamente, por la ecuación:

$$\chi_B(y) = \sup_{x \in X} \text{mín} [\chi_A(x), \chi_R(x,y)] \qquad (3.5)$$

$\forall y \in Y$.

Ahora asumimos que R es una relación difusa sobre XxY, y que A' y B' son conjuntos difusos sobre X e Y respectivamente. Entonces, si R y A' vienen dados, podemos obtener B' mediante la ecuación:

$$B'(y) = \sup_{x \in X} \text{mín} [A'(x) , R(x,y)] \qquad (3.6)$$

$\forall y \in Y$, que es una generalización de (3.5) obtenida por el reemplazo de las funciones características en (3.5) con las funciones de pertenencia correspondientes. Esta ecuación, que también puede ser escrita de forma matricial como

B'=A' o R

es llamada la *regla composicional de inferencia.*

La relación difusa empleada en (3.6) no es usualmente dada directamente, sino en alguna otra forma.
En el caso de la relación R que está representada por una proposición difusa condicional p de la forma

p: Si X es A, entonces Y es B

tenemos que R esta determinada $\forall x \in X$, $\forall y \in Y$ por la fórmula

$$R(x,y) = \delta [A(x) , B(y)] \qquad (3.7)$$

donde δ denota una implicación difusa.
Usando la relación R obtenida de la proposición p dada por (3.7), y dada otra proposición q de la forma

q: X es A'

podemos concluir que Y es B' por la regla composicional de inferencia (3.6). Este procedimiento es llamado un *modus ponens generalizado.*

Viendo la proposición p como una regla y la proposición q como un hecho, el modus ponens generalizado es expresado por el esquema siguiente:

Regla:	Si X es A, entonces Y es B	
Hecho:	X es A'	(3.8)
Conclusión:	Y es B'	

En este esquema, B' es calculado por (3.6), y R en esta ecuación está determinado por (3.7). Observar que (3.8) se transforma en el modus ponens clásico cuando los conjuntos son los clásicos y A'=A, B'=B.

III.A.2.e) Implicaciones difusas

La operación lógica de implicación es tan esencial para el razonamiento aproximado como lo es para el razonamiento clásico dentro de la lógica clásica bivaluada. En general, una *implicación difusa* δ es una función de la forma

$\delta: [0,1] \times [0,1] \rightarrow [0,1]$

la cual, para cualquier par de valores de verdad a,b de las proposiciones difusas dadas p,q respectivamente, define el valor de verdad $\delta(a,b)$ de la proposición condicional "si p entonces q". Esta función debería ser una extensión de la implicación clásica $p \Rightarrow q$ del dominio restringido $\{0,1\}$ al dominio $[0,1]$ de valores de verdad en lógica difusa.

Todas las implicaciones difusas son obtenidas generalizando el operador de implicación de la lógica clásica. Es decir, colapsan en la implicación clásica cuando los valores de verdad son restringidos a 0 y 1. A continuación se dan cuatro implicaciones difusas posibles.

1. $\delta(a,b)=máx(1-a,b)$
2. $\delta(a,b)=1-a+ab$
3. $\delta(a,b)=mín(1,1-a+b)$
4. $\delta(a,b)=máx(1-a,mín(a,b))$

En la literatura se pueden encontrar otras definiciones posibles, incluso categorizadas. Ver, por ejemplo, [KLI/95].

III.B. Introducción a los Sistemas Expertos

III.B.1. Introducción

Se puede definir un sistema experto como *un programa de computación basado en conocimiento que emula a un experto humano en la resolución de un problema significativo en un dominio específico.* [CAR/87]

Ante la ocurrencia de un problema determinado, se debe recurrir a cierto conocimiento para poder resolverlo correctamente. Normalmente el problema es muy bien conocido y es posible, etapas de relevamiento y análisis mediante, desarrollar un programa de computación que permita resolverlo. Para llegar a este objetivo sólo es necesario recurrir a las herramientas y técnicas de análisis, diseño y programación tradicionales.

Existe una categoría de problemas que excede a la descripta anteriormente. En este caso, su resolución no depende simplemente del seguimiento de un conjunto de instrucciones de un lenguaje de programación procedural. Aquí se hace necesario que el programa que resuelva el problema tenga un comportamiento *inteligente*. Es decir, idealmente el programa debe poder seguir los pasos de razonamiento de un humano para poder lograr su cometido. Se supone que el humano en cuestión tiene un alto conocimiento del problema y de los pasos necesarios para resolverlo. La diferencia con la categoría anterior reside en que el proceso de resolución y el conocimiento involucrado no están completamente modelados, además se requiere de un proceso de desarrollo incremental para incorporar al sistema el conocimiento del experto en la materia y los procesos de razonamiento que utiliza para resolverlo.

Volviendo a la definición inicial de sistema experto, encontramos en ella tres conceptos fundamentales que hacen a la correcta comprensión de esta rama de aplicaciones: *programa de computación basado en conocimiento*, *emulación de un experto humano* y *resolución de un problema en un dominio específico*.

Sistema Experto: Programa de computación basado en conocimiento.

Se dice que un programa de computación está basado en conocimiento cuando el conocimiento que utiliza se encuentra expresado de una forma que no depende de una interpretación externa que se haga del mismo.

Normalmente, en los programas de computación procedurales, el conocimiento se encuentra expresado en el flujo de control del programa y en los comentarios que en él se escribieron. Este último conocimiento no participa en absoluto del funcionamiento del programa y cumple una función meramente accesoria.

En un programa basado en conocimiento, todo el conocimiento expresado en él influye en su curso de ejecución participando activamente de su comportamiento.

Emulación de un experto humano.

Un elemento fundamental que participa del desarrollo de un sistema experto es el experto humano. El experto es "alguien que posee una gran cantidad de conocimiento que le permite resolver ciertos problemas de manera más eficaz que la mayoría de las personas". [CAR/87]

Cuando hablamos de expertos no nos referimos simplemente a conocedores o estudiosos de un tema determinado, sino a personas que dada su gran experiencia en un campo determinado han adquirido un alto conocimiento del mismo. Tanto un cirujano del corazón de muchos años de trabajo en el quirófano o un mecánico de autos de larga experiencia en su actividad pueden ser expertos en sus respectivas áreas.

El conocimiento del experto proviene de dos ámbitos distintos. Por un lado, del conocimiento público, que proviene de la bibliografía especializada. Por otro lado, del conocimiento privado, que consiste de reglas adquiridas por el experto a través de años de experiencia. Estas reglas, llamadas *heurísticas*, son aquellas más difíciles de extraer del experto para incorporar al sistema en desarrollo. Son este tipo de reglas las modeladas luego de un proceso incremental de relevamiento durante el cual el ingeniero en conocimiento, a través de sucesivas reuniones con el experto, intenta comprender. Las reglas heurísticas constituyen la herramienta principal que el experto humano utiliza para resolver los problemas.

La pericia del experto no depende de la complejidad de los problemas con los que trata. Tanto para problemas complejos como para problemas que carecen de una profunda base teórica es necesario el conocimiento del experto en la materia, ya que en ambos casos son igualmente fundamentales las reglas prácticas que el experto fue adquiriendo con el tiempo.

El objetivo de un sistema experto es poder simular el comportamiento de un experto humano para resolver un problema. La simulación no consiste sólo de la utilización de los conocimientos privados y públicos del experto sino también en apelar a la interacción con el usuario. De esta forma, un sistema experto puede realizar preguntas para solicitar datos adicionales necesarios para la resolución del problema, explicar el por qué de una pregunta o el camino de razonamiento llevado a cabo para llegar a una conclusión determinada.

El objetivo de la emulación del experto humano en la resolución de problemas, lleva al sistema experto a la posibilidad de equivocarse. Este es un resultado esperable ya que mientras se supone un cierto grado de falibilidad (mínimo) en un experto humano también se esperan resultados similares por parte del sistema que se encuentra emulándolo.

Resolución de un problema en un dominio específico.

El proceso de desarrollo de los sistemas expertos lleva a que éstos puedan ser explotados al máximo en su funcionalidad cuando se los lleva a tratar sólo un dominio específico de conocimiento. [NEB/91]

De esta forma, el proceso de ingeniería de conocimiento, entrevistas con el experto, mejoramiento del conocimiento del sistema, se dedican al proceso de resolución de problemas dentro de una única área.

Por ejemplo, dentro de la compleja área de la medicina moderna, se han desarrollado sistemas expertos abocados a tratar ramas específicas de enfermedades: enfermedades infecciosas, medicina interna, etc.

Usos de los sistemas expertos

Casi cualquier campo del conocimiento humano es susceptible de incorporar la aplicación de un sistema experto. Desde la medicina moderna hasta la exploración petrolífera, pasando por el diagnóstico de fallas en maquinarias o plantas industriales.

Los sistemas expertos pueden ser aplicados para resolver problemas de diagnóstico, diseño, optimización, lingüística, etc.

Usualmente los sistemas expertos son clasificados en base a su dominio, el mecanismo de representación de conocimiento, el método de inferencia y determinadas características especiales. Una clasificación basada en el uso que se le da al sistema experto se puede encontrar en [HAL/91].

III.B.2. Entorno de los Sistemas Expertos

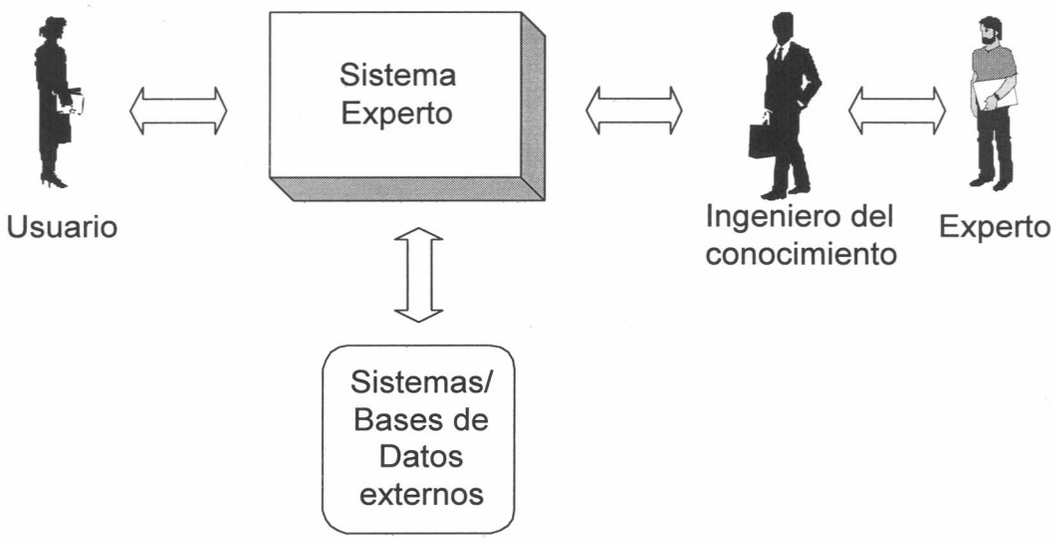
El entorno de los sistemas expertos durante las etapas de desarrollo y uso está constituido principalmente por los **usuarios**, los **expertos del dominio**, **los ingenieros de conocimiento** y potencialmente **sistemas externos** con los cuales pueda llegar a interactuar.

Los expertos y los ingenieros de conocimiento son los participantes fundamentales de los procesos de desarrollo y mantenimiento del sistema experto. Los ingenieros de conocimiento tienen como misión fundamental la de interactuar con el experto humano. De esta manera intentan comprender el conocimiento que éste utiliza para resolver problemas con el fin de modelarlo y así proveer una representación formal del mismo de modo de ser utilizada en la construcción del sistema experto.

Los usuarios del sistema experto interactúan con él una vez construido y puesto en funcionamiento. De todos modos, se requiere una continua participación de los expertos e ingenieros de conocimiento con el fin de corregir errores e incorporar nuevo conocimiento.

Como ocurre con los sistemas convencionales, es posible que el sistema experto deba interactuar con otros sistemas a partir de los cuales necesite obtener datos o enviárselos. También puede surgir la necesidad de interactuar con bases de datos externas que provean información sobre la cual deba operar el sistema experto.

El ámbito de funcionamiento de un sistema experto puede ejemplificarse en el siguiente esquema:



III.B.3. Arquitectura de un Sistema Experto

En el siguiente gráfico vemos un ejemplo de arquitectura de un sistema experto: [CAR/87]

Se puede resumir la arquitectura de un sistema experto en tres componentes principales: la **base de conocimientos**, el **motor de inferencias** y la **memoria de trabajo**.

La **base de conocimientos** contiene el conocimiento utilizable por el sistema experto. Este conocimiento consiste de hechos conocidos - válidos en todo momento de utilización del sistema - y reglas. Las reglas se incorporan a la base de conocimientos a partir del trabajo llevado a cabo por el ingeniero de conocimiento junto con el experto del dominio. Las reglas contienen los conocimientos públicos y privados del experto.

La **memoria de trabajo** almacena la información que el sistema experto utiliza durante el proceso de resolución.
Un elemento de importancia que se encuentra en la memoria de trabajo son las explicaciones de las conclusiones intermedias a las que arriba el sistema experto durante el proceso de resolución. Las explicaciones contienen la cadena de reglas recorrida para llegar a una conclusión determinada. Con esta información el sistema experto puede mostrar al usuario qué reglas utilizó para elaborar una respuesta a su consulta.

El **motor de inferencias** representa la "inteligencia" del sistema. Es este módulo el que genera los resultados del sistema experto a partir de la base de conocimientos y la memoria de trabajo.
Dentro del motor de inferencias, el activador de reglas encadena las reglas entre sí para concluir nuevo conocimiento.

El encadenamiento de reglas puede realizarse hacia adelante o hacia atrás dependiendo de las características de los datos de entrada y de la cantidad posible de conclusiones existentes.

En el encadenamiento hacia adelante se parte de los datos suministrados por el usuario - almacenados en la memoria de trabajo - y se determina cuáles reglas son válidas a partir de éstos obteniéndose nuevas conclusiones. Luego se reinicia el ciclo incorporando previamente las conclusiones obtenidas en el paso anterior. Si en algún momento se obtiene dentro del conjunto de conclusiones una expresión que coincide con el objetivo buscado en la consulta, el proceso se detiene y se informa al usuario los resultados obtenidos. Cuando en algún paso se agotan todas las reglas existentes sin poder obtener nuevas conclusiones, se termina el proceso informando al usuario que no fue posible llegar al objetivo buscado.

En el encadenamiento hacia atrás, el encadenador parte del objetivo buscado, determina las reglas cuyas conclusiones coinciden con él y para alguna de estas busca a su vez otras reglas cuyas conclusiones coincidan con los antecedentes de aquella. En caso de tener éxito en este proceso con todos los antecedentes, el ciclo continua aplicando los mismos pasos con las reglas incorporadas. En algún momento, para alguna de las reglas cuyos antecedentes se busca validar puede no encontrarse otra regla que concluya de la forma requerida. En ese caso, el sistema experto recurre al usuario para consultarle por la veracidad del antecedente. La respuesta del usuario es almacenada en la memoria de trabajo.

Si en algún paso de este proceso todos los antecedentes de las reglas recorridas se encuentran confirmados por la conclusión de otra regla o por la información del usuario, se ve que se ha concluido el objetivo pedido y se informa al usuario del éxito de la consulta. En caso contrario, se informa que no se ha podido concluir el objetivo solicitado.

Un elemento importante en este proceso es el método que se utiliza para determinar qué regla incorporar en cada momento. Tanto para el encadenamiento hacia atrás como para el encadenamiento hacia adelante, se debe contar con una estrategia que indique cuál regla considerar.

Esta estrategia se denomina "estrategia de resolución de conflictos" e influye en la eficacia y la eficiencia del sistema experto.

La conveniencia de utilizar uno u otro método de encadenamiento está determinada por las características de los datos de entrada y por la diversidad de conclusiones posibles.

Es aconsejable utilizar encadenamiento hacia atrás cuando el número de conclusiones posibles es acotado. De esta forma se pueden considerar el conjunto de conclusiones para aplicar las reglas hacia atrás y ver si se llega a verificar alguna de ellas.

Mientras que en un sistema experto dedicado a diagnosticar enfermedades las conclusiones posibles son escasas y por lo tanto se hace conveniente aplicar encadenamiento hacia atrás, en los sistemas expertos dedicados a resolver problemas de diseño las respuestas válidas pueden llegar a ser infinitas. En casos como este último se hace necesario aplicar un encadenamiento hacia adelante.

Dentro del motor de inferencias, encontramos también al **trazador de explicaciones**. Este módulo se encarga de registrar en la memoria de trabajo los datos necesarios para poder explicar la cadena de resoluciones llevada a cabo para llegar a una conclusión determinada.

El trazador de explicaciones también se encarga de dar explicaciones en todo momento en que el usuario lo requiera.

III.B.4. Desarrollo posible, justificado y apropiado

Existen una serie de características de los problemas que es conveniente analizar con el fin de determinar la viabilidad del proyecto de desarrollo de un sistema experto para su solución.

El objetivo principal del análisis es reconocer los aspectos principales del problema que respaldan la idea del desarrollo de un sistema experto para resolverlo.

Se puede hablar de tres ejes de análisis con respecto al desarrollo de un sistema experto para la resolución de un problema específico dentro de un dominio determinado. El desarrollo del sistema experto debe resultar **posible**, **justificado** y **apropiado**.

Desarrollo posible

Hay un conjunto de condiciones que deben cumplirse para poder hacer posible el desarrollo de un sistema experto:

• **Existencia de un experto:** Esta es prácticamente la condición básica que se debe cumplir para llevar a cabo el desarrollo. Debe existir un experto humano que sea capaz de resolver el problema en cuestión.

Al referirnos a la existencia del experto lo hacemos en el marco de la definición de experto hecha previamente. O sea que debe existir una o más personas que posean el conocimiento especializado sobre el problema.

Podemos encontrarnos ante situaciones que no nos lleven a la certeza de la existencia de un experto humano. Existen ámbitos donde una gran cantidad de personas poseen un mediano conocimiento del problema, en estos casos nos damos cuenta de que no existe un conocimiento altamente especializado y concentrado en unos pocos que sea necesario difundir entre los demás. En otros ambientes puede ocurrir que un gran número de personas posea un alto grado de conocimiento del problema. Tampoco en estos casos encontramos una alta concentración de conocimiento que necesite ser difundido.

• **Posibilidad de estructurar el conocimiento del experto:** Es posible que en un ambiente determinado exista efectivamente el experto, aunque también puede ocurrir que el mismo no sea capaz de dar una estructura a su propio conocimiento. Esto puede deberse a la falta de capacidad del experto en llevar a cabo esta tarea o a la propia naturaleza del conocimiento que maneja.

• **Concordancia en las soluciones entre distintos expertos:** Con el motivo de tener la posibilidad de corroborar la correctitud del funcionamiento del sistema experto, se hace necesario que en caso de contar con más de un experto en la materia, todos ellos concuerden en las soluciones a los problemas planteados.

• **Dependencia del sentido común:** Es necesario, para hacer posible el desarrollo de un sistema experto, que la resolución del problema no dependa exclusivamente del uso del sentido común. El sentido común no es un ámbito de conocimiento accesible para los sistemas expertos.

• **Grado de dificultad de las tareas:** Se hace necesario también que la tarea a encomendarse al sistema experto no posea una complejidad excesiva. Los problemas a tratar por los sistemas expertos deben ser de una naturaleza tal que puedan ser comprendidos por iniciados en la materia y que no deba requerirse meses o años de trabajo para resolverlos por parte de un experto humano.

Desarrollo justificado

Es importante poder determinar, previamente al desarrollo de un sistema experto, si el mismo tendrá algún tipo de utilidad o proveerá algún tipo de provecho en el sentido económico, científico, etc.

Existe una serie de aspectos que ayudan a determinar el grado de justificación del proyecto:

• **Pérdida del experto:** Es posible justificar el desarrollo en casos en que el experto abandonará la organización o cambiará de sector y no es posible contar con alguien que lo reemplace.

• **Escasa disponibilidad de expertos humanos:** Puede ocurrir que se carezca de especialistas en un área determinada o que exista un aumento en su demanda que provoque que los especialistas existentes no alcancen para cubrirla. En ambos casos encontramos una justificación para el desarrollo de un sistema experto.

• **Necesidad de distribuir el conocimiento:** Puede hacerse necesario en una organización que cierto conocimiento sea requerido en diversos lugares, de modo que el/los experto/s disponibles no puedan cubrir las necesidades planteadas. También puede ocurrir que se desee descentralizar ciertas actividades y por lo tanto se haga necesario difundir el conocimiento existente sobre cierto dominio de problemas.

• **Necesidad de contar con el conocimiento en horarios muy amplios, o en ambientes hostiles a la presencia humana.**

• **Requerimiento de consistencia en las soluciones planteadas a problemas similares:** Es útil contar con sistemas expertos que provean la posibilidad de uniformizar la provisión de soluciones en distintos lugares y a través del tiempo.

Desarrollo apropiado

Se puede hablar de *desarrollo apropiado* de un sistema experto cuando el enfoque de la misma solución a través de sistemas convencionales no sea conveniente.

Hablamos de desarrollo apropiado cuando lo importante del proceso de resolución del problema planteado está basado en la aplicación de heurísticas y de conocimiento altamente especializado.

El camino para llegar a la solución del problema, en estos casos, debe ser a través de la acumulación de evidencias tanto a favor como en contra de cada posible conclusión. Estamos hablando de casos en que no es posible trazar un único camino a la solución a través de etapas fijas.

III.B.5. Representación del conocimiento

El problema de la representación del conocimiento constituye un aspecto crucial en la construcción de un sistema experto. Este punto está relacionado con la definición inicial que se ha hecho sobre esta clase de sistemas en cuanto a que son sistemas basados en conocimiento. Para que las bases del conocimiento se encuentren correctamente implementadas será necesaria una adecuada representación del conocimiento.

Se puede definir la representación del conocimiento como "un conjunto de convenciones sintácticas y semánticas que hace posible definir cosas". Los distintos lenguajes de representación constituyen distintos tipos de estas convenciones.

Se pueden citar tres aspectos fundamentales que se requieren de los lenguajes de representación: deben tener **expresividad**, **poder heurístico** y **conveniencia notacional**.

En cuanto a tener **expresividad**, nos referimos a que el lenguaje debe permitir representar todo el conocimiento relevante que el experto maneja para resolver los problemas específicos.

El **poder heurístico** de un lenguaje da una medida de hasta qué punto el mismo permite utilizar en forma eficiente el conocimiento representado para resolver los problemas.

Finalmente, la **conveniencia notacional** se refiere a que el lenguaje tenga una sintaxis clara, fácil de escribir y de comprender.

Los distintos paradigmas de representación de conocimiento se pueden agrupar en tres grandes grupos: **basados en reglas**, **basados en objetos estructurados** y **basados en lógica**.

Representación del conocimiento basada en reglas

La representación del conocimiento en este modelo se realiza a través de reglas. Cada regla es una o más conclusiones y conjunto de precondiciones que deben cumplirse para darse por válidas las conclusiones:

SI
$P_1, P_2, \ldots P_n$
ENTONCES
$C_1, C_2, \ldots C_n$

Por ejemplo, las premisas P_i y las conclusiones C_i pueden ser pares de elementos del tipo 'atributo - valor' como por ejemplo *edad-empleado = 26*.

Este esquema de representación tiene como ventajas el poseer un alto poder expresivo y su simplicidad de escritura y comprensión.

El principal problema de este modelo está relacionado con su imposibilidad de poder representar el conocimiento según nociones de grupos de objetos (clases) y relaciones entre ellos. Normalmente, el conocimiento resulta representado en forma desordenada.

Representación del conocimiento basada en objetos estructurados

Este esquema, al contrario del anterior, enfoca el conocimiento en base a los objetos que representa.

Los distintos métodos de representación basados en objetos estructurados definen objetos en base a su pertenencia a grupos o clases de objetos. A su vez, se plantea la idea de herencia de propiedades y procedimientos entre estos elementos. También es posible representar excepciones entre los objetos y sus clases, de modo de permitir indicar los casos en que ciertos objetos no posean todas las características de su clase.

La ventaja de este modelo es su enfoque a favor de la representación del conocimiento en forma estructurada de acuerdo a cómo están estructurados los objetos en el dominio.

Las limitaciones de este modelo surgen de la complejidad de las representaciones para casos reales.

La categoría de los objetos estructurados incluyen a las **redes semánticas** y los **frames.**

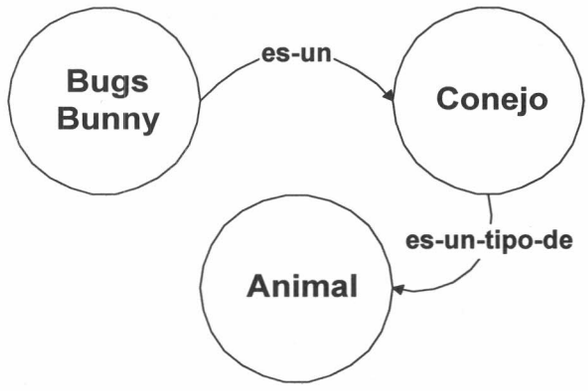
Redes semánticas

Las **redes semánticas** son básicamente grafos, en donde los nodos representan conceptos y los ejes representan relaciones entre esos conceptos. A su vez, los nodos pueden ser *individuales* - representando descripciones de instancias de objetos - o *genéricos* - que representan descripciones de clases de objetos.

Las relaciones entre los nodos se realizan a través de ejes. Entre los ejes, tenemos dos clases principales, los *es-un* - que ligan nodos individuales con nodos genéricos y los *es-un-tipo-de* - que ligan dos nodos genéricos entre sí.

Como los nodos genéricos pueden tener distinto grado de especificidad, se logra así la estructura de red.

Ejemplo:



A través de la relación transitiva es-un-tipo-de y aplicando la herencia de propiedades de los nodos, es posible recorrer los distintos nodos de la jerarquía e inferir conocimiento.

En el ejemplo, podríamos deducir que "Bugs Bunny" es un animal.

El problema de la deducción en las redes semánticas consiste en que el significado asociado a la red depende del intérprete de la misma, siendo difícil hablar de métodos generales de deducción.

Frames

Podemos ver al modelo de frames como un caso más sofisticado que el de las redes semánticas. Los nodos de las redes semánticas están aquí en la forma de *frames* que constituyen una descripción estructurada de un objeto o de una clase de objetos.

Los frames tienen asociados atributos denominados *slots* los cuales especifican distintos tipos de propiedades.

El detalle asociado a los slots incluye *valores* - que establece la clase de valores que puede adquirir el atributo especificado - *restricciones* - que especifica distintos tipos de restricciones que deben cumplir los valores posibles del atributo - y *procedimientos agregados* - que indican cómo usar el atributo-.

Los slots, a su vez, pueden hacer referencia a características de los miembros de la clase a la que representa el frame - *m-slots* - o a características específicas del objeto o clase que el frame simboliza.

Ejemplo:

FRAME: Astros	
INCLUIDO-EN:	Objetos-estelares
SUB-CLASES:	(Estrellas, Planetas, Satélites)
M-SLOT:	Masa
CLASE-DE-VALORES:	Entero
UNIDAD-DE-MEDIDA:	Tonelada
VALOR-MINIMO:	0
M-SLOT:	Luminosidad
CLASE-DE-VALORES:	Real
FRAME: Planetas	
ESPECIALIZACIÓN-DE:	Astros
M-SLOT:	Sistema Planetario
CLASE-DE-VALORES:	String
M-SLOT:	Habitado?
CLASE-DE-VALORES:	Lógico
DEFAULT:	No
FRAME: Mercurio	
MIEMBRO-DE:	Planetas
P-SLOT:	Sistema Planetario
valor:	Sistema Solar
P-SLOT:	Luminosidad
valor:	0.9
P-SLOT:	Masa
valor:	1 e 11

El problema con la inferencia en frames ya se presentaba en redes semánticas. También aquí tenemos un significado que depende del intérprete. Sin embargo en los frames encontramos estructuras que incluyen una semántica más definida.

Representación del conocimiento basada en lógica

La lógica de primer orden surge como un formalismo válido para representar el conocimiento.

Esta representación se basa en la utilización de predicados y reglas de la lógica de primer orden.

La resolución de problemas, en este ámbito, se lleva a cabo como un proceso de demostración de teoremas basado en el principio de resolución y la unificación de variables.

Este formalismo cuenta con un amplio poder expresivo, facilitado principalmente por la posibilidad del uso de los cuantificadores existencial y universal. El aspecto fundamental a destacar es que se cuenta con una "teoría de verdad" que respalda al modelo, es decir que se encuentran completamente formalizados los métodos de validación de conocimiento.

Como contraparte, encontramos en este formalismo grandes dificultades de implementación debido a los problemas de complejidad computacional de los algoritmos de resolución para la lógica de primer orden.

III.B.6. Herramientas de construcción de sistemas expertos.

Las implementación de sistemas expertos puede realizarse de diversas maneras. Se puede llevar a cabo una implementación utilizando lenguajes tradicionales (C, Pascal, etc.) o apelar a lenguajes como Lisp y Prolog, los cuales se encuentran más cerca de cumplir con los requerimientos de representación del conocimiento nombrados anteriormente.

A medida que se fue avanzando en la construcción de distintos sistemas expertos y llegando a un modelo generalizado de arquitectura común para todos ellos - como el visto más arriba - se comenzó a construir esqueletos de sistemas expertos. Estos sistemas constan de un motor de inferencias desarrollado para funcionar con distintas bases de conocimiento referentes a dominios de distintos ámbitos. Dicha concepción provino principalmente de sistemas expertos ya desarrollados de los cuales se extrajo su motor de inferencias para poder utilizarlo con otras bases de conocimiento.

Estos sistemas que constituyen cáscaras vacías de sistemas expertos fueron denominados "shells".

Las diferencias principales entre el desarrollo de sistemas expertos utilizando shells o lenguajes de programación residen principalmente en la flexibilidad y tiempos de obtención de resultados.

Mientras que el uso de los lenguajes de programación para el desarrollo de los sistemas expertos proveen una alta flexibilidad en el momento de implementación, los shells son herramientas que poseen gran parte de su estructura de inferencias ya desarrollada pero no es tan fácil adaptarla a los requerimientos del dominio de problemas a resolver.

Por otro lado, los shells proveen una base ya establecida para el desarrollo y permiten construir rápidamente prototipos de nuevos sistemas expertos utilizando el motor de inferencias y las interfaces ya desarrollados.

III.B.7. Construcción de los Sistemas Expertos.

El proceso de construcción de un Sistema Experto consta de diversas etapas, la mayoría de las cuales se asemejan a las ya conocidas de los sistemas tradicionales. En caso de elegirse un shell determinado como herramienta de desarrollo, el ciclo será básicamente un proceso de prototipación.

La etapa que hace distintivo el ciclo de construcción de los sistemas expertos con respecto a los sistemas tradicionales es la que se dedica a la adquisición del conocimiento.

Adquisición del conocimiento

El proceso de adquisición del conocimiento está principalmente dedicado a extraer por parte del ingeniero de conocimiento el conocimiento privado del experto a través de una serie de entrevistas personales.

Previamente a llevar a cabo las entrevistas, el ingeniero de conocimiento debe asimilar el manejo del conocimiento público del experto. De esta forma se facilita un mejor acercamiento con el entrevistado. Esta tarea involucra consultar la bibliografía disponible sobre el tema, diccionarios de términos relacionados, etc. El objetivo principal es adquirir un vocabulario común con el experto.

Una tarea importante a realizar con el experto consiste del análisis de casos. En este trabajo se enfrenta al experto con casos de problemas que ya se han planteado previamente e incluso han sido resueltos por él mismo. Durante el análisis el ingeniero de conocimiento puede observar la línea de razonamiento llevada a cabo por el experto, los datos considerados y la solución hallada.

La utilidad de esta tarea está dada por el hecho de que el experto puede llegar a explicar en forma general muchas de las reglas que aplica para resolver problemas habituales, pero recién al enfrentarlo con problemas reales se hace evidente que el mecanismo que aplica para resolverlos va más allá de las reglas explicitadas y tiene que ver con conocimiento que generalmente ni el mismo experto del dominio puede racionalizar y explicar fácilmente.

Estudiando la resolución por parte del experto de los casos seleccionados es posible extraer reglas asociadas y luego en pasos subsiguientes llegar a reglas más generales que se apliquen a los casos analizados y cualquier otros que se puedan presentar en el futuro.

La técnica aplicada en estos casos consiste de la observación por parte del ingeniero del conocimiento del proceso de resolución de casos particulares por parte del experto y luego analizando con él los caminos elegidos para llegar a la solución final solicitándole al experto además que explique desde su propio punto de vista cómo razona a medida que va resolviendo el problema.

Para un mejor encaminamiento del proceso de adquisición del conocimiento, es recomendable iniciar en esta etapa el desarrollo de una versión preliminar del sistema experto para comenzar a involucrar al experto en el proceso y llevar a cabo un análisis crítico de los resultados que se comienzan a obtener.

III.B.8. Ejemplos de sistemas expertos

Proyecto ExTra (Diagnóstico)

Objetivo y marco del problema

ExTra es un sistema experto cuyo objetivo es colaborar con médicos dedicados al tratamiento de pacientes en etapa de postoperatoria de transplantes. ExTra fue desarrollado en Alemania como producto de la colaboración de distintos organismos especializados de los ámbitos privados y públicos. El objetivo principal de ExTra es el de colaborar

con los médicos de un centro de transplantes. Los problemas del tratamiento de pacientes transplantados están relacionados con varios factores. Por un lado los pacientes a ser tratados pueden sufrir tanto problemas de rechazo a órganos transplantados como infecciones debidas a la supresión de reacciones inmunológicas. Se torna necesario poder determinar, dada una complicación, si la misma se trata de un rechazo o de una infección. Por otro lado, el cuidado de los pacientes se complica debido a la continua fluctuación de la dotación de médicos. De este hecho se hace necesario que los médicos ingresantes dispongan de los conocimientos de los médicos salientes.

Requisitos del problema

A partir de la definición general del problema, podemos destacar los siguientes puntos salientes:

•Necesidad de consultas sobre distintos pacientes durante largos períodos. Se debe poder introducir nuevos datos para cada paciente, solicitar su análisis e iniciar, en caso de ser necesario, la consecuente consulta.

•El sistema debe poder dar hipótesis sobre las causas de las complicaciones y proponer diagnósticos.

•Debe ser posible la introducción de nuevo conocimiento por parte de los médicos sin necesidad de la intervención de un ingeniero de conocimiento. Debe existir, por lo tanto, una interface de usuario fácil de manejar y con una completa interface de explicación.

Proceso de desarrollo

Para el desarrollo de ExTra se utilizaron dos herramientas distintas en etapas sucesivas: ExTool y Babylon. La primera de ellas fue desarrollada paralelamente al proyecto.

El desarrollo de ExTra transcurrió a través de distintas etapas, en cada una de ellas de fueron desarrollando prototipos de sistemas expertos cuya funcionalidad se fue acercando cada vez más a la buscada.

Inicialmente se trabajó sobre la base de conocimientos del sistema experto Mycin (de diagnóstico de enfermedades infecciosas). Para este trabajo se desarrolló la herramienta ExTool.

En un segundo paso se realizaron las tareas de relevamiento del conocimiento. Durante esta etapa se hicieron reuniones entre dos ingenieros de conocimiento y dos expertos en la materia. De esta forma se fijó una terminología unitaria, y se logró entender los procedimientos llevados a cabo por los médicos para llegar a un diagnóstico y reconocer los principales parámetros del problema.

En la etapa subsiguiente se amplió la base de conocimientos y se implementaron cinco diagnósticos principales. En esta etapa los expertos trabajaron directamente con la herramienta ExTool. En una etapa posterior se pasó a implementar la base de conocimientos sobre la herramienta Babylon. En 1988 ExTool pasó a la fase de prueba clínica.

Descripción de solución

Base de Conocimientos

A partir de los formularios utilizados en la clínica, se reconoció el conjunto de parámetros clínicos a partir de los cuales los médicos elaboran los diagnósticos. Los parámetros clínicos incluyen el historial médico del paciente, resultados de análisis clínicos, datos sobre el donante del transplante, etc.

Cada uno de estos parámetros clínicos fue modelado utilizando frames y atributos de frames.

Existe una categoría de atributos denominados atributos complejos cuyos valores se obtienen a través de la aplicación de reglas. Promedios o sumas de ciertos valores dentro de un período constituyen ejemplos de atributos complejos. Los llamados estados médicos del paciente son también ejemplos de atributos complejos.

Generación y valoración de hipótesis

A partir de la utilización de reglas en sentido forward es posible obtener un conjunto de hipótesis adecuadas. También el usuario puede introducir hipótesis en el sistema.

Las hipótesis a las que llega el sistema constituyen instancias de frames con atributos especiales. Estos atributos indican evidencia positiva, evidencia negativa, síntomas ocultos, riesgo actual, etc. Existe un conjunto de hipótesis activas que es evaluado en cada sesión. Durante el proceso de evaluación puede llegar a desactivarse o activarse alguna hipótesis.

El proceso de evaluación de hipótesis consta de tres pasos: en un primer paso se calculan los valores indicativos para cada hipótesis activa. A continuación se determina el grado de aceptación para cada hipótesis a través de la asignación de un valor entre 0 y 1.

Finalmente se obtiene el estado de cada hipótesis pudiendo esta valuación adoptar tres posibles valores: *aceptado* - cuando la hipótesis se considera segura - , *considerado* - cuando se necesitan más medidas de diagnóstico para aceptarlo y *activo*, significando esto último sólo una evaluación a considerarse en la siguiente sesión. Teniendo cualquiera de estos tres estados, la hipótesis sigue activa respecto a futuras sesiones. También una hipótesis puede adoptar los estados de *rechazado* o *curado* en cuyos casos la vuelve inactiva.

Resultados

El sistema experto ExTra adoptó el estado de prototipo a principios de 1989. A ese momento la base de conocimientos constaba de 55 frames, 231 atributos y 474 reglas.

Luego de la última fase de la implementación, la base de conocimientos pudo continuar siendo ampliada con la única participación del especialista médico.

Por otro lado, el sistema experto no llegó a la situación de poder proponer medidas diagnósticas o terapéuticas.

Otros desarrollos

METAL

Metal es un traductor de textos técnicos. Se han desarrollado varias versiones lingüísticas: Alemán-Inglés, Ingés-Alemán, Holandés-Francés, Francés-Holandés, Español-Alemán, etc.

En los años '70 ya existía un sistema prototipo de traductor del alemán al inglés. A mediados de los '80, se comenzó a desarrollar el núcleo del software de METAL, mejorando con respecto al prototipo anterior los componentes lingüísticos y la integración y extensión de los diccionarios del sistema. Finalmente, en 1988 se lanzó como producto la versión del sistema de traducción alemán-inglés.

Paralelamente a este proceso se desarrollaron traductores para otras parejas de idiomas.. De esta manera se crearon centros en distintos países de Europa (Bélgica, España, etc.) para trabajar con las distintas versiones de acuerdo al lenguaje local. Este proceso provocó la necesidad de variar algunos componentes del sistema de modo de poder combinarlos para lograr las distintas versiones de traductor para las diversas parejas de idiomas.

El sistema experto METAL ha sido desarrollado utilizando una base de conocimientos basada en reglas. Las características propias del proceso de traducción hacen que sea necesario poder especificar distintos tipos de reglas para referir a los diversos niveles del lenguaje (palabra, frase, etc.). De esta manera surge una estructura común de reglas para representar el conocimiento de cada uno de los niveles del proceso de traducción.

METAL cuenta además con un módulo denominado METALSHOP. Este módulo fue desarrollado con el objetivo de contar con una serie de herramientas para proveer una adecuada interface tanto con el experto como con el usuario.

Mycin

Mycin constituye un punto importante en la historia de los desarrollos de sistemas expertos. Fue construido con el objetivo de diagnosticar enfermedades infecciosas y recomendar terapias apropiadas.

Mycin se desarrolló en Lisp y se encuentra basado en un sistema de reglas con encadenamiento hacia atrás utilizando razonamiento inexacto. El sistema evolucionó hasta manejar alrededor de 400 reglas.

El desarrollo de Mycin originó otros dos sistemas importantes. Por un lado **Emycin** (Empty Mycin) surgió como el esqueleto de Mycin para poder ser utilizado en el desarrollo de otros sistemas expertos. Por otro lado, se creó **Teiresias** para proveer una interface apropiada entre el experto humano y el sistema.

Xcon

El sistema experto Xcon fue desarrollado en la Universidad de Carnegie Mellon. El objetivo del sistema es el de resolver el problema de configuración de equipos de una línea de computadoras. Xcon resuelve tanto el tema de la integración como el de la distribución espacial.

El prototipo de Xcon contó originalmente con 250 reglas y fue desarrollado utilizando un lenguaje de construcción de sistemas expertos denominado **Ops 4**. La empresa fabricante de los equipos se incorporó luego al proyecto permitiendo su posterior evolución.

Genesis

El sistema Genesis fue desarrollado inicialmente en la universidad de Stanford. El objetivo del sistema es el de diseño de procedimientos de genética molecular.

El mecanismo de representación del conocimiento de Genesis es el de frames, utilizando encadenamiento hacia adelante y hacia atrás.

Genesis es utilizado actualmente por 500 científicos.

Delta

El desarrollo de Delta fue realizado por la empresa norteamericana General Electric. El objetivo del sistema es de asistir al personal de mantenimiento en el diagnóstico y reparación de un conjunto amplio de fallas en locomotoras Diesel.

La iniciativa del desarrollo se debió a la necesidad de reemplazar un empleado de la compañía que había acumulado una alta experiencia en la materia.

Delta no sólo asiste en la resolución de problemas sino que también cuenta con medios para realizar capacitación del personal técnico acerca de las herramientas a utilizar.

El sistema fue desarrollado originalmente en Lisp migrándoselo posteriormente al lenguaje Forth.

La evolución del Delta lo llevó a manejar hasta 1200 reglas con un grado del 80% de efectividad.

III.C. Introducción a los Sistemas Expertos Difusos

III.C.1. Imprecisión en sistemas expertos

Existen numerosas fuentes de imprecisión e incerteza en un dominio de sistema experto.

Durante el proceso de inferencia realizado por el sistema experto, puede surgir interacción con el usuario. Durante la misma, una pregunta imprecisa puede llevar a una respuesta también imprecisa. Un sistema experto debe ser capaz de formular preguntas en una manera que capture los significados vagos cuando éstos aparecen. El sistema experto también debe ser capaz de aceptar y proveer una interpretación para las respuestas que contienen cierta incertidumbre, y necesita poder explicar adecuadamente su razonamiento con la información incierta que ha recolectado.

El proceso de adquisición de conocimiento es también bastante impreciso. Es probable que el conocimiento adquirido no capture exactamente el del experto, especialmente desde el punto que el experto no está habitualmente enterado de todas las herramientas que usa en su proceso de razonamiento. El mismo conocimiento con el que él razona puede contener incertidumbre. Esto debe ser efectivamente capturado si se va a emular el proceso de razonamiento del experto.

Con respecto al proceso de razonamiento del experto, el mismo suele tener factores de incerteza. Es decir, la forma en que el conocimiento es usado para hacer inferencias y decidir sobre una solución a un problema no es generalmente un proceso preciso. Se hace necesario contar con una metodología confiable para capturarlo.

Mientras que el ingeniero de conocimiento captura el proceso de razonamiento, se pueden producir desviaciones. Lo que el ingeniero de conocimiento ve como proceso de razonamiento puede no coincidir exactamente con el proceso verdadero del experto. Dado que el experto no está siempre enterado del proceso de razonamiento actual que usa, las diferencias pueden no ser descubiertas. Los mayores errores podrán ser encontrados, pero sin embargo el sistema experto debe ser capaz de recuperarse con éxito de las pequeñas inconsistencias que sobrevivan, con la ayuda del mismo experto.

El lenguaje de representación del conocimiento también introducirá alguna incertidumbre en el sistema experto. Si el conocimiento no es expresado en algún lenguaje formal, el significado no puede ser interpretado exactamente. Dado que los ingenieros de conocimiento no han sido capaces de desarrollar o hacer uso universal de un adecuado lenguaje formal de representación, un esquema de representación de conocimiento debe ser elegido de manera tal que capture adecuadamente el conocimiento de un dominio en particular. Dado que el esquema que es elegido puede no proveer un marco exacto para la representación del conocimiento del experto, el sistema experto debe ser capaz de tratar con esta imprecisión en una manera consistente. Las técnicas de razonamiento difuso pueden proveer la base para representar la imprecisión inherente en el conocimiento del experto. Puede ser usado en conjunción con la teoría de probabilidades y la teoría evidencial de Dempster-Shafer, dependiendo del caso.

La información incompleta también agrega incertidumbre al sistema. No se puede extraer siempre toda la información que maneja el usuario. En este caso, se desea igualmente llegar a una conclusión si es posible, aún si la conclusión es menos cierta. El razonamiento de sentido común acerca de la realidad incluye muchas inferencias a partir de información incompleta. Este es un problema que recién ha sido empezado a ser atacado pero parece encajar dentro del razonamiento difuso. En cualquier evento, un sistema experto necesita ser suficientemente robusto para superar casos en los cuales existe información incompleta. Esto puede ser realizado mediante el razonamiento sobre las incompletitudes en el conocimiento. La incorporación de técnicas difusas en un sistema experto debería permitir que se apliquen alguno de estos métodos.

Los datos que el sistema experto adquiere mientras trata de resolver un problema usualmente provienen de humanos, por lo tanto no están libres de error y no se puede esperar una precisión absoluta. Aún si la respuesta es elaborada con precisión, puede no ser una respuesta binaria. Como fue mencionado antes, es extremadamente importante que un sistema experto de propósito general sea capaz de aceptar e interpretar apropiadamente la información imprecisa provista por el usuario del sistema. En casos donde el sistema experto recibe datos de aparatos sensores o algún tipo de equipamiento, el ya conocido problema del ruido aparece y agrega imprecisión a la ecuación. Además, el equipamiento puede estar desajustado. La interfase del sistema experto al equipamiento externo debe ser capaz de aceptar que las lecturas son imprecisas o difusas y efectivamente tratar este caso.

Existe también incertidumbre en la base de conocimientos donde puede haber implicaciones débiles, dadas por el experto. Al traducir la oración del experto a la implicación, se puede perder el significado de alguna manera. El sistema

experto debe poder aceptar y usar las implicaciones débiles junto con otras fuentes de información para llegar a una conclusión acerca del problema bajo investigación.

Otro tipo de incertidumbre aparece en la colección de conocimiento de diferentes fuentes, expertos, literatura, etc. Pueden existir bancos de conocimiento conflictivos, redundantes, subsumidos o faltantes. En sistemas basados en reglas, este tipo de incertidumbre ha llevado a la compilación de las reglas dentro de una red donde pueden ser examinadas para estos problemas. Parte del conocimiento faltante se hace evidente cuando el sistema es probado. Las piezas de conocimiento que subsumen a otras pueden ser encontradas por el ingeniero de conocimiento o a través de las trazas de la operación del sistema. Los conflictos también son normalmente encontrados mediante las pruebas del sistema, pero es posible que una función difusa, que opere en la base de conocimientos, pueda ser diseñada y usada para indicar este tipo de conflictos.

III.C.2. La evolución hacia los sistemas expertos difusos

Se debe contar con un método apropiado para tratar la imprecisión en un sistema experto, de manera tal de que éste pueda tener éxito en transformarse en una herramienta útil. También necesita ser natural para que el conocimiento fluya libremente desde el experto. Actualmente hay muy pocas formas diferentes en que la imprecisión pueda ser manejada en un sistema experto. Las que son completas tienden a permitir que la imprecisión y las incertezas sean manejadas en un grado pequeño o en una forma poco flexible. La mayoría de los métodos para manejar imprecisiones están basados en probabilidades. Es interesante notar que los expertos usualmente no piensan en valores probabilísticos, sino en términos tales como *mucho, usualmente, siempre, algunas veces*, etc. A pesar de que es posible darles a estos términos valores probabilísticos, también existe la posibilidad de usar valores difusos que capturen el significado en una manera natural.

A continuación, se verán brevemente dos ejemplos de sistemas expertos que tratan con incertidumbre: MYCIN y CASNET.

MYCIN introdujo el concepto de factores de certeza. Este concepto ha sido ampliamente aplicado, de muchas formas, para manejar la imprecisión en sistemas expertos. Cada regla MYCIN tiene una fuerza, llamado factor de certeza, que se ubica en el intervalo [0,1]. Cuando una regla es disparada, su premisa es evaluada y un valor numérico entre -1 y 1 es asociado con la premisa. Si el valor de la premisa está afuera del intervalo formado por el valor previo (p.ej. 0.2 y -0.2 respectivamente), entonces la parte de la acción de la regla es evaluada y la conclusión es obtenida con una certeza que es igual al valor de la premisa por el factor de certeza de la regla. La evidencia por hipótesis es dividida en medidas de creencia y descreencia. La medida de creencia es un valor en el rango [0,1], y la medida de descreencia está en el rango [-1,0]. Una hipótesis es creída o descreída si la hipótesis está por arriba o por debajo de la respectiva cota.

CASNET es un sistema experto basado en una red semántica. Cada nodo (o estado, como es conocido en este contexto) en la red tiene un peso hacia adelante y un peso hacia atrás asociado con él. Estos pesos unen con mayor o menor fuerza a los nodos, dando así el concepto de causalidad entre ellos. Esto posibilita que la red sea recorrida en un sentido o en otro. Los pesos corresponden a las siguientes interpretaciones de fuerza causal: algunas veces, frecuentemente, usualmente, casi siempre, y siempre. Las reglas son usadas para asociar las observaciones con estados. Estas (las reglas) tienen un valor de confianza asignado, entre -1 y 1. Han sido definidos métodos para combinar valores de confianza. A cada estado se le asocia un valor de certeza. Estos denotan la creencia que existe de que el estado ha ocurrido. Los estados o nodos son considerados confirmados cuando el factor de certeza está por arriba de cierta cota. Las observaciones generalmente son tests. La observación con el resultado positivo o negativo más grande es usada para confirmar o negar un estado. La combinación de observaciones y caminos confirmados parcialmente permiten que se realice la inferencia bajo incertidumbre y se obtengan conclusiones.

Estos métodos y otros son razonablemente efectivos en casos especiales y en dominios específicos. De hecho, muchas variaciones del método de MYCIN para tratar incertezas están actualmente en uso. Los métodos difusos pueden, en ciertas áreas, reemplazar los métodos clásicos y proveer un mejor desempeño. En otros pueden trabajar en convivencia con métodos basados en probabilidades. La necesidad de un método teórico para tratar incertezas nos lleva a la teoría de conjuntos difusos. Antes de profundizar en este aspecto, debe ser mencionada la teoría de evidencia de Dempster-Shafer, la cual está basada en probabilidades. Un problema con esta teoría es que la regla de combinación de evidencia puede hacer muy grande la medida de certeza de un hecho, si se usa una normalización para eliminar o esconder una contradicción. Este método es realmente valioso en un dominio en el cual la mayoría de la incerteza puede ser efectivamente especificada por probabilidades. Sin embargo, donde el dominio no se presta para las estimaciones o medida de probabilidades, la teoría de conjuntos difusos ofrece una alternativa. Donde los conjuntos difusos o los

términos lingüísticos describen mejor a los items de un dominio, las teorías de conjuntos posibles y necesarios pueden ser aplicadas más naturalmente que las probabilidades.

III.C.3. Sistemas Expertos Difusos

Por sistema experto difuso se entiende un sistema experto que incorpora conjuntos difusos y/o lógica difusa dentro de su proceso de razonamiento y/o esquema de representación del conocimiento. En las últimas décadas se han desarrollado muchos sistemas expertos que incorporan técnicas difusas.

La base de conocimientos de un sistema experto difuso basado en reglas, está representada usualmente por un conjunto de reglas difusas, que conectan antecedentes con consecuentes, premisas con conclusiones, o condiciones con acciones. Generalmente tienen la forma de "Si A, entonces B", donde A y B son conjuntos difusos.

El motor de inferencias de un sistema experto difuso generalmente opera sobre una serie de reglas y efectúa inferencias difusas. Existen dos aproximaciones para evaluar las reglas relevantes en un sistema experto difuso, al igual que en los sistemas expertos convencionales. La primera es *data-driven* y es ejemplificada por el modus ponens generalizado. En este caso, los datos disponibles son suministrados al sistema experto, el cual los usa para evaluar las reglas relevantes y deducir todas las conclusiones posibles. Un método alternativo de evaluación es *goal-driven*; es ejemplificado por el modus tollens generalizado de la inferencia lógica. Aquí, el sistema experto busca por datos especificados en las partes IF de las reglas que guiarán al objetivo; estos datos son encontrados en la base de conocimientos, o bien preguntando al usuario. Dado que el método *data-driven* procede de las partes IF a las partes THEN en la cadena a través de las reglas, se llama comunmente *encadenamiento hacia adelante*. Similarmente, dado que el método *goal-driven* procede hacia atrás, desde las partes THEN (objetivos) a las partes IF en su búsqueda por los datos requeridos, se lo conoce como *encadenamiento hacia atrás*. El encadenamiento hacia atrás tiene la ventaja de la velocidad, dado que sólo las reglas que guian al objetivo necesitan ser evaluadas. Más aún, si existen datos difíciles de obtener, y éstos son solo potencialmente necesarios, entonces el método de encadenamiento hacia atrás es claramente superior.

Las teorías de conjuntos difusos y lógica difusa utilizadas, entre otros, en la construcción de sistemas expertos difusos, son fuertes y se encuentran bien fundadas. Estas teorías existen hace más de 25 años y han mostrado ser muy útiles en muchas aplicaciones; por ejemplo de control. La lógica difusa es empleada, por ejemplo, en el control de trenes en Japón. Se ha implementado exitosamente mediante un controlador experto difuso, que se encuentra actualmente on-line en la ciudad de Sendai. El mismo provee cambios más suaves en la velocidad de los trenes que sistemas no difusos nunca han alcanzado. En este sistema se usan variables lingüísticas difusas para hacer cálculos internos y proveer la estrategia de control.

La base teórica detrás de las técnicas difusas permite tratar la incertidumbre con una buena fundamentación teórica. La teoría usada apropiadamente permite que esquemas de razonamiento difuso sean desarrollados y aplicados a un amplio espectro de problemas sin cambios mayores. Un conjunto consistente y unificado de métodos puede ser desarrollado para razonar con incertezas en los sistemas expertos.

Ejemplos

Existe un sistema lógico de lingüística difusa desarrollado por Whalen y Schott, que utiliza reglas de producción para sugerir técnicas apropiadas de predicciones de ventas. Este sistema experto usa probabilidades y trabaja con encadenamiento hacia atrás. Empieza con las posibilidades de todas las conclusiones igualadas a uno e intenta reducir las posibilidades hasta que encuentra un conjunto irreductible.

SPERILL II, un sistema experto que trabaja con daños a estructuras, usa conjuntos difusos para representar los datos imprecisos.

FLOPS es un sistema experto difuso basado en reglas. La entrada al sistema es un vector cuyos componentes son todos conjuntos difusos. Como salida produce un conjunto difuso de conclusiones. La información de entrada es comparada mediante patrones contra reglas para proveer un conjunto difuso de reglas disparables, las cuales son ejecutadas. Este proceso continua hasta que no se encuentran más reglas disparables, y luego produce un conjunto de conclusiones difusas. FLOPS ha sido vendido comercialmente por muchos años, y ha sido usado con éxito para modelar muchos dominios diferentes.

El sistema Z-II es un shell de sistema experto difuso basado en reglas, desarrollado por K. S. Leung y W. Lam (ver [LEU/88]), que trata efectivamente con incertezas e imprecisiones. Permite cualquier combinación de términos difusos y normales, e incertezas. Emplea lógica difusa y números difusos para su razonamiento inexacto. Ha sido usado para construir muchos sistemas expertos; los dominios han sido diagnóstico médico, psicoanálisis y análisis de riesgos. Según los mismos expertos, se puede expresar el conocimiento de una manera natural: con términos lingüísticos difusos. Una regla de ejemplo es la siguiente:

SI tu interés en analizar el cuerpo humano es alto,
ENTONCES tu interés general en la medicina debería ser alto (0.95).

En tanto los términos lingüísticos sean definidos por alguna función de pertenencia razonable, éste es un formato de representación de conocimiento muy efectivo para la información imprecisa e incierta.

Se han desarrollado sistemas expertos difusos de propósito general. Un ejemplo de este tipo de sistemas puede encontrarse en [HAL/91].

Otro tipo importante de sistema para el cual los conjuntos difusos proveen una base conveniente es el razonamiento de sentido común. La *"theory of usuality"* propuesta por Zadeh provee una herramienta para incorporar algo de información de sentido común en los sistemas expertos. El concepto de *usuality* hace referencia a eventos que son usualmente verdaderos o tienen una alta probabilidad de ocurrencia. Una proposición puede tener implícitamente un *usualmente*. Por ejemplo, *la nieve es blanca, Buenos Aires es húmeda, los días ventosos son buenos para navegar* son proposiciones cuyo *usualmente* implícito puede ser interpretado como un cuantificador difuso. Este último es básicamente una proporción difusa. Esta teoría puede ser aplicada a situaciones ordinarias y proveer un método para efectivamente representar conocimiento acerca de eventos o items que son generalmente verdaderos. Esto incluye muchos conceptos de sentido común.

La teoría de conjuntos difusos provee un método natural para tratar con los términos lingüísticos con los cuales un experto describe mejor un dominio de problema. El uso de esta teoría en sistemas expertos ha causado una evolución de sistemas.

IV. ESTUDIO DEL PROBLEMA Y DE SU SOLUCIÓN

Introducción

En esta sección se hará una reseña de las características principales del problema planteado por el personal de N.A.S.A. y a continuación se expondrá el estudio de la solución propuesta, tal como fue planteada finalmente al organismo. En secciones posteriores a la presente, serán considerados tanto los aspectos de diseño e implementación de la solución planteada como los fundamentos teóricos correspondientes.

IV.A. Etapas preliminares

Los primeros pasos en el desarrollo del proyecto transcurrieron durante las entrevistas iniciales con los responsables del mismo por parte del organismo.

En dichas entrevistas iniciales se esbozaron los puntos principales del proyecto. Existía ya un proyecto global, del cual la aplicación final formaría parte, que consistiría de un **Sistema de Ayuda al Operador** para los operadores de la Central Nuclear Embalse en la provincia de Córdoba. El proyecto estaría dedicado a brindar una herramienta a los operadores de la central para que colabore con ellos en las tareas de detección de eventos anormales en el reactor y del seguimiento de los pasos a seguir para su posterior mitigación.

Tal como se ha citado previamente, la solución al problema del diagnóstico de eventos anormales fue subdividida inicialmente por los responsables del proyecto en N.A.S.A. en dos partes:

La primera parte, que es la que se desarrolla en este trabajo, enfocaría la sección de los POEAs dedicadas al diagnóstico de los eventos anormales y la segunda parte estaría relacionada con la sección de los POEAs que informan al operador sobre los pasos a seguir para mitigar el evento detectado.

La justificación de la necesidad de contar con una aplicación que brindara esta solución se basaba en que los operadores de la CNE necesitaban de una herramienta que colaborara con ellos en los procesos de detección y mitigación de eventos anormales. La base de esta idea estaba dada por el hecho de que el facilitar ambos procedimientos a los operadores contribuiría a asegurar la confiabilidad de sus conclusiones y haría más seguro su trabajo en la central; y por supuesto esto incrementaría la seguridad en el funcionamiento general de la planta.

IV.B. Estudio del problema

IV.B.1. Entrevistas con los expertos

Una vez definido y asignado al grupo de trabajo la parte correspondiente del proyecto global planteado por N.A.S.A., de detección de eventos anormales, el trabajo inicial se dedicó al análisis de los POEAs. Éstos constituyen la documentación principal que utilizan los operadores de la central para realizar su tarea habitual. El análisis de los POEAs se llevó a cabo a través de una serie de entrevistas con los expertos.

Dadas las características de los diagramas de los POEAs, el análisis de los mismos derivó en dos puntos principales: el significado de los símbolos presentes en ellos y la información brindada por los diagramas, la cual es utilizada por los operadores durante el proceso de diagnóstico.

El objetivo principal de las entrevistas fue el de interiorizarse de la documentación utilizada por los operadores de la central. Siendo el objetivo del desarrollo el crear un sistema experto de ayuda, esta documentación es una fuente importante e imprescindible de información para la creación de la base de conocimientos y para el diseño de la interface de la aplicación utilizadas por el sistema.

Las entrevistas con los expertos se realizaron a través de una serie de reuniones durante las cuales se grabó en cinta el contenido de las mismas para facilitar el posterior re-análisis.

Grabar las conversaciones con los expertos fue de utilidad para poder analizar posteriormente la información recolectada durante las reuniones, repasar el vocabulario de los expertos y poder registrar su contenido por escrito, de manera de evitar posteriores contradicciones.

Los elementos principales que se analizaron en las entrevistas fueron los siguientes:

- Conceptos básicos sobre el funcionamiento del reactor: Con el objetivo de conocer la función que debería cumplir la aplicación y las características del entorno operativo para el que sería desarrollada, el grupo de N.A.S.A. explicó una serie de conceptos básicos sobre el funcionamiento de un reactor nuclear basándose específicamente en el de CNE.

- Especificación y alcance del sistema: Se definió el alcance buscado para el sistema. Se aclaró la subdivisión inicial que se realizó sobre la solución al problema (subsistema de diagnóstico y subsistema de seguimiento). Se convino además en que esta separación inicial en dos proyectos surgió exclusivamente a partir del criterio aplicado por los responsables del proyecto en N.A.S.A.

- Colaboración de la aplicación con el operador de la sala de control: El grupo de N.A.S.A. especificó al equipo de desarrollo el tipo de ayuda o colaboración que la aplicación debería prestar a los operadores. El objetivo de la aplicación sería extraer conclusiones exclusivamente a partir de la base de conocimientos y de la información proveniente de las señales del reactor. El sistema de ayuda no tendría que solicitar al operador ningún tipo de información adicional para realizar sus inferencias.

- Entorno de trabajo en la CNE: En las entrevistas con el personal de N.A.S.A. el grupo de desarrollo se interiorizó sobre las características de la sala de control de la central Embalse. Se reseñó el equipamiento existente en la central y el trabajo habitual de los operadores.

- POEAs. Estudio de su configuración: El objetivo principal de las entrevistas fue el de relevar el conocimiento de los expertos sobre los POEAs. A través de dichas entrevistas, el grupo de desarrollo se interiorizó sobre los distintos aspectos de la simbología presente en los diagramas de POEAs. También se analizaron los mecanismos presentes en la central para obtener la información sobre el estado del reactor y los medios principales existentes en la sala de control cuya función es analizar dicha información para presentarla a los operadores (sensores del reactor, equipo de computadoras, manejo de señales, etc.).

IV.B.2. Requerimientos planteados por N.A.S.A.

La necesidad específica de N.A.S.A. con respecto al subsistema de diagnóstico consistía básicamente en contar con un equipo en el cual pudiera especificarse de alguna manera el conocimiento de los expertos - especialmente el volcado en los POEAs - sobre el proceso de detección de eventos anormales. El sistema, de esta manera, estaría en funcionamiento en forma permanente en la sala de control principal de la CNE, recibiendo información de los sensores del reactor, con el fin de que los operadores obtuvieran del mismo el conocimiento del estado del reactor en todo momento.

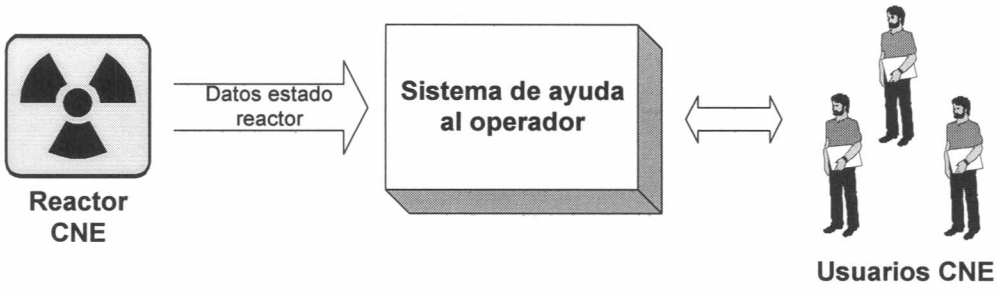
Ante la ocurrencia de un evento anormal, el sistema debería informarlo al operador poniendo a su vez en conocimiento del mismo las causas que lo llevaron a concluir la existencia del evento anormal.

Los POEAs serían la fuente principal de información sobre el proceso de detección de eventos anormales. Los expertos responsables deberían ser capaces, además, de incorporar al sistema nueva información proveniente de POEAs desarrollados en el futuro.

Debido a que el equipamiento no se encontraba aún preparado en la sala de control de la central, se convino en que la aplicación a desarrollar obtuviera los datos de los controles de la central a partir de un almacenamiento interno de pruebas de la computadora donde se ejecutará la aplicación. Una vez que el sistema se pusiera en funcionamiento en el entorno definitivo, se alimentaría dicho almacenamiento con los datos reales de la planta. Asimismo, se convino que la comunicación con el subsistema de seguimiento de los procedimientos de operación para mitigar los eventos anormales no será desarrollada en el presente trabajo.

IV.B.3. Definición preliminar del sistema

Una vez descriptas las características generales de los requerimientos de N.A.S.A. con respecto al subsistema asignado, es posible exponer una idea preliminar del objetivo planteado.



Existirá un conjunto de usuarios del sistema de detección. Estos usuarios se distinguirán según su función en *expertos* y *operadores*.

Los usuarios expertos utilizarán una interface especialmente diseñada para que éstos puedan especificar al sistema el conocimiento referido a la detección de los eventos anormales ocurridos en el reactor. Dicho conocimiento de los expertos se volcará de acuerdo a un lenguaje diseñado para tal fin.

Dada la naturaleza de la especificación de la documentación que constituye la base del sistema de detección - los POEAs - el lenguaje a utilizar por los expertos deberá tener dos características fundamentales: guardar en su estructura de definición una relación apropiada con el dominio del problema y permitir la especificación difusa de los estados de las señales.

Los usuarios operadores del sistema se dedicarán a atender periódicamente la información presentada a ellos sobre el resultado del monitoreo cíclico del estado de la planta en base al conocimiento ingresado oportunamente por los expertos. Ante la presencia de un evento anormal, el sistema presentará la información necesaria para que los operadores puedan conocer la falla detectada, dando lugar, en caso de que el operador así lo decida, al módulo correspondiente dedicado al seguimiento del proceso de mitigación del evento anormal.

El sistema obtendrá la información del reactor a través de la lectura de una base de información que contendrá los datos correspondientes a los sensores que se encuentran distribuidos en la planta. La base de información será alimentada oportunamente por un sistema dedicado a ese fin. Este objetivo execederá al objetivo del presente trabajo.

Limites y alcances

Comunicación Reactor - Sistema de ayuda al operador

El Sistema de detección no se detendrá en los aspectos específicos de la comunicación con los elementos del reactor que entreguen la información del reactor. El problema de la obtención de los datos no involucrará a este desarrollo sino que el sistema consultará una base de datos interna, que en la puesta en producción definitiva deberá ser alimentada por una tercera aplicación.

Usuarios del sistema

Como fue mencionado anteriormente, los usuarios del sistema se caracterizarán según dos grupos principales. Los operadores de la planta utilizarán el sistema habitualmente para conocer el estado del reactor y estar al tanto de los eventos anormales que se puedan presentar. Existirán uno o varios usuarios calificados que serán los encargados de actualizar el conocimiento del sistema que será utilizado para examinar los resultados obtenidos de la planta y determinar su estado y la presencia de eventos anormales.

Obtención de los resultados

El análisis de los datos por parte del sistema de detección no estará acotado por márgenes de tiempo. Esto último se corresponde con un requerimiento de un sistema de tiempo real. Este objetivo adicional puede constituir la base de una futura ampliación al sistema de detección. En definitiva, el proceso de inferencia en base al conocimiento incorporado por los expertos se extenderá en el tiempo de acuerdo a la cantidad de datos e inferencias que se deban analizar.

Información proporcionada a los operadores

La función asignada al sistema se cumplirá utilizando exclusivamente como base de conocimientos la información volcada por los expertos junto con los datos recabados de la planta. El sistema no solicitará información adicional a los operadores y en caso de detectar un evento anormal se dedicará a explicar detalladamente las bases de su conclusión.

El sistema tampoco sugerirá pasos a seguir ya que deberá mostrar a los operadores las diversas conclusiones a las que ha llegado sin ambigüedades y de la manera más clara posible.

IV.C. Estudio de una solución al problema

IV.C.1. Desarrollo de un sistema experto

La solución del problema involucra una tarea de representación del conocimiento especificado en el sistema original de POEAs.

La naturaleza de la información del sistema original hace que una automatización del proceso de diagnóstico deba considerar la creación de un sistema que maneje conocimiento proveniente de la especificación actual más la información proporcionada por los expertos y la experiencia adquirida por los operadores.

En la sección "**IV. D.** Aplicación de Sistemas Expertos al problema de la CNE" se explican detalladamente los fundamentos que justifican la decisión de la creación de un sistema experto para resolver el problema.

IV.C.2. Representación del conocimiento

La especificación misma del problema hace que sea "natural" encontrar una relación directa entre cada procedimiento de operación y un conjunto de reglas. Esta relación indica que una representación basada en reglas surga como lo más apropiado para solucionar el problema de representación. En general, las precondiciones de los POEAs se pueden dividir en subconjuntos de precondiciones que se pueden representar como nuevas reglas de menor complejidad, facilitando de esta forma la especificación del conocimiento. Sin embargo, se hace necesaria la búsqueda de algún otro tipo de sistema experto que también pueda servir como alternativa al problema de diagnóstico.

Un sistema experto basado en casos (CBR - case based reasoner) es, en principio, una propuesta viable. Sin embargo, un CBR debe contar con una base de conocimientos que contenga una historia suficientemente abundante de casos que pueda cubrir todos los eventos anormales existentes, al menos al momento de su puesta en funcionamiento. Pero en el problema de la CNE no se encuentra dicha experiencia de casos, justamente por la falta de experiencia adquirida ante eventos anormales que suceden con muy poca frecuencia. Esta es la razón principal por la cual no se puede adoptar como solución un CBR. Dicho argumento fue discutido en las entrevistas mantenidas con NASA, y se llegó a la misma conclusión.

Un sistema experto con frames está basado en la representación de los elementos del problema a través de conjuntos o clases a los que se le asignan características y se los relaciona. Sin embargo, esta orientación es adecuada para representar problemas de naturaleza taxonómica, y no un problema como el descripto de detección y diagnóstico.

IV.C.3. Aspectos difusos de la solución

El conocimiento presente en el sistema de diagnóstico incluye aspectos de razonamiento aproximado que hacen viable la utilización de la lógica difusa. La información presente en el sistema original de POEAs está claramente especificada: no hay ambigüedades, falta de información ni tampoco datos redundantes. Sin embargo, la naturaleza misma del diagnóstico en la verificación del estado del reactor, más la experiencia adquirida por los operadores de la planta y sumado al conocimiento específico de los expertos dan como resultado ciertas heurísticas y formas de diagnosticar eventos anormales que sí conllevan aspectos de razonamiento aproximado. También, la referencia a los estados de las variables en la especificación de los procedimientos de operación indican ciertos factores que no se pueden representar mediante la lógica clásica bivalente. Esto lleva a utilizar la lógica difusa como una herramienta apropiada y necesaria a la hora de conseguir una representación y manipulación apropiada de los datos del problema.

Se puede pensar que la teoría de conjuntos difusos puede servir a los fines de modelizar el problema también de la siguiente manera. Si asumimos que los estados de la planta son elementos de un universo U, dado un POEA específico $\mathcal{P}$, decimos que este último está representado en nuestro modelo por un conjunto difuso P. Cada estado de la planta tiene asociado un valor de pertenencia a P: habrá estados en los cuales el valor es nulo y otros en los cuales es

el máximo posible. Aquellos estados en los cuales se observa una tendencia al evento anormal P, tendremos un valor de pertenencia intermedio o "difuso".

Con el fin de especificar el mecanismo de representación del conocimiento en el sistema experto, es necesario obtener un lenguaje que incluya las características indicadas previamente. Esto es, un lenguaje con aspectos difusos que permita representar mediante reglas el conocimiento del experto acerca de los procedimientos de operación.

A los efectos de demostrar la efectividad del lenguaje en cuanto a su poder expresivo con respecto a la semántica buscada, se hace necesario buscar una formalización teórica que sustente la implementación computacional.

El lenguaje a desarrollar deberá estar estrechamente vinculado con la forma de representación de conocimiento elegida para el problema. Es decir, deberá tener el poder expresivo suficiente para permitir representar todo el conocimiento presente en el sistema original de POEAs, y de la manera más aproximada al experto como sea posible.

También se deberá contemplar el aspecto computacional, dado que finalmente se implementará un sistema automatizado. Es decir, el lenguaje deberá ser computacionalmente tratable, y será necesario considerar un balance entre poder expresivo y tratamiento computacional.

La propuesta finalmente adoptada como implementación incluye el desarrollo de un sistema experto basado en reglas que utiliza aspectos de lógica difusa para razonar.

Inclusión de componentes difusos

El lenguaje a desarrollar deberá contemplar alguna manera para expresar el conocimiento aproximado. Con este fin, se propusieron, entre otros, tres conceptos básicos: incidencias, confianzas y modificadores.

Las incidencias surgieron debido al hecho de que en una misma regla, en algunos casos era necesario dar mayor importancia a determinadas precondiciones sobre otras, según lo conversado durante las reuniones mantenidas con NASA. Esto llevó a que se propongan incidencias para asignar factores de importancia para la veracidad de cada precondición de una regla en el cálculo de la veracidad de la conclusión. Es decir, la veracidad del consecuente de una regla se calcula en función de las veracidades de sus precondiciones y sus respectivas incidencias.

Las confianzas de las reglas representan el grado de confiabilidad que tiene el experto que define y escribe las reglas en cada una de éstas. Estas surgieron debido a que se observó que determinadas reglas tenían un mayor grado de certidumbre que otras con respecto a lo que concluían. De esta manera, este elemento adicional del lenguaje permite al experto expresar su propia confianza en cada regla con respecto a las heurísticas y datos que él mismo conoce.

Los modificadores constituyen extensiones lingüísticas que actúan sobre la semántica de los predicados. Estos permiten lograr un mayor acercamiento al lenguaje propio de los expertos, logrando que se modifique el grado de verdad de los predicados en las reglas a través del uso de adverbios (*algo, muy, poco, etc.*). Los modificadores deben ser definidos oportunamente por los expertos mediante funciones matemáticas, como fue explicado anteriormente en el capítulo "**III.A.** Introducción a la lógica difusa".

IV.D. Aplicación de Sistemas Expertos al problema de la CNE

IV.D.1. Introducción

A partir de la definición del problema presentado en la CNE, analizaremos las características de una solución al mismo basada en la aplicación de un sistema experto, como fue decidido en el capítulo anterior.

El objetivo es mostrar los requerimientos del problema al momento del comienzo del desarrollo, reconocer las características del mismo que lo vinculan a soluciones relacionadas con el desarrollo de sistemas expertos y reconocer los aspectos principales de los sistemas expertos que se explicaron anteriormente que se pueden aplicar a la presente solución.

IV.D.2. Objetivo de la aplicación de un Sistema Experto

Posteriormente a dividirse la solución al problema de la detección y mitigación de eventos anormales, por parte de los responsables del proyecto en NASA, se llevó a cabo la definición de la solución específica para el problema de la detección de eventos anormales.

Mientras que para la etapa de ayuda al operador en la tarea de mitigación del evento anormal detectado (basándose siempre en lo especificado en el POEA correspondiente) se reduce a un problema de seguimiento de una

serie de pasos claramente explicitados, el problema de la ayuda al operador para la determinación del procedimiento de mitigación a aplicar está más relacionado con el área de razonamiento.

Se pueden destacar las siguientes características principales del problema de la detección de eventos anormales; donde un sistema experto podría responder satisfactoriamente:

- Existe documentación sobre el conocimiento del experto: A pesar de la existencia de un experto en el tema, se ha desarrollado una documentación (POEAs) cuyo objetivo es cubrir el conocimiento del experto para que el mismo esté disponible a los operadores durante su trabajo en la sala de control.

A menudo, en los ámbitos donde se lleva a cabo el desarrollo de sistemas expertos, el conocimiento que se desea relevar se encuentra únicamente en poder del mismo experto. En este caso, existe documentación que es la realmente utilizada por los usuarios y a partir de ésta y con ayuda del equipo de expertos, es posible llevar a cabo el proceso de ingeniería de conocimiento.

- Ayuda al operador de la central: La idea de contar con un sistema que colabore con los operadores de la sala de control principal es el aspecto sobre el que más se han interesado los responsables del proyecto en la CNE.

El concepto de "ayuda al operador" está relacionado con el hecho de que el sistema experto a desarrollarse podrá proveer la información que la documentación de POEAs ya brinda pero en menor tiempo y con menor probabilidad de errores o confusiones. Si tenemos en cuenta que el operador debe consultar las páginas de un manual de operaciones en momentos en que se presentan eventos inesperados en la central, se hace evidente que la tensión y la ansiedad de los operadores humanos los lleve a equivocarse en el momento de estudiar la(s) causa(s) del evento anormal.

Es en este punto donde se hace evidente la importancia de la existencia de un sistema experto de ayuda. El sistema podrá determinar qué evento anormal se ha producido utilizando conocimiento incorporado previamente en base a la documentación de los POEAs.

- Historial de problemas en la central: Como se ha descripto previamente, parte del proceso de ingeniería del conocimiento consiste en estudiar soluciones dadas a eventos anormales que han ocurrido anteriormente. En el caso particular del problema de la Central Nuclear de Embalse, se cuenta con una escasa historia de eventos anormales producidos. Debido a esto, el proceso de ingeniería del conocimiento debe restringirse casi en su totalidad al estudio del diseño de los POEAs.

- Incorporación de futuros (y posibles) eventos anormales: Existiendo la documentación necesaria para analizar los controles de la sala y detectar los eventos anormales, es válido preguntarse dónde yace la conveniencia de desarrollar un sistema experto para este ámbito de trabajo.

Un aporte fundamental que puede brindar la incorporación de un sistema experto al ámbito de operación de CNE, es la posibilidad de brindar una herramienta que permita definir los POEAs existentes más los POEAs nuevos que se desarrollen con el tiempo. También, contar con el medio para realizar simulaciones de sucesos de eventos anormales, y a partir de su estudio analizar la definición de nuevos POEAs. De esta manera se lograría acelerar el proceso de reconocimiento de posibles eventos anormales en la CNE agilizando la actualización y revisión a la que es sometida frecuentemente el sistema de POEAs.

IV.D.3. Desarrollo posible, justificado y apropiado.

A continuación analizaremos el desarrollo del sistema experto para CNE desde los puntos de vista presentados con anterioridad (en "**III. B.** Introducción a los Sistemas Expertos"). Veremos la posibilidad, justificación y conveniencia de la aplicación de un sistema experto para el problema de la CNE.

Posibilidad de desarrollo del sistema experto.

Para considerar este punto recurriremos a los aspectos planteados previamente y analizaremos con cada uno de ellos el caso particular del desarrollo del sistema experto.

•**Existencia del experto:** Según lo descripto en la presente documentación, sabemos que el dominio del problema que se está tratando es el de la detección de eventos anormales en la planta de la Central Nuclear Embalse.De acuerdo a la información recabada en las reuniones iniciales con los responsables del proyecto en N.A.S.A. nos hemos

encontrado con que el conocimiento detallado sobre la tarea de reconocimiento de eventos anormales en la planta se halla concentrado principalmente en la persona del Ing. Batistic, quien luego de años de experiencia y de entrenamiento ha adquirido el conocimiento necesario sobre el problema.

Dicho experto de CNE ha reelaborado en dos ocasiones - junto con sus colaboradores - el conocimiento sobre la tarea de detección de eventos con el fin de obtener una documentación que sirva de ayuda a los operadores de la central. Esta reelaboración ha dado finalmente como resultado los documentos llamados POEAs que sintetizan los procedimientos esenciales necesarios para poder llevar a cabo un diagnóstico de problemas de la planta.

Encontramos aquí dos grupos diferenciados: el experto y los operadores de la planta. La actividad de estos últimos está limitada sólo a interpretar la información brindada por los POEAs ante un evento anormal.

Finalmente, vemos que la existencia de los POEAs no elimina la importancia del experto ya que es él mismo quien los ha redactado y uno de los que puede dar la autorización necesaria para modificarlo, ya sea en su formato original o en la futura base de conocimientos del sistema experto.

•**Posibilidad de estructurar el conocimiento del experto:** La posibilidad por parte del experto de estructurar su conocimiento se encuentra presente en este problema. El mejor ejemplo de esta característica lo presenta la existencia de los POEAs. Los POEAs, en cuanto a documentación basada en el conocimiento del experto para ser utilizada por los operadores, surgen gracias a que el experto es capaz de formalizar su conocimiento.

•**Concordancia en las soluciones entre distintos expertos:** El problema del diagnóstico de eventos anormales se encuentra estudiado y analizado por expertos del país de origen de la planta (Canadá) y por los expertos de N.A.S.A., quienes han concurrido a cursos de entrenamiento. Además, los documentos que forman la base del conocimiento del sistema experto, los POEAs, han sido revisados por los especialistas luego de su redacción.

Por lo tanto, las soluciones que plantean los expertos a cada una de las anormalidades que pueden surgir en la planta nuclear, cuentan con una amplia concordancia por parte de todos ellos.

•**Dependencia del sentido común:** Como se ha visto en la descripción de los POEAs, el análisis de los mismos ante una situación de emergencia depende mayormente del seguimiento de los diagramas de flujo provistos en la documentación más la utilización del conocimiento provisto por la experiencia adquirida.

Vemos que un sistema experto dedicado al seguimiento de las fallas de la planta no depende en forma alguna del uso del sentido común.

•**Grado de dificultad de las tareas:** El nivel de complejidad de la tarea a realizar por el sistema experto no es excesivamente alta. Esto está demostrado por el desarrollo de los POEAs. El hecho de haberse desarrollado la documentación que permite sintetizar la información correspondiente a la tarea del diagnóstico de fallas muestra que el nivel de complejidad no es suficientemente alto para eliminar la posibilidad de aplicar un sistema experto para su resolución.

Desarrollo justificado

El interés de N.A.S.A. por el proyecto nació de la necesidad de contar con una aplicación que sirva de ayuda a los operadores de la central para detectar fallas en el funcionamiento del reactor. En este punto, la colaboración principal a prestar por el sistema experto es la de detectar fallas con mayor rapidez y confiabilidad que un operador humano.

Si consideramos que actualmente los operadores utilizan documentación para ayudarse en la detección de eventos anormales, podemos notar que este hecho es contraproducente en casos de emergencia en la central, momentos en los cuales pueden surgir numerosas fallas simultáneas (algunas falsas) y provocar confusión en los operadores responsables llevando a una respuesta equivocada por parte de ellos.

A diferencia de las razones que se han enumerado anteriormente que pueden justificar un desarrollo, en este caso no nos encontramos con la necesidad de difundir conocimiento o suplantar a un experto sino con la de dotar de conocimiento a una herramienta que potencie las capacidades ya existentes en los usuarios con el fin de asegurar que sus tareas se realicen en forma confiable y segura.

Desarrollo apropiado

Para considerar el desarrollo del sistema experto como apropiado al problema planteado, debemos verificar principalmente la conveniencia de la solución al problema utilizando este tipo de sistemas.

Vemos por las características del problema, que el mismo se corresponde a uno cuya solución depende de la utilización de conocimiento especializado (POEAs, conocimiento de los expertos).

El requerimiento de contar con evidencias a favor y en contra de cada conclusión se ve reflejado en la existencia de distintos POEAs, para los cuales es necesario acumular certezas de modo de poder determinar la detección de alguno de ellos.

Otro punto a favor del vínculo de este problema con un problema de conocimiento se da al observar que los expertos han modelado la información relacionada con la detección de eventos anormales a través de una estructura similar a la de reglas.

La estructura de reglas de los POEAs hacen conveniente el enfoque de la solución basándose en un problema de conocimiento.

V. DESARROLLO TEÓRICO

Introducción

El sistema experto planteado en este trabajo tiene el propósito específico de servir de solución al problema de diagnóstico de eventos anormales en la CNE. Con este objetivo, el grupo de trabajo ha desarrollado un lenguaje lógico teórico que constituye el fundamento básico del lenguaje de reglas desarrollado para la aplicación. Dicho lenguaje se ha basado en otro lenguaje de primer orden que se encuentra definido en [LLO/87]. El mismo fue enriquecido sintáctica y semánticamente. Para esto último, se tomaron en cuenta los aportes de los trabajos detallados en [LEE/71] y [LEE/72].

En esta sección se explicará el desarrollo del lenguaje teórico denominado **LDSE** (**L**ógica **D**ifusa para **S**istemas **E**xpertos). Se enfocarán sus niveles sintácticos y semánticos llegando finalmente a su transformación a un lenguaje de programación lógica. El término LDSE se aplicará tanto al lenguaje mismo como a cada uno de sus componentes.

A continuación se detallan las características del lenguaje LDSE a través de seis puntos principales. En la primera parte, **definición del lenguaje LDSE**, se definirán la sintaxis y la semántica del lenguaje. En este punto se establecerán las definiciones de los componentes básicos incorporando un conjunto de aspectos específicos de la lógica difusa a las características comunes de un lenguaje de primer orden. Finalmente se llega a la noción de consecuencia lógica reflejando la consistencia de las definiciones semánticas previas.

En la segunda parte, **pasaje a forma clausal en el lenguaje LDSE**, se muestra una representación del lenguaje LDSE restringida al propósito específico de su implementación computacional. Finalmente se muestra el método de conversión de la representación teórica a su correspondiente restringida (computacional).

A continuación, en el punto siguiente, **método de inferencia en el lenguaje LDSE**, se explicará la restricción a la consecuencia lógica -llamada consecuencia sintáctica o derivación- aplicada a la versión restringida del lenguaje LDSE desarrollada en el punto anterior. También se prueba que ésta es correcta (preserva la noción de consecuencia lógica).

En **comparación entre consecuencia sintáctica LDSE y clásica**, se representará un programa lógico clausal LDSE utilizando un lenguaje clausal clásico de primer orden no difuso. Además del ejemplo brindado, se dará un método general de conversión que permitirá representar cualquier programa lógico LDSE en el lenguaje clausal específico utilizado en la implementación.

En el anteúltimo punto, **características del lenguaje en la aplicación**, se detallará qué características del lenguaje clausal teórico son mantenidas en el lenguaje propio de la aplicación, incluyendo sus aspectos prácticos.

Finalmente, en **comentarios al desarrollo**, se verán una serie de acotaciones adicionales al desarrollo presentado.

V.A. Definición del lenguaje LDSE

V.A.1. Sintaxis

Definición (1):

Un lenguaje £ se dice que es **LDSE** si está representado por la siguiente estructura:

$£ = < V, C, F, P, M, \{\neg, \wedge\}, \{\forall\}, \{(,), \{, \}, \sigma\} >$

donde cada componente representa:

① conjunto infinito numerable de variables $V = \{v_h\}_{h \in H}$
② conjunto de constantes $C = \{c_i\}_{i \in I}$
③ conjunto finito de símbolos de función $F = \{<f_j, n(j)>\}_{j \in J}$
④ conjunto finito de símbolos de predicado $P = \{<p_k, n(k)>\}_{k \in K}$
⑤ conjunto finito de símbolos modificadores de predicados $M = \{m_l\}_{l \in L}$
⑥ conjunto de conectivos lógicos
⑦ conjunto de símbolos cuantificadores
⑧ conjunto de símbolos de precedencia y auxiliares

La función 'n', que está ligada a los símbolos de función y de predicados, representa la aridad de cada uno de éstos y devuelve un número natural.

Definición (2):

Se define el conjunto de los **términos** (T) de la siguiente manera:

① $v_h \in V \Rightarrow v_h \in T$
② $c_i \in C \Rightarrow c_i \in T$
③ $t_1 \in T$ y ... y $t_{n(j)} \in T$ y $<f_j, n(j)> \in F \Rightarrow f_j(t_1, ..., t_{n(j)}) \in T$
④ T es cerrado por las construcciones anteriores.

Definición (3):

Se define el conjunto de los **predicados modificados** (PM) de la siguiente manera:

① $<p_k, n(k)> \in P \Rightarrow <p_k, n(k)> \in PM$
② $<p, a> \in PM$ y $m_l \in M \Rightarrow <m_l(p), a> \in PM$
③ PM es cerrado por las construcciones anteriores.

Definición (4):

Se define el conjunto de los **átomos** (A) -también llamados **fórmulas atómicas**- de la siguiente manera:

① $t_1 \in T$ y ... y $t_a \in T$ y $<p, a> \in PM \Rightarrow p(t_1, ..., t_a) \in A$
② A es cerrado por la construcción anterior.

Definición (5):

Se define el conjunto de las **fórmulas bien formadas** (W) de la siguiente manera:

① $F \in A \Rightarrow F \in W$
② $F \in A$ y $i \in [0,1] \Rightarrow \{F,i\} \in W$
③ $x \in [0,1] \Rightarrow \sigma_x \in W$
④ $F \in W \Rightarrow (\neg F) \in W$

⑤ $F \in W$ y $G \in W \Rightarrow (F \wedge G) \in W$
⑥ $F \in W$ y $x \in v \Rightarrow ((\forall x)\ F) \in W$
⑦ W es cerrado por las construcciones anteriores.

Las fórmulas bien formadas de la definición ② podrían expresarse en lenguaje natural como "'F' tiene importancia (o incidencia) contextual 'i'". Las fórmulas bien formadas de la definición ③ equivalen a fórmulas asociadas a predicados constantes (de aridad cero) con una veracidad asociada dada 'x'.

Definición (6):

Se definen los siguientes nuevos conectivos lógicos y cuantificadores en función de los anteriores:

① $(F \vee G) =_{def} (\neg((\neg F) \wedge (\neg G)))$
② $(F \rightarrow G) =_{def} ((\neg F) \vee G)$
③ $((\exists x)\ F) =_{def} (\neg((\forall x)\ (\neg F)))$

Para simplificar la escritura de fórmulas bien formadas se definen las siguientes reglas de supresión de paréntesis:

① Los operadores están ordenados de mayor a menor por su precedencia de la siguiente manera: $\neg$, $\wedge$, $\vee$, $\rightarrow$.
② Los operadores binarios '$\wedge$' y '$\vee$' son asociativos a izquierda, mientras que el operador binario '$\rightarrow$' lo es a derecha.
③ Los cuantificadores '$\forall$' y '$\exists$' tienen ambos tan alta precedencia como el operador '$\neg$' (esto apunta a la eliminación de los paréntesis exteriores a la fórmula).
④ La formación sucesiva de fórmulas precedidas por cuantificadores, permite eliminar los paréntesis exteriores de aquellas subfórmulas interiores.
⑤ Los paréntesis exteriores de cualquier fórmula que no es subfórmula de otra, pueden eliminarse.

V.A.2. Semántica

Definición (7):

Se define una **estructura adecuada** para el lenguaje £ como:

① Un conjunto D ($D \neq \varnothing$) llamado el universo de la estructura
② Un mapeo de cada $c_i \in C$ a un $c_i^D \in D$
③ Un mapeo de cada $<f_j, n(j)> \in F$ a una función $f_j^D : D^{n(j)} \rightarrow D$
④ Un mapeo de cada $<p_k, n(k)> \in P$ a una función $p_k^D : D^{n(k)} \rightarrow [0,1]$
⑤ Un mapeo de cada $m_l \in M$ a una función $m_l^D : D^{dom^*}$
(donde $D^{dom^*} =_{def} \cup_{i>0}((D^i \rightarrow [0,1]) \rightarrow (D^i \rightarrow [0,1]))$
⑥ Un mapeo de cada $<p, a> \in PM$ a una función $PM_p^D : D^a \rightarrow [0,1]$ definida así:
① $<p_k, n(k)> \in P \Rightarrow PM_{pk}^D = p_k^D$ (como fue dicho antes)
② $m_l \in M$ y $<p, a> \in PM \Rightarrow PM_{ml(p)}^D = m_l^D(PM_p^D)$

Aquí se nota que los predicados representan conjuntos difusos (o mejor dicho, relaciones difusas n-arias sobre el universo D, donde la función 'n' devuelve su aridad), ya que tienen una semántica similar. El concepto de relación difusa se encuentra explicado en el capítulo "**III. A.** Introducción a la lógica difusa".

Definición (8):

Se define una **interpretación de variables** como una función $\lambda : v \rightarrow D$. Una interpretación de variables es una asignación semántica de cada variable a un determinado valor del universo de la estructura.

Definición (9):

Se define una **interpretación de términos** en función de una interpretación λ de variables como una función $T_\lambda : T \rightarrow D$ definida así:

① $v_h \in V \Rightarrow T_\lambda(v_h) = \lambda(v_h)$
② $c_i \in C \Rightarrow T_\lambda(c_i) = c_i^D$
③ $t_1 \in T$ y ... y $t_{n(j)} \in T$ y $<f_j, n(j)> \in F \Rightarrow T_\lambda(f_j(t_1, ..., t_{n(j)})) = f_j^D(T_\lambda(t_1), ..., T_\lambda(t_{n(j)}))$

Una interpretación de términos es una asignación semántica de cada término a un determinado valor del universo de la estructura a partir de una interpretación de variables dada.

Definición (10):

Se define una **interpretación de fórmulas** en función de una interpretación T_λ de términos como una función $V_\lambda : W \rightarrow [0,1]$ definida así:

① $V_\lambda(p(t_1,\ldots,t_a)) = PM_p^D(T_\lambda(t_1), ..., T_\lambda(t_a))$
② $V_\lambda(\{F,i\}) = \text{norm}(i)(V_\lambda(F))$
③ $V_\lambda(\sigma_x) = x$
④ $V_\lambda((\neg F)) = \text{neg}(V_\lambda(F))$
⑤ $V_\lambda((F \wedge G)) = \otimes(V_\lambda(F), V_\lambda(G))$
⑥ $V_\lambda(((\forall x)\ F)) = \otimes^+_{w \in D}\{V_\lambda(F[^w/_x])\}$

donde 'neg' representa a la función:

$\text{neg} : [0,1] \rightarrow [0,1]$
$\text{neg}(x) =_{def} 1-x$

y donde "$F[^p/_x]$" representa la sustitución de todas las variables 'x' que aparecen libres en 'F' por el término 'p', usando una definición análoga de sustitución y de variables libres que a la de la lógica de primer orden.

Una interpretación de fórmulas es una asignación semántica de cada fórmula a un determinado valor de verdad en el intervalo [0,1] a partir de una interpretación de términos dada.

Funciones Auxiliares

La inclusión de la función 'norm' en esta definición tiene como objetivo "normalizar" los valores de verdad de los átomos de una fórmula con sus grados de incidencia a un valor con una incidencia "standard". El tipo de esta función es el siguiente:

$\text{norm} : [0,1] \rightarrow ([0,1] \rightarrow [0,1])$

El primer parámetro del dominio (utilizando abuso de notación) representa la incidencia del átomo en el contexto, y el segundo parámetro representa el valor de verdad del átomo. La función devuelve como resultado un valor de verdad normalizado del átomo matizado por esa importancia en su contexto. Una explicación más detallada de las incidencias puede encontrarse en el capítulo "**IV.** Estudio del problema y de su solución". No existen características visibles que deba poseer la función 'norm'.

La versión de la función 'norm' especificada en esta tesis tiene las siguientes propiedades:

① norm(i) es continua $\forall i$ (salvo en los extremos de i)
② norm(i) es creciente $\forall i$ $(\text{norm}(i)'(x) \geq 0\ \forall x)$
③ $\text{norm}(i)^{-1} = \text{norm}(1-i)\ \forall i$

Una función 'norm' que cumple con lo anterior puede ser:

$\text{norm}(i)(v) = v^{g(i)}$

donde g debería cumplir con lo siguiente:

① $g : [0,1] \rightarrow R_{\geq 0}$
② g es continua $\forall i$ (salvo en los extremos de i)

③ g es creciente $\forall i\ (g'(x) \geq 0\ \forall x)$
④ $g(0) = 0$
⑤ $g(0,5) = 1$
⑥ $\lim_{i \to 1^+} g(i) = +\infty$
⑦ $g(1-i) = 1/g(i)$ (si $g(i) \neq 0$)

Dos posibles ejemplos para la función 'g' pueden ser:

① $g(i) = 1/(1-i)-1$
② $g(i) = tg(i*\pi/2)$

Ver que por el punto ⑥ -analíticamente hablando-, la función 'norm' debe cumplir estas propiedades:

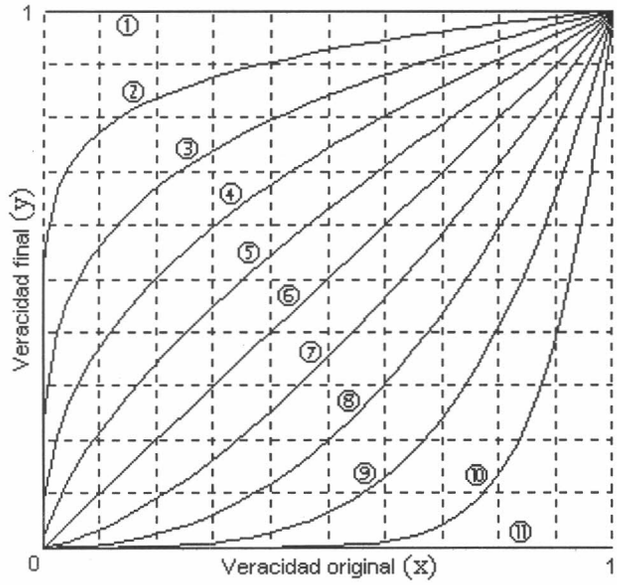
① $norm(1)(v) = 0$ (para $v \in [0,1)$)
② $norm(1)(1) = 1$

Finalmente, la función 'norm' que utilizaremos de aquí en adelante será la siguiente:

$$norm(i)(v) = \begin{cases} v^{1/(1-i)-1} & \text{si } i \neq 1 \text{ y } (v \neq 0 \text{ o } i \neq 0) \\ 1 & \text{si } i = 1 \text{ y } v = 1 \\ 0 & \text{si } i = 1 \text{ y } v \neq 1 \text{ o } v = 0 \text{ y } i = 0 \end{cases}$$

Vale la pena notar que esta definición de 'norm' cumple la propiedad "$norm(i)^{-1} = norm(1-i)\ \forall i$", como fue dicho anteriormente. Esto indica una "simetría" de la función entre los valores de 'i' interpretados como más "permisivos" (cada vez más a medida que se acercan a 0) y los valores de 'i' interpretados como más "estrictos" (cada vez más a medida que se acercan a 1) respecto a los valores de 'i' interpretados como más "indiferentes o realistas" (cada vez más a medida que se acercan a 0,5). Es deseable que la función norm sea creciente para cada valor de 'i' y también derivable.

En el gráfico a continuación se muestran varias funciones de incidencia, donde varían su valor de 'i' desde 0 hasta 1. Estas son:



Función de incidencia

$y = norm(i)(x)$

① $y = F(x) = norm(0)(x)$
② $y = F(x) = norm(0,1)(x)$
③ $y = F(x) = norm(0,2)(x)$
④ $y = F(x) = norm(0,3)(x)$
⑤ $y = F(x) = norm(0,4)(x)$
⑥ $y = F(x) = norm(0,5)(x)$
⑦ $y = F(x) = norm(0,6)(x)$
⑧ $y = F(x) = norm(0,7)(x)$
⑨ $y = F(x) = norm(0,8)(x)$
⑩ $y = F(x) = norm(0,9)(x)$
⑪ $y = F(x) = norm(1)(x)$

La función '$\otimes^+$', utilizada en la definición de interpretación de fórmulas, está definida de la siguiente manera:

$\otimes^+ : \wp([0,1]) \rightarrow [0,1]$ (de aquí en adelante se utilizará a $\wp$ como la función de partes finitas)
$\otimes^+(\{x_1, x_2, x_3, ..., x_n\}) =_{def} \otimes(... \otimes(\otimes(x_1, x_2), x_3), ..., x_n)$

Se espera que la función '⊗' cumpla las siguientes propiedades:

❶ $\otimes : [0,1] \times [0,1] \rightarrow [0,1]$
❷ $\otimes(x, y) \leq \text{mín}(x, y)$ $\quad \forall x,y \in [0,1]$
❸ $\otimes(x, x) = x$ $\quad \forall x \in [0,1]$ (Idempotencia)
❹ $\otimes(x, \otimes(y, z)) = \otimes(\otimes(x, y), z)$ $\quad \forall x,y,z \in [0,1]$ (Asociatividad)
❺ $\otimes(x, y) = \otimes(y, x)$ $\quad \forall x,y \in [0,1]$ (Conmutatividad)

Las interpretaciones de fórmulas construídas a partir de los nuevos conectivos lógicos y cuantificadores dan como resultado lo siguiente:

$I((F \vee G)) =_{def} I((\neg((\neg F) \wedge (\neg G)))) = neg(\otimes(neg(I(F)), neg(I(G))))$
$I((F \rightarrow G)) =_{def} I(((\neg F) \vee G))$
$I(((\exists x)\ F)) =_{def} I((\neg((\forall x)\ (\neg F)))) = neg(\otimes^{+}_{w \in D}\{neg(I(F[^{w}/_{x}]))\})$

Un ejemplo para la función '⊗' que cumple las cinco propiedades puede ser:

⊗ = mín (mínimo entre dos números reales)

Esta aplicación nos da los siguientes resultados:

$I((F \wedge G)) = \text{mín}(I(F), I(G))$
$I(((\forall x)\ F)) = \text{mín}^{+}_{w \in D}\{I(F[^{w}/_{x}])\}$
$I((F \vee G)) =_{def} I((\neg((\neg F) \wedge (\neg G)))) = neg(\text{mín}(neg(I(F)), neg(I(G)))) =$
$= \text{máx}(I(F), I(G))$
$I((F \rightarrow G)) =_{def} I(((\neg F) \vee G)) =_{def} \text{máx}(neg(I(F)), I(G))$
$I(((\exists x)\ F)) =_{def} I((\neg((\forall x)\ (\neg F)))) = neg(\text{mín}^{+}_{w \in D}\{neg(I(F[^{w}/_{x}]))\}) =$
$= \text{máx}^{+}_{w \in D}\{I(F[^{w}/_{x}])\}$

Otro ejemplo posible para '⊗' que suele usarse con fines probabilísticos es:

⊗ = * (producto de dos números reales)

Sin embargo esta función no cumple la idempotencia (punto ❸). Esta definición de '⊗' provoca que no se cumplan propiedades necesarias definidas posteriormente. Es por eso que para el presente lenguaje usaremos "⊗ = mín". Otras definiciones pueden encontrarse, por ejemplo, en [KLI/95].

Definición (11):

Dos fórmulas bien formadas F y G se dice que son **equivalentes** (o bien $F \equiv G$) sii se cumple:

$I(F) = I(G)$ $\forall I$ interpretación de fórmulas.

Se cumplen las siguientes propiedades acerca de la equivalencia de fórmulas bien formadas:

① $F \equiv (\neg(\neg F))$
② $(\neg(F \vee G)) \equiv ((\neg F) \wedge (\neg G))$
③ $(\neg(F \wedge G)) \equiv ((\neg F) \vee (\neg G))$
④ $(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$
⑤ $(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$

V.A.3. Concecuencia lógica

Sea $\alpha = 0{,}5$. En lo sucesivo, se hará referencia a un par genérico fórmula-valor (F,x), cuyo sentido será dar una semántica clásica a la interpretación del par fórmula-valor al compararlo con relaciones no difusas. Dicha semántica clásica se logra a través de una comparación entre la interpretación difusa de la fórmula y el valor asociado a la fórmula.

Definición (12):

Una interpretación I de fórmulas se dice que es **modelo** de un par fórmula-valor (F,x) (o que **satisface** el par (F,x)) si y sólo si $I(F) \geq x$ (considerando $x \geq \alpha$).

Definición (13):

Una interpretación I de fórmulas se dice que **falsifica** un par fórmula-valor (F,x) si y sólo si $I(F) \leq neg(x)$ (considerando $x \geq \alpha$).

Definición (14):

Un par fórmula-valor (F,x) se dice que es **tautología** si y sólo si I satisface (F,x) para toda I interpretación de fórmulas.

Definición (15):

Un par fórmula-valor (F,x) se dice que es **contradicción** si y sólo si I falsifica (F,x) para toda I interpretación de fórmulas. Dicho de otra forma, el par (F,x) no posee modelos.

Definición (16):

Un conjunto de pares fórmula-valor $\{(F_i,v_i)\}_{i \in N}$ se dice que es **válido** si y sólo si para toda interpretación I de fórmulas y todo índice j, I satisface (F_j,v_j).

Definición (17):

Un conjunto de pares fórmula-valor $\{(F_i,v_i)\}_{i \in N}$ se dice que es **insatisfactible** si y sólo si para toda interpretación I de fórmulas existe un indice j tal que I falsifica (F_j,v_j).

Acorde a las definiciones anteriores, se puede notar que existen interpretaciones 'I' de fórmulas que satisfacen un par (F,α) y a su vez, también lo falsifican. Esto ocurre cuando $I(F)=\alpha$. Si $x>\alpha$ también se puede notar que existen interpretaciones 'I' de fórmulas que ni satisfacen ni falsifican pares (F,x). Esto ocurre cuando $neg(x) \leq I(F) \leq x$.

Definición (18):

Sea $\Gamma = \wp(W \times [0,1])$ (donde W es el conjunto de las fórmulas bien formadas)
Se define la **consecuencia lógica** como una relación clásica $\models_{LDSE} \subseteq \Gamma \times (W \times [0,1])$ de la siguiente forma:

$$\models_{LDSE} =_{def} \{ (\Gamma,(F,x)) / (I(w) \geq v \geq \alpha \quad \forall (w,v) \in \Gamma) \Rightarrow I(F) \geq x \geq \alpha \quad \forall I \text{ interpretación de fórmulas } \}$$

En lugar de usar $(\Gamma,(F,x)) \in \models_{LDSE}$, usaremos $\Gamma \models_{LDSE} (F,x)$. En todo par $(X,Y) \in (W \times [0,1])$ que aparece en la definición de consecuencia lógica , el valor 'X' representa una fórmula, y el valor 'Y' representa un valor de verdad hipotético mínimo asociado a la fórmula 'X' que es independiente de las interpretaciones involucradas en la consecuencia lógica, asumiendo que esta última se cumple.

Esta definición de consecuencia lógica, como otras definiciones semánticas anteriores, están basadas en la definición de R. C. T. Lee (Ver [LEE/72]). En esa definición, Lee clasificó a todos los valores de verdad (los elementos del intervalo [0,1]) en dos grandes grupos: los valores verdaderos (cuyo valor es mayor o igual a 0,5) y los valores falsos (cuyo valor es menor o igual a 0,5), manteniendo todas las propiedades de la lógica clásica. Este valor límite (también llamado umbral) está aquí representado con la constante 'α'.

Lee sólo redefinió la semántica de los operadores '$\wedge$', '$\neg$' y '$\forall$', como se definió en **V.A.2**, y los valores "verdadero" y "falso", los cuales no son ya considerados como valores de verdad concretos sino como un conjunto de valores. Tampoco tuvo que cambiar las definiciones de modelo y de consecuencia lógica. En su definición de consecuencia lógica, la fórmula que es deducida debe considerarse verdadera bajo aquellas interpretaciones que hacen verdaderas a todas las fórmulas del conjunto.

La definición de consecuencia lógica definida en este trabajo surge como una extensión de la de Lee. En esta definición nos detenemos en las interpretaciones que no sólo hacen verdaderas a las fórmulas sino que además el valor de verdad de la fórmula bajo esa interpretación es mayor o igual a un número asociado a ella, que debe ser verdadero (mayor o igual a α). Es inútil usar este valor como extensión a la definición de consecuencia lógica en la lógica clásica, ya que existe un único valor de verdad "verdadero", y éste debería ser el único posible.

Es importante notar que las relaciones "modelo", "falsifica", "tautología", "contradicción", "válido", "insatisfactible", y "consecuencia lógica" están definidas a través de relaciones bivaluadas. Esto hace que las relaciones antes definidas también sean bivaluadas.

Se cumplen las siguientes propiedades acerca de la consecuencia lógica:

① $\{(P,x)\} \models_{LDSE} (P,y)$ (con $x \geq y \geq \alpha$) (Amplitud)

② $\{(P \wedge Q,x)\} \models_{LDSE} (R,y) \Leftrightarrow$
$\{(P,x), (Q,x)\} \models_{LDSE} (R,y)$ (con $x, y \geq \alpha$) (Disgregación)

③ $\Gamma \models_{LDSE} R \Rightarrow \Gamma \cup S \models_{LDSE} R$ (Debilitamiento de precondición/monotonía)

④ $\Gamma \models_{LDSE} (P,x) \Rightarrow \Gamma \models_{LDSE} (P \vee Q,x)$ (Fortalecimiento de postcondición)

⑤ $\{P^*_i\}_{i \in M} \models Q^* \Rightarrow \{(P_i,1)\}_{i \in M} \models_{LDSE} (Q,1)$ (Extensión respecto a la lógica clásica)
(con X^* fórmulas de la lógica clásica y X fórmulas LDSE "equivalentes" a X^*)

⑥ $\Gamma \models_{LDSE} (P,x) \Leftrightarrow \Gamma \cup \{(\neg P,x)\}$ es insatisfactible (Reducción al absurdo)

Teorema (1):

Dado un conjunto de fórmulas bien formadas Γ, una fórmula bien formada F y una veracidad x es indecidible determinar que $\Gamma \models_{LDSE} (F,x)$.

Este teorema está basado en el teorema de la incompletitud de Gödel usado para expresar la indecidibilidad de la lógica clásica de primer orden.

Debido a esta gran limitación, nos vemos forzados a reformular la sintaxis de las fórmulas de una forma más restringida que el conjunto W. El objetivo es poder aplicar de una forma sintáctica y computacional el criterio de consecuencia lógica.

V.B. Pasaje a forma clausal en el lenguaje LDSE

V.B.1. Fórmulas restringidas

Definición (19):

Se define el conjunto de las **cláusulas** (S) de la siguiente manera:

① $a_1 \in A$ y ... y $a_m \in A$ y $p_1 \in A$ y ... y $p_k \in A$ y $i_1 \in [0,1]$ y ... y $i_k \in [0,1]$ y $c \in [\alpha,1]$ y $m,k \geq 0 \Rightarrow$
$<<a_1, ..., a_m :- \{p_1,i_1\}, ..., \{p_k,i_k\}>, c> \in \mathbf{S}$
② S es cerrado por la construcción anterior.

Si para una cláusula se cumple que m>0, k=0 y c=1 entonces esa cláusula es llamada **hecho**; si cumple que m>0 y k>0, la cláusula es llamada **regla**. Vale la pena notar que estos dos "subconjuntos" no conforman una partición de S.

Las cláusulas son versiones simplificadas (en función del valor de verdad o semántica) de las fórmulas que son implicaciones. A cada uno de los 'p_j' se los suele llamar "precondición j-ésima" de la cláusula; a cada uno de los 'i_j' se los llama "incidencia" o "importancia" de la "precondición j-ésima asociada p_j" en la implicación, y a cada uno de los 'a_k' se los conoce como "postcondición" o "conclusión" k-ésima de la cláusula, mientras que el valor 'c' es considerado como confianza de la cláusula. Vale la pena notar que las incidencias usadas en las cláusulas son tomadas como valores propios característicos de la cláusula misma, y no como en las fórmulas, donde cada una es considerada como propia de la fórmula asociada a ella.

A los átomos que son conclusiones de la regla también se los llama literales positivos, mientras que a los átomos que son precondiciones se los llama literales negativos.

Definición (20):

Se define la **conjunción clausal LDSE**,que define el conjunto de los **programas lógicos LDSE** o conjunto PL, de la siguiente manera:

① $C \in \mathbf{S} \Rightarrow C \in PL$
② $C_1 \in PI$ y $C_2 \in PI \Rightarrow (C_1 \wedge C_2) \in PL$
③ PL es cerrado por las construcciones anteriores.

También se suele llamar "programa lógico LDSE" a un conjunto de cláusulas.

Desde el punto de vista semántico, el conjunto S de las cláusulas es un subconjunto de W. Esta afirmación se aclara en la definición de semántica de una cláusula (definida a continuación), donde la veracidad de una cláusula será equivalente a la semántica de una fórmula (que depende de la cláusula) bajo esa misma interpretación. Este conjunto de cláusulas es aquel al que se hacía referencia en el final del punto anterior como restricción del conjunto de fórmulas, para poder realizar la consecuencia sintáctica.

Definición (21):

Se define una **interpretación de cláusulas** (en función de una interpretación T_λ de términos) como una función $V'_\lambda : \mathbf{S} \rightarrow [0,1]$ definida así:

$V'_\lambda(<<a_1, ..., a_m :- \{p_1,i_1\}, ..., \{p_k,i_k\}>, c>) =_{def}$
$V_\lambda((\forall v_1)(\forall v_2) ... (\forall v_n)(\{p_1,i_1\} \wedge ... \wedge \{p_k,i_k\} \wedge \sigma_c \rightarrow (a_1 \vee ... \vee a_m))) =$
$V_\lambda((\forall v_1)(\forall v_2) ... (\forall v_n)(a_1 \vee ... \vee a_m \vee \neg\{p_1,i_1\} \vee ... \vee \neg\{p_k,i_k\} \vee \neg\sigma_c))$
(donde $\{v_1, v_2, ..., v_n\} = FV(a_1) \cup ... \cup FV(a_m) \cup FV(p_1) \cup ... \cup FV(p_k)$)

donde 'FV' representa la función "variables libres", que dada una fórmula, devuelve el conjunto de variables que aparecen libres en ella.

De la definición anterior se puede verificar que las cláusulas son sentencias (no poseen variables libres) -en realidad es porque su fórmula equivalente es sentencia-. También se puede verificar que las confianzas representarían una precondición adicional de la cláusula cuya veracidad es siempre constante.

La definición de semántica asociada al operador '$\wedge$' de conjunciones clausales es análoga a la del operador '$\wedge$' de fórmulas.

Sabiendo que ahora vamos a trabajar sobre cláusulas y no sobre fórmulas, deberá ser necesario encontrar un método para convertir fórmulas en una conjunción de cláusulas, para que así se pueda aplicar su consecuencia sintáctica. Sin embargo, éste método no garantizará para cualquier fórmula que la conjunción de cláusulas-resultado sea semánticamente equivalente a ella. Este método está basado en transformaciones sucesivas de la fórmula original (basándose en propiedades que garantizan equivalencia semántica en la mayoría de los casos), hasta llegar a una fórmula que pueda ser convertida en una o más cláusulas, según la definición anterior. Esto es similar al método de pasaje de fórmulas a cláusulas en la lógica clásica.

V.B.2. Conversión fórmulas-cláusulas

Se utilizará el siguiente método de pasaje a forma clausal. Éste transforma una fórmula que es sentencia a través de pasos sucesivos a otra fórmula que es lógicamente equivalente a alguna conjunción de cláusulas. Para ponerlo en práctica, se deberá aplicar las siguientes reglas de transformación a todas las subfórmulas que cumplan las condiciones necesarias, en el orden indicado:

Nota: La subfórmula a transformar debe unificar con la subfórmula de la izquierda del signo $\equiv$ de la regla, tranformándose en la subfórmula de la derecha del mismo signo.

① $((\Diamond x)\ F) \equiv ((\Diamond y)\ [^{y}/_{x}]F)$ $\quad (x \in \cup_{w \in Ctx(F)} Var(w),\ y \notin Var(F),\ y \notin \cup_{w \in Ctx(F)} Var(w),\ \Diamond = \{\forall, \exists\})$

(Renombrar por variables nuevas a las variables ligadas de la fórmula. Las variables a renombrar serán aquellas con igual nombre que otras variables de distintos alcances de cuantificadores. Esto sólo garantiza no confundir esas variables ligadas con otras que estén al alcance de otro cuantificador). Aquí Ctx(F) devuelve un conjunto de fórmulas que representan el contexto de la fórmula F dentro de la fórmula principal.

② $(P \rightarrow Q) \equiv ((\neg P) \vee Q)$

(Eliminar el operador '$\rightarrow$' de implicación. Éste no aparece en la fórmula final la cual es lógicamente equivalente a una conjunción de cláusulas).

③ $(\neg(P \wedge Q)) \equiv ((\neg P) \vee (\neg Q))$
$(\neg(P \vee Q)) \equiv ((\neg P) \wedge (\neg Q))$
$(\neg((\forall x)\ P)) \equiv ((\exists x)\ (\neg P))$
$(\neg((\exists x)\ P)) \equiv ((\forall x)\ (\neg P))$

(Desplazar las negaciones '$\neg$' hacia los átomos. Finalmente las negaciones deben aplicarse solamente a átomos con importancia contextual).

④ $(\neg(\neg P)) \equiv P$

(Eliminar las negaciones '$\neg$' innecesarias).

⑤ $(P \vee (Q \wedge R)) \equiv ((P \vee Q) \wedge (P \vee R))$

(Desplazar las diyunciones '$\vee$' hacia los átomos y desplazar las conjunciones '$\wedge$' más "lejos" de los átomos. Con esto, la estructura de la fórmula se asemeja -exceptuando a los cuantificadores- a una conjunción de disyunciones, que es el formato base de una conjunción de cláusulas).

⑥ $((\forall x)\ (P \wedge Q)) \equiv (((\forall x)\ P) \wedge ((\forall x)\ Q))$

(Desplazar los cuantificadores universales '$\forall$' que ocurren fuera de las cláusulas, hacia ellas. Con esto nos aseguramos que las cláusulas sólo van a estar conectadas con operadores de conjunción '$\wedge$', con la excepción de aquellos cuantificadores '$\exists$' con un alcance mayor a una subfórmula-cláusula y de los operadores '$\forall$' en cuyo cuerpo aparece alguna fórmula con un cuantificador $\exists$).

⑦ $(P \vee ((\forall x)\ Q)) \equiv ((\forall x)\ (P \vee Q))$ $\quad$ (si $x \notin FV(P)$)
$(((\forall x)\ P) \vee Q) \equiv ((\forall x)\ (P \vee Q))$ $\quad$ (si $x \notin FV(Q)$)

(Desplazar los cuantificadores universales '$\forall$' que ocurren dentro de las cláusulas, hacia la subfórmula más externa de esa cláusula. Con esto nos aseguramos que las fórmulas principales asociadas a las cláusulas se escriban con cuantificadores universales '$\forall$'. Las condiciones necesarias para poder aplicar esta regla se van a cumplir, ya que en el punto ① se renombraron variables ligadas para evitar conflictos).

⑧ $((\forall x)\ P) \equiv P \qquad (\text{si } x \notin FV(P))$

(Eliminar los cuantificadores universales ∀ innecesarios).

⑨ $((\forall v1)\ (\forall v2)\ ...\ (\forall vn)\ (\exists x)\ P) \equiv ((\forall v1)\ (\forall v2)\ ...\ (\forall vn)\ [^{f(v1, v2, ..., vn)}/_{x}]P)$

(Eliminar los cuantificadores existenciales '∃' dentro de cada cláusula, ya que no deben utilizarse. Encontramos el problema de que esta eliminación no preserva la semántica porque cuando se expresa la fórmula $((\exists x)\ P)$ se está diciendo que para algún elemento del dominio 'x' vale 'P'. No se aclara, sin embargo, cuáles ni cuántos elementos del dominio hacen valer 'P'. Al quitar el cuantificador '∃' se está expresando que hay un elemento del dominio que hace valer 'P', y que éste es un término con un símbolo de función "nuevo" f en función de otros términos que son todas las variables ligadas por los cuantificadores universales '∀' anteriores. Al eliminar los cuantificadores '∃', podrían quedar liberados algunos cuantificadores '∀' cuyo alcance abarcaba más de una subfórmula-cláusula, en los que anteriormente no podía aplicarse la regla ⑥. Si este caso ocurre, luego de la eliminación de los cuantificadores existenciales, la regla ⑥ debe ser aplicada).

⑩ $(\neg\sigma_n) \equiv \sigma_{neg(n)}$

$P \equiv (P \vee \sigma_0)$

$(\sigma_n \vee \sigma_m) \equiv \sigma_{neg(\otimes(neg(n),\ neg(m)))} = \sigma_{neg(mín(neg(n),\ neg(m)))} = \sigma_{máx(n,\ m)}$

$\sigma_n \equiv (\neg\sigma_{neg(n)})$

(Manipular el valor de la confianza en cada cláusula. Estas transformaciones quitan negaciones de las confianzas, agrega el valor de confianza en aquellas cláusulas que no los tienen, quita valores de confianza en aquellas cláusulas que tienen más de una, y niegan el valor de la confianza, para poder llegar al valor final de la confianza que debe estar negado).

V.B.3. Restricciones

Es lógico ver que en la conjunción clausal final no deben existir átomos con incidencia ni tampoco átomos negados sin incidencia en ninguna de las cláusulas. En el caso en que se presente esta situación se dirá que no se podrá pasar la fórmula en cuestión a forma clausal.

Definición (22):

Una cláusula LDSE se dice que es **de Horn** si tiene un único literal positivo (m=1). Una conjunción clausal LDSE -y un programa lógico LDSE- se dice también que es **de Horn** si se expresa a través de cláusulas LDSE de Horn.

Es importante que el programa lógico LDSE final (obtenido a partir de la aplicación de las reglas del pasaje a forma clausal) sea de Horn y que todas sus cláusulas sean hechos o reglas. De no ser así no se podrá aplicar sobre el programa el método de derivación para obtener su consecuencia sintáctica. Igualmente, la veracidad ligada a las cláusulas que son reglas deberá ser 1.

Existe otra restricción ligada a la estructura adecuada para el lenguaje. La semántica de todo modificador de predicados $m \in M$ debe poseer las siguientes restricciones adicionales:

① $\forall m\ \ I(p_1(t_{11}, ..., t_{1n})) = I(p_2(t_{21}, ..., t_{2k})) \Rightarrow I(m(p_1)(t_{11}, ..., t_{1n})) = I(m(p_2)(t_{21}, ..., t_{2k}))$

② $\forall m\ \ \exists m^{-1} \in M \,/\, I(m^{-1}(m(p))(t_1, ..., t_m)) = I(m(m^{-1}(p))(t_1, ..., t_m)) = I(p(t_1, ..., t_m))$

③ $\forall m\ \ I(p_1(t_{11}, ..., t_{1n})) > I(p_2(t_{21}, ..., t_{2k})) \Rightarrow I(m(p_1)(t_{11}, ..., t_{1n})) > I(m(p_2)(t_{21}, ..., t_{2k}))$

La restricción ① sugiere que para cualquier modificador de predicados $m \in M$ existirá también otra función semántica simplificada m' declarada y definida a través de la siguiente propiedad:

$m' : [0,1] \rightarrow [0,1]$

$m'(I(p(t_1, ..., t_n))) =_{def} I(m(p)(t_1, ..., t_n)) \qquad \forall I$ interpretación de fórmulas

La restricción ② indica que estas funciones simplificadas deben ser biyectivas, mientras que la restricción ③ indica que deben ser monótonas estrictamente crecientes. También por estas restricciones se ve que vale que $m'(0) = 0$ y $m'(1) = 1$.

Las propiedades más importantes derivadas de esta restricción son las siguientes (recordar que α fue definido en **V.A.3.**):

$\{(p(t_1, ..., t_n),x)\} \models_{LDSE} (m(p)(t_1, ..., t_n),m'(x))$ (Aumentación)
(con $p \in PM$, $x \geq \alpha$ y $m'(x) \geq \alpha$)
$\{(m(p)(t_1, ..., t_n),x)\} \models_{LDSE} (p(t_1, ..., t_n),m^{-1}'(x))$ (Disminución)
(con $p \in PM$, $x \geq \alpha$ y $m^{-1}'(x) \geq \alpha$)

donde se puede notar que la disminución es un caso particular de las aumentación. También se puede notar que se cumple que $m^{-1}'(x) = m'^{-1}(x)$.

Si la estructura adecuada del lenguaje no define semánticamente a los modificadores con esta restricción, tampoco se podrá realizar la consecuencia sintáctica.

Hasta ahora sólo vimos cómo convertir una fórmula de la base de conocimientos del lenguaje LDSE en una o más cláusulas (conectadas con una conjunción) de la base, pero no hablamos sobre la transformación de los valores asociados a las fórmulas. Para ello se utiliza la propiedad de disgregación (mencionada en el punto **V.A.3.**) que permite "propagar" este valor a cada una de las subfórmulas separadas por conjunciones. Cabe recordar que estas subfórmulas son semánticamente equivalentes a cláusulas, donde finalmente todas tendrán su propio valor asociado, aunque éstos van a ser todos iguales entre sí.

V.C. Método de inferencia en el lenguaje LDSE

En esta sección explicaremos el método de inferencia, para ver si para un programa lógico LDSE 'Γ' dado (en formato de conjunto de cláusulas LDSE de Horn) junto a las veracidades de sus átomos y un átomo 'F' con veracidad 'x', se cumple que $\Gamma \vdash_{LDSE} (F,x)$.

El operador de consecuencia sintáctica $\vdash_{LDSE}$ está representado como una relación clásica:

$$\vdash_{LDSE} \subseteq \mathcal{P}(\mathbf{s} \times [0,1]) \times (\mathbf{A} \times [0,1]).$$

De su representación se puede notar también que el dominio de la consecuencia sintáctica es un caso particular del dominio de la consecuencia lógica, ya que **s** es un caso particular de W, en los primeros parámetros de ambos, y A es un caso particular de W, en los segundos parámetros.

Una definición general de $\vdash_{LDSE}$ que abarca a fórmulas (más general que cláusulas) es la siguiente:

$$\{(x_1,v_1), ..., (x_n,v_n), (y_1 \wedge ... \wedge y_n \rightarrow y,1)\} \vdash_{LDSE} (z, \text{Mín}(\{v_1, ..., v_n\}))$$
$$(\text{si } v_j \geq \alpha \ \ \forall 1 \leq j \leq n \ \ y$$
$$\exists \phi_1, ..., \phi_n, \phi \ / \ x_1\phi_1 = y_1\phi_1 \wedge ... \wedge x_n(\phi_n \circ ... \circ \phi_1) = y_n(\phi_n \circ ... \circ \phi_1) \wedge$$
$$y(\phi \circ \phi_n \circ ... \circ \phi_1) = z(\phi \circ \phi_n \circ ... \circ \phi_1))$$

donde los 'ϕ_i' y 'ϕ' representan sustituciones, los '$x\phi$' representan la aplicación de la sustitución 'ϕ' al término 'x' lo que devuelve un nuevo término, y 'o' representa el operador de composición de sustituciones. Todas estas definiciones y operaciones son análogas a las de la lógica clásica. Las veracidades a las que se le aplica el mínimo en la fórmula deducida corresponden a las veracidades de cada una de las precondiciones de la regla, considerando que valen los otros pares fórmula-valor en el conjunto.

Es conveniente aclarar que esta definición de consecuencia sintáctica refleja una idea de "modus ponens generalizado" - como ya fue mencionado en el punto **III.A.2.** -, ya que si el dominio de v_j se restringe a {0,1} se tiene el caso clásico. En el capítulo "**III.A.** Introducción a la lógica difusa", se da una explicación más detallada de los operadores posibles a la hora de obtener el valor de verdad de la cláusula inferida.

Una definición más particular de $\vdash_{LDSE}$ que abarca a cláusulas en lugar de fórmulas, que es la que vamos a usar, es la siguiente:

$$\{(<<x_1 :->,1>,v_1), ..., (<<x_n :->,1>,v_n), (<<y :- \{y_1,i_1\}, ..., \{y_n,i_n\}>,c>,1)\}$$
$$\vdash_{LDSE} (<<z :->,1>, \text{Mín}(\{\text{norm}(i_1)(v_1), ..., \text{norm}(i_n)(v_n), c\}))$$
$$(\text{si norm}(i_j)(v_j) \geq \alpha \ \ \forall 1 \leq j \leq n \ y \ c \geq \alpha \ \ y$$
$$\exists \phi_1, ..., \phi_n, \phi \ / \ x_1\phi_1 = y_1\phi_1 \wedge ... \wedge x_n(\phi_n \circ ... \circ \phi_1) = y_n(\phi_n \circ ... \circ \phi_1)$$
$$y(\phi \circ \phi_n \circ ... \circ \phi_1) = z(\phi \circ \phi_n \circ ... \circ \phi_1))$$

Se puede ver que las cláusulas-hecho que aparecen son semánticamente equivalentes a fórmulas que son átomos. Esto ocurre por lo siguiente:

$$V'_\lambda(<<x :->,1>) = V_\lambda(x \vee \neg\sigma_1) = V_\lambda(x \vee \sigma_0) = V_\lambda(x)$$

Ahora la veracidad de la "precondición j-ésima" de la regla no será 'v_j', sino que es "norm(i_j)(v_j)". Vale la pena notar también que la confianza de una regla es el único valor no unificable con otros pares cláusula hecho-valor que forma parte de la precondición de ésta, y que la existencia de incidencias en las precondiciones de la regla no inciden en el proceso de unificación de las precondiciones de ésta.

Teorema (2):

$\vdash_{LDSE}$ es correcto ($\Gamma \vdash_{LDSE} R \Rightarrow \Gamma \models_{LDSE} R$).

Demostración: (de la definición general de $\vdash_{LDSE}$ que abarca a fórmulas)

Supongamos que I es una interpretación de fórmulas que cumple lo siguiente: ($\alpha>0$)

$I(X_1)\geq v_1$ (considerando $v_1\geq\alpha$) y
...
$I(X_n)\geq v_n$ (considerando $v_n\geq\alpha$) y
$I(X_1\wedge...\wedge X_n\rightarrow Y)\geq 1$

donde las fórmulas X_1, ..., X_n e Y representan lo siguiente:

$X_1 = x_1\phi_1 = y_1\phi_1$
...
$X_n = x_n(\phi_n \circ ... \circ \phi_1) = y_n(\phi_n \circ ... \circ \phi_1)$
$Y = y(\phi \circ \phi_n \circ ... \circ \phi_1) = z(\phi \circ \phi_n \circ ... \circ \phi_1)$

Se quiere ver que $I(Y) \geq Mín(\{I(X_i)\}) \geq \alpha$.

Por definición de interpretación de fórmulas vale que $I(X_1\wedge...\wedge X_n\rightarrow Y)=I(\neg(X_1\wedge...\wedge X_n)\vee Y)=$ $máx(I(\neg(X_1\wedge...\wedge X_n)), I(Y))=máx(neg(I(X_1\wedge...\wedge X_n)), I(Y))=máx(neg(Mín(\{I(X_i)\})), I(Y))$, y por la hipótesis anterior se deberá cumplir que $máx(neg(Mín(\{I(X_i)\})), I(Y))\geq 1$. Al ser 1 el supremo de valores de interpretaciones, entonces deberá valer que $máx(neg(Mín(\{I(X_i)\})), I(Y))=1$, lo que es equivalente a decir que $neg(Mín(\{I(X_i)\}))=1$ o $I(Y)=1$ y que es equivalente a $Mín(\{I(X_i)\})=0$ o $I(Y)=1$.

Al ser por hipótesis $I(X_1)\geq v_1$ y ... y $I(X_n)\geq v_n$ entonces valdrá que $1\geq Mín(\{I(X_i)\})\geq Mín(\{v_i\})\geq\alpha>0$. Por lo tanto, nunca valdrá que $Mín(\{I(X_i)\})=0$, y por lo tanto deberá valer que $I(Y)=1\geq Mín(\{I(X_i)\}) \geq \alpha$ como quería demostrarse. ■

La demostración anterior sugiere que cualquier valor de verdad entre α y 1 ligado a la fórmula deducida hace que la derivación sea correcta (siempre que $\alpha>0$). Sin embargo existe una motivación para que se elija que este valor sea el mínimo de las veracidades de los antecedentes de la regla. Esto se debe a que se considera que el valor de veracidad del consecuente es el valor de veracidad calculado del antecedente, que es una conjunción de átomos "verdaderos". Esto hace que se calcule el mínimo de las veracidades de esos átomos para obtener ese valor. Para que se pueda asegurar que este valor sea mayor o igual a α se tomó la función Mín (mínimo) en lugar de $\otimes^+$, ya que esta última función puede llegar a devolver un valor menor a α.

La definición anterior no es la única usada como consecuencia sintáctica. Las propiedades de aumentación y disminución (mencionadas al final del punto anterior) también son utilizadas como consecuencia sintáctica. O sea que también vale lo siguiente:

$\{(p(t_1, ..., t_n),x)\} \vdash_{LDSE} (m(p)(t_1, ..., t_n),m'(x))$ (con $p\in PM$, $x\geq\alpha$ y $m'(x)\geq\alpha$)
$\{(m(p)(t_1, ..., t_n),x)\} \vdash_{LDSE} (p(t_1, ..., t_n),m^{-1}'(x))$ (con $p\in PM$, $x\geq\alpha$ y $m^{-1}'(x)\geq\alpha$)

La correctitud de estas dos últimas consecuencias sintácticas sale de verificar que cumplen la consecuencia lógica. Más adelante hablaremos sobre la completitud de la consecuencia sintáctica $\vdash_{LDSE}$.

V.D. Comparación entre consecuencia sintáctica LDSE y clásica

V.D.1. Conversión programa lógico LDSE-programa lógico clásico

Ahora que tenemos definido el criterio de consecuencia sintáctica sobre el lenguaje LDSE clausal veremos de qué forma se lo puede comparar (y hasta representar) con la consecuencia sintáctica clausal clásica.

Un lenguaje LDSE con los siguientes componentes:

v_{LDSE}, c_{LDSE}, F_{LDSE}, P_{LDSE}, M_{LDSE}

tendrá las siguientes características de un lenguaje clausal clásico de primer orden:

$v = v_{LDSE} \cup Var$ (donde Var contiene a todas las variables nuevas propias de la conversión)
$c = c_{LDSE} \cup R \cup \{p / <p, a> \in P_{LDSE}\} \cup M_{LDSE} \cup \{listvac\}$
$F = F_{LDSE} \cup \{<conslist, 2>\}$
$P = \{<umbral, 1>, <min2, 3>, <norm, 3>, <g, 2>, <cortarUlt, 3>, <aplica, 3>,$
$<aplicaInv, 3>, <quitaComunes, 4>, <desMod, 4>, <demPrecInc, 5>, <demPrecs, 3>,$
$<atomo, 4>, <atomoReal, 4>, <modificador, 3>, <modificadorInv, 3>, <hecho, 1>,$
$<regla, 1>, <<, 2>, <>=, 2>, <=, 2>, <≠, 2>, <^, 3>, <-, 3>, </, 3>\} \cup$
$\{<m'_{rel}, 2> / m \in M_{LDSE}\}$

Los conjuntos con subíndice LDSE corresponden a conjuntos ligados a lenguajes LDSE, mientras que los que no tienen subíndice, corresponden a sus conjuntos análogos ligados a lenguajes de la lógica clásica. Todas las relaciones identificadas con operadores son aquellas asociadas a relaciones ya conocidas, ligadas a los operadores relacionales y a funciones matemáticas transformadas en relaciones. Las relaciones de la forma m'_{rel} son las que provienen de las funciones resumidas m' ligadas a los modificadores. Todas estas relaciones tienen una semántica propia, no definida en el programa lógico clásico. 'R' representa el conjunto de los números reales.

La consecuencia sintáctica LDSE sobre un programa lógico LDSE (elemento del conjunto $\mathcal{P}(S_{LDSE})$) y sobre un átomo LDSE con una veracidad (que puede ser un valor real perteneciente al intervalo $[\alpha,1]$ o una variable) con el siguiente formato:

$\{(h_1,v_1) ..., (h_y,v_y), r_1, ..., r_z\} \vdash_{LDSE} (p,v)$

puede transformarse en la siguiente consecuencia sintáctica clásica sobre un programa lógico clásico (elemento del conjunto $\mathcal{P}(S)$) y sobre un átomo clásico:

$(Mot \cup convMod(m_1) \cup ... \cup convMod(m_x) \cup \{convHec(h_1,v_1), ..., convHec(h_y,v_y), convReg(r_1), ..., convReg(r_z)\}) \vdash convGoal(p,v)$

donde los h_i representan hechos (con sus respectivas veracidades v_i), los r_i reglas y modificadores de predicados $\{m_1, ..., m_x\} \in M_{LDSE}$.

Como abuso de notación se utilizará el término $[x_1, x_2, ..., x_n]$ como sinónimo del término $[x_1 | [x_2 | [... | [x_n | []] ...$]]] que es a su vez una versión resumida del verdadero término que es $conslist(x_1, conslist(x_2, ..., conslist(x_n, listvac)...))$. Lo mismo ocurre con el término [], que es sinónimo de listvac.

Nota: En la especificación de la traducción (a continuación), las letras en cursiva representan traducción textual, las letras normales representan parámetros y las letras en negrita representan el uso de definiciones auxiliares.

Mot (motor) representa un conjunto de cláusulas comunes a cualquier conversión. Este conjunto es el siguiente:

Mot : $\mathcal{P}(S)$
Mot = {
umbral(0.5).,
min2(X,Y,X) :- >=(Y,X).,

```
min2(X,Y,Y) :- <(Y,X).,
norm(1.0,V,0.0) :- <(V,1.0).,
norm(1.0,1.0,1.0).,
norm(0.0,0.0,0.0).,
norm(0.0,V,1.0) :- <(0.0,V).,
norm(I,V,R) :- <(I,1.0), <(0.0,I), g(I,R2), ^(V,R2,R).,
g(I,R) :- -(1.0,I,R1), /(1.0,R1,R2), -(R2,1.0,R).,
cortarUlt([X],X,[]).,
cortarUlt([X|[Y|Ys]],Z,[X|Zs]) :- cortarUlt([Y|Ys],Z,Zs).,
aplica([],V,V).,
aplica([M|Ms],VI,VF) :- aplica(Ms,VI,VF2), modificador(M,VF2,VF), umbral(Alfa),
    >=(VF,Alfa).,
aplicaInv([],V,V).,
aplicaInv([M|Ms],VI,VF) :- modificadorInv(M,VI,VI2), umbral(Alfa), >=(VI2,Alfa),
    aplicaInv(Ms,VI2,VF).,
quitaComunes([],Mu,[],Mu).,
quitaComunes([X|Mo],[],[X|Mo],[]).,
quitaComunes([X|Mo],[Y|Mu],MoF,MuF) :- cortarUlt([X|Mo],U,RMo),
    cortarUlt([Y|Mu],U,RMu), quitaComunes(RMo,RMu,MoF,MuF).,
quitaComunes([X|Mo],[Y|Mu],[X|Mo],[Y|Mu]) :-
    cortarUlt([X|Mo],UMo,RMo), cortarUlt([Y|Mu],UMu,RMu), ≠(UMo,UMu).,
desMod(Mo,Mu,VI,VF) :- quitaComunes(Mo,Mu,MoF,MuF),
    aplicaInv(MoF,VI,V), aplica(MuF,V,VF).,
demPrecInc(Ms,P,Ts,I,VD) :- atomoReal(Ms,P,Ts,VDa), norm(I,VDa,VD), umbral(Alfa),
    >=(VD,Alfa).,
demPrecs(C,[],C).,
demPrecs(C,[[Ms,P,Ts,I]|PCs],VD) :-
    demPrecInc(Ms,P,Ts,I,VDp), demPrecs(C,PCs,VDr), min2(VDp,VDr,VD).,
atomo(Ms,P,Ts,VD) :- hecho([Ms,P,Ts,VD]).,
atomo(Ms,P,Ts,VD) :- regla([[Ms,P,Ts],PCS,Conf]), demPrecs(Conf,PCS,VD).,
atomoReal(Ms,P,Ts,VD) :- atomo(M2s,P,Ts,VDa), desMod(M2s,Ms,VDa,VD).
}
```

ConvMod representa una función que dado un modificador de predicados, devuelve un par de cláusulas clásicas. Esta función está definida así:

convMod : $M_{LDSE} \rightarrow \wp(S)$
convMod(m) = { *modificador(*m, *X, RX)* :- m'_{rel}*(X, RX).*,
*modificadorInv(*m, *X, RX)* :- m'_{rel}*(RX, X).*}

ConvHec representa una función que dado un hecho con su respectiva veracidad, devuelve un hecho clásico. Esta función está definida así:

convHec : $S_{LDSE}|_{hechos}$ x $[\alpha,1] \rightarrow S$
convHec($m_1(...(m_k(p))...)(t_1, ..., t_a)$ con veracidad v) =
hecho([[m_1, ..., m_k*]*, p, *[*t_1, ..., t_a*]*, v*]).*
(con p∈{p / <p, a>∈P_{LDSE}})

ConvReg representa una función que dado una regla -que se asume tiene veracidad 1- devuelve una cláusula clásica (en realidad un hecho). Esta función está definida así:

convReg : $S_{LDSE}|_{reglas} \rightarrow S$
convReg($m_1(...(m_k(p))...)(t_1, ..., t_a)$:-
pc_1 con incidencia i_1, ..., pc_r con incidencia i_r (y confianza c)) =
regla([[[m_1, ..., m_k*]*, p, *[*t_1, ..., t_a*]], [***generarPrec(**pc_1**,** i_1**)**, ..., **generarPrec(**pc_r**,** i_r**)***], c]).*
(con p∈{p / <p, a>∈P_{LDSE}})

generarPrec : A_{LDSE} x $[0,1] \rightarrow T$

generarPrec($m_1(...(m_k(p))...)(t_1, ..., t_a)$, i) = *[[*$m_1$, ..., m_k*]*, p, *[*t_1, ..., t_a*]*, i*]*
(con $p \in \{p / <p, a> \in P_{LDSE}\}$)

ConvGoal representa una función que dado un goal (constituído por un átomo con una veracidad) devuelve un átomo clásico correspondiente al goal clásico. Esta función está definida así:

convGoal : A_{LDSE} x ($[\alpha,1] \cup V_{LDSE}$) $\rightarrow$ A
convGoal($m_1(...(m_k(p))...)(t_1, ..., t_a)$ con veracidad v) =
atomoReal([[m_1, ..., m_k*]*, p, *[*t_1, ..., t_a*]*, v*])*
(con $p \in \{p / <p, a> \in P_{LDSE}\}$)

El operador | utilizado sobre el conjunto S_{LDSE} representa una restricción (subconjunto) dada respecto a ese conjunto. Se utiliza para restringir la primer coordenada del dominio de la función convHec a las cláusulas LDSE que son hechos y el dominio de la función convReg a las cláusulas LDSE que son reglas. Como abuso de notación utilizamos una sintaxis más natural para los elementos del dominio de las funciones convHec, convReg y convGoal.

Cabe aclarar que la consecuencia sintáctica mencionada en el punto anterior queda reflejada dentro de la definición del predicado **atomoReal** en las definiciones de reglas, mientras que la aumentación (que también se usa como consecuencia sintáctica), queda reflejada dentro de la definición del predicado **aplica** del motor. La disminución (que también se usa como consecuencia sintáctica) queda reflejada en el predicado **aplicaInv**.

Luego de haber especificado el método de conversión de cláusulas LDSE a cláusulas clásicas, estamos en condiciones de hablar sobre la completitud de la consecuencia sintáctica $\vdash_{LDSE}$. La consecuencia sintáctica $\vdash_{LDSE}$ es completa por dos razones: la consecuencia sintáctica clásica es completa (ver [NIL/87]), y además se puede verificar que existe una correspondencia de uno a uno entre las respuestas deducidas sintácticamente de un conjunto de cláusulas LDSE y las respuestas deducidas sintácticamente del conjunto de cláusulas clásicas obtenidas como resultado de aplicar el método de conversión de cláusulas antes mencionado a las cláusulas LDSE anteriores. Justamente, las respuestas deducidas a partir de las cláusulas LDSE, convertidas a clásicas, coinciden con las deducidas a partir de las cláusulas clásicas equivalentes (considerando siempre a una cláusula verdadera si su valor de verdad es mayor a α). Al ser completa la lógica clásica, también lo será la lógica LDSE.

V.D.2. Ejemplo

A continuación se detalla un ejemplo de programa clausal LDSE con su conversión al programa clausal clásico usando el criterio antes descripto. El programa se escribe en lenguaje Prolog, y el programa LDSE original se encuentra al comienzo marcado como texto de comentario. También se indica para ambos casos un ejemplo de consulta al programa.

/* Ejemplo de programa difuso LDSE de primer orden

Programa:

```
generoso(dante) con veracidad 0.9.
dadivoso(dante,X) con veracidad 0.7.
algo(bastante(creyente))(dante,X) con veracidad 0.7.
divino(dios) con veracidad 1.0.

muy(bastante(pobre))(X) :-
  generoso(X) con incidencia 0.7,
  algo(bastante(dadivoso))(X,projimo) con incidencia 0.5
  (y confianza 0.9).
entra_en(X,cielo) :-
  pobre(X) con incidencia 0.4,
  muy(algo(creyente))(X,Y) con incidencia 0.3,
  divino(Y) con incidencia 1.0
  (y confianza 0.8).
entra_en(X,infierno) :-
  poco(pobre)(X) con incidencia 0.3
```

```
  (y confianza 0.5).
entra(X) :-
  entra_en(X,Y) con incidencia 0.5
  (y confianza 1.0).

Consecuencia sintáctica LDSE:

algo(entra_en)(X1,X2) con veracidad Y.

Resultado de la veracidad/falsedad de la consecuencia sintáctica:

X1 = dante
X2 = cielo
Y = 0.8        (más respuestas)

X1 = dante
X2 = infierno
Y = 0.5        (más respuestas)

No hay (más) respuestas
*/

umbral(0.5).

min2(X,Y,X) :- Y >= X.
min2(X,Y,Y) :- Y < X.

norm(1.0,V,0.0) :- V < 1.0.
norm(1.0,1.0,1.0).
norm(0.0,0.0,0.0).
norm(0.0,V,1.0) :- 0.0 < V.
norm(I,V,R) :- I < 1.0, 0.0 < I, g(I,R2), R is V ^ R2.

g(I,R) :- R is (1.0 / (1.0 - I)) - 1.0.

cortarUlt([X],X,[]).
cortarUlt([X|[Y|Ys]],Z,[X|Zs]) :- cortarUlt([Y|Ys],Z,Zs).

aplica([],V,V).
aplica([M|Ms],VI,VF) :- aplica(Ms,VI,VF2), modificador(M,VF2,VF),
  umbral(Alfa), VF >= Alfa.

aplicaInv([],V,V).
aplicaInv([M|Ms],VI,VF) :- modificadorInv(M,VI,VI2), umbral(Alfa),
  VI2>=Alfa, aplicaInv(Ms,VI2,VF).

quitaComunes([],Mu,[],Mu).
quitaComunes([X|Mo],[],[X|Mo],[]).
quitaComunes([X|Mo],[Y|Mu],MoF,MuF) :-
  cortarUlt([X|Mo],U,RMo), cortarUlt([Y|Mu],U,RMu),
  quitaComunes(RMo,RMu,MoF,MuF).
quitaComunes([X|Mo],[Y|Mu],[X|Mo],[Y|Mu]) :-
  cortarUlt([X|Mo],UMo,RMo), cortarUlt([Y|Mu],UMu,RMu), UMo \== UMu.

desMod(Mo,Mu,VI,VF) :- quitaComunes(Mo,Mu,MoF,MuF), aplicaInv(MoF,VI,V),
  aplica(MuF,V,VF).
```

```
demPrecInc(Ms,P,Ts,I,VD) :- atomoReal(Ms,P,Ts,VDa), norm(I,VDa,VD),
  umbral(Alfa), VD >= Alfa.

demPrecs(C,[],C).
demPrecs(C,[[Ms,P,Ts,I]|PCs],VD) :-
  demPrecInc(Ms,P,Ts,I,VDp), demPrecs(C,PCs,VDr), min2(VDp,VDr,VD).

atomo(Ms,P,Ts,VD) :- hecho([Ms,P,Ts,VD]).
atomo(Ms,P,Ts,VD) :- regla([[Ms,P,Ts],PCs,Conf]), demPrecs(Conf,PCs,VD).

atomoReal(Ms,P,Ts,VD) :- atomo(M2s,P,Ts,VDa), desMod(M2s,Ms,VDa,VD).

/* ----- */

modificador(muy,X,RX) :- RX is X ^ 2.0.
modificadorInv(muy,X,RX) :- RX is X ^ (1.0 / 2.0).
modificador(algo,X,RX) :- RX is X.
modificadorInv(algo,X,RX) :- RX is X.
modificador(poco,X,RX) :- RX is X ^ (1.0 / 2.0).
modificadorInv(poco,X,RX) :- RX is X ^ 2.0.
modificador(bastante,X,RX) :- RX is X ^ 1.5.
modificadorInv(bastante,X,RX) :- RX is X ^ (1.0 / 1.5).

hecho([[],generoso,[dante],0.9]).
hecho([[],dadivoso,[dante,X],0.7]).
hecho([[algo,bastante],creyente,[dante,X],0.7]).
hecho([[],divino,[dios],1.0]).

regla([[[muy,bastante],pobre,[X]],[[],generoso,[X],0.7],[[algo,bastante],
  dadivoso,[X,projimo],0.5]],0.9]).
regla([[[],entra_en,[X,cielo]],[[],pobre,[X],0.4],[[muy,algo],
  creyente,[X,Y],0.3],[[],divino,[Y],1.0]],0.8]).
regla([[[],entra_en,[X,infierno]],[[[poco],pobre,[X],0.3]],0.5]).
regla([[[],entra,[X]],[[],entra_en,[X,Y],0.5]],1.0]).

/*
Consecuencia sintáctica Prolog:

atomoReal([algo],entra_en,X,Y).

Resultado de la veracidad/falsedad de la consecuencia sintáctica:

X = [dante,cielo]
Y = 0.8        (más respuestas)

X = [dante,infierno]
Y = 0.5        (más respuestas)

No hay (más) respuestas
*/
```

V.E. Características del lenguaje en la aplicación

V.E.1. Características principales

El lenguaje implementado en la aplicación no es exactamente LDSE; por un lado es una extensión mientras que por otro es una limitación. En realidad, representa un lenguaje LDSE tipificado sin inferencia de tipos, clausal y restringido.

Un lenguaje **LDSE tipificado** está representado por la siguiente estructura:

$$\pounds = < Z, v, c, F, P, M, \{\neg, \wedge\}, \{\forall\}, \{(,), \{, \}, \sigma\} >$$

donde cada componente representa:

① conjunto finito de tipos $Z = \{z_g\}_{g \in G}$
② conjunto infinito numerable de variables $v = \{<v_h, t(h)>\}_{h \in H}$
③ conjunto de constantes $c = \{<c_i, t(i)>\}_{i \in I}$
④ conjunto finito de símbolos de función $F = \{<f_j, \{t_1(j), ..., t_{n(j)}(j)\}, t(j)>\}_{j \in J}$
⑤ conjunto finito de símbolos de predicado $P = \{<p_k, \{t_1(k), ..., t_{n(k)}(k)\}>\}_{k \in K}$
⑥ conjunto finito de símbolos modificadores de predicados $M = \{m_l\}_{l \in L}$
⑦ conjunto de conectivos lógicos
⑧ conjunto de símbolos cuantificadores
⑨ conjunto de símbolos de precedencia y auxiliares

La función 't' ligada a los símbolos de variables y de constantes representa el tipo de cada una de ellas; la ligada a los símbolos de función representa el tipo de la imagen de ésta. La función 't_i' ligada a los símbolos de función y de predicados representa el tipo del "parámetro i-ésimo" de cada uno de ellos.

Las siguientes son las definiciones que difieren de las de un lenguaje LDSE:

Definición (2b):

Se define el conjunto de los **términos** (T) de la siguiente manera:

① $<v_h, t(h)> \in v \Rightarrow <v_h, t(h)> \in T$
② $<c_i, t(i)> \in c \Rightarrow <c_i, t(i)> \in T$
③ $<t_1, z_1> \in T$ y ... y $<t_a, z_a> \in T$ y $<f_j, \{z_1, ..., z_a\}, z> \in F \Rightarrow <f_j(<t_1, z_1>, ..., <t_a, z_a>), z> \in T$
④ T es cerrado por las construcciones anteriores.

Definición (3b):

Se define el conjunto de los **predicados modificados** (PM) de la siguiente manera:

① $<p_k, \{t_1(k), ..., t_{n(k)}(k)\}> \in P \Rightarrow <p_k, \{t_1(k), ..., t_{n(k)}(k)\}> \in PM$
② $<p, z> \in PM$ y $m_l \in M \Rightarrow <m_l(p), z> \in PM$
③ PM es cerrado por las construcciones anteriores.

Definición (4b):

Se define el conjunto de los **átomos** (A) -también llamados **fórmulas atómicas**- de la siguiente manera:

① $<t_1, z_1> \in T$ y ... y $<t_a, z_a> \in T$ y $<p, \{z_1,..., z_a\}> \in PM \Rightarrow p(<t_1, z_1>, ..., <t_a, z_a>) \in A$
② A es cerrado por la construcción anterior.

Definición (7b):

Se define una **estructura adecuada** para el lenguaje $\mathfrak{L}$ a lo siguiente:

① Un mapeo de cada $z_g \in \mathbf{Z}$ a un conjunto z_g^D ($z_g^D \neq \emptyset$) llamado el universo del tipo
② Un mapeo de cada $\langle c_i, z_i \rangle \in \mathbf{c}$ a un $c_i^D \in z_i^D$
③ Un mapeo de cada $\langle f_j, \{z_1, ..., z_a\}, z \rangle \in \mathbf{F}$ a una función $f_j^D : z_1^D \times ... \times z_a^D \rightarrow z^D$
④ Un mapeo de cada $\langle p_k, \{z_1, ..., z_a\} \rangle \in \mathbf{P}$ a una función $p_k^D : z_1^D \times ... \times z_a^D \rightarrow [0,1]$
⑤ Un mapeo de cada $m_l \in \mathbf{M}$ a una función $m_l^D : Z_{in}^{dom*}$
(donde $Z_{in}^{dom*} =_{def} \cup_{i>0}\{((t_1^D \times ... \times t_i^D \rightarrow [0,1]) \rightarrow (t_1^D \times ... \times t_i^D \rightarrow [0,1])) / t_j \in Z \ \forall 1 \leq j \leq i\}$)
⑥ Un mapeo de cada $\langle p, \{z_1, ..., z_a\} \rangle \in \mathbf{PM}$ a una función $PM_p^D : z_1^D \times ... \times z_a^D \rightarrow [0,1]$

La tipificación sólo es considerada como una validación para la construcción sintáctica de términos y fórmulas. Los conjuntos ligados a los tipos no influyen en el cálculo de las interpretaciones semánticas de ellas.

La restricción del lenguaje implementado respecto al lenguaje LDSE tipificado se refleja en las características de sus componentes, a saber:

$$\mathfrak{L} = < \mathbf{Z} \cup \{R\}, \mathbf{v}, \mathbf{c} \cup \{\langle c, R \rangle / c \in R\}, \emptyset, \mathbf{P} \cup \{\langle =, \{t, t\} \rangle, \langle \neq, \{t, t\} \rangle, \langle \geq, \{t, t\} \rangle, \langle \leq, \{t, t\} \rangle, \langle >, \{t, t\} \rangle, \langle <, \{t, t\} \rangle\}_{t \in Z \cup \{R\}}, \mathbf{M}, \{\neg, \wedge\}, \{\forall\}, \{(,), \{, \}, \sigma\} >$$

Cabe aclarar que aquí se utilizó un abuso de notación, ya que la expresión 'R' se usó para representar, además del tipo (real), su universo o interpretación -el mapeo- que sería el conjunto de los números reales. Lo mismo se utilizó sobre las expresiones que representan los números, ya que se utilizó el mismo símbolo como constante y como su semántica (significado), que es el número real asociado. Vale la pena notar que el lenguaje implementado no posee símbolos de función. En futuras versiones, el lenguaje podría extenderse con estos nuevos símbolos.

En la aplicación, entre los términos aparece una clase nueva de ellos, llamados señales. Estas señales pueden llegar a comprenderse como "constantes con un valor inicial fijo". Cada una representa una medición física específica de una entidad que conforma el estado general de la planta en un instante dado. Esta medición -que representa su valor asociado- es una constante que permanece fija dentro del mismo instante, es decir, dentro del tiempo necesario para completar un ciclo de demostración de una lectura. La incorporación de nuevos términos amplía el concepto de apareamiento de términos durante la demostración, para encontrar reglas aplicables. Las señales unifican con cualquier variable. También unifican con aquella constante que coincida con su valor asociado. Por útlimo, unifican con otras señales que tengan asociado la misma constante.

Los operadores relacionales son difusos, y las relaciones asociadas a ellos se pueden llegar a ver también como sobrecargadas (para todos los tipos) e infijas. La semántica ligada a estas relaciones asociadas a los operadores relacionales difusos es considerada como infija, y se detallará más adelante. Aparece una restricción en la sintaxis de las fórmulas generadas por las relaciones asociadas a los operadores relacionales difusos, ya que ambos parámetros-términos no pueden ser constantes a la vez.

En la aplicación, las reglas conforman el conocimiento del experto y son ingresadas previamente. Pero como parte del conocimiento del experto, se puede escribir conocimiento muy similar a los hechos. Estos "hechos" constan de reglas sin precondiciones, donde su confianza pasaría a ser el valor de veracidad de ese hecho. Este conocimiento no sufre cambios durante el proceso de demostración. Sin embargo, otros hechos están ligados a aseveraciones directas sobre las señales que representan el estado general de la planta en un instante dado. En otros sistemas expertos, el concepto de hecho es dinámico; en cambio, en la aplicación esto no existe. Sin embargo, existe algo muy similar que son valores asignados a las señales para un instante determinado, que varían en función de las lecturas del estado del reactor.

El concepto de instanciación es análogo al de otros sistemas expertos que manipulan variables como términos. Sin embargo la aplicación utiliza un criterio específico al tratar de demostrar un átomo cuyo predicado es un operador relacional difuso y alguno de sus dos términos corresponde a una variable sin instanciar. Si los dos términos son de tipo R (números reales), entonces el átomo falla; si fueran de un tipo de lista de valores, la o las variables sin instanciar toman todos los posibles valores de constantes del tipo (dominio) asociado, y se demuestra cada combinación del átomo original con las sustituciones aplicadas, por separado (con un estilo de demostración por generate-and-test). Para poder instanciar estas variables con todos sus posibles valores para luego probar, es necesario que dentro del conocimiento esté especificado cuáles son todos los términos constantes de cada tipo. Es evidente que para realizar ésto, es necesario manipular tipos en tiempo de demostración, y estos tipos van a formar parte de la identificación de todos los términos

utilizados (incluídas las variables). Las señales no tienen sustituciones asociadas luego de unificaciones exitosas (es decir, no se comportan como variables).

Otra característica es que la representación de las reglas implementadas pueden ser más amplias que las cláusulas teóricas. Las precondiciones de estas reglas están constituídas por átomos con incidencias conectados entre sí con operadores de conjunción y disyunción, pudiendo mantener además más de un nivel de paréntesis. Sin embargo, internamente la precondición de una regla es llevada a una forma normal disyuntiva de conjunciones de átomos con incidencias (donde esta precondición es transformada a través de reglas ya mencionadas antes que preservan la semántica difusa, como por ejemplo las versiones difusas de las leyes de De Morgan). El formato de las reglas "intermedias" implementadas ya tranformadas es el siguiente:

$$<<q :- \{p_{11}, i_{11}\}, ..., \{p_{1k}, i_{1k}\} \vee \{p_{21}, i_{21}\}, ..., \{p_{2l}, i_{2l}\} \vee ... \vee \{p_{m1}, i_{m1}\}, ..., \{p_{mt}, i_{mt}\}>, c>$$

donde '$\vee$' representa el operador de disyunción en el contexto del antecedente de la regla. Sin embargo, esta regla se puede descomponer en varias subreglas de la regla original -preservando la semántica- de la siguiente forma:

$$<<q :- \{p_{11}, i_{11}\}, ..., \{p_{1k}, i_{1k}\}>, c>$$
$$<<q :- \{p_{21}, i_{21}\}, ..., \{p_{2l}, i_{2l}\}>, c>$$
$$...$$
$$<<q :- \{p_{m1}, i_{m1}\}, ..., \{p_{mt}, i_{mt}\}>, c>$$

donde ahora sí mantienen la sintaxis de las cláusulas teóricas.

Otra característica de la implementación es que para el proceso de demostración no se ingresa el átomo para verificar si es consecuencia sintáctica de las cláusulas y los hechos. La misma aplicación demuestra aquellos átomos que son conclusiones de todas las reglas, cuyos predicados asociados son llamados "principales". En la aplicación, aquellas reglas cuyo átomo conclusión está compuesto de un predicado principal, también serán indicadas como principales (con un valor de sólo lectura).

Una característica adicional del lenguaje implementado es el control de la recursión, donde el motor de inferencias falsifica la demostración de un átomo cuya regla aplicada (en realidad regla-subregla) ya fue aplicada en una instancia anterior de la traza de la demostración. Esta es una propiedad netamente práctica.

La aplicación, al caer dentro de la clasificación de sistema experto, necesita además incorporar un mecanismo de registro de los motivos y explicaciones correspondientes a la demostración de cada una de las conclusiones.

V.E.2. Operadores relacionales difusos

De todos los operadores relacionales usados, los más significativos son el operador '=' (igualdad entre dos elementos del mismo tipo), el operador '>' (mayor entre dos números -con extensión a cualquier tipo-) y '<' (menor entre dos números -con extensión a cualquier tipo-), sobre cuya semántica difusa hablaremos a continuación.

La declaración de la semántica de todos estos operadores será la siguiente:

$$\_\oplus\_ : t \times t \rightarrow [0,1]$$ (donde '$\oplus$' es el operador relacional, y 't' es el dominio de los elementos)

Un ejemplo muy genérico de definición del operador '=' es el siguiente:

$$x = y =_{def} ND_x(y)$$

donde ND_x representa una función semántica ligada a un concepto similar al de número difuso x (que es un conjunto difuso). Sin embargo, no es necesario que esta función semántica cumpla el punto 3 de la definición de número difuso. Para repasar el concepto de número difuso, ver el capítulo "**III.A.** Introducción a la lógica difusa".

El uso de números difusos da la idea de existencia de un orden entre los elementos. Vamos a asumir que el orden entre los elementos del dominio D lo da una función $\mathbf{orden_D}$, que dado un elemento de ese dominio, devuelve un número real. Esta función debe reflejar monotonía estricta (a mayor valor en el dominio, mayor deberá ser su valor de orden

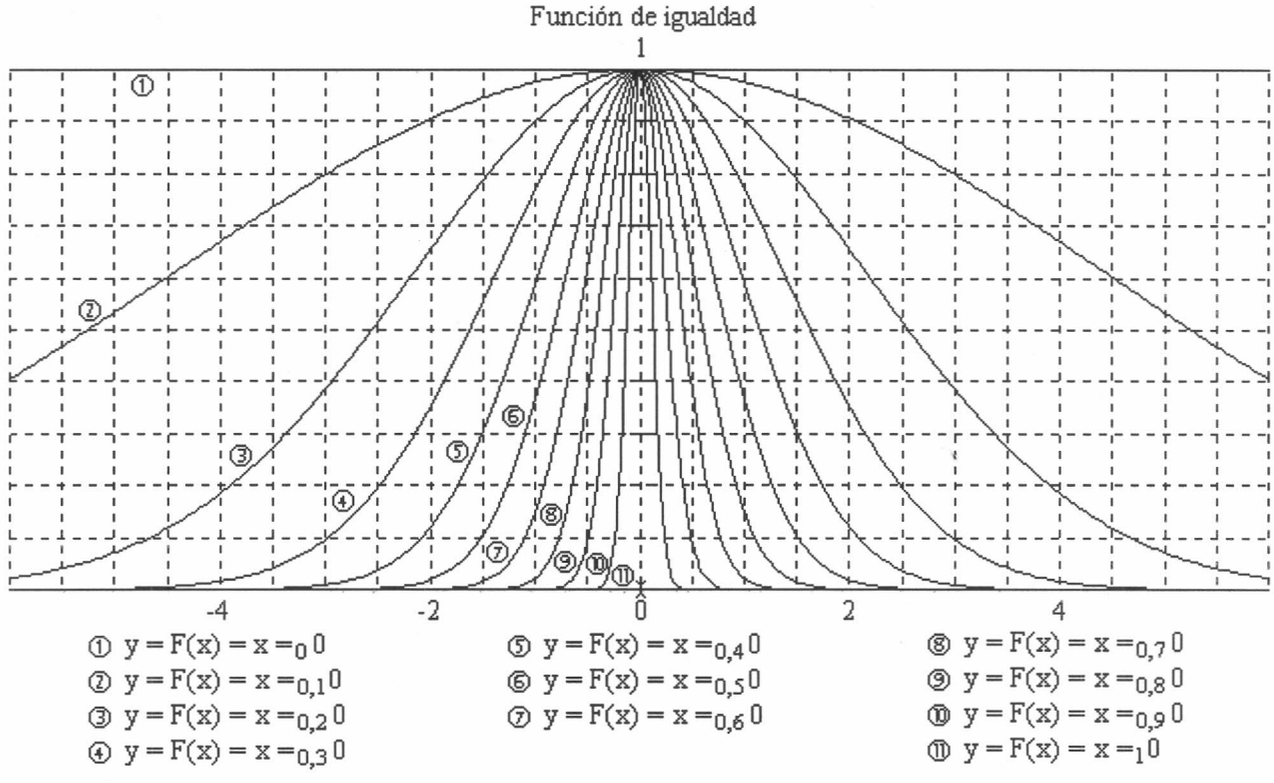
asociado). Dos elementos del dominio considerados "cercanos" cumplirán que la diferencia entre la aplicación de la función orden a cada uno de ellos será menor que si no lo fueran; si esos dos elementos son "lejanos", su diferencia será mayor. Dicho de otra manera, la función 'orden' refleja la idea de la existencia de un orden total dentro de los elementos de cualquier tipo. Sobre el tipo de los números reales siempre valdrá que $orden_R(x) = x$.

Una definición propia de la semántica del operador '=' difuso es la siguiente:

$$x =_\varepsilon^D y =_{def} \begin{cases} 1 & (\text{si } \varepsilon=0) \\ e \text{ ^ } -((orden_D(x) - orden_D(y)) / tg((1-\varepsilon)*\pi/2))^2 & (\text{si } \varepsilon\in(0,1)) \\ 1 \ (\text{si } x =^D y); \quad 0 \ (\text{si } x \neq^D y) & (\text{si } \varepsilon=1) \end{cases}$$

donde 'ε' (ε∈[0,1]) representa un factor de estrictez (que también podría pensarse como grado de difusidad) interpretado de una forma similar que una incidencia. En la tercera definición de la igualdad difusa (caso ε=1), los operadores '$=^D$' y '$\neq^D$' representan los operadores relacionales clásicos de igualdad y desigualdad sobre los elementos del dominio D. Si el dominio en cuestión es R (reales), el supraíndice R será omitido. Es necesario aclarar que esta función no cumple con el punto 3 de la definición de número difuso.

En el siguiente dibujo vemos graficada a la función $F_\varepsilon(x) =_{def} x =_\varepsilon 0$, para varias instancias de ε que varían desde 0 hasta 1. Esta función tiene forma de "campana con un valor de media 0". Aquí se ve que cada vez que se tiende el valor de ε a 1, la función tiene un comportamiento cada vez más estricto (tiende a la función de igualdad clásica).



Esta función puede generalizarse utilizando una "media" de valor y (y∈R) y definiéndola como $Ig_\varepsilon(x, y) =_{def} F_\varepsilon(x-y)$.

Se tomó esta función como semántica de la igualdad difusa porque cumple que a medida que los elementos del dominio se acercan (alejan) al valor a comparar, la veracidad de la igualdad será cada vez mayor (menor). También cumple la propiedad deseable que es "simétrica par" respecto al valor a comparar, y que es continua y derivable (salvo en el caso que ε=1, que representa la igualdad clásica). Otra propiedad que se cumple es que no se encuentran intervalos de valores del dominio con derivada cero, lo que muestra una característica más difusa que bivaluada / multivaluada. Esta función tiene su origen en la función de densidad de la distribución normal (que se puede encontrar en la literatura clásica

de probabilidades), donde el valor a comparar se asemeja a su esperanza y el valor de ε a su varianza. Sin embargo, la función aquí utilizada fue ligeramente modificada para adaptarla a los objetivos del trabajo.

Para las definiciones de los otros operadores difusos de orden se puede utilizar el concepto de "filtro". Un ejemplo muy genérico de definición de un operador '~' de orden es el siguiente:

$$x \sim y =_{def} FTR_x(y)$$

donde FTR_x representa la función semántica ligada a un filtro dado. Una función f cumple la propiedad de ser filtro cuando su dominio corresponde al de los números reales y se cumple que es monótona, continua y además:

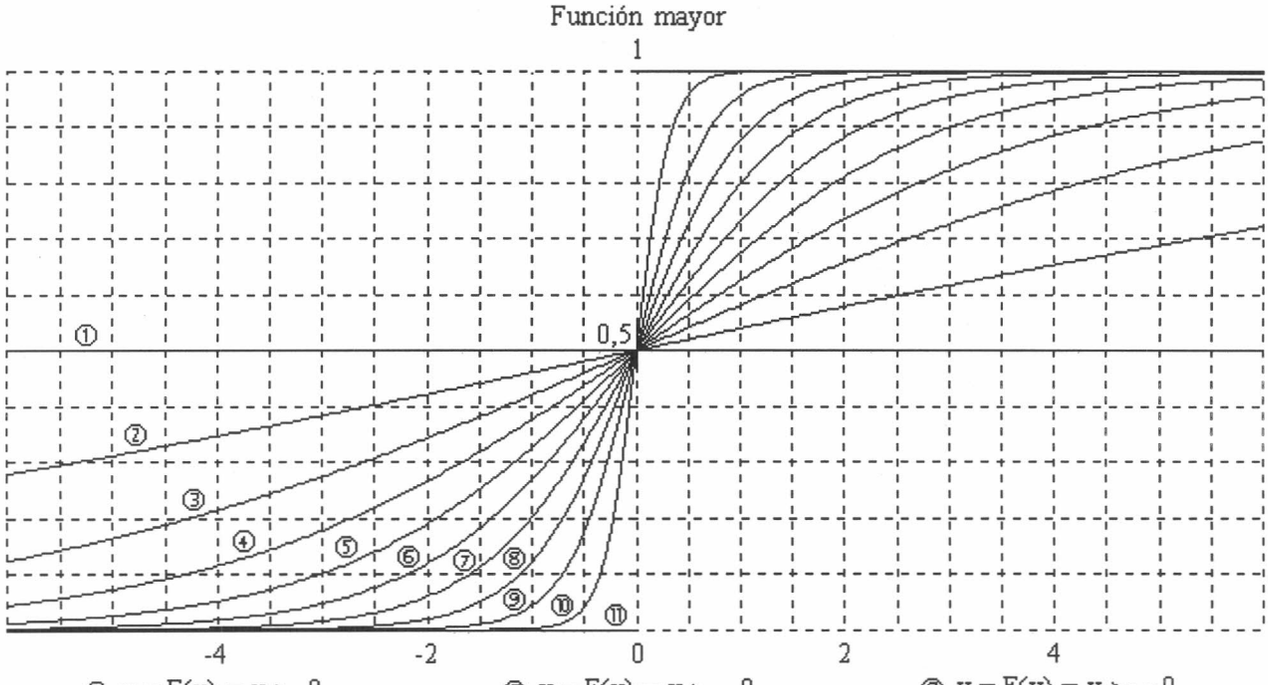
$$\lim_{x \to +\infty} f(x) = 1 - \lim_{x \to -\infty} f(x)$$

Una definición propia de la semántica del operador '>' difuso es la siguiente:

$$x >_\varepsilon^D y =_{def} \begin{cases} 0{,}5 & (\text{si } \varepsilon=0) \\ 1 \,/\, (1 + e \,\hat{}\, -(\operatorname{tg}(\varepsilon * \pi/2) * (\text{orden}_D(x) - \text{orden}_D(y)))) & (\text{si } \varepsilon \in (0,1)) \\ 1 \text{ (si } x >^D y);\ 0 \text{ (si } x <^D y);\ 0{,}5 \text{ (si } x =^D y) & (\text{si } \varepsilon=1) \end{cases}$$

donde en este caso 'ε' representa lo mismo que en la función anterior. Como en el caso anterior, también omitiremos el dominio si es R. Aquí los operadores '$>^D$', '$<^D$' y '$=^D$' representan los operadores relacionales clásicos de mayor, menor e igualdad sobre los elementos del dominio D.

En el siguiente dibujo vemos graficada a la función $F_\varepsilon(x) =_{def} x >_\varepsilon 0$, para varias instancias de ε que varían desde 0 hasta 1. Aquí se ve, como en el caso anterior, que cada vez que se tiende el valor de ε a 1, la función tiene un comportamiento cada vez más estricto (tiende a la función mayor clásica).



① $y = F(x) = x >_0 0$
② $y = F(x) = x >_{0,1} 0$
③ $y = F(x) = x >_{0,2} 0$
④ $y = F(x) = x >_{0,3} 0$
⑤ $y = F(x) = x >_{0,4} 0$
⑥ $y = F(x) = x >_{0,5} 0$
⑦ $y = F(x) = x >_{0,6} 0$
⑧ $y = F(x) = x >_{0,7} 0$
⑨ $y = F(x) = x >_{0,8} 0$
⑩ $y = F(x) = x >_{0,9} 0$
⑪ $y = F(x) = x >_1 0$

Las definiciones de los otros operadores difusos son las siguientes:

$$x \neq_\varepsilon^D y =_{def} 1 - (x =_\varepsilon^D y)$$

$x \geq_\varepsilon^D y =_{def} máx(x >_\varepsilon^D y, x =_\varepsilon^D y)$
$x <_\varepsilon^D y =_{def} 1 - (x \geq_\varepsilon^D y)$
$x \leq_\varepsilon^D y =_{def} 1 - (x >_\varepsilon^D y)$

Se tomó esta función como semántica del mayor difuso por razones similares a la elección de la igualdad difusa. Se cumple que a medida que los elementos del dominio son cada vez menores (mayores) al valor a comparar, la veracidad del mayor será cada vez menor (mayor). También cumple la propiedad deseable que es "simétrica impar" respecto al valor a comparar, y que es continua y derivable (salvo en el caso que $\varepsilon=1$, que representa al mayor clásico). Otra propiedad que se cumple es que no se encuentran intervalos de valores del dominio con derivada cero, lo que, al igual que el operador de igualdad, muestra una característica más difusa que bivaluada / multivaluada. Otra característica deseable es que si colapsáramos esta función a la lógica clásica bivaluada el resultado será la función mayor clásica. Por colapsar se entiende mapear el resultado de cada valor verdadero de los elementos del dominio a 1 y falso a 0, considerando al valor 0,5 como límite entre los valores verdaderos y falsos. Esto no ocurre con el igual difuso ya visto antes, ya que en la igualdad clásica la partición de los elementos verdaderos no representa un intervalo infinito, sino un conjunto unitario con veracidad 1 asociada a ese valor, y no contiene un "elemento frontera" con veracidad 0,5 en la igualdad difusa. Esta función está basada en una de las posibles funciones sigmoideas utilizadas en redes neuronales (que se puede encontrar en la literatura clásica de redes neuronales), donde el valor ε se asemeja a su factor de escalonamiento β. Sin embargo, la función aquí utilizada también fue ligeramente modificada para adaptarla a los objetivos del trabajo.

Vale la pena notar que en las definiciones de '$\neq$' y '$<$' anteriores no se usó la función 'neg' con el respectivo umbral. Esto se determinó con el objetivo de que la semántica de las relaciones inversas no sean complementarias respecto al umbral (o sea que se pueda cumplir que los valores de verdad de un término y de su negación sean verdaderos -o falsos- a la vez).

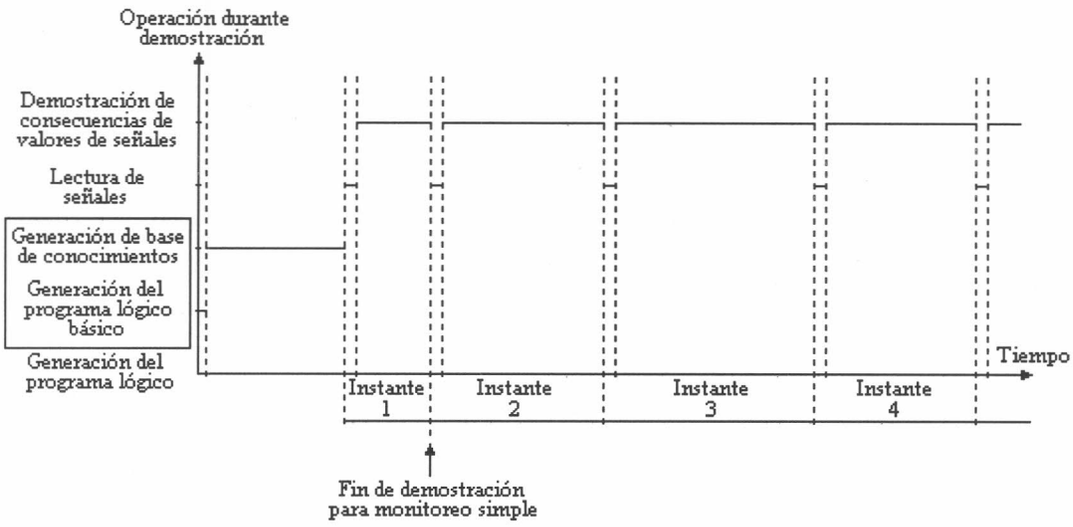
V.E.3. Formato del programa generado

Al comienzo de cada proceso de demostración (de uno o más instantes consecutivos del estado del reactor), se genera un archivo de conocimientos. Este archivo que refleja el conocimiento es un programa lógico que inmediatamente será tomado como programa fuente por un motor de demostración (máquina virtual lógica) para demostrar si los átomos con predicados asociados consultables (principales) que son consecuencias de reglas se derivan sintáctimante de ese programa fuente. Dentro del programa lógico pueden distinguirse tres secciones, a saber:

① Parte general básica (motor básico): es común a todos los programas lógicos generados, y se compone del motor central y de predicados generales de uso durante el proceso de demostración. Equivale a una versión refinada del conjunto 'Mot', mencionado en el método de conversión de un programa difuso LDSE en su equivalente clásico.
② Parte específica del conocimiento: involucra a las definiciones de reglas, de modificadores (con sus modificadores inversos) y de los dominios con sus constantes asociadas. Esta información (junto con la anterior) es considerada estática durante el proceso completo de demostración (sin importar cuántos instantes -lecturas- serán contemplados en ésta). Equivale a los predicados 'modificador', 'modificadorInv', 'regla' (ahora llamado 'datoreg') y un predicado de dominios con sus constantes asociadas, llamado 'dominio'.
③ Parte específica de un instante (lectura): involucra a los valores tomados por las señales dentro de un instante dado. Involucra al predicado 'variables', que indica cuáles son las señales y cuál es el valor asociado (como término constante) a cada una de ellas. Se asume que dentro del mismo instante, las señales no cambian su valor. No forma parte del archivo físico de conocimientos, ya que esta información varía dentro de la demostración.

A continuación presentamos un diagrama de tiempos de las operaciones presentes en la demostración. Aquí se ve que los tiempos de monitoreo (demostración de consecuencias de valores de señales) en cada uno de los instantes son variables. También lo son los tiempos de lecturas de señales.

Diagrama de tiempos



V.F. Comentarios al desarrollo

V.F.1. Relación lógica difusa LDSE-lógica clásica

Tal como se describió en el punto **V. A. 3.**, se ha tomado en cuenta para el desarrollo del lenguaje LDSE una serie de definiciones hechas por Lee ([LEE/71] y [LEE/72]) en sus trabajos sobre lógica difusa. Los conceptos de Lee se basan en dividir los valores de verdad en dos grupos: verdaderos y falsos, donde el valor de verdad es tomado en cuenta sólo si es verdadero. Esta división sugiere que el lenguaje no es puramente difuso, ya que subyace una clasificación binaria de los valores de verdad. Con el objetivo de poder expresar un modelo consistente, el desarrollo del lenguaje LDSE adoptó las definiciones aportadas por Lee.

Como se dijo antes, la lógica LDSE representa por definición una extensión de la lógica clásica, aunque sin embargo parecen existir casos en que esto no ocurre, u ocurre lo contrario. Un ejemplo claro lo da el hecho de que la implementación final del lenguaje LDSE se basa en una representación de la lógica clásica. Esto sugiere una mayor expresividad de la lógica clásica con respecto a la lógica LDSE, que es lo contrario de lo antes mencionado. Lo que ocurre en este caso es que la lógica clásica donde se implementa el lenguaje LDSE representa un nivel superior de discurso (metalenguaje) que generaliza al nivel normal del discurso, dado por el lenguaje LDSE. El mismo planteo surge en el ámbito de las funciones cuando se analiza si éstas en general son casos particulares de relaciones o es lo contrario.

Algo similar ocurre con las definiciones presentadas de consecuencia lógica y consecuencia sintáctica. Éstas son definiciones no difusas pese a que se basan en asignaciones de valores difusos de verdad a las fórmulas o cláusulas. A pesar de que existen definiciones o propiedades difusas, permanentemente se hacen aseveraciones sobre esas propiedades (se cumple o no se cumple esa propiedad para un determinado elemento del dominio con una determinada veracidad), que son condiciones con un valor de verdad bivalente, lo que refuerza lo dicho anteriormente.

Debido a que el lenguaje LDSE se implementa sobre la lógica clásica, el método de resolución utilizado es el de Robinson (ver [NIL/87]), al igual que la noción de cláusula nula, que determina la veracidad de la consecuencia sintáctica respecto al goal inicial (convertido del goal LDSE original).

V.F.2. Límite establecido entre valores verdaderos y falsos

En el lenguaje LDSE, en base a la propuesta de Lee, se ha establecido un umbral entre los valores de verdad verdaderos y falsos. Dicho umbral es el valor 0,5 (α). La operación entre valores de verdad en esta lógica mantiene la coherencia al hacer que un valor verdadero (mayor o igual a 0,5) se transforme en falso al ser negado (menor o igual a 0,5) y viceversa.

Sin embargo, en caso de modificar el umbral se deberá tener en cuenta el proveer una función semántica distinta para que la operación de negación mantenga esta coherencia en función del nuevo valor. De esta manera se podrá seguir logrando que la negación de valores de verdad verdaderos (mayores al nuevo umbral) devuelvan valores de verdad falsos (menores al nuevo umbral) y viceversa.

En caso de utilizar el valor 0,5 como umbral, la función ligada a la semántica de la negación:

$$neg(x) =_{def} 1-x$$

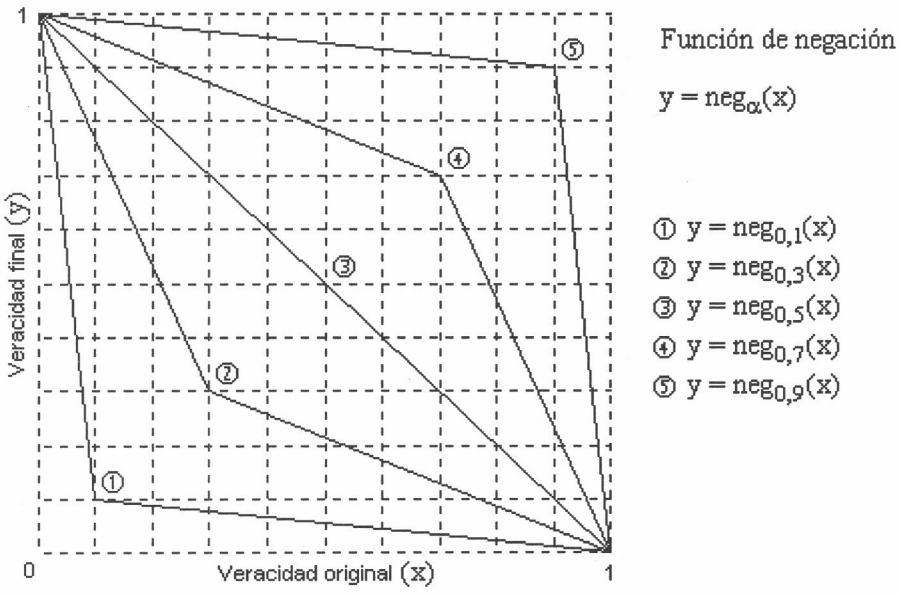
cumple con la propiedad buscada.

Se puede definir una función genérica de negación que sirva para un valor de umbral 'α' fijo (pero no necesariamente igual a 0,5) dentro del intervalo (0,1). Esta función es la siguiente:

$$neg_{\alpha}(x) =_{def} \begin{cases} (\alpha-x)*(1-\alpha)/\alpha+\alpha & (\text{si } x \leq \alpha) \\ (1-x)*\alpha/(1-\alpha) & (\text{si } x > \alpha) \end{cases}$$

Se puede ver que la función se comporta como la función semántica de la negación para el caso en que el umbral sea 0,5. A medida que hacemos tender el umbral a 0 (cero), el sistema se comportará de una forma cada vez más difusa; si lo hacemos tender a 1 (uno), el sistema se comportará de una forma más clásica (bivaluada).

En el siguiente dibujo vemos graficada a la función $F_\alpha(x) =_{def} neg_\alpha(x)$, para varias instancias de α que varían desde 0 hasta 1 (sin contar a ambos).



La función 'neg' de negación especificada en esta tesis debería tener las siguientes propiedades:

① neg_α es continua y estrictamente decreciente
② $neg_\alpha(0) = 1$ $\quad \forall\alpha\in(0,1)$
③ $neg_\alpha(1) = 0$ $\quad \forall\alpha\in(0,1)$
④ $neg_\alpha(\alpha) = \alpha$ $\quad \forall\alpha\in(0,1)$
⑤ $neg_\alpha(neg_\alpha(x)) = x$ $\quad \forall\alpha\in(0,1), x\in[0,1]$

Esta es la única función semántica que es necesario redefinir si consideramos lo expresado anteriormente. Las otras funciones semánticas básicas (las asociadas a los operadores '$\wedge$' y '$\forall$' considerando $\otimes$ = mín) no necesitan ser alteradas. Esto ocurre porque si alguno de los valores es menor al umbral (es "falso"), el resultado será a lo sumo ese valor, lo que dará un valor menor al umbral ("falso"); si todos los valores son mayores o iguales al umbral (tienen valores "verdaderos"), el resultado final será el mínimo de todos ellos, que será mayor o igual al umbral (dará un valor "verdadero"). Las funciones semánticas asociadas a los operadores definidos a partir de los operadores básicos no cambian.

En el caso de aplicar esta variante del umbral a todo el desarrollo, lo único que será necesario cambiar de él para mantener la consistencia es reemplazar todas las apariciones del valor 0,5 asociado a valores de umbral, por el valor 'α' del umbral, y reemplazar todas las apariciones de la función 'neg' por la función 'neg_α' definida recientemente. Al incorporar esta variante en la aplicación sólo será necesario realizar el primer reemplazo (el del umbral). La aplicación contempla esta variante.

V.F.3. Independencia de las incidencias

Todos los valores que reflejan un grado de verdad y pueden variar entre 0 y 1 necesitan ser mayores o iguales a 0,5 (o al umbral elegido) para ser considerados. Entre ellos se encuentran las confianzas, los valores de verdad asociados a fórmulas (en todos los contextos), y las cotas de mínimo valor verdadero en las definiciones semánticas y en las

propiedades. Sin embargo, las incidencias pueden tener cualquier valor entre 0 y 1 sin ser rechazadas. Esto ocurre porque no se realiza esta división entre incidencias "verdaderas" y "falsas" en función de su valor. Vale la pena notar que la función 'norm' (asociada a la semántica de las incidencias) no está definida usando la función 'neg', lo que le da independencia respecto al valor del umbral.

V.F.4. Lógica "trivalente" generada por algunas definiciones

Existen algunas definiciones semánticas usadas para definir la consecuencia lógica (como por ejemplo las definiciones de "satisface" y "falsifica") que para la lógica clásica son complementarias. Sin embargo, para la lógica LDSE no lo son, aunque estas definiciones tampoco sean difusas. Es posible que existan interpretaciones que ni satisfagan ni falsifiquen un par fórmula-valor de verdad. Es por esto que estas definiciones reflejan la idea de un "tercer estado" y rompen con algunas características clásicas bivalentes observadas en la lógica LDSE.

V.F.5. Relación confianza de reglas-veracidad de hechos

Ya se ha mencionado antes que las reglas no tienen un valor de verdad asociado (en realidad siempre es uno para cualquiera de ellas), pero tienen un valor fijo de confianza. Los hechos, sin embargo, no tienen una confianza (en realidad siempre es uno, considerando que son un caso particular de cláusulas), pero tienen un valor de verdad "dinámico" determinado en la demostración. Pero se puede decir que existe una relación entre las veracidades dinámicas de los hechos y las confianzas fijas de las reglas. Esto se aclara en la definición del predicado '*demPrecs*' del motor de conversión programa LDSE-programa clásico, que dice que el valor de verdad del átomo-conclusión de una regla sin precondiciones (lo que se asemeja mucho a un átomo) es la misma confianza.

V.F.6. Monotonía de los programas LDSE

Los programas LDSE, al igual que los programas lógicos clásicos "puros", son monótonos -la inserción de nuevos hechos y/o reglas no impide la demostración válida de los átomos antes deducidos-. Esto se debe a que se mantiene una estructura de motor semejante al de la lógica clásica (en realidad **usa** el de la lógica clásica), la cual cumple con la propiedad de monotonía. Además, al valer la propiedad de amplitud, podemos decir que si luego del agregado de la nueva información se deduce un átomo idéntico con veracidad mayor, el anterior se seguirá deduciendo por esta propiedad; si luego del agregado se deduce un átomo idéntico con veracidad menor, por esta propiedad el nuevo átomo será redundante.

Toda esta explicación parece repetitiva ya que la consecuencia sintáctica aplicada en los programas LDSE es una restricción de la consecuencia lógica de las fórmulas LDSE, sobre la cual se cumple la monotonía. Lo que se pretende es aclarar que en la implementación del motor de demostración para programas LDSE no existe ningún principio de lógicas no monótonas. Por ejemplo, no se usa el principio de "negación por falla" presente en algunas implementaciones no puras de lenguajes lógicos clásicos como Prolog, que aceptan átomos negados como precondiciones, y con esto rompen con la idea de cláusula clásica de Horn, y de monotonía.

V.F.7. Conjunto variable de cuantificadores difusos no contemplado en el lenguaje

La inserción de modificadores de predicados ofrece un mayor poder de expresividad de fórmulas del lenguaje LDSE respecto al lenguaje de la lógica clásica, y esto es deseable. Por esta razón, el lenguaje LDSE podría contener en su definición un conjunto de cuantificadores difusos que se podrían agregar a los ya conocidos '$\forall$' y '$\exists$' (ahora con una semántica difusa). Algunos ejemplos de estos cuantificadores podrían ser: pocos, algunos, muchos, frecuentes, etc. A éstos a su vez se le podrían llegar a aplicar modificadores, para obtener "cuantificadores modificados". Sin embargo, esta opción no fue tomada en cuenta para la definición del lenguaje debido a la dificultad de transformación de una fórmula de estas características a su cláusula equivalente durante el pasaje a la forma clausal.

V.F.8. Cálculo de veracidad del consecuente en reglas

De la definición de consecuencia sintáctica se puede ver cómo se calcula la veracidad del consecuente de una regla en función de la veracidad de sus antecedentes, de las incidencias de cada uno de los antecedentes, y de la confianza de la regla. Hemos visto que el criterio usado es el de normalización individual de la veracidad de antecedentes respecto a su incidencia, para luego tomar el mínimo de esos valores, junto a la confianza. Sin embargo este criterio tomado para calcular la veracidad del consecuente puede no ser el único. Otro criterio para calcular la veracidad del consecuente podría consistir en realizar un promedio de las veracidades de los antecedentes ponderado por sus incidencias. Otro criterio podría ser utilizar la operación de producto en lugar del mínimo entre veracidades. Estas variantes de cálculo no deben ser descartadas, aunque no reflejan una representación tan consistente e independiente del contexto como el cálculo original. Para una visión mayor de las funciones correspondientes a los operadores de implicación, ver el capítulo "**III.A.** Introducción a la lógica difusa".

V.F.9. Existencia de diferentes niveles de lenguaje

Dentro de la aplicación, el proceso de demostración está asociado a un proceso (desde el punto de vista del análisis). Este proceso se puede descomponer en dos subprocesos: el de transformación de la base de conocimientos en un formato de consulta intermedia, y el de la demostración misma de la consulta generada por el proceso anterior a partir de la o las lecturas de señales del reactor. La implementación de todos los procesos del sistema fue realizada con un lenguaje imperativo-orientado a eventos. Sin embargo el proceso de demostración propiamente dicho no fue implementado con el mismo lenguaje, sino que representa otro lenguaje de programación por sí mismo, no desarrollado en este trabajo, que interpreta a la consulta intermedia. Es decir, la consulta intermedia que es devuelta por el proceso de transformación (el proceso de transformación es un programa o código fuente) representa un dato (código objeto) para el proceso de transformación, y es tomado por el proceso de demostración mismo (que es un intérprete de programas), por lo que se puede decir que para éste, la consulta intermedia es un programa o código fuente. El proceso de demostración mismo es un intérprete de programas lógicos.

Como consecuencia de la existencia de dos lenguajes dentro de la misma aplicación, se logra aprovechar las características de un lenguaje especializado para aplicaciones de sistemas expertos (intérprete de programas lógicos) en el núcleo de la aplicación. Complementariamente se utiliza un lenguaje imperativo-orientado a eventos para implementar las rutinas de mantenimiento de la base de conocimientos del sistema.

VI. DESARROLLO DEL SISTEMA DE AYUDA AL OPERADOR DE LA CNE

Una vez culminadas las entrevistas iniciales, durante las cuales se obtuvo un acercamiento a la estructura de los POEAs y a la información que éstos manejan, se pasó a la etapa de desarrollo de una versión preliminar (prototipo) del sistema experto.

Los puntos principales que se consideraron para desarrollar el prototipo de sistema experto fueron los siguientes:

- Definición de tipos de datos y señales: Se vio necesario que el prototipo contara con una interface de definición de variables. El usuario podría de esta manera definir los controles de la central (cuyos estados son considerados en los POEAs) como *señales* de las reglas del sistema experto.

También se incorporó en el prototipo una interface de definición de *tipos de señales*. Los tipos de señales refieren a los distintos tipos de controles de la central. De esta manera es posible definir clases de controles a las cuales se puedan ligar cada una de las señales, de modo que éstas adopten sus características. Todas las señales definidas poseerían las características de su tipo asociado.

- Definición de un lenguaje de representación de los POEAs: El esquema de representación adoptado para los POEAs y para la información adicional proveniente de los expertos es el de reglas. Se consideró que el esquema de reglas es el esquema apropiado para la representación del conocimiento relacionado con la detección de eventos anormales.

En relación con los otros esquemas de representación estudiados, es el de reglas el que cuenta con el nivel de expresividad necesario para este desarrollo. Tanto el esquema de frames como el de redes semánticas proveen un poder expresivo excesivo para el nivel deseado para esta aplicación, como fue explicado anteriormente.

Se definió un esquema de reglas con lógica difusa para manejar la imprecisión propia del proceso de reconocimiento de eventos anormales.

El lenguaje de representación fue desarrollado con su correspondiente fundamento teórico cuyo estudio se ha reseñado en el capítulo anterior.

- Interface de usuario: El prototipo se desarrolló con una interface que permite la definición de tipos de datos, señales y reglas. El objetivo fue incorporar las opciones suficientes como para poder definir los principales POEAs y experimentar de esa manera con ellos, a partir de datos provisorios.

La herramienta fue diseñada de modo de poder extender la base de conocimientos de una manera sencilla, contando con una apropiada interface hombre-máquina que permita al experto agregar nuevas reglas, en la medida en que ésto se haga necesario. Dichas reglas tienen, desde el punto de vista del experto, una gramática que se encuentra especificada en el lenguaje definido a tal efecto. Dicho lenguaje fue diseñado teniendo en cuenta las características propias del lenguaje del experto y del dominio en general.

- Proceso de diagnóstico de eventos anormales: El prototipo contó también con la opción correspondiente que active el análisis de los datos de entrada a través de las reglas ingresadas.

El proceso de deducción se llevó a cabo transformando en forma automática las reglas cargadas por los expertos en un módulo escrito en lenguaje *Prolog*. Posteriormente se lleva a cabo la deducción en el ambiente Prolog y luego se obtienen los resultados en el entorno del sistema de ayuda.

La demostración del funcionamiento del prototipo del sistema experto a los responsables del proyecto en N.A.S.A. sirvió para completar el conocimiento de los POEAs y testear la correctitud de la herramienta contra la solución buscada.

Algunos detalles de la implementación final

El acercamiento provisto por el desarrollo del prototipo permitió continuar las entrevistas con los expertos con el fin de acomodar las características de la implementación final a los requerimientos planteados por ellos.

Como resultado del examen que los usuarios y los expertos realizaron sobre el prototipo, se convino mejorar los tiempos de proceso, facilitar algunos aspectos de la interface de usuario e incorporar la opción de poder armar distintas bases de conocimiento con el fin de elegir aquella que se desea utilizar en el momento correspondiente.

El proceso de detección de eventos fue mejorado con el fin de permitir al usuario detenerlo en caso de necesidad. También se mejoró la presentación final de los resultados de modo de hacerla más accesible y legible al operador. Por último, se superó la limitación que tenía el sistema experto respecto de la cantidad de reglas que podía tener la base de conocimientos, así como también de la cantidad de señales que había en el archivo de lecturas. En las primeras versiones del sistema, se producían fallos en el sistema cuando había una sobrecarga de reglas y/o señales.

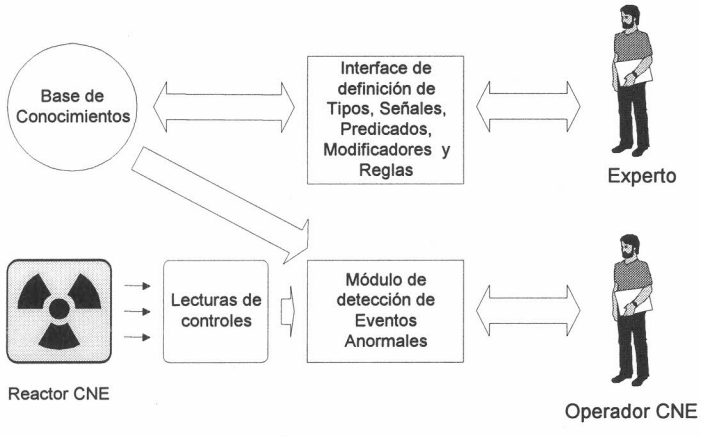
Por otro lado, en esta etapa se completó la implementación del lenguaje difuso propuesto agregando la definición de *predicados* y *modificadores*.

Para una completa revisión de las mejoras y cambios propuestos y llevados a cabo, ver el apéndice de minutas, que se encuentra en la documentación de desarrollo entregada con el presente trabajo.

VI.A. Características de la implementación del sistema experto.

El sistema experto de Ayuda al Operador para la Central Nuclear Embalse consta básicamente de una interface de usuario para ingresar tipos de datos, señales, predicados, modificadores y reglas, más un motor de inferencias que obtiene un conjunto de conclusiones sobre el estado del reactor utilizando los datos ingresados provenientes de los controles de la central.

La estructura se puede ver en la siguiente figura:



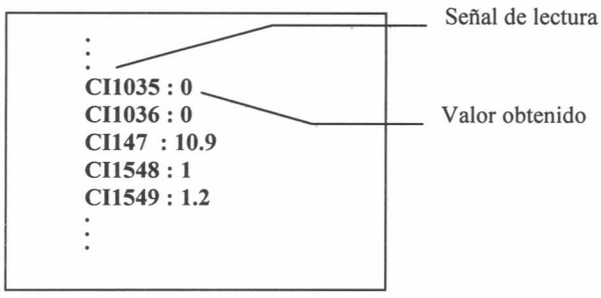
VI.A.1. Entrada de los datos de los controles de la planta

Con el fin de facilitar la adaptación del sistema a la entrada de datos directamente de los controles de la central, en lugar de una entrada de datos interna con los fines del testeo, se convino que los datos se tomaran de una serie de archivos que deberían ser alimentados convenientemente a partir del momento en que se desee incorporar al sistema en el entorno definitivo.

El objetivo de contar con una serie de archivos alimentados por las lecturas de los controles de la central, se debe a la necesidad de poder mantener todas las lecturas que se obtienen mientras se realiza el proceso de detección de eventos anormales.

La serie de almacenamientos que es utilizada como entrada de datos para el sistema experto está constituida por un conjunto de archivos con el nombre **'Lnnnnnnn.TXT'**. El prefijo 'L' sirve para identificar el archivo, mientras que el resto del nombre lo forma un número de hasta 7 (siete) dígitos que sirve para mantener el orden de la secuencia de entrada.

Cada uno de los archivos de lectura está compuesto por una secuencia de líneas, cada una de las cuales contiene el nombre de una señal junto al valor registrado:



La ubicación de las señales dentro del archivo no es relevante ya que el orden de utilización está dado por el orden de aplicación de las reglas.

VI.A.2. Base de conocimientos

La base de conocimientos se encuentra implementada sobre una base de datos relacional. Es sobre esta base de datos relacional donde los usuarios realizan los cambios correspondientes a los tipos de datos, señales, predicados, modificadores y reglas del sistema.

VI.A.3. Programa Prolog

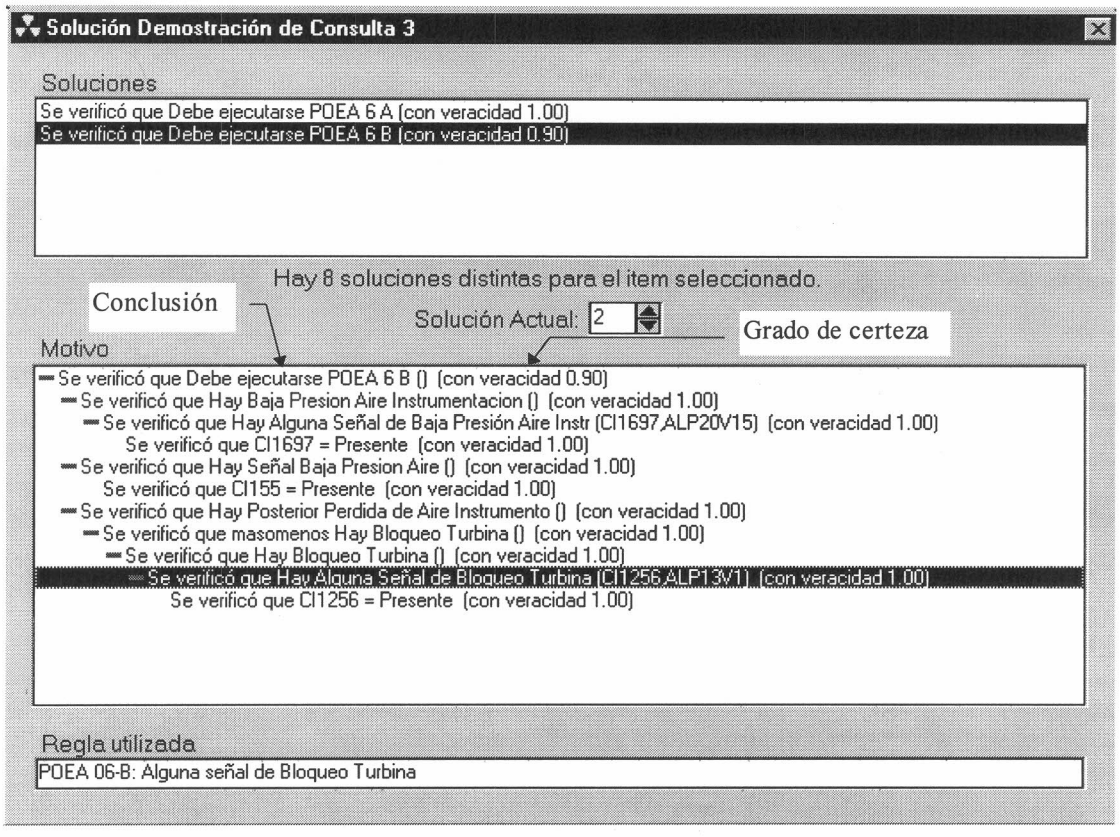
En el momento en que el usuario solicita iniciar el proceso de detección de eventos anormales, se genera un programa en lenguaje Prolog que contiene tanto el motor de inferencias como las reglas generadas a partir de la base de conocimientos con la que se está operando.

Es este programa Prolog el que realiza el proceso de inferencia propiamente dicho y devuelve los resultados al programa llamador.

VI.A.4. Resultados del programa de inferencias

La comunicación entre el programa Prolog generado y la interface con el usuario se encuentra establecida a través de una serie de archivos de nombre **'Mnnnnnnn.TXT'** que contienen cada una de las conclusiones extraídas en la última sesión de consulta realizada. En cada uno de estos archivos se almacena la conclusión junto con la cadena de reglas aplicadas para llegar a ella (traza de demostración). Como se puede llegar a más de una conclusión en cada consulta, y es el operador quien debe aplicar su criterio para elegir el diagnóstico más apropiado, el nombre del archivo contiene un espacio para un número de secuencia de hasta 7 (dígitos).

El siguiente es un ejemplo de una conclusión que un operador podría recibir del sistema, durante una sesión de consulta:



Las soluciones son distinguidas entre sí según la conclusión final a la que se arribó. En la lista superior (**Soluciones**) se muestran las distintas conclusiones obtenidas. Como cada una de ellas puede tener más de una *cadena de inferencia* o *traza de demostración* por medio de la cual se obtiene la conclusión, es posible seleccionar cuál de todas ellas se desea ver (**Solución Actual**).

Una vez seleccionada la solución particular, en la lista **Motivo** se puede explorar la cadena de inferencias a través de la cual se llegó a la solución actual de la conclusión que se está examinando.

La explicación de una solución, además de agregados para facilitar su lectura, se compone de tres componentes básicos, los cuales son: **la conclusión**, **la veracidad de la conclusión** y **el nombre de la regla aplicada**.

La explicación adopta una estructura jerárquica con el fin de asemejarse al árbol de reglas aplicadas para llegar a la conclusión final. Esto le sirve al operador de la Central, quien puede focalizar lo más importante (la conclusión) y si eventualmente necesitara una explicación de por qué se llegó a la misma, puede solicitarlo desplegando el subárbol correspondiente a las condiciones que se dieron para que se arribe a dicha conclusión. Por ser una estructura de árbol, el operador puede realizar este procedimiento hasta el nivel deseado.

En el ejemplo, la conclusión final a la que se llegó es **Debe ejecutarse POEA 6 B**, la veracidad de la conclusión es: **0.9** y la regla aplicada para llegar a dicha conclusión es **'POEA 06- B: Alguna señal de Bloqueo Turbina'**.

Descendiendo en la jerarquía, se encuentra que los antecedentes de la regla **'POEA 6 B'** cuya validez llevó a su conclusión son: **Hay Baja Presion Aire Instrumentacion**, **Hay Señal Baja Presion Aire** y **Hay Posterior pérdida de aire de instrumento**. Para cada uno de estos antecedentes, se muestra de la misma forma que para la conclusión principal, su veracidad, la regla aplicada y sus respectivos antecedentes.

Las *hojas* del árbol de conclusiones las forman, generalmente, los antecedentes cuya veracidad se comprueba a partir de la información proveniente de las lecturas del reactor.

Es importante mencionar que tanto la primer lista (Soluciones), como la lista implícita formada por cada una de las soluciones actuales (lista de las distintas cadenas de demostración), se encuentran ordenadas. El orden adoptado fue el siguiente.

Dado que por cada solución presente en la primer lista puede haber muchas cadenas de demostración (donde cada una de estas últimas llegan a la misma conclusión pero eventualmente con un grado de veracidad distinto), el criterio que se adoptó fue ordenar descendentemente, para cada solución presente en la primer lista, sus distintas

cadenas de demostración por el grado de veracidad correspondiente a cada una de éstas. Luego, se obtiene el máximo grado de veracidad (es decir, el grado de veracidad de la primer cadena de demostración) y ese valor de verdad es el que se le asigna a la solución presente en la primer lista que agrupa a dichas cadenas de demostración. Este proceso se repite para cada solución presente en la primer lista de Soluciones. Por último, se ordena descendentemente la primer lista de soluciones por el grado de verdad asignado a cada una de éstas. Por lo tanto, cuando se selecciona una solución de la primer lista, las *n* distintas cadenas de demostración correspondientes estarán ordenadas descendentemente desde la *solución actual 1* hasta la *solución actual n* por su grado de veracidad.

Notar que no se podría haber adoptado un criterio de orden total, ya que los datos a ordenar poseen una estructura de árbol. Sin embargo, con el orden parcial especificado, se logró que el operador de la Central, quien recibe los datos demostrados, pueda evaluar lo más importante primero: es decir, que aquellas conclusiones con mayor grado de veracidad sean observadas primero tanto en la lista de soluciones como también en la lista de cadenas de demostración.

VI.B. Problemas encontrados y soluciones propuestas

VI.B.1. Representación del conocimiento basada en reglas / basada en casos

La representación de la información correspondiente a los POEAs de la CNE se encuentra realizada en base a un esquema de reglas. Previamente a esta decisión ya se había descartado la posibilidad de utilizar otros esquemas de representación como las Redes Semánticas y los Frames ya que proveen un poder expresivo que excede a los fines del desarrollo encarado.

Se analizó, sin embargo, la posibilidad de utilizar un esquema basado en casos para la aplicación. La idea de un esquema basado en casos consiste básicamente de una base de datos de casos de eventos anormales conocidos junto con el estado completo de todas las señales de la central. De esta manera, el sistema experto contrastaría (utilizando lógica difusa) el estado de las señales al momento de la consulta contra la base de casos que se encuentra almacenada. Posteriormente se evaluarían los grados de equivalencia de cada uno de los casos analizados con el estado en que se encuentra la planta para llegar así a la conclusión de la ocurrencia de un evento anormal.

La objeción principal contra la posibilidad de implementar una base de casos para la aplicación provino del hecho de que no se cuenta en CNE con un historial suficientemente grande de eventos producidos. Debido a esto, se decidió descartar esta opción de implementación.

VI.B.2. Tiempos de proceso de inferencia

El hecho de que el sistema experto se encuentre desarrollado para evaluar la presencia de eventos anormales en una central nuclear lleva a pensar inicialmente en restricciones de tiempo muy duras para su funcionamiento.

La aclaración inicial por parte de los responsables del proyecto en N.A.S.A., consistió en que el objetivo para este desarrollo inicial no exigiría las restricciones de tiempo correspondientes a las de un verdadero sistema de tiempo real.

De esta forma, el desarrollo de la aplicación se llevó a cabo con una orientación abocada al tratamiento de los problemas de representación del conocimiento y de funcionamiento del proceso de razonamiento del motor de inferencias.

VI.B.3. Archivo de lecturas

El objetivo final planeado por N.A.S.A. para la aplicación es el de instalarla en el mismo ambiente de trabajo de la sala de control de la CNE. De esta manera, la aplicación obtendría directamente de la planta la información sobre los estados de los distintos controles y llevaría a cabo el proceso de detección de eventos en base a dicha información.

Debido a que N.A.S.A. ha planeado inicialmente esta aplicación para los fines del testeo de su utilidad y eficiencia, no se encontraba durante su desarrollo el entorno necesario en la central para poder testear su funcionamiento 'in-situ'.

De esta manera, se convino entre el grupo de desarrollo y el personal responsable del proyecto en N.A.S.A. en acordar una interface de comunicación entre el sistema experto y la aplicación que se dedique a recolectar la

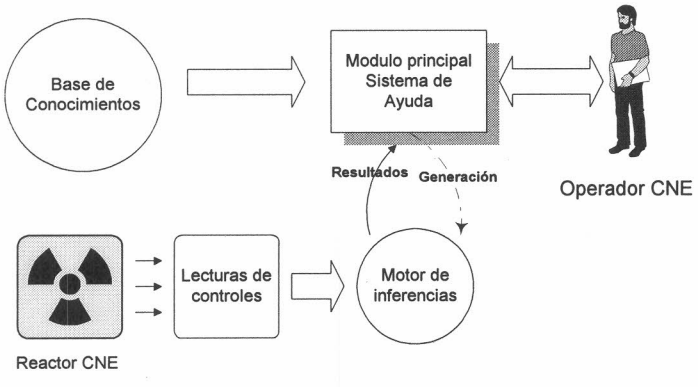
información de los controles, de manera que esta interface sirva tanto para la aplicación en funcionamiento en una PC stand-alone en etapa de desarrollo y testeo como cuando finalmente se la implemente en la sala de control de la CNE.

La interface acordada consiste de una serie de archivos de texto ASCII que contienen la información de los estados de los controles de la planta.

Como existe la posibilidad de que se acumulen varias lecturas de los controles de la planta mientras se está evaluando otra llegada previamente, se estableció que los archivos que contienen los estados de las variables se nombren en forma secuencial de modo que se mantenga una cola de lecturas entrantes para analizar.

VI.B.4. Estructura del programa Prolog

El sistema se encuentra dividido en dos sub-aplicaciones. La primera se encuentra implementada utilizando un lenguaje procedural siendo su objetivo el servir de interface con los expertos y los usuarios de la central.



La segunda sub-aplicación se ha desarrollado en el lenguaje Prolog. Su función es realizar las inferencias respecto al estado de la planta utilizando la información recolectada en base a los controles y reglas especificados oportunamente por el experto.

El programa Prolog, generado previamente a comenzar el proceso de verificación, consta de tres partes: el estado de las señales del reactor, el motor de inferencias y las reglas.

Mientras que el motor de inferencias es un código fijo en el programa y su estructura no depende de las reglas generadas, las reglas Prolog se generan en base a las reglas que se encuentran almacenadas en la base de conocimientos al momento de ordenar la verificación. El estado de las señales depende del archivo de lectura correspondiente a cada instante de demostración.

POEAS.ARI (Programa de inferencias generado)

Estados de las señales
Motor de inferencias
. . . **Reglas** . . .

Es importante señalar que se ha logrado resolver satisfactoriamente el problema del manejo de un gran volumen de información de monitoreo por parte del motor de inferencias, y la devolución de los resultados al programa llamador. Esto incluye tanto una base de conocimientos extensa como un archivo de lecturas con gran cantidad de señales. Versiones preliminares del sistema experto tenían una limitación con respecto a la cantidad de reglas que se podían especificar en la base de conocimientos y con respecto a la cantidad de señales disponibles en los archivos de lectura. El problema era el mal funcionamiento del sistema, que también se observaba cuando, a pesar de haber pocas reglas y/o pocas señales, se obtenían muchas soluciones (es decir, un árbol extenso de soluciones). Dado que las soluciones son devueltas al programa llamador con una estructura de árbol (donde cada nodo es una regla aplicada), se optó devolver nodo por nodo en cada solución (al programa llamador). Luego, este último (el programa llamador) rearma en forma interna el árbol, y lo formatea para presentarlo al operador (en caso de haber alguna solución). Esta técnica permitió que el sistema finalmente implementado y entregado funcione perfectamente con bases de conocimientos relativamente grandes, de aproximadamente cien reglas y más de docientas señales, que son, de una manera muy aproximada, lo que va a ser necesario en la implementación final en la(s) computadora(s) de la C.N.E.

VI.B.5. Acceso a las señales en el programa Prolog

El acceso a los estados de las señales que participan de las reglas se realiza poniendo a disposición de la memoria del programa Prolog los valores de todas las señales cuyo valor se reporta en el archivo de lecturas que se está analizando.

De esta forma, el motor de inferencias posee disponibles en todo momento en su memoria de trabajo los valores de todas las señales con que debe operar.

El programa Prolog recibe los valores de todas las señales a través de una estructura de arreglo a la que accede secuencialmente en cada momento en que necesita acceder al valor de alguna de ellas.

VI.C. Creación de la Base de Conocimientos

La creación de la base de conocimientos del trabajo aquí presentado involucró varias sesiones con los responsables de N.A.S.A. . Algunas de ellas fueron documentadas y se pueden encontrar en el apéndice correspondiente (minutas). El vocabulario necesario para interpretar a los expertos fue adquirido en el transcurso mismo de las entrevistas, y consultando la bibliografía suministrada oportunamente por los responsables del proyecto en N.A.S.A. . La base de conocimientos finalmente acumuló más de cien reglas, e involucró más de docientas cincuenta señales provenientes del reactor para dar cuenta de los catorce POEAs con que cuenta actualmente la C.N.E.

En general, la creación de la base de conocimientos involucra una tarea dinámica con el experto. Esta tarea pertenece al área general del problema de *adquisición del conocimiento*, la cual es un tema de un campo de estudio relativamente nuevo conocido como *ingeniería de conocimiento*.

La adquisición del conocimiento generalmente es un cuello de botella en el desarrollo de sistemas expertos. Típicamente, el ingeniero de conocimiento debe "sentarse" con un experto, en varias sesiones largas, y extraer el conocimiento del dominio que posee el experto para usar en el sistema experto.

El ingeniero de conocimiento debe tener cierto entendimiento del área del experto para que pueda entender e interpretar correctamente el vocabulario del mismo. El experto usualmente no será capaz de proveer heurísticas generales para la resolución de problemas. Generalmente, deben presentarse problemas de ejemplo y las heurísticas deben explicarse en el proceso de resolución del experto. Las heurísticas iniciales que son recolectadas tienden a ser incompletas y no muy correctas por lo general. Por lo tanto, el proceso de recolección de conocimiento es incremental y coincide con el desarrollo y pruebas del sistema experto.

Las entrevistas llevadas a cabo con el personal de N.A.S.A. tuvieron las características descriptas anteriormente. Se repasaron en varias oportunidades las heurísticas de detección de eventos; se fue adquiriendo un vocabulario específico del dominio; se acudió a distintos ejemplos para aclarar los procesos de resolución del problema. El proceso de recolección del conocimiento a través de las entrevistas también fue un proceso incremental que coincidió con el desarrollo de las primeras versiones del sistema experto.

Para desarrollar un sistema experto, debe encontrarse uno o más expertos pacientes y bien predispuestos para el proceso de adquisición del conocimiento. Las sesiones iniciales con el experto son un proceso intenso de búsqueda de la información relevante y cómo acceder a ella. Las sesiones siguientes tienen por objeto superar las incompletitudes en el conocimiento del sistema y los errores que puedan haberse deslizado. El conocimiento incorrecto puede aparecer debido al hecho de que el ingeniero de conocimiento debe hacer muchas interpretaciones e incluso desechar cierta información al adquirir las heurísticas del experto. Si muchos expertos contribuyen con conocimiento, ocasionalmente también se contradicen uno al otro en cierto grado. El ingeniero de conocimiento es responsable de despejar las inconsistencias que puedan surgir en la base de conocimientos.

En el caso del trabajo aquí presentado, el grupo de desarrollo tuvo necesariamente que hacer ciertas interpretaciones sobre la lógica que hay detrás de los diagramas lógicos correspondientes a la documentación de los POEAs. Esto fue necesariamente documentado en las minutas que se elaboraban luego de cada reunión, y después de enviar las mismas a los responsables de N.A.S.A., éstos enviaban la minuta elaborada por ellos, con las respuestas y correcciones necesarias en el caso que hiciera falta (a veces este intercambio era a la inversa). De esta forma, el grupo de desarrollo podía gradualmente ir despejando las dudas, y lograr un proceso incremental de adquisición del conocimiento. Esto sirvió para no dejar inconsistencias en la base de conocimientos final.

VI.D. Pruebas del sistema

Para las pruebas del sistema se utilizó una PC 486 DX4 de 100 Mhz con 16 Mb de RAM.

Estas pruebas fueron realizadas sobre 16 (dieciseis) archivos de lecturas. Las primeras 13 (trece) (numeradas de 2 a 14) corresponden a la demostración de los poeas 2 a 14, respectivamente. Las últimas tres pruebas (numeradas de 15 a 17) son lecturas que demuestran más de un POEA simultáneamente.

La tabla siguiente resume la información obtenida de las pruebas. Cada archivo de lectura se puede encontrar en el directorio donde se encuentra el sistema (instalado a partir del diskette suministrado con este trabajo).

Las celdas en blanco corresponden a una misma lectura, por lo tanto el valor es el mismo que la celda superior inmediata.

Número Lectura	Cantidad de Señales	POEA(s) demostrado(s)	Tiempo (en segundos)	Cantidad de Soluciones	Mínima Veracidad	Máxima Veracidad
2	1	2	62	1	1.00	1.00
3	4	3-C2	61	1	1.00	1.00
4	22	4	153	108	0.59	1.00
5	17	5-0	71	8	1.00	1.00
6	23	6A	68	4	1.00	1.00
		6B		8	0.90	0.90
7	9	7	77	9	1.00	1.00
8	6	8	67	1	0.99	0.99
9	16	9	65	1	0.99	0.99
10	15	10	64	1	0.99	0.99
11	11	11	64	4	0.96	0.96
12	21	12	65	1	0.90	0.90
13	8	13-0	70	3	0.91	0.91
14	8	14-4	69	1	0.97	0.97
15	44	2	162	1	1.00	1.00
		3-C2		1	1.00	1.00
		4		108	0.59	1.00
		5-0		8	1.00	1.00
16	55	6A	121	4	1.00	1.00
		6B		16	0.90	0.90
		7		9	1.00	1.00
		8		32	0.99	0.99
		9		32	0.99	0.99
17	64	10	77	2	0.99	0.99
		11		8	0.96	0.96
		12		2	0.90	0.90
		13-0		3	0.91	0.91
		14-4		1	0.97	0.97

Una de las primeras cosas que se pueden observar a partir de la información volcada en esta tabla, es que el tiempo de procesamiento no es directamente proporcional con la cantidad de señales. Se ve que a pesar de que la lectura 15 tiene 44 señales, el tiempo que se consume para procesar esta lectura es mucho mayor (más del doble) al tiempo que se consume en procesar la lectura 17, que cuenta con más señales. Esto se debe a la forma en que las reglas se disparan por el motor de inferencias, y la manera que las reglas se encadenan a partir de los datos disponibles (señales). Lo mismo pasa con la relación cantidad de señales/cantidad de soluciones obtenidas. Esta relación tampoco es proporcional, por los mismos motivos que antes: las señales disponibles son las que disparan determinadas reglas, y puede darse el caso de que

pocas señales se encuentren involucradas en muchas reglas o viceversa. De esta manera, vemos que por ejemplo, para la lectura 15, las 44 señales disponibles participan en más reglas que las 64 señales disponibles en la lectura 17.

VII. CONCLUSIONES

En base a consideraciones sobre la naturaleza del proyecto, los resultados obtenidos y los posibles proyectos futuros que pueden tenerse en cuenta, se presenta a continuación una breve evaluación sobre el trabajo llevado a cabo.

VII.A. Trabajo realizado

El presente trabajo involucró el desarrollo de un **sistema experto** que utiliza un **lenguaje difuso** diseñado para el dominio particular del problema de detección de **eventos anormales** en una central nuclear. El sistema implementado finalmente fue tomado por NASA como una primera etapa en el camino hacia la implementación definitiva. Desde este punto de vista, el trabajo aquí presentado constituye un prototipo. El mismo, una vez terminado, se convirtió en una alternativa viable para NASA.

El desarrollo del proyecto involucró distintas disciplinas.

Por un lado, se aplicó **metodología estructurada** y **prototipación** para el desarrollo de aspectos tales como la interface de usuario, diseño de la base de datos y módulos de actualización de la base de datos. En otro aspecto se llevó a cabo un proceso de **ingeniería de conocimiento** que incluyó una serie de entrevistas con los expertos.

Por último, fue necesaria la investigación, el estudio y el **desarrollo formal de un lenguaje** que adoptara las características deseadas de los requerimientos de representación del conocimiento estudiado a través de las entrevistas.

La alternativa de utilizar un sistema experto para NASA en este proyecto fue desarrollada y justificada en el presente trabajo. Se llegó a la conclusión de que en este caso, un sistema experto es posible, justificado y apropiado.

La fundamentación teórica del presente trabajo constituye una base principal donde descansa la validez de las inferencias hechas por el sistema experto. Desde la especificación misma de las reglas del sistema, hasta la acción que consiste en la elaboración de respuestas en el lenguaje propio del operador, el sistema experto se encuentra justificado por elementos formales explicados a lo largo del presente documento.

VII.B. Logros principales

VII.B.1. Automatización del proceso de detección y diagnóstico de fallas

El principal objetivo del presente trabajo ha sido automatizar un sistema manual de detección y diagnóstico de fallas. En esta etapa de automatización se han tenido en cuenta la documentación escrita, el conocimiento de los expertos y la experiencia adquirida por los operadores.

El nuevo sistema colabora con los operadores en las fases más críticas del proceso de diagnóstico en la central. Éste último tiene lugar precisamente en el momento en que la falla se manifiesta de alguna forma y los operadores deben reconocer su origen de la manera más rápida posible.

El sistema logrado presenta un entorno de trabajo que permite a los operadores obtener la información correspondiente a la eventual falla presentada sin tener que consultar los manuales. De esta manera se acelera el crítico proceso de mitigación.

VII.B.2. Representación del conocimiento

Una parte esencial del estudio del problema y del desarrollo de su solución ha tenido que ver con el análisis del esquema de representación del conocimiento adoptado para los POEAs. El estudio ha tenido en cuenta los distintos elementos presentes en la documentación.

Las entrevistas realizadas con los expertos de NASA han significado un aporte fundamental para el desarrollo de la etapa de relevamiento del conocimiento. Se comprobó que el lenguaje propio del experto corresponde a un lenguaje enriquecido por vocabulario específico del dominio del problema. Esto hizo necesario profundizar sobre ciertos conceptos que los expertos manejaban con mucha soltura, y que eran totalmente ajenos al equipo de desarrollo. En este sentido, las grabaciones y posterior reproducción y análisis de las entrevistas con los expertos han constituido una herramienta sumamente útil para desentrañar el lenguaje propio y la terminología utilizada en el dominio del problema por los expertos de NASA.

Finalmente, y como consecuencia del estudio de las distintas alternativas, se ha reelaborado el esquema de representación de la información de los POEAs, llevando la misma a un sistema de reglas basadas en el conocimiento expresado en el sistema anterior, más el conocimiento aportado por los expertos.

VII.B.3. Simulación

La estructura de diseño del sistema experto permite que se lo pueda alimentar con datos de control provenientes de distintas fuentes. La información puede consistir de datos genuinos del reactor o de datos creados artificialmente para simular comportamientos anómalos o recrear eventos pasados.

De esta forma es posible utilizar al sistema para recrear escenarios pasados o hipotéticos con el objetivo de poder evaluar su eficiencia en la detección de eventos anormales. En este proceso pueden intervenir los expertos realizando cambios en las reglas almacenadas para testear posibles mejoras en el conocimiento incorporado al sistema.

VII.B.4. Lenguaje de primer orden con características difusas

El lenguaje difuso creado cuenta con una fundamentación teórica. Para lograr ésto se ha partido de la creación de un lenguaje de primer orden basado en el dominio que fue enriquecido sintácticamente con elementos difusos para darle mayor expresividad semántica. De esta manera se consiguió una aproximación mayor al dominio del problema.

Luego el lenguaje fue derivado, con la fundamentación adecuada, a una forma clausal para adaptarlo a las caracteristicas computacionales del sistema. Esta forma del lenguaje fue finalmente la utilizada para que el experto de la CNE exprese su conocimiento en el sistema. A su vez, se encontró un método para convertir un programa expresado en este lenguaje clausal en un lenguaje clásico de programación lógica, de modo de poder implementar el motor de inferencias.

El resultado de esta etapa fue un lenguaje caracterizado por aspectos específicos que lo hacen adecuado para ser utilizado en el marco del problema de la CNE.

VII.C. Trabajos futuros

Si bien es cierto que NASA planteó una etapa posterior al presente trabajo, también se reconoció la utilidad y eficacia del sistema obtenido. Esto refleja el hecho de haber planteado una solución viable y correcta para el problema de la CNE. Sin embargo, surge del trabajo realizado la posibilidad de continuarlo en distintas direcciones con el fin de mejorar el mismo.

VII.C.1. Sistema de tiempo real

El sistema experto desarrollado para NASA no cuenta con las características de un sistema de tiempo real. El sistema no ajusta la respuesta de sus procesos de inferencia a restricciones de tiempo.

Dada la visión de los responsables de NASA con respecto al sistema experto considerándolo como prototipo, se han tratado principalmente los problemas relacionados con representación del conocimiento y definición de un lenguaje de especificación.

Los aspectos relacionados con el cumplimiento de la entrega de resultados por parte del sistema dentro de lapsos de tiempo preestablecidos constituyen un marco interesante para tener en cuenta en un desarrollo posterior. En definitiva, esta última característica se constituye en un requerimiento esencial para poder considerar la puesta en operación del sistema experto dentro de la CNE.

VII.C.2. Entorno gráfico de simulación y estudio

Aprovechando la característica ya mencionada del sistema sobre la posibilidad de utilizarlo para simular eventos posibles o recrear eventos ya ocurridos, es interesante considerar la opción de explotar al máximo este camino.

Un objetivo interesante para seguir podría ser la realización de un entorno gráfico de simulación que permita combinar las características de inferencia ya desarrolladas en el sistema con una interface que refleje visualmente los componentes del reactor que están siendo monitoreados.

VII.C.3. Implementación final en las computadoras de la C.N.E.

Sería necesario que la versión final del sistema experto contara con una función de autoverificación de sus condiciones operativas, y que alerte al operador cuando no esté disponible ya sea por fallas internas o externas. Esta última función la debería cumplir aún cuando no se está frente a un evento anormal. Sin embargo, cuando la falta de datos es parcial, el sistema debería continuar en servicio si el operador lo desea.

Por último, sería conveniente jerarquizar ciertos parámetros del sistema experto, por ejemplo las reglas o las mismas señales del reactor. Con una jerarquización relativa de las señales del reactor, se asegura la integridad de los elementos combustibles del reactor y se evita la emisión de material radiactivo al exterior. La razón se puede explicar con un ejemplo. Si se detecta un evento anormal que involucra ciertos parámetros (señales) de una jerarquía determinada, y durante el proceso de mitigación otro parámetro crítico (con una jerarquía superior) supera su valor de referencia, entonces el sistema debería informar ésto al operador y reemplazar la página del diagrama lógico correspondiente (en ejecución) por la nueva página (correspondiente al nuevo evento anormal detectado). Esta jerarquización (de señales) esta hecha y se puede encontrar en [BAT/2]. Se debe considerar que se acaba de describir lo que debería hacer el sistema global finalmente implementado en la C.N.E. (sistema de ayuda al operador + sistema de seguimiento y mitigación de los eventos anormales).

VII.C.4. Mejora en el lenguaje de interface Hombre - Máquina.

Una mejora posible a la versión implementada del sistema experto en este sentido podría tener lugar en el ámbito de la visualización de la base de conocimientos como en la explicación de los resultados de las inferencias. Tanto en la interface de mantenimiento del sistema, que comprende la especificación de reglas, predicados, señales, tipos y modificadores, como en la interface de explicación de los resultados obtenidos por el sistema, debería poder lograrse un nivel de aproximación mayor al lenguaje natural. Esto ayudaría tanto al experto en la definición de las reglas del sistema como al operador que obtendría una explicación más comprensible de los eventos anormales detectados por el sistema.

En el caso de la visualización de la base de conocimientos, se pueden analizar medios alternativos para poder ver las reglas incorporadas a través de una especificación más cercana al lenguaje natural del experto. En el caso de la explicación de los resultados, es posible estudiar formas para que el usuario pueda especificar la sintaxis de dichas explicaciones, incorporando sus propias estructuras sintácticas y los vocablos a utilizar para expresar los grados de certeza de las conclusiones.

VIII. BIBLIOGRAFÍA - REFERENCIAS

	Título	Autor(es)	Fuente/Editorial
[BAT/1]	Sistemas Computarizados de Ayuda a la Operación	J. Batistic, M. Goldstein, H.A. Palamidessi, R. Pereira da Luz	Comisión Nacional de Energía Atómica
[BAT/2]	Propuesta de Sistema de Ayuda al Operador Aplicado a Eventos Anormales en la Central Nuclear Embalse	J. A. Batistic	Comisión Nacional de Energía Atómica
[BAT/3]	Embalse NGS: Abnormal event procedures development lifecycle	J. A. Batistic	Comisión Nacional de Energía Atómica
[CAR/87]	Sistemas Expertos y Representación del Conocimiento	Raúl Jorge Carnota y Alberto Daniel Teszkiewicz	EBAI
[HAL/91]	Fuzzy Expert Systems	Lawrence O. Hall, Abraham Kandel, Mordechay Schneider	CRC Pr, 1991
[KLI/95]	Fuzzy Sets and Fuzzy Logic - Theory and Applications	George J. Klir and Bo Yuan	Prentice Hall, 1995
[LEE/71]	Some Properties of Fuzzy Logic	Richard C. T. Lee and Chin-Liang Chang	Information and Control 19, 1971
[LEE/72]	Fuzzy Logic and the Resolution Principle	Richard C. T. Lee	Journal ACM, Vol. 19, 1972
[LEU/88]	Fuzzy Concepts in Expert Systems	K. S. Leung and W. Lam	Computer, September 1988
[LLO/87]	Foundations of Logic Programming	John Wylie Lloyd	Springer Verlag, 1987
[NEB/91]	Sistemas Expertos	Nebendahl, Dieter	Siemmens - Marcombo, 1991
[NIL/87]	Principios de Inteligencia Artificial	Nilsson, Nils J.	Diaz de Santos, Edición Castellano, 1987
[ZAD/65]	Fuzzy Sets	L. A. Zadeh	Information and Control 8, 1965
[ZAD/75]	The Concept of a Linguistic Variable and its Application to Aproximate Reasoning - III	L. A. Zadeh	Information Sciences 9, 1975
[ZAD/77]	Theory of Fuzzy Sets	L. A. Zadeh	Encyclopedia of Computer Science and Technology, 1977
[ZAD/88]	Fuzzy Logic	L. A. Zadeh	Computer, April 1988

INDICE