UNIVERSIDAD DE DEUSTO

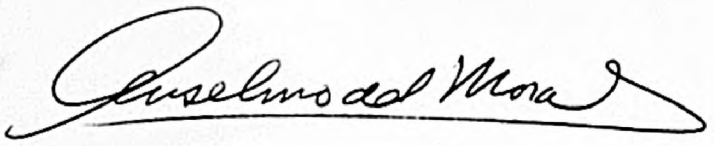
Facultad de Informática

VALIDACION DE BASES DE CONOCIMIENTOS EN SBC PARA DIAGNOSTICO

Tesis doctoral presentada por Verónica Canivell Castillo

Dirigida por el Dr. Anselmo del Moral Bueno

El Director



El Doctorando

Bilbao, Abril 1991

AGRADECIMIENTOS

Al Dr. Anselmo del Moral Bueno, Director de esta Tesis Doctoral, que me ha introducido en el estudio de la Inteligencia Artificial y me ha guiado durante el desarrollo de este trabajo.

A Emilio Uriarte Asteinza, médico, por su apoyo y sus valiosos consejos sobre las referencias médicas.

Al Dr. Ulises Cortés, profesor titular de la U.P.C., que me ha proporcionado ideas y sugerencias muy importantes.

A Enric Plaza, Carlos Sierra y Beatriz López, del Centro de Estudios Avanzados de Blanes, que me han proporcionado parte del material de trabajo y me han enseñado el funcionamiento del sistema MILORD.

A la Dra. Asunción Barredo Fuentes, que me ha dado numerosas ideas y consejos.

A Beatriz Galán, Iciar Rodriguez, Manuel Sesé, Pablo Ochoa de Retana y Ramón Puga, por su colaboración en este trabajo.

Y a todos aquellos que con su ayuda y apoyo han hecho posible la realización de esta Tesis.

RESUMEN

El objetivo de este trabajo es presentar un estudio sobre la validación de Bases de Conocimientos para SBCs en el área de diagnóstico. Se examinan los diferentes tipos de validación y los métodos empleados actualmente para llevarlas a cabo. Además de proponen tres modelos de medidas de evaluación para BC dinámicas, así como su desarrollo y los resultados obtenidos al aplicarlos a la BC PNEUMONIA utilizada por el sistema MILORD.

ABSTRACT

The objetive of this Thesis is to propose a method for validating knowledge bases for knowledge-based systems used for diagnosis purposes. First, the different types of existing validation methods are analysed together with a study on the necessity for such validation. Three models of measuring the evaluation of dynamic knowledge bases are then proposed, as well as developed for the knowledge base PNEUMONIA used in the MILORD system.

A mis padres,

INDICE

CAPITULO 1

INTRODUCCION

Hoy en día, los sistemas basados en el conocimiento (SBC) están adquiriendo gran importancia en campos tales como la industria, la medicina, la investigación científica, etc.. Las razones de ello son:

- El uso de conocimiento no estructurado.

- La necesidad de manejar una gran cantidad de datos, informaciones, etc..

- La capacidad de los SBC para entrenar a nuevos expertos.

- La necesidad de aprender más sobre un tema mientras se organiza el conocimiento para desarrollar los SBC.

- Las reducciones de costos que proporcionan los SBC.

- El deseo de captar el conocimiento colectivo para que no se pierda con los cambios individuales.

Así, los SBC constituyen actualmente un objeto de investigación en plena expansión.

Una de las áreas donde hoy día se están realizando grandes avances es el campo de la decisión médica, sea ésta diagnóstica, terapéutica o preventiva. No obstante hay que poner de relieve que aunque se están obteniendo resultados muy interesantes, algunos médicos no tienen gran confianza en estos sistemas, por la dificultad que conlleva evaluar la calidad de un diagnóstico propuesto por "una máquina". Sin embargo, son numerosos los casos de diagnóstico o de terapéutica en los que un médico necesita una ayuda, la cual, podría ser aportada por la informática, y más concretamente, por los sistemas basados en el conocimiento.

Desde hace algunos años, los investigadores que trabajan en el desarrollo de sistemas orientados a dar ayuda en la toma de decisiones en medicina, utilizan métodos de Inteligencia Artificial (IA) que han conducido a la realización de los SBC en este área. Estos sistemas se interesan por la representación del conocimiento médico y por los procesos en los que se utiliza dicho conocimiento, y efectúan deducciones simples y sólidas que hacen su "razonamiento" comprensible para los usuarios médicos. Las aproximaciones mediante IA intentan dejar atrás los límites ligados a los métodos puramente estadísticos, y presentan elementos capaces de resolver y razonar los problemas planteados en los cuales puede incluso intervenir información

incompleta. Por otro lado, estos sistemas son interactivos, permitiendo intercambios de información entre el usuario y la máquina en un lenguaje cercano al del usuario que los utiliza. Esto constituye un argumento a favor de una mejor aceptación por parte de los usuarios no especialistas en informática, los cuales pueden pedir justificaciones de los consejos propuestos por el sistema.

Los campos de aplicación de los SBC en el ámbito médico son numerosos. Por ejemplo:

- La ayuda al diagnóstico: el sistema asume el papel de consultor utilizado una y otra vez por el médico con el objeto de enfocar o contrastar los posibles diagnósticos derivados de los datos obtenidos en los exámenes de los pacientes.

- La ayuda a la terapéutica: este tipo de aplicación es mucho más solicitado aunque son muchos los que expresan cierta inquietud en este campo. Los consejos ofrecidos por estos sistemas pueden ayudar al médico a prescribir con certeza en función del diagnóstico, pero también adaptar la prescripción al caso concreto del paciente, ajustar la posología, conocer y tener en cuenta las contraindicaciones, las sinergías, los efectos potencializadores, etc..

- La enseñanza. Es un dominio privilegiado. Para enseñar una materia, hay que conocerla. Un SBC de ayuda a la enseñanza debe contener obligatoriamente un subsistema de ayuda al diagnóstico y/o terapéutica. Además debe adaptarse al nivel de conocimiento y "de inteligencia" de su interlocutor. Por lo tanto, debe poseer conocimientos de pedagogía.

- La ayuda a la investigación. Además de lo expuesto, los SBC pueden elaborar hipótesis, conocidas o desconocidas. En este último caso, las hipótesis se convierten en temas de investigación.

- La ingeniería biológica y médica: la interpretación "automática" de las exploraciones funcionales, los signos fisiológicos, teniendo en cuenta los datos específicos del paciente, etc..

Los principales problemas planteados actualmente en la elaboración y utilización de estos sistemas son:

- La representación del conocimiento: Las estructuras a adoptar deben ser a la vez generales, potentes y fáciles de utilizar.

- La calidad y el soporte teórico del motor de inferencia: La lógica proposicional actualmente muy utilizada, la lógica de primer orden restringido, de primer orden completo, etc..

- El interface hombre-máquina: El lenguaje natural de fácil uso, complejo de analizar, gran consumidor de recursos informáticos, etc..

- La elección del razonamiento exacto o la intervención del razonamiento aproximado. En el último caso se plantean las cuestiones de si hay que introducir métodos numéricos y quedarse con el cálculo simbólico, o bien de si el razonamiento por analogía en cálculo simbólico es "la solución" del tratamiento correcto de lo incierto.

Es preciso también referirse a los problemas de ética que pueden plantear los SBC. El acto médico se descompone esquemáticamente en cuatro etapas:

- El examen del paciente.

- La elaboración del diagnóstico.

- La prescripción terapéutica.

- El seguimiento de la evolución.

Los SBC aportan una ayuda clara en la realización del acto médico, pero no examinan al paciente. Este examen es el más humano de los componentes del acto médico.

El uso de los SBC se sitúa, previsiblemente, entre el de las enciclopedias bien utilizadas y el del especialista humano. El hombre, el médico, de ninguna manera puede ser reemplazado por la máquina; se trata, por tanto, de un temor sin fundamento. Pero se puede decir que los hábitos médicos van a evolucionar. El uso de la informática médica y de los SBC influirá de forma clave en esta evolución. Por consiguiente, corresponde a los hombres, en particular a los médicos, asimilar los métodos nuevos y utilizarlos en el momento oportuno.

Una vez planteada la necesidad de los SBC como ayuda para los especialistas médicos en sus tareas de emisión de diagnóstico, prescripción terapéutica, etc., se va a considerar un aspecto de vital importancia para el desarrollo de los SBC: la validación de su base de conocimientos (BC).

Concretamente en medicina, dados los problemas de desconfianza que plantea la difusión de un SBC en la comunidad médica, esta validación es difícil y debe apoyarse

en una metodología irreprochable. Si se desea que estos sistemas sean útiles y accesibles, hay que poder asegurar que el médico que siga sus consejos proporcione la mejor terapéutica posible.

Hay que reconocer que, hasta el momento actual, existen muy pocos sistemas validados según una metodología rigurosa, bien codificada y que presente garantías suficientes. De forma general, las prestaciones de los sistemas actuales son evaluadas en circunstancias particulares que introducen factores de pérdida de objetividad que falsean la apreciación de las posibilidades del sistema ante casos reales, así como el nivel de pericia de la base de conocimientos. Lo más frecuente es que estos sistemas sean evaluados a partir de casos de pacientes obtenidos de un único servicio hospitalario, por usuarios que han participado en la elaboración del sistema y de su base de conocimientos, según criterios que no son suficientes para asegurar su aceptabilidad.

Además de la validación de la BC, es preciso considerar también la evaluación del motor de inferencia, el interface hombre-máquina, el modo de representación del conocimiento y sus facilidades de actualización, la calidad de las explicaciones suministradas, las conclusiones intermedias que intervienen en el razonamiento del sistema, el tiempo de respuesta del sistema, la duración de una consulta, los aspectos ergonómicos como por ejemplo la utilización de

terminales gráficos, etc. Es muy difícil establecer si cierto enfoque de la representación del conocimiento o cierto mecanismo de inferencia es más adecuado o no, que otro para implementar una aplicación dada. La distinción entre la naturaleza del conocimiento y las posibilidades que ofrece la representación adoptada no siempre aparece de forma clara en los SBC. Las restricciones y los límites de los formalismos que utilizan no siempre son explícitos [FIES84]. Finalmente todos los criterios de evaluación no tienen la misma importancia según las aplicaciones, las motivaciones y los servicios esperados por los diferentes tipos de usuarios. Esquemáticamente, estos pueden clasificarse en tres grupos principales:

- El usuario destinatario no es un experto del dominio y utiliza el sistema para obtener rápidamente un consejo sagaz sobre un problema difícil de resolver. Por ejemplo, un especialista en medicina general interroga a un sistema para obtener una opinión terapéutica sobre el tratamiento de un diabético. Este tipo de usuario toma muchas decisiones y dedica a cada una de ellas un tiempo mínimo. Por ello, la facilidad de acceso al consejo, su rapidez, claridad, la forma que utiliza para expresarlo y las posibilidades de diálogo son criterios que deben considerarse y que son de primera importancia para que el sistema sea aceptable.

- El usuario es un experto que comprueba el conocimiento del sistema y eventualmente puede modificarlo o incrementarlo. En este caso serán más importantes las ayudas al dominio del conocimiento puesto que el experto toma pocas decisiones y todas necesitan tiempo y reflexión profundas.

- El usuario no es un experto y conoce mal el dominio. Por ejemplo, es el caso de un enfermo que utiliza el sistema para vigilar su enfermedad o un estudiante que lo utiliza con fines de aprendizaje. Las explicaciones deben poder adaptarse al nivel del interlocutor. De igual modo, en este caso hay que subrayar la importancia del vocabulario utilizado en las preguntas y en la formulación de éstas, así como en las respuestas.

Un estudio de validación de un sistema debe, por tanto, abordar la evaluación de la base de conocimientos y de los diferentes componentes del sistema. Esta Tesis se va a centrar en los aspectos referentes a la validación de la BC, y el objetivo que se persigue es proponer tres modelos de medición para la evaluación de bases de conocimientos. Para ello, se ha realizado un estudio basado en el sistema de diagnóstico MILORD [SIER89], y en una de sus aplicaciones a la medicina, PNEUMONIA [VERD89a], que se utiliza para diagnosticar neumonías adquiridas extrahospitalariamente. Con esta investigación se pretende mejorar el conocimiento sobre el comportamiento de dicho sistema así como de otros similares dentro del área de diagnóstico médico.

La Tesis se ha estructurado de la siguiente forma:

En primer lugar, se da una visión global de los conceptos fundamentales de los sistemas basados en el conocimiento.

En segundo lugar se trata la validación de SBC, realizando un repaso de los diferentes tipos de validación existentes.

En tercer lugar se estudian las características fundamentales del sistema MILORD, gracias al cual se ha podido llevar a cabo este trabajo, así como de la base de conocimientos PNEUMONIA, que contiene la información con la que se trabaja.

A continuación se pasa a considerar el estado actual de la evaluación de las bases de conocimientos.

Finalmente se presentan tres modelos de medidas de evaluación aplicables a las bases de conocimientos dinámicas y sus aportaciones en el campo de la evaluación como mejora para un mayor conocimiento del comportamiento de los SBC.

En la tabla 1.1. se representan los puntos de atención del trabajo de investigación y los capítulos en que se encuentran.

TRABAJO DE INVESTIGACION	CAPITULO
Visión global de los Sistemas Basados en el Conocimiento	2
Validación de los SBC	3
Estudio del sistema MILORD y de la BC PNEUMONIA	4
Evaluación de las BC	5
Modelos de Medidas de Evaluación aplicables a las BC	6

Tabla 1.1.

Estructura de la tesis

CAPITULO 2

FUNDAMENTOS DE LOS SISTEMAS BASADOS EN EL CONOCIMIENTO

2.1.- INTRODUCCION: CARACTERIZACION Y OBJETIVOS

Los sistemas basados en el conocimiento se encuentran entre las aplicaciones informáticas más excitantes surgidas en los últimos años. Estos sistemas utilizan la pericia o experiencia de uno o más individuos para resolver gran variedad de problemas, tales como diagnóstico de fallos de equipo, diagnóstico de enfermedades, diseño de nuevos equipos, etc..

Los sistemas basados en el conocimiento se distinguen de los programas convencionales en varios aspectos importantes. Aunque ninguna de las características citadas a continuación faltan en su totalidad en otros tipos de software bien diseñados, todas ellas juntas describen una clase de programas diferente.

Un sistema basado en el conocimiento es un programa que:

a) Razona con conocimiento de un dominio específico que puede ser expresado simbólica y matemáticamente.

b) Usa métodos específicos del dominio que son heurísticos (plausibles) y a la vez pueden ser algorítmicos (ciertos).

c) Funciona como los especialistas en sus áreas de problema.

d) Hace que se entienda tanto lo que sabe como las razones para sus respuestas.

e) Es flexible.

Las características (a) y (b) - razonamiento simbólico y métodos heurísticos - definen los sistemas basados en el conocimiento como programas de Inteligencia Artificial (IA).

El punto (d) - un sistema explicando sus razonamientos - y el (e) - la provisión de flexibilidad - son citados y realizados menos frecuentemente que los anteriormente vistos. Se incluyen aquí para indicar su importancia en el diseño y en la implantación de cualquier SBC. Son importantes cuando se está diseñando el SBC y cuando se usa.

Durante el diseño y la implantación, no todo el conocimiento necesario está al alcance, porque ni siquiera los especialistas pueden decir lo que un programa necesita saber. Debido a esto, los sistemas basados en el conocimiento se construyen de forma incremental. Es importante entender la terminología común a los especialistas y usuarios. Entender la base de conocimientos estática permite decidir lo que se necesita incorporar para mejorar la representación. Entender las dinámicas del razonamiento es importante para decidir qué cambiar. La flexibilidad es necesaria para permitir que los cambios se realicen fácilmente. Las explicaciones ayudan a los diseñadores y usuarios finales a entender las razones para las conclusiones obtenidas. Esta capacidad es especialmente importante cuando los usuarios finales aceptan la responsabilidad legal, moral y financiera de las acciones consecuentes de las recomendaciones de los programas.

2.2.- FUNDAMENTOS

Todos los programas de IA, incluidos los sistemas basados en el conocimiento, parten de una buena representación y utilización del conocimiento. El modelo conceptual en la resolución de problemas, en IA, es la búsqueda. Aunque de forma inmediata es clara y simple, esta fórmula no indica cómo lograr una solución eficiente y

segura. El número de posibles soluciones puede ser astronómico. Así pues, la consideración exhaustiva de las alternativas está fuera de lugar. La mayoría de los sistemas basados en el conocimiento usan heurísticas para evitar la búsqueda exhaustiva. Para áreas de problemas en las que los expertos poseen un conocimiento más eficiente y seguro resulta razonable que lo que el experto sepa sea codificado para uso del programa. Esta es una de las presunciones fundamentales de la Ingeniería del Conocimiento, el arte de construir SBCs extrayendo el conocimiento de los expertos [HAYE83].

Desde el punto de vista funcional, un SBC puede considerarse como un intermediario entre el experto humano que le ha transmitido su saber, y el usuario humano, que hace la consulta con la intención de resolver un problema, y que a la vez va a adquirir parte de la destreza del experto.

Sin embargo, es en la estructura donde los SBC se diferencian más claramente de los programas tradicionales. Los SBC se construyen mediante módulos, entre los que podemos distinguir fundamentalmente los representados en la Figura 2.1. y que a continuación se mencionan:

- *La Base de Conocimientos*, donde se representa el conocimiento.

- *El Motor de Inferencia o estructura de control*, que decide qué reglas de la BC van a ser activadas según la información que el usuario vaya suministrando.

- *La Base de Hechos*, que es la memoria de trabajo del sistema.

- *El Módulo de Interacción con el Usuario*, que facilita el diálogo entre el hombre y la máquina.

- *El Módulo de Actualización de la Base de Conocimientos*, que se utiliza para poder modificar, añadir o eliminar reglas de la BC con el objeto de mantenerla al día.

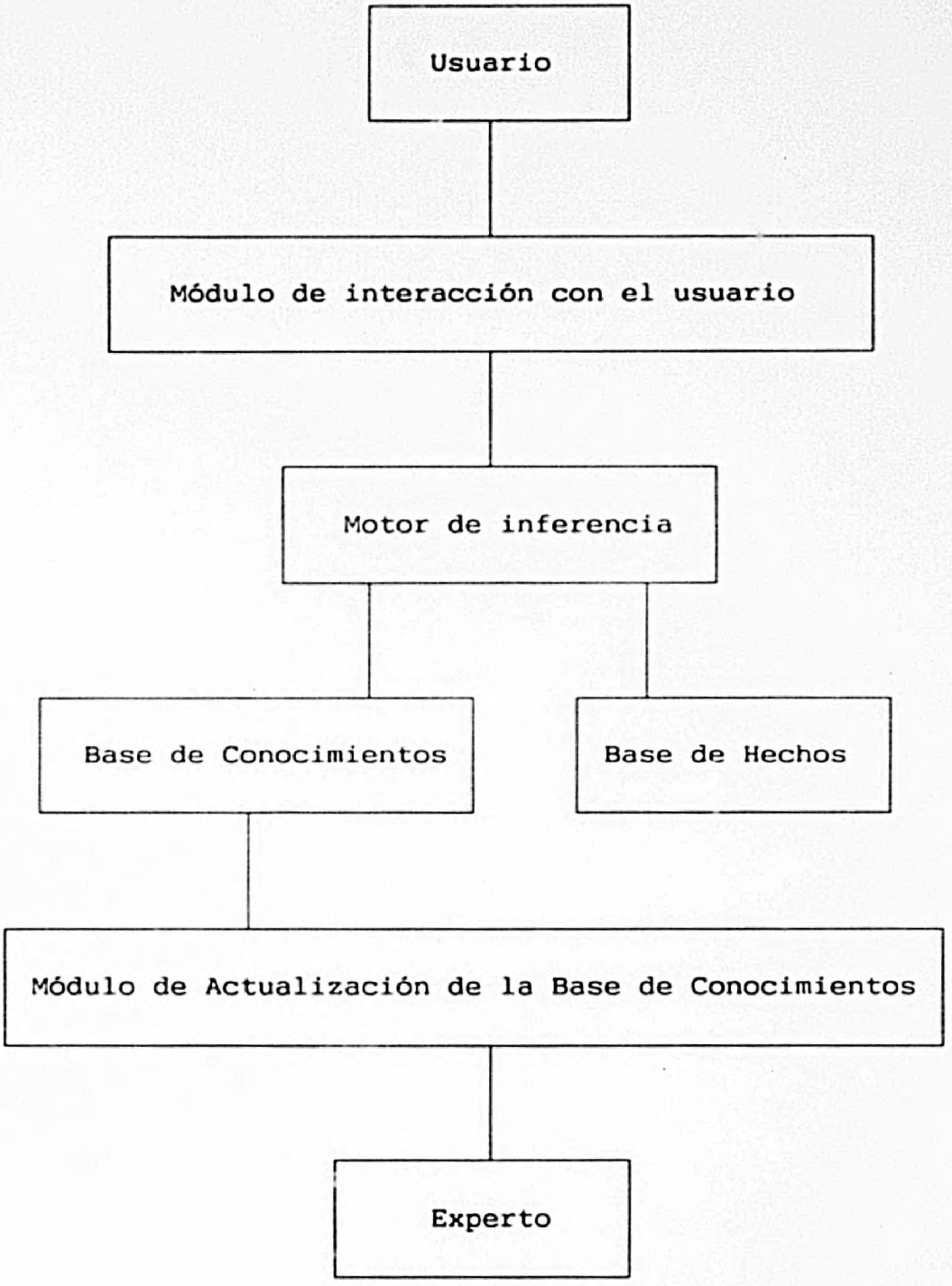


Figura 2.1.

Módulos de un SBC.

2.2.1.- REPRESENTACION

Los sistemas basados en el conocimiento usan el conocimiento específico del dominio de aplicación, con un sistema de control ejercido por el motor de inferencia.

En los primeros años de la IA, John McCarthy consideró que el principio más importante en la representación del conocimiento era el conocimiento declarativo [McCa58]. Más adelante, Winograd coincidió con esta teoría [WINO72]. Este principio implica que el conocimiento debe ser codificado explicitamente en un programa inteligente, de manera que permita que los otros programas razonen sobre él.

Algunos procedimientos arbitrarios escritos en FORTRAN o LISP, por ejemplo, no se pueden explicar o editar mediante otros programas (aunque pueden ser compilados y ejecutados), mientras que esto sí es posible en los que están escritos con una representación del tipo atributo-valor, estructuras de registros, u otras estructuras de datos más complejas.

Bajo cierta interpretación se puede considerar que una base de conocimientos es una base de datos (BD). Las diferencias esenciales entre ambas son la flexibilidad y complejidad de las relaciones.

Las investigaciones actuales que sobre IA y BD, (denominados en ocasiones sistemas de BD expertos [KERS89] [PARS89]), están reduciendo estas diferencias. Una base de conocimientos requiere un modelo de organización además de una estructura de datos para su implantación. Estas dos partes, en conjunto, constituyen la representación del conocimiento en un programa de IA.

Los elementos del conocimiento necesarios para la resolución de problemas pueden organizarse alrededor de los objetos primarios (conceptos) del área del problema, o alrededor de las acciones (incluyendo relaciones de inferencia) entre esos objetos. Por ejemplo, en medicina se puede pensar primero sobre las relaciones entre las manifestaciones y las enfermedades, y las relaciones entre las enfermedades y las acciones terapéuticas; y segundo, sobre los conceptos así asociados. Este modelo, está centrado en el conocimiento que permite las inferencias y acciones a desarrollar - el cómo conocer. Por otra parte, también se podría organizar el conocimiento médico primero alrededor de la clasificación de las enfermedades y de la clasificación de sus manifestaciones; y segundo, alrededor de las reglas de inferencia que relacionan manifestaciones y enfermedades. Este segundo modelo, se concentra en lo que podría denominarse qué es conocimiento. Estas dos perspectivas conceptuales se conocen como modelos de representación del conocimiento guiados por los datos y guiados por la meta.

Para cada tipo de representación, es posible identificar una unidad primitiva y una acción primitiva. La unidad primitiva, en el caso de representaciones **guiadas por los datos**, es el hecho (por ejemplo, la temperatura de congelación del agua es 0º C). Los hechos o datos primitivos se emparejan con las reglas del tipo "SI...ENTONCES...", disparándose entonces la parte acción de la regla.

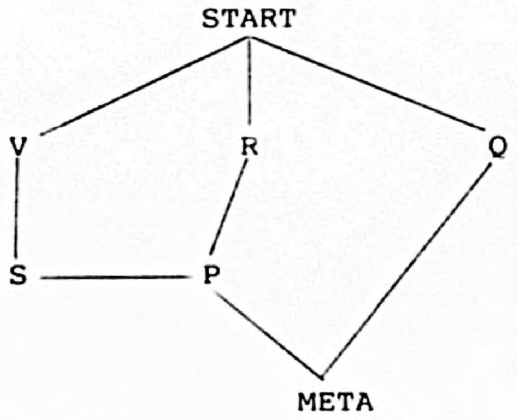
A continuación se presenta una búsqueda simple sobre un conjunto de producciones expresadas como implicaciones del cálculo de proposiciones. La estrategia de resolución de conflictos es realmente simple, se dispara la regla activada que se ha disparado menos recientemente o nunca y en el caso de empates, se elige la primera. La ejecución se para cuando se ha llegado a la meta.

CONJUNTO DE PRODUCCIONES
1.- P ∧ Q —> META
2.- R ∧ S —> P
3.- W ∧ R —> Q
4.- T ∧ U —> Q
5.- V —> S
6.- START —> V ∧ R ∧ Q

Traza de la ejecución:

NUMERO ITERACION	MEMORIA DE TRABAJO	CONJUNTO CONFLICTO	RE-GLA
0	START	6	6
1	START, V, R, Q,	6, 5	5
2	START, V, R, Q, S	6, 5, 2	2
3	START, V, R, Q, S, P	6, 5, 2, 1	1
4	START, V, R, Q, S, P, META	6, 5, 2, 1	PARADA

ESPACIO EXPLORADO DURANTE LA EJECUCION



Recíprocamente, en un sistema del tipo guiado por la meta, la unidad primitiva es dicha meta, emparejándose en este caso, la meta con la parte derecha de las reglas hasta llegar a los hechos primitivos.

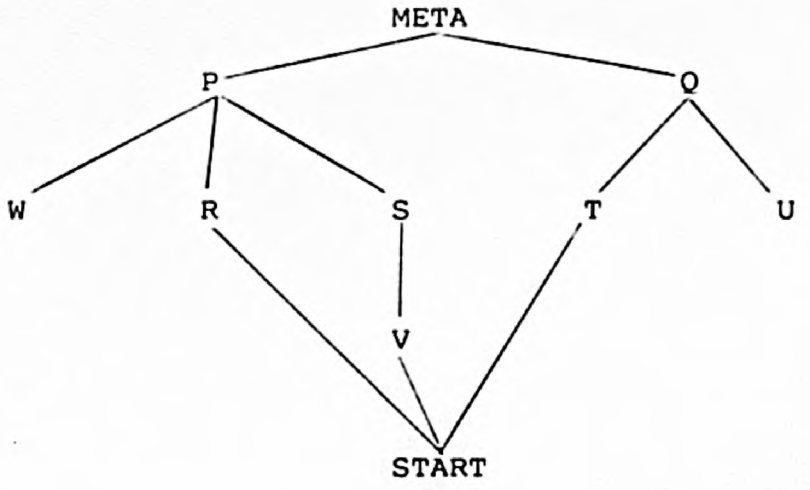
Por ejemplo:

CONJUNTO DE PRODUCCIONES
1.- P ∧ Q —> META
2.- R ∧ S —> P
3.- W ∧ R —> P
4.- T ∧ U —> Q
5.- V —> S
6.- START —> V ∧ R ∧ Q

Traza de la ejecución:

NUMERO ITERACION	MEMORIA DE TRABAJO	CONJUNTO CONFLICTO	RE-GLA
0	META	1	1
1	META, P, Q	1, 2, 3, 4	2
2	META, P, Q, R, S	1, 2, 3, 4, 5	3
3	META, P, Q, R, S, W	1, 2, 3, 4, 5	4
4	META, P, Q, R, S, W, T, U	1, 2, 3, 4, 5	5
5	META, P, Q, R, S, W, T, U, V	1, 2, 3, 4, 5 6	6
6	META, P, Q, R, S, W, T, U, V, START	1, 2, 3, 4, 5 6	PA RA DA

ESPACIO EXPLORADO POR LA EJECUCION



Un objeto puede verse como un conjunto estructurado de hechos - Minsky popularizó el uso de objetos de representación (llamados frames) para IA [MINS75]. Smalltalk [GOLD83] fue uno de los primeros lenguajes que mostraron el poder de los objetos como estructuras y el poder de un entorno de programación con gráficos integrados. Muchos de los SBC comercializados contienen un componente orientado al objeto [STEF86].

La aplicación de la acción primitiva en representaciones guiadas por los datos se denomina a menudo disparar una regla, es decir, si la parte condición de un regla es cierta en una situación, entonces las acciones especificadas en la parte acción de la regla se llevan a cabo (se dispara la acción).

Este estilo de programación comenzó llamándose sistemas de producción, popularizado por los trabajos de Newell en los años 60 pero propuestos, por primera vez, por Post en 1943 [POST43].

Dado que la programación de sistemas basados en reglas, a veces, involucra hacer deducciones, se ha discutido que las distintas formas de la lógica valdrían para su uso a la hora de razonar en los SBC. Los sistemas simples han usado lógica proposicional, los más complejos han usado lógica de predicados de primer orden y se está investigando la utilización de lógicas de orden superior para expresar las

relaciones entre creencias, relaciones temporales, necesidad e información incierta [McDE80]. Igualmente, algunos sistemas incorporan la lógica difusa propuesta por Zadeh [ZADE65].

En las representaciones **guiadas por la meta**, la acción primitiva se denomina **enviar un mensaje**: si se necesita realizar una acción, (por ejemplo, se necesita el valor de un atributo), se envía una petición al objeto que puede llevar a cabo la acción (por ejemplo, calcular o concluir el valor). El resultado es la ejecución de una acción arbitraria, que puede incluir la obtención de inferencias. El estilo de la programación orientada al objeto fue definido por Hewitt [HEWI77].

En términos de estructuras de datos, los objetos son muy parecidos a los registros. Cada objeto tiene un número de campos fijos. Sin embargo, difieren de los registros en que se les pueden añadir nuevos campos durante su proceso. Los objetos se dividen en dos tipos: instancias y clases. Las instancias son individuos o casos particulares de un dominio. Las clases representan conjuntos de instancias que definen las características de los individuos. Las clases se organizan generalmente en jerarquías, llamadas taxonomías, según las diferentes relaciones. Las relaciones más comunes son la especialización, es decir subclase ó relación "ES-UN", y la relación "PARTE-DE". Los sistemas orientados al objeto

permiten que se codifiquen relaciones arbitrarias pero también proporcionan ayudas eficientes para una o dos relaciones específicas como las anteriormente mencionadas.

Los mecanismos de representación deben ser lo suficientemente expresivos para indicar, breve y claramente, qué es conocimiento y cómo conocer. Esta expresividad tiene implicaciones a la hora del diseño y en el tiempo de ejecución. Uno de los problemas claves para los diseñadores de SBC es la gestión de la complejidad. Los mecanismos de representación pobres fuerzan a los diseñadores a codificar la información de forma poco clara, lo cual origina dificultades para la ampliación y explicación del comportamiento de los SBC. Los mecanismos de representación que permiten una compilación y estructura de conocimiento eficientes reducen las necesidades de tiempo y memoria en la ejecución.

Por ejemplo, un lenguaje orientado al objeto permite establecer la información una sola vez en una clase abstracta y accederla (por herencia) desde un gran número de subclases. Un mecanismo de representación que no permita esto, obliga a los diseñadores a afrontar la complejidad de repetir la misma información muchas veces. Esto puede producir inconsistencias, y dificultades para actualizar la información. También implica un mayor coste de memoria. En el

momento de ejecución, cada una de las informaciones codificadas por separado debe considerarse individualmente, retardando por tanto el proceso de ejecución.

Los modelos guiados por los datos y guiados por la meta son los dos extremos de un espectro de posibilidades de representación. Los programas de IA contemporáneos a menudo usan modelos de representación heterogéneos (por ejemplo, asociando reglas simples con la expresividad de los objetos).

Los esquemas de representación extensibles facilitan el desarrollo incremental de los SBC, que es necesario cuando no existe una especificación completa de si el problema o el conocimiento requieren una solución. Cuando se añaden conceptos, atributos y relaciones, el diseñador no debe ser forzado a recodificar partes sustanciales del conocimiento codificadas con anterioridad.

La experiencia demuestra que las representaciones modulares y declarativas son muy útiles para los SBC. Algunos tipos de información son más difíciles de codificar en los modelos guiados por los datos, mientras que otros tipos de información son más difíciles de codificar en los modelos guiados por la meta. Por ejemplo, una secuencia de acciones es más difícil de codificar en un modelo guiado por los datos. Lo mismo ocurre cuando la información es esencialmente estática, tal como descripciones estructurales o causales. Por otra parte, las representaciones guiadas por la meta no

tienen ningún mecanismo de inferencia salvo el de herencia (aunque los pueden soportar y muchos sistemas comerciales tienen un componente orientado a las reglas).

Resumiendo, no hay una respuesta a la pregunta: "¿qué método de representación es el mejor?". Los SBC contemporáneos usan gran variedad de métodos pero tienden a integrarlos en una configuración uniforme. Como los sistemas cada vez son más complejos, será más difícil mantener una visión uniforme.

2.2.2.- RAZONAMIENTO

Para utilizar apropiada y eficientemente los datos de una base de conocimientos y alcanzar algún propósito, tal como diagnosticar una enfermedad, se requieren métodos de inferencia. Lógicamente hablando, las dos reglas de inferencia más utilizadas en la resolución de problemas son *modus ponens* (Si A implica B y conocemos A, entonces inferimos B) y *modus tollens* (si A implica B y conocemos no B, entonces inferimos no A). La primera algunas veces se denomina "regla de la cadena", porque las inferencias se pueden encadenar en una secuencia de deducciones:

A

A ---> B

B ---> C

C ---> D

Por tanto, D

Además de estas dos reglas simples, se suelen usar reglas de cuantificación (por ejemplo, si todas las As son Bs y x es A, entonces x es B). Unas pocas reglas de inferencia simples dirigiendo la resolución de un problema, una base de conocimientos con muchos hechos especiales y las relaciones sobre el área del problema, pueden proporcionar la destreza en la que se basa la representación.

Algunos SBC (por ejemplo, aquellos escritos en PROLOG) usan "demostradores de teoremas" para determinar la verdad o falsedad de las proposiciones y para asociar variables con el fin de obtener proposiciones ciertas. Otros utilizan sus propios intérpretes para incorporar más de lo que los demostradores de teoremas pueden proporcionar - capacidades para controlar el orden de las inferencias, razonamiento estratégico y el razonamiento ante la incertidumbre. La mayoría de los SBC basados en reglas han utilizado intérpretes de reglas especializados que no se basan directamente en ninguna lógica.

Existe una gran parte del conocimiento de "sentido común" al que las lógicas convencionales no pueden dar cabida. Esto se debe a que el conocimiento de sentido común implica típicamente mucha incertidumbre. Por ejemplo, considérese la sentencia:

"Si el coche es grande y bonito,
entonces probablemente será caro"

Esta declaración está llena de incertidumbre ya que según el individuo que haga la declaración, las características 'grande', 'bonito', 'caro' pueden tener diferentes grados de certeza. Las lógicas convencionales no admiten grados de verdad y por tanto, no pueden utilizarse para representar y razonar con declaraciones de este tipo.

La lógica borrosa o difusa [ZADE65] puede adaptar tal incertidumbre mediante un enfoque "semántico" [ZADE83], que es bastante distinto del utilizado en la lógica convencional.

Los intérpretes y los compiladores eficientes de PROLOG, han llegado a ser válidos sólo recientemente [COLM83]. Sin embargo, también reflejan la necesidad de estilos de inferencia más flexibles y el control de las estrategias que dirigen el orden de las inferencias.

2.2.2.1.- CONTROL DEL ORDEN DE LAS INFERENCIAS Y PREGUNTAS

Desde un punto de vista lógico, el orden en el que se derivan los nuevos hechos es irrelevante, si se consideran todas las consecuencias lógicas de los hechos iniciales. Por razones programáticas, los SBC a menudo necesitan ser selectivos sobre qué hechos considerar y qué consecuencias perseguir. Por ejemplo, a menudo el espacio y el tiempo están limitados y podría ser importante desarrollar una línea de razonamiento a seguir. De esta forma, los SBC pueden estar organizados en torno a tres modelos de razonamiento diferentes: hacia delante (forward), hacia atrás (backward) y oportunístico (opportunistic).

El razonamiento hacia adelante desde los datos hasta las conclusiones, se utiliza cuando el coste o la inconveniencia de datos concurrentes es bajo y hay relativamente pocas hipótesis que explorar. Un sistema de encadenamiento hacia adelante comienza a partir de un conjunto de hechos y va manejando las reglas para obtener al final, ciertas conclusiones. Las condiciones de fin varían desde finalizar con la primera hipótesis plausible hasta finalizar solamente cuando no se puedan añadir más conclusiones.

Emparejar las premisas de todas las reglas en una base de conocimientos con cada situación nueva puede ser muy caro cuando hay muchas reglas y muchas situaciones nuevas creadas

por inferencia de hechos nuevos. Las reglas suelen contener variables que se pueden asociar de forma que sea posible emparejar esas premisas con alguna situación.

Hay intérpretes de reglas, tal como OPS5 [BROW85], que proporcionan mecanismos para la compilación de la reglas y procedimientos para emparejarlos. Los grupos de reglas (llamados "set", "tasks" or "control blocks" - conjunto, tarea o bloque de control-) también se usan para controlar el foco o centro de atención de los SBC con el propósito de interaccionar con usuarios de forma más comprensible. Por ejemplo, en un sistema médico, se ayudará a los usuarios a entender mejor los razonamientos si las peticiones de datos están organizadas en grupos de reglas que:

a) realizan diagnósticos de enfermedades;

b) se centran en el historial del paciente o en los exámenes del laboratorio;

c) recomiendan terapias.

El razonamiento hacia atrás no requiere que todos los datos relevantes estén disponibles al comenzar las inferencias, trabajando con hipótesis. Funciona mejor cuando el usuario proporciona muchos datos y cuando es relevante el orden en el que se solicitan. Un ejemplo clásico es el MYCIN. Tiene un sistema de encadenamiento hacia atrás que comienza

con una hipótesis y va haciendo preguntas sobre qué hechos (premisas de las reglas) serían necesarios para verificarla, con el fin de conocer si la hipótesis es cierta. Algunos de estos hechos son conocidos porque se dan como datos iniciales, otros pueden conocerse preguntando al usuario e incluso otros se obtendrán considerándolos hipótesis y desarrollando un proceso de encadenamiento hacia atrás. Las condiciones para finalizar pueden variar desde acabar con la primera hipótesis cierta (o suficientemente cierta) que se encuentre hasta finalizar cuando todas las posibles hipótesis han sido exploradas.

El razonamiento oportunistico combina algunos elementos de los dos anteriores. Es útil cuando el número de posibles inferencias es muy grande, cuando no es probable que surja una única línea de razonamiento y cuando el sistema de razonamiento debe responder al recibir nueva información. Conocidos los nuevos datos, se pueden realizar nuevas inferencias y obtenidas las nuevas conclusiones, se pueden hacer nuevas preguntas sobre los datos específicos. De esta forma, un sistema de razonamiento oportunistico ayuda a extraer unos pocos datos de una masa confusa. El elemento clave de un sistema tal es una agenda de acciones con un esquema asociado que permite decidir sobre las acciones a llevar a cabo (por ejemplo, elegir las reglas a aplicar, aplicarlas hacia delante o hacia atrás y qué objeto será el centro de atención). Tales decisiones son difíciles de tomar en los sistemas de encadenamiento hacia adelante y hacia

atrás. Un prototipo con éxito basado en este modelo es el sistema HASP [NII82], en el que los datos acústicos de los sensores localizados en el océano proporcionan información sobre el tipo y la situación de los barcos. Como la recepción de datos no es uniforme, se deben revisar las hipótesis. En cada revisión, aparecen nuevas ambigüedades que se resuelven volviendo a procesar los datos antiguos o buscando nuevas señales.

2.2.2.2.- UTILIZACION DE ESTRATEGIAS EXPLICITAS

Los tres modelos de razonamiento vistos son estrategias primitivas que necesitan refinamiento y coordinación para que constituyan una estrategia de decisión compleja como la necesaria para un sistema de diagnóstico médico. Para los sistemas basados en el conocimiento es importante representar explícitamente dichas estrategias, sobre todo cuando éstas pueden cambiar. Las meta-reglas del MYCIN (solución a este problema a finales de los años 70), representan las estrategias de razonamiento sobre el conocimiento como reglas [BUCH84]. Se diferencian de las otras reglas del dominio del conocimiento en que se hace referencia a estas reglas en alguna de sus premisas o conclusiones:

SI contexto médico

Y hay reglas que mencionan el hecho A

y hay reglas que mencionan el hecho B

ENTONCES reglas que mencionan el hecho A antes que a otros.

Las estrategias también pueden representarse como un conjunto de pasos a llevar a cabo [CLAN86]. La representación explícita de las estrategias del conocimiento puede mejorar los sistemas para explicar su propio comportamiento.

2.2.2.3.- RAZONAMIENTO CON INCERTIDUMBRE

El razonamiento ante la duda es esencial en áreas de problema fuera de la lógica y las matemáticas en las que la información es incompleta y/o errónea. En medicina, por ejemplo, es raro tener una certeza absoluta de poseer todos los datos o de la precisión de los datos. En los SBC se usan varios métodos que tratan de la incertidumbre procedente de los datos, las asociaciones de datos y conclusiones, y de la combinación de las dos anteriores.

Los métodos principales de razonamiento con incertidumbre son:

1.- **Abstracción:** Asume que la duda es pequeña y se puede ignorar. Este método es muy simple y eficiente de usar. Sin embargo, hay muchos problemas que requieren mayor precisión para estimar la incertidumbre.

2.- **Teorema de Bayes:** Usa las probabilidades anteriores y posteriores a un hecho para representar los datos y las asociaciones inciertas, y calcula nuevas probabilidades a partir de ellos [GORR70]. Este método se basa en un formalismo sólido pero requiere datos de frecuencias o estimaciones subjetivas para las muchas combinaciones de sucesos.

3.- **Lógica difusa:** Representan la incertidumbre de proposiciones tales como "Pepe es alto" con una distribución de valores; entonces, razonan sobre combinaciones de distribuciones [ZADE79]. Se basa en conceptos lingüísticos normales. Es operacionalmente más complejo puesto que la incertidumbre se propaga a través de distribuciones de valores.

4.- **Tablas de criterio:** Asigna categorías y pesos a las cláusulas de las reglas basándose en su importancia relativa al obtener conclusiones; entonces se elegirá una conclusión si en cada categoría hay un número suficiente de cláusulas ciertas [KULI82]. Es un mecanismo simple y muy rápido

operacionalmente; sin embargo, al perder graduación entre las categorías, disminuye el poder expresivo de razonamiento en algunas áreas complejas del problema.

5.- **Factores de certeza (FC):** Asigna un número a cada proposición y a las asociaciones entre ellas, que representan probabilidades o conjuntos de probabilidades; entonces, se aplican fórmulas para determinar los factores de certeza por creencias inferidas [BUCH84]. Estos cálculos se han utilizado frecuentemente y han demostrado tener una representación formal en teoría probabilística. Como se basan en medidas de creencias, los efectos de añadir nuevas relaciones o cambiar los factores de certeza son difíciles de predecir.

2.2.3.- DESARROLLO DE LA BASE DE CONOCIMIENTOS

En la construcción de SBCs, el proceso de adquirir el conocimiento en una BC, se considera como un "cuello de botella" [HAYE83]. Generalmente, este proceso requiere dos personas o equipos: un experto, cuyo conocimiento se va a reflejar parcialmente en la BC, y un ingeniero del conocimiento, que entreviste al experto para organizar su conocimiento en estructuras de datos que se utilizarán en los programas. La representación del SBC resultante depende de este proceso que es difícil y lleva mucho tiempo. Esto se

acentúa con el hecho de que el diseño de la BC a menudo integra el conocimiento de varios expertos porque considera que el de uno solo podría pasar por alto algunos supuestos implícitos.

Gran parte del proceso de ingeniería del conocimiento es ingeniería pura. Hay varios asuntos fundamentales involucrados en los pasos del proceso.

1.- Durante el primer paso, el ingeniero de conocimiento debe emparejar las características del problema propuesto con las de los métodos de solución conocidos. Desafortunadamente, no hay una buena clasificación de problemas o métodos de solución ni un buen criterio para decidir si hay emparejamiento.

2.- El segundo paso fundamental es la programación preliminar en la que se construyen una serie de prototipos experimentales, primero como un concepto total y posteriormente como fracciones del conocimiento del experto que muestra la parte del problema que puede resolverse con dicho conocimiento, codificados en un entorno específico. Dos temas a considerar son:

a) Formular una estructura conceptual precisa, incluida la terminología, que permita añadir conocimiento incrementalmente.

b) Interaccionar eficientemente con el experto para obtener información del problema relevante para el SBC.

3.- El tercer paso fundamental en el desarrollo de la base de conocimientos, es ampliar la competencia del sistema. Este paso es el que más tiempo requiere (varias personas-años) pero es relativamente llevadero si los dos pasos anteriores se han realizado bien. Una dificultad es la anticipación de las características de los usuarios finales y su contexto de utilización. Otra es decir qué hechos y relaciones son relevantes y cuales no, para la representación del sistema y el entendimiento del contexto. Los modelos competentes para tomar esta decisión (y para la ingeniería del conocimiento en general) se denominan **dirigido por el modelo** (model directed) y **dirigido por el caso** (case directed). En el primero, la base del conocimientos se desarrolla siguiendo las líneas del modelo, o teoría, del área del problema. En el último, se desarrolla respondiendo a los errores que se muestran en los casos del test de resolución. Los ingenieros de conocimiento usan ambos ya que ninguno es completamente adecuado por sí solo. Sea cual sea la combinación de métodos de desarrollo que se usen, no hay un criterio de finalización claro. Estos problemas se agravan con las continuas ediciones y modificaciones de la base de conocimientos.

4.- El último paso del proceso es la ingeniería del software que asegure que el sistema se ajusta al entorno del usuario final, a sus necesidades,..., etc. La dificultad en este paso

es diferente para cada SBC. Para finalizar debe mencionarse el hecho de que el éxito de una aplicación requiere algo más que el desarrollo de una base de conocimientos.

2.2.4.- AUTOEXPLICACION

Uno de los criterios que definen a los sistemas basados en el conocimiento es su habilidad para explicar sus conclusiones. Las explicaciones se centran en mostrar la línea de razonamiento, típicamente una secuencia de reglas, que se dirige a una conclusión particular. Normalmente, esto se haría en lenguaje natural [BUCH84]. El usuario podría preguntar al sistema cuestiones del tipo: "¿cómo ...?", "¿por qué ...?".

En los primeros SBC, el interface era un subconjunto del lenguaje natural casi exclusivamente, pero el poder y bajo coste de los gráficos han derivado en la tendencia hacia interfaces gráficos, por ejemplo, el sistema STEAMER usado para entrenar al personal naval [HOLL84] [SLAG90]. Los sistemas contemporáneos a menudo incluyen ambos tipos de interfaces, por ejemplo, el DRILLING ADVISOR SYSTEM [RAUC86].

Las líneas de razonamiento se pueden mostrar como gráficos que permiten usar la interacción para explorar las posibles líneas de razonamiento alternativos, por ejemplo, el sistema GUIDON-WATCH [RICH85]. Quizás esto prueba el hecho de que las facilidades de explicación son como las facilidades de depuración (debugging) de los programas sofisticados y a menudo se utilizan como tales. Como todo buen sistema de depuración, permite a los programadores y usuarios examinar el sistema en funcionamiento en términos de alto nivel, más que en términos de instrucciones máquina a bajo nivel que se están ejecutando. Las bases de conocimiento actuales tienden a realizar ajustes de registros a la hora de almacenar los datos [SMIT85], lo cual se puede utilizar para aumentar las explicaciones. Se está investigando la posibilidad de que el propio SBC use esta información.

El término "explicación" también puede incluir la inspección de la base de conocimientos estática. Las representaciones orientadas al objeto y las facilidades gráficas sofisticadas aumentan la capacidad de un dominio especializado para comprender lo que se ha codificado [SMIT87].

Podríamos decir que también un usuario de un programa convencional de FORTRAN puede examinar la base de conocimientos del programa, pero solamente del modo en que se ha codificado el programa (normalmente, con un editor de texto). Sin embargo, un hecho que diferencia a los SBC de los

programas convencionales es su capacidad para ser interrogados en tiempo de ejecución: mientras que un programa convencional sólo puede examinarse estáticamente, un SBC puede examinarse también dinámicamente.

2.2.5.- VALIDACION

Se puede juzgar a un sistema basado en conocimientos desde muchas dimensiones. Las tres más importantes son: la computacional, la psicológica "mirar y sentir" (look and feel) y la de actuación (performance). Las características computacionales incluyen velocidad, memoria requerida, extensibilidad y portabilidad. Las psicológicas incluyen facilidad de uso, autoexplicacón, naturalidad y ayuda en tiempo real. Las de actuación incluyen el alcance de la competencia, el porcentaje de soluciones positivas y negativas falsas, y el tiempo o dinero ahorrados. Algunos tipos de validación incluyen evaluaciones de la base de conocimientos estática mientras otras implican observar el programa en funcionamiento, por ejemplo, facilidad de utilización o estadísticas sobre su fiabilidad.

Las validaciones formales de sistemas expertos raramente se publican. Una excepción es la validación formal del sistema MYCIN [BUCH84]. En este estudio, se revisaron las

recomendaciones de terapias realizadas por MYCIN y por nueve personas (desde especialistas hasta estudiantes de medicina) para varios pacientes elegidos aleatoriamente. Se llegó a la conclusión de que las recomendaciones indicadas por MYCIN no se distinguían de aquellas dadas por los especialistas. En la práctica, los sistemas basados en el conocimiento se validan de igual forma que el software convencional. Los encargados del desarrollo demuestran que el nuevo sistema resuelve cierta variedad de problemas difíciles antes de entregarlo a los usuarios finales [O'KEE87]. Los usuarios finales prueban el nuevo sistema en el contexto para un gran número de casos diferentes. A menudo, este proceso lo realizan en paralelo con el método antiguo. Es una manera de detectar los errores que puedan existir y corregirlos. Una vez probada la efectividad del programa, este se utilizará rutinariamente.

En los programas convencionales, se comprueba cada rama de cada subrutina con diferentes valores de las variables para asegurarse de que todas las partes del programa trabajan como deben. En los SBC, cada elemento de la base de conocimientos se examina independientemente, pero al igual que con las subrutinas, los errores suceden en las interacciones entre los elementos. Dichos errores se descubren con tests empíricos (se ejecuta el programa con ejemplos de problemas aleatorios y se determina qué casos se han resuelto correctamente y cuales no). En caso de no contar con un análisis lógico completo que demuestre que la BC y el motor de inferencias son correctos, se debe analizar la

representación empíricamente. Sin embargo, se debe determinar el nivel aceptable de error de cualquier tipo según los costes de error de cada tipo frente a los beneficios de las soluciones correctas.

CAPITULO 3

VALIDACION DE

BASES DE CONOCIMIENTOS

3.1.- INTRODUCCION

Los sistemas basados en el conocimiento deben depurarse y validarse como cualquier otro tipo de software. Las bases de conocimientos utilizan un estilo declarativo que separa el conocimiento del control. Esto hace posible una mejor comprobación semántica que en los sistemas de software convencionales donde el conocimiento y el control se encuentran mezclados. A medida que los avances en hardware y software hacen posible la construcción de grandes sistemas basados en reglas, la importancia de la validación de la base de conocimientos se hace aún mayor.

Como con la mayoría del software, la revisión sintáctica es útil para evitar problemas graves durante la ejecución. El encargado de desarrollo necesita información sobre la naturaleza y localización de los errores. Esta información es análoga a los mensajes de error de compilación resultantes de errores de sintaxis en los lenguajes de programación ordinarios.

Un método común de validación de software consiste en ejecutar ejemplos con resultados conocidos y comparar el resultado del programa con las respuestas correctas. Este método puede utilizarse también con sistemas basados en el conocimiento, aunque el número de casos que se deben comprobar puede ser muy grande si el dominio es complejo.

Uno de los elementos en los que se diferencian los sistemas basados en el conocimiento de otros sistemas software, es que la mayoría de ellos dependen, al menos en parte, del uso de reglas para la representación del conocimiento, y el comportamiento del sistema depende de como interaccionan las reglas entre sí. Si falta una regla, el encadenamiento fallará, tal como un programa ordinario puede fallar si le falta una subrutina. Sin embargo, el hecho de que un programa basado en conocimientos falle a causa la falta de una regla puede no ser tan obvio. Hay otros muchos

problemas potenciales en los sistemas basados en reglas, tales como reglas contradictorias o reglas que nunca se activan porque necesitan información que nunca podrán llegar a obtenerse.

Los tipos de validación que se pueden aplicar dependen de las reglas y de la estructura de la base de conocimientos.

La definición precisa de validación de SBC no es una tarea fácil, pero es posible decir que principalmente tiene que ver con una estructura correcta y un comportamiento adecuado de los SBC. De acuerdo con esos dos deseos, se van a considerar los siguientes aspectos [BREN89]:

- Verificación estructural o estática
- Verificación funcional o dinámica
- Evaluación

3.2.- VERIFICACION ESTRUCTURAL O ESTATICA

Una base de conocimientos puede ser considerada como una estructura de datos que será interpretada por el motor de inferencias. Esta estructura de datos debe cumplir algunas convenciones impuestas por el motor de inferencias, de forma

que se asegure su correcto funcionamiento. La verificación estructural debe comprobar que se lleven a cabo esas convenciones obligatorias.

Un análisis estático de las reglas puede detectar muchos problemas potenciales existentes en una base de conocimientos.

La estructura del SBC depende básicamente del formalismo de representación utilizado en el SBC. Por tanto, la verificación estructural puede verse como un proceso de comprobación de un conjunto de propiedades del SBC ligadas fuertemente al formalismo de representación. Estas propiedades son independientes del contenido del SBC, pueden comprobarse utilizando únicamente métodos estructurales, y esos métodos pueden trasladarse de un SBC a otro, suponiendo que ambos utilicen el mismo formalismo de representación.

Los problemas de la base de conocimientos sólo se pueden detectar si la sintaxis de las reglas es suficientemente restrictiva como para permitir que uno examine dos reglas y determine las situaciones existentes en las cuales las dos reglas pueden tener éxito, si los resultados de aplicarlas pueden ser los mismos, si son conflictivos o si no están relacionados. En los lenguajes de reglas que permiten una sintaxis muy poco restrictiva se hace difícil o imposible implementar los algoritmos que se verán a continuación.

3.2.1.- VALIDACION DE LA CONSISTENCIA

Mediante el análisis estático de la semántica de las reglas, un verificador de reglas puede detectar:

a) reglas redundantes,
b) reglas conflictivas o contradictorias,
c) reglas subsumidas por otras reglas,
d) condiciones SI innecesarias,
e) cadenas de reglas circulares.

3.2.1.1.- REGLAS REDUNDANTES

Dos reglas son redundantes si tienen éxito en la misma situación y llegan a las mismas conclusiones. Esto significa que las partes SI de las dos reglas son equivalentes, y una o más conclusiones son también equivalentes. Las partes SI de dos reglas pueden ser equivalentes sólo si cada parte tiene el mismo número de condiciones y cada condición en una parte es equivalente a una condición de la otra parte. Dado que en la regla se permiten variables, dos condiciones son equivalentes si son unificables [CUEN86].

Por ejemplo, si se considera:

(1) SI ?X tiene una tos ronca, Y
?X tiene una respiración difícil
ENTONCES el tipo-de-enfermedad de ?X es
CRUP-LARINGEO

(2) SI ?Y tiene una respiración difícil, y
?Y tiene una tos ronca
ENTONCES el tipo-de enfermedad de ?Y es
CRUP-LARINGEO

?X e ?Y representan variables que se asignarán a una persona. Estas dos reglas serían redundantes incluso si usasen diferentes variables e incluso si sus condiciones SI estuviesen en distinto orden.

3.2.1.2.- REGLAS CONFLICTIVAS O CONTRADICTORIAS

Dos reglas son conflictivas si ambas tienen éxito en la misma situación pero llegando a conclusiones conflictivas. Esto significa que las partes SI de las dos reglas son equivalentes pero una o más conclusiones son contradictorias.

Más formalmente, utilizando la notación del cálculo de predicados, la regla "p(x) —> ¬(q(x))" es contradictoria con la regla "p(x) —> q(x)".

Por ejemplo, si se consideran las reglas siguientes:

(1) SI ?X tiene una tos ronca, Y
?X tiene una respiración difícil
ENTONCES tipo-de-enfermedad de ?X es
CRUP-LARINGEO

(2) SI ?X tiene una tos ronca, Y
?X tiene una respiración difícil
ENTONCES tipo-de-enfermedad de ?X es
BRONQUITIS

Las dos reglas son conflictivas porque dada la misma información, una regla concluye que la enfermedad es CRUP y la otra que es BRONQUITIS.

3.2.1.3.- REGLAS SUBSUMIDAS

Se considera que una regla está subsumida por otra si las dos reglas tienen las mismas conclusiones, pero una contiene restricciones adicionales en algunas situaciones en

las que se aplica. Esto significa que una o más conclusiones son equivalentes pero la parte SI de una regla contiene menos restricciones y/o condiciones que la parte SI de la otra regla.

Más formalmente, utilizando la notación del cálculo de predicados, la regla "(p(x) ∧ q(y)) —> r(z)" está subsumida por la regla "p(x) —> r(z)". Siempre que una regla más restrictiva tenga éxito y la regla menos restrictiva también lo tenga, resulta una redundancia.

Por ejemplo, si se consideran las siguientes reglas:

(1) SI ?X tiene puntos rosas en la piel, Y
?X tiene fiebre
ENTONCES tipo-de-enfermedad de ?X es
SARAMPION

(2) SI ?X tiene puntos rosa en la piel
ENTONCES tipo-de-enfermedad de ?X es
SARAMPION

En este caso se puede decir que la regla (1) está subsumida por la regla (2) ya que la regla (2) sólo necesita una parte de la información para concluir SARAMPION. Siempre que la regla (1) tenga éxito, la regla (2) también lo tendrá.

3.2.1.4.- CONDICIONES SI INNECESARIAS

Dos reglas contienen condiciones SI innecesarias si las reglas tienen las mismas conclusiones y la condición SI de una regla está en conflicto con la condición SI de la otra regla, y todas las demás condiciones SI en las dos reglas son equivalentes. Utilizando la notación del cálculo de predicados, si tenemos dos reglas, (1) "$(p(x) \wedge q(y)) \rightarrow r(z)$", (2) "$(p(x) \wedge \neg(q(y)) \rightarrow r(z)$", la condición q(y) en ambas reglas es innecesaria. Estas dos reglas podrían combinarse como "$(p(x) \wedge (q(y) \vee \neg(q(y))) \rightarrow r(z)$". La condición "$(q(y) \vee \neg(q(y))$" resulta **VERDAD**, por tanto la regla se convierte en "$p(x) \rightarrow r(z)$". En este caso la condición SI innecesaria, indica que sólo una regla es necesaria.

(1) SI ?X tiene puntos rosas en la piel, Y
?X tiene fiebre
ENTONCES tipo-de-enfermedad de ?X es
SARAMPION

(2) SI ?X tiene puntos rosas en la piel, Y
?X no tiene fiebre
ENTONCES tipo-de-enfermedad de ?X es
SARAMPION

En este caso, la segunda condición SI es innecesaria. Las dos reglas pueden refundirse en una sola.

Hay un caso especial que se da cuando hay dos reglas con la misma conclusión, una de ellas contiene una sola condición SI, y está en conflicto con otra regla que contiene dos o más condiciones. Utilizando la notación del cálculo de predicados, si tenemos dos reglas, (1) "(p(x) ∧ q(y)) —> r(z)", (2) "¬(q(y)) —> r(z)", entonces la segunda condición SI de la regla (1) es innecesaria, pero ambas reglas son necesarias y se pueden reducir a: (1) "(p(x)) —> r(z)", y (2) "¬(q(y)) —> r(z)".

3.2.1.5.- REGLAS CIRCULARES

Un conjunto de reglas es circular si el encadenamiento de las mismas en el conjunto forma un ciclo. Utilizando la notación del cálculo de predicados, si se tiene un conjunto de reglas tal como: (1) "p(x) —> q(x)", (2) "q(x) —> r(x)", (3) "r(x) —> p(x)", y la meta es r(A), donde A es una constante, entonces el sistema entrará en un bucle infinito en tiempo de ejecución, a menos que el sistema tenga una forma especial de manejar las reglas circulares. Esta definición también incluye la posibilidad de una única regla formando un ciclo circular (es decir, "p(x) —> p(x)").

Por ejemplo, si se considera el siguiente conjunto de reglas:

(1) SI temperatura de ?X > 36.5
ENTONCES ?X tiene fiebre

(2) SI ?X tiene fiebre, Y
?X tiene puntos rosas en la piel
ENTONCES tipo-de-enfermedad de ?X es SARAMPION

(3) SI tipo-de-enfermedad de ?X SARAMPION
ENTONCES temperatura de ?X > 36.5

Dada la meta "tipo de enfermedad es SARAMPION", este conjunto de reglas provocarían un bucle infinito.

3.2.2.- VALIDACION DE LA COMPLETITUD

El desarrollo de un sistema basado en el conocimiento es un proceso iterativo en el cual el conocimiento se codifica, se comprueba, se añade, se cambia y se refina. El conocimiento fluye desde el experto hasta la base de conocimientos a través de un intermediario (un ingeniero de conocimiento). Este proceso iterativo a menudo deja huecos

o lagunas en la base de conocimientos, que tanto al ingeniero de conocimientos como al experto se les ha pasado inadvertido durante el proceso de adquisición del conocimiento. Además, al crecer el número de reglas, se hace imposible comprobar todos los posibles caminos a través del sistema. Hay cuatro situaciones indicativas de estos huecos (es decir, reglas que faltan) en una base de conocimientos:

- valores de atributo no referenciados,
- metas que van por caminos sin salida,
- conclusiones inalcanzables,
- condiciones SI sin salida.

Cualquiera de estas cuatro condiciones pueden indicar que faltan reglas.

Una de las características de algunas conchas de sistemas basados en conocimientos (ej. EMYCIN [VANM81], KEE, y LES [NGUY85]) que facilitan la detección de huecos, es que permiten al ingeniero de conocimientos establecer los tipos de atributos que define. Para cada atributo, se pueden definir un conjunto de propiedades, incluyendo si el usuario puede preguntar por el valor, y un conjunto de valores que puede tomar (valores legales). Se ha reconocido que en el área de ingeniería del software, ésta es una práctica de programación excelente.

Las propiedades de atributos que se implantan más frecuentemente son:

- restricción de valores de atributo aceptables,

- que el usuario pueda preguntar por el valor de un atributo,

- que el atributo tenga un único valor o que sea multivaluado.

A continuación se describe cómo usar estas propiedades para encontrar posibles huecos y errores en la base de conocimientos.

3.2.2.1.- VALORES DE ATRIBUTO NO REFERENCIADOS

Los valores de atributo no referenciados se dan en aquellas situaciones en las que algunos valores del conjunto de valores posibles de un atributo no están referenciados por ninguna condición SI de las reglas. Es decir, los valores legales de un conjunto están referenciados parcialmente. Un atributo parcialmente referenciado, puede impedir que el sistema llegue a una conclusión o provocar que llegue a una conclusión errónea al encontrar un valor de atributo no

referenciado en tiempo de ejecución. Los valores de atributo no referenciados pueden indicar también que faltan reglas. Por ejemplo, suponiendo un atributo TEMPERATURA cuyo conjunto de valores legales es {alta, normal, baja}, y que los valores "alta" y "normal" se utilizan en las condiciones SI de las reglas pero no así el valor "baja", entonces el verificador de reglas debería avisar al ingeniero de conocimientos de que "baja" no se utiliza, y éste tendría que decidir si falta una regla que incluya el valor "baja", o si dicho valor debería ser eliminado del conjunto de valores legales.

3.2.2.2.- VALORES DE ATRIBUTO ILEGALES

Un valor de atributo ilegal se da cuando una regla referencia a un valor de atributo que no pertenece al conjunto de valores legales. Normalmente este error se produce por una mala definición en los términos de las reglas.

Por ejemplo, supóngase el atributo TEMPERATURA cuyo conjunto de valores legales es {alta, media, baja}. Si existe una regla con una condición tal como:

SI temperatura de ?X es muy alta ...

... ENTONCES temperatura de ?X es media

El verificador de reglas debería avisar al ingeniero de conocimientos de que el valor "muy alta" es un valor de atributo ilegal para "temperatura".

3.2.2.3.- CONCLUSIONES INALCANZABLES

En un sistema de producción guiado por la meta, una conclusión de una regla debe o bien emparejarse con una meta o bien con la parte condición de otra regla. Si no hay emparejamiento para la conclusión, se dice que ésta es inalcanzable. Por ejemplo, supóngase que existe la regla:

SI temperatura de ?X > 36.5

ENTONCES ?X tiene fiebre

Si la condición "?X tiene fiebre" no aparece en ninguna parte SI de ninguna regla y no es parte de una meta, entonces la estrategia de control debería avisar al ingeniero de conocimientos de que dicha conclusión es inalcanzable.

3.2.2.4.- CONDICIONES "SI" SIN SALIDA Y METAS QUE VAN POR CAMINOS SIN SALIDA

Para alcanzar una meta, o submeta, en un sistema guiado por la meta, es preciso que:

- pueda determinarse la verdad de la meta preguntando al usuario, o bien,

- la meta coincida con una conclusión de una de las reglas del conjunto.

Si no se satisface ninguno de estos requisitos, entonces la meta no se puede alcanzar (es una meta que nos lleva a un camino sin salida). De forma similar, las condiciones SI de una regla deben también cumplir alguna de las dos condiciones, o en caso contrario serán "condiciones SI sin salida". Por ejemplo, supóngase la siguiente meta:

tipo-de-enfermedad del paciente es SARAMPION,

Si el atributo "tipo-de-enfermedad" no se puede solicitar al usuario y no hay ninguna regla que concluya este hecho, entonces debe considerarse que la meta nos lleva a un camino sin salida.

3.2.3.- COMPROBACION DE REGLAS GUIADAS POR DATOS

Hasta este punto, se han considerado sólo las reglas guiadas por la meta. La comprobación de la consistencia y completitud de las reglas guiadas por datos, es muy similar a la de las reglas guiadas por la meta. La detección de reglas conflictivas, reglas redundantes, reglas subsumidas, cadenas de reglas circulares, condiciones SI sin salida, valores de atributo no referenciados, y valores de atributo ilegales, se realiza de la misma forma que se ha descrito anteriormente.

La detección de conclusiones inalcanzables no se aplica en reglas guiadas por datos ya que los objetivos no coinciden con las conclusiones.

3.2.4.- EFECTO DE LA INCERTIDUMBRE EN LAS COMPROBACIONES

El método para manejar la incertidumbre en una base de conocimientos es un tópico muy debatido. El formalismo clásico para cuantificar la incertidumbre es la teoría de la probabilidad (Bayesiana), pero también existen otras

alternativas. Entre ellas se pueden señalar la teoría de la certeza, la teoría de la posibilidad y la teoría de la evidencia de Dempster-Shafer.

El hecho de permitir que las reglas de un sistema lleguen a conclusiones con cierto grado de certeza y permitir que los datos sean introducidos con un factor de certeza asociado, modifica las definiciones vistas anteriormente, en la siguiente forma:

- **CONFLICTO:** En un conjunto de reglas que utilizan factores de certeza, es muy común que aparezcan dos reglas que se produzcan en la misma situación pero con diferentes conclusiones. A menudo, dados los mismos síntomas, el experto puede desear que haya varias conclusiones con distintos grados de certeza.

- **REDUNDANCIA:** Las reglas redundantes pueden ocasionar graves problemas. Pueden provocar que la misma información se cuente dos veces, originando un incremento erróneo en el peso de las conclusiones.

- **SUBSUNCION:** La subsunción se utiliza a menudo en conjuntos de reglas con factores de certeza. El ingeniero de conocimiento diseña frecuentemente reglas tales que las más restrictivas aportan mayor peso a las conclusiones realizadas por las menos restrictivas.

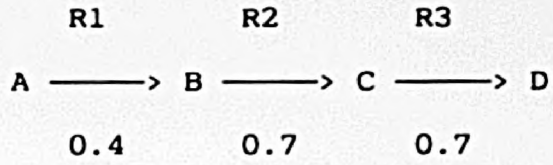
- **CONDICIONES SI INNECESARIAS:** Las condiciones SI que se consideran innecesarias cuando las reglas concluyen con certeza absoluta, pueden ser necesarias cuando se trabaja con factores de certeza. El ingeniero de conocimiento puede desear concluir un valor con diferente factor de certeza. Si las reglas concluyen con el mismo factor de certeza, entonces las condiciones SI seguirán siendo innecesarias.

- **VALORES DE ATRIBUTO NO REFERENCIADOS:** Los factores de certeza no afectan a esta definición ni a la forma de detectar los valores de atributo no referenciados.

- **VALORES DE ATRIBUTO ILEGALES:** Los factores de certeza no afectan a esta definición ni a la forma de detectar los valores de atributo ilegales.

- **CONDICIONES SI SIN SALIDA:** La detección de condiciones SI sin salida (o metas inalcanzables) es más compleja cuando se utilizan factores de certeza con umbrales a considerar. De este modo, una meta que lleva a un camino sin solución puede darse si existe una cláusula ENTONCES que concluye con un factor de certeza inferior que el umbral.

Por ejemplo, si se supone que hay un camino de razonamiento lineal de tres reglas (R1, R2, y R3) donde A se le pedirá al usuario y D es la meta que inicia la línea de razonamiento:



Si se conoce A con certeza, D sólo podrá conocerse con un FC de (0.4)(0.7)(0.7) = 0.19. Si este FC es menor que el umbral utilizado en el sistema, entonces D sería una meta que nos lleva a un camino sin salida.

- **CONCLUSIONES INALCANZABLES:** La detección de conclusiones inalcanzables en un conjunto de reglas con factores de certeza es también mucho más compleja. Una conclusión de una regla puede ser inalcanzable incluso si su parte SI coincide con una conclusión de otra regla. Esto puede ocurrir si la conclusión que coincide con una de las condiciones SI no puede determinarse con un FC superior al umbral. Por ejemplo, si se suponen las reglas:

R1: SI A ENTONCES B (FC = 0.1)

R2: SI B ENTONCES C (FC = 1.0)

Si la única forma de determinar B es mediante la regla R1, entonces la conclusión de la regla R2 será inalcanzable ya que, si A se conoce con certeza, C no puede determinarse con un FC superior a 0.2.

- CADENAS DE REGLAS CIRCULARES: Los factores de certeza no afectan a la detección de reglas circulares. Sin embargo, debe señalarse que los FC pueden originar que se rompan cadenas circulares de reglas, si el FC de una conclusión es menor que el umbral.

3.3.- VERIFICACION FUNCIONAL O DINAMICA

La verificación funcional o dinámica contribuye a la seguridad sobre el comportamiento adecuado del SBC con respecto al problema que se está intentando resolver. Un sistema basado en el conocimiento, como cualquier programa de ordenador, se entiende como una función que toma algunas entradas (argumentos) y genera una salida (resultado). Esta funcionalidad establece una relación entre las entradas y las salidas de los SBC.

La validación del funcionamiento de los SBC se dirige a asegurar que la salida del SBC cumpla con la semántica del dominio. Puesto que un SBC puede considerarse como un programa mas los datos especificados por las estructuras del conocimiento en la BC, su validación equivale a la validación de la conducta del programa y del contenido de la BC.

La verificación funcional de un SBC comprende el siguiente ciclo (Figura 3.1.): detección de resultados incorrectos, análisis del contenido de la BC, y modificación de las estructuras de la BC de forma adecuada con el objeto de corregir el funcionamiento del SBC. Este ciclo es parte del proceso de adquisición del conocimiento, normalmente denominado refinamiento del conocimiento. La fase de refinamiento no se caracteriza por la adquisición de conjuntos completos de estructuras de la BC, sino por la incorporación, eliminación y modificación de estructuras de conocimiento y de componentes de esas estructuras.

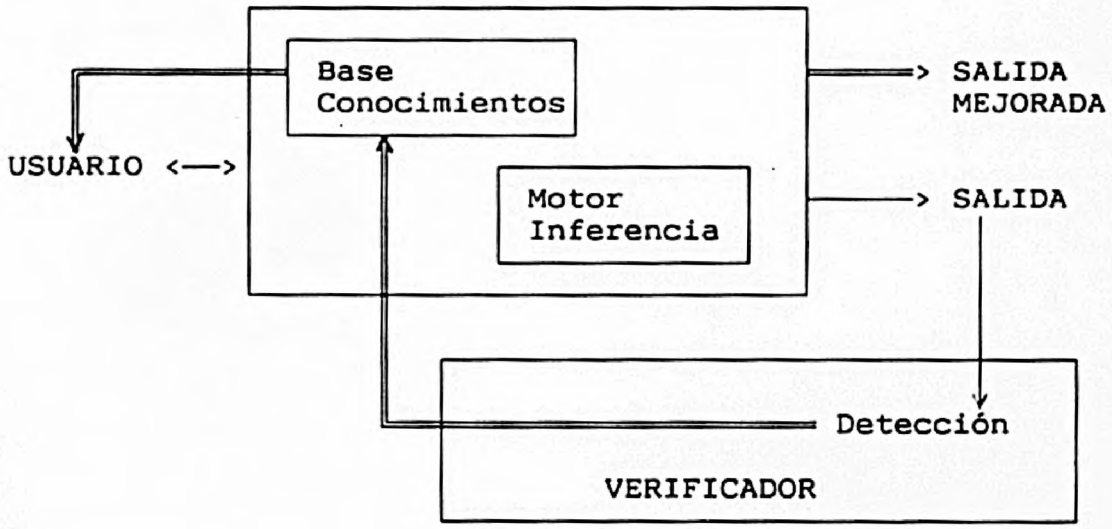


Figura 3.1.

Ciclo de verificación funcional.

3.4.- EVALUACION

El objetivo de la evaluación es proporcionar criterios para la medición de las características de los sistemas basados en conocimientos. Estos criterios deben contemplar aspectos tales como la calidad, la complejidad del SBC, la dificultad para alcanzar una solución, etc. Estos criterios pueden ser también útiles para comparar sistemas basados en conocimientos.

CAPITULO 4

DESCRIPCION DEL SISTEMA MILORD

4.1.- INTRODUCCION

MILORD es un sistema basado en el conocimiento cuyas herramientas son dos motores de inferencia (uno de encadenamiento hacia delante y otro de encadenamiento hacia atrás) y un módulo de explicación. Su arquitectura se decidió a partir de la experiencia adquirida en la resolución de problemas, sobre todo en el campo de la medicina. Sus elementos arquitectónicos más relevantes son [SIER89]:

- Su simplicidad como factor determinante para ser utilizado por los expertos.

- El razonamiento aproximado como elemento determinante de la arquitectura.

- La credibilidad de las aplicaciones.

- La modularización como forma de definir grandes aplicaciones.

- La definición de niveles de conocimiento como una ayuda a la estructuración de problemas mal definidos.

- Descripción formal de la semántica del lenguaje.

4.2.- REPRESENTACION DEL CONOCIMIENTO EN MILORD

4.2.1.- ENTIDADES BASICAS: HECHOS, REGLAS, MODULOS, ESTRATEGIAS Y CONTEXTOS

MILORD consta de cinco tipos de objetos: hechos, reglas, módulos, estrategias y contextos. Hay dos tipos de reglas que dependen de sus conclusiones: reglas y meta-reglas.

CONTEXTO: La base de conocimientos (BC) está particionada en un conjunto de contextos. Cada uno de ellos está especializado en una tarea particular, por ejemplo

diagnóstico, tratamiento, etc. Cada contexto está compuesto por varios módulos, uno de ellos contiene las meta-reglas que controlan las estrategias de búsqueda a través de los otros módulos.

MODULOS: Un módulo es un conjunto de reglas agrupadas según diversos criterios: condiciones similares, conclusiones iguales, ..., etc. Por ejemplo, en una aplicación de diagnóstico médico, cada módulo tiene un conjunto de diagnósticos y tratamientos con los que se relacionan las reglas. Los diágnosticos y tratamientos pueden estar compartidos por diferentes módulos, permitiéndoles representar diversos pasos hacia los mismos objetivos. Cada módulo tiene un conjunto de meta-reglas que supervisa que las reglas se puedan aplicar en él. Estas sólo se aplican a la reglas del módulo al que pertenecen. El resultado, en cada módulo, es un valor de certeza asociado a cada diagnóstico o tratamiento a los que se refiere dicho módulo.

ESTRATEGIAS: Una estrategia es un conjunto de módulos a evaluar y un conjunto de módulos rechazados. Las estrategias se construyen con meta-reglas y se generan dinámicamente en ejecución, dependiendo de las meta-reglas que se puedan utilizar. Al evaluar una estrategia, el sistema recorre secuencialmente todos los módulos incluidos en ella y que han de ser visitados.

REGLAS Y META-REGLAS: Las reglas son los componentes básicos de un módulo. Están compuestas por un conjunto de condiciones, hechos, predicados sobre los hechos o cálculos sobre los valores numéricos de los hechos. Cada condición tendrá un valor como resultado de su evaluación. Las reglas tienen una conclusión que es un predicado sobre un objeto dado: un hecho (en el caso de la conclusión de una regla), una regla o un módulo (en el caso de la conclusión de una meta-regla). Cada regla tiene un valor de certeza asociado que permite la propagación de la certeza desde las condiciones a las conclusiones. Evidentemente, sólo tienen sentido cuando las conclusiones se refieren al valor de certeza de un hecho pero no cuando la conclusión es una decisión binaria.

Los ejemplos a los que se hará referencia están relacionados con PNEUMONIA (una aplicación de diagnóstico médico utilizando MILORD).

Las meta-reglas son de tres tipos:

- Orientadas a reglas: Aquellas que supervisan que las reglas se puedan aplicar en un módulo. Dependiendo de los hechos contextuales, algunas reglas son trasladadas del conjunto activo de reglas: por ejemplo, las reglas que deducen algún tipo de enfermedad viral dependen del análisis de sangre y no podrán aplicarse a menos que los resultados del análisis estén disponibles.

- **Orientadas a módulos:** Se refieren a la estrategia global de selección de subobjetivos. Tras la aplicación de cada módulo el conjunto de meta-reglas modifica la estrategia, añadiendo o eliminando módulos de la lista de objetivos; por ejemplo, la conclusión de una enfermedad bacteriana probablemente no alterará los módulos que no conciernen a enfermedades de este tipo.

- **Orientadas a estrategias:** Son las responsables de combinar las estrategias, cuando se genera más de una, en una única. Usan el dominio y el control del conocimiento en sus premisas.

HECHOS: Son los objetos más elementales del sistema. Hay cuatro tipos:

- **Booleanos:** Pueden tomar dos valores: verdadero ó falso.

- **Difusos:** El valor es un grado representado por un número lingüístico difuso o una etiqueta lingüística difusa.

- **Numéricos:** El valor del hecho es un número.

- **Subconjunto:** El valor del hecho es un subconjunto de un conjunto dado de posibles valores.

En la definición de las reglas es necesario examinar el tipo de hecho así como los predicados. MILORD mantiene el control de las reglas que se deducen y de aquellas que le pertenecen, como parte de la premisa. Esto es necesario para garantizar un encadenamiento hacia adelante y hacia atrás de una forma eficiente.

4.2.2.- VERIFICACION DE LA BC

La verificación lógica requiere la selección de cualquier situación en la BC que pudiese causar un comportamiento impropio del motor de inferencias. Como se ha considerado en el capítulo 3, estas situaciones pueden ser errores lógicos (reglas contradictorias o circulares), falta de conocimiento (conclusiones inalcanzables) o duplicación de conocimiento (reglas redundantes, condicionales SI innecesarias, reglas asumidas). En la lógica booleana, todas estas situaciones se detectan con un análisis de reglas estático. En los sistemas difusos se requieren más extensiones.

4.2.3.- ORGANIZACION DE LA RED SEMANTICA

Existe una estructura jerárquica evidente entre los objetos definidos en MILORD. Cada elemento tiene en este sentido algunas propiedades heredadas. Un hecho hereda los nombres de las reglas con las que se relaciona. Las reglas y los hechos heredan el nombre del módulo en el que aparecen, etc.. Esta estructura se puede ver como una red semántica de tres niveles. En cada nivel los objetos tienen interrelaciones que expresan prioridad, subsunción, inclusión por parte de la cadena propietaria, especificación, etc..

Estas relaciones se obtienen a partir de un análisis sintáctico de la BC que se representa mediante el editor que tiene MILORD [GODO88]. Junto con estas estructuras explícitas, existen relaciones entre estos objetos, dadas por las meta-reglas. En los siguientes ejemplos, se establece una red semántica de forma que permite al sistema respetar las restricciones impuestas por la BC. En las figuras 4.1., 4.2., y 4.3., las líneas gruesas representan las relaciones inferidas a partir de la información dada en la definición de la BC.

El siguiente ejemplo muestra una inclusión mediante las cadenas de propiedades. En este caso el tipo de propiedad es "es-una-clase-de".

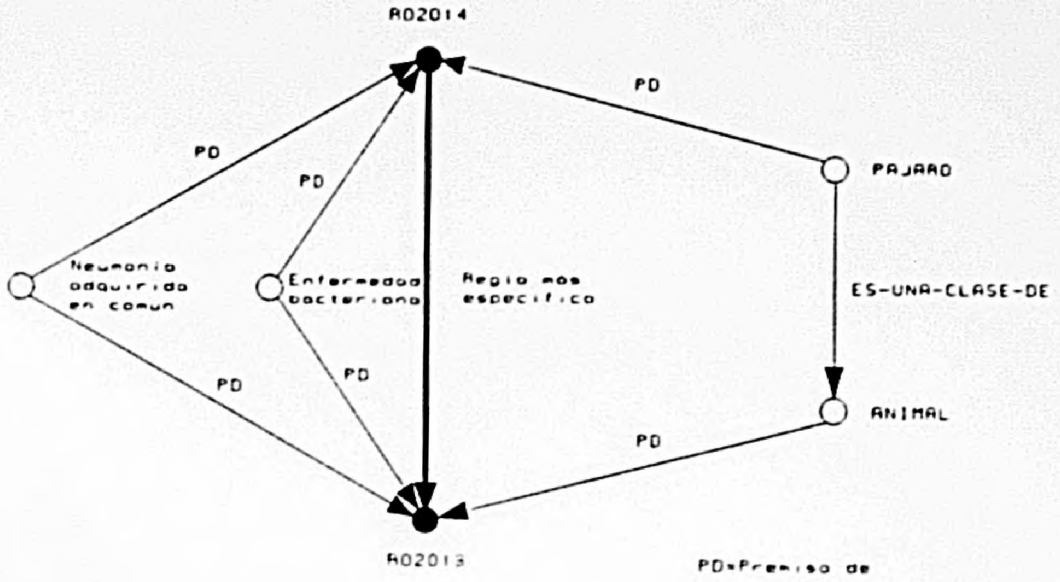


Figura 4.1.

Representación esquemática de las relaciones entre R02013 y R02014.

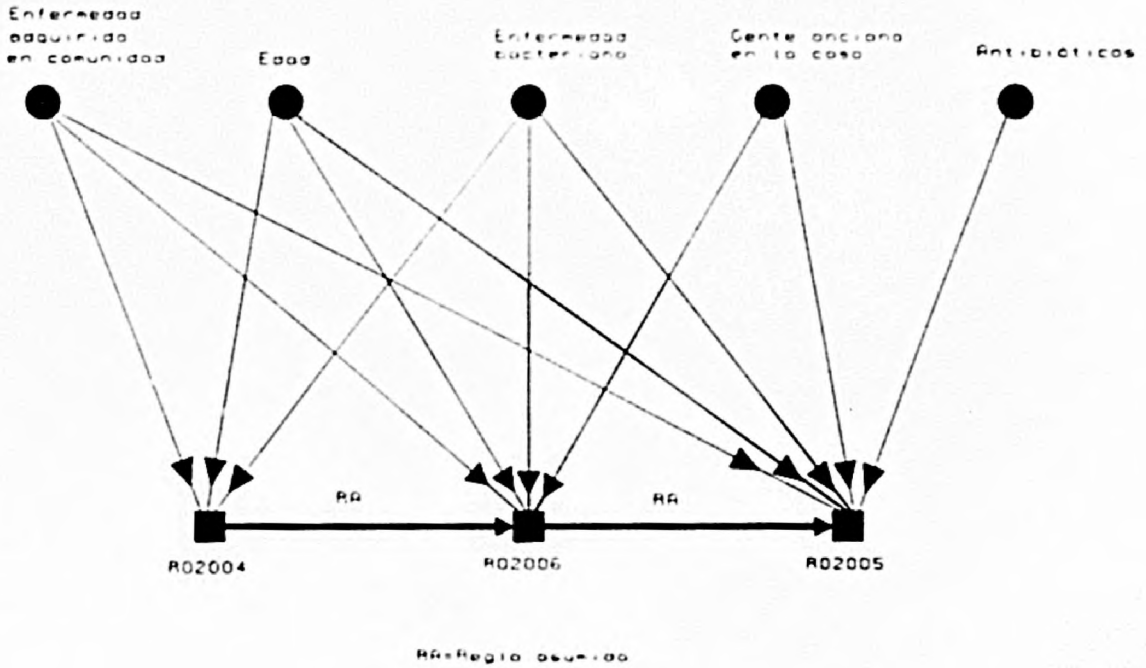


Figura 4.2.

Subsunción de reglas (todas las uniones son PD).

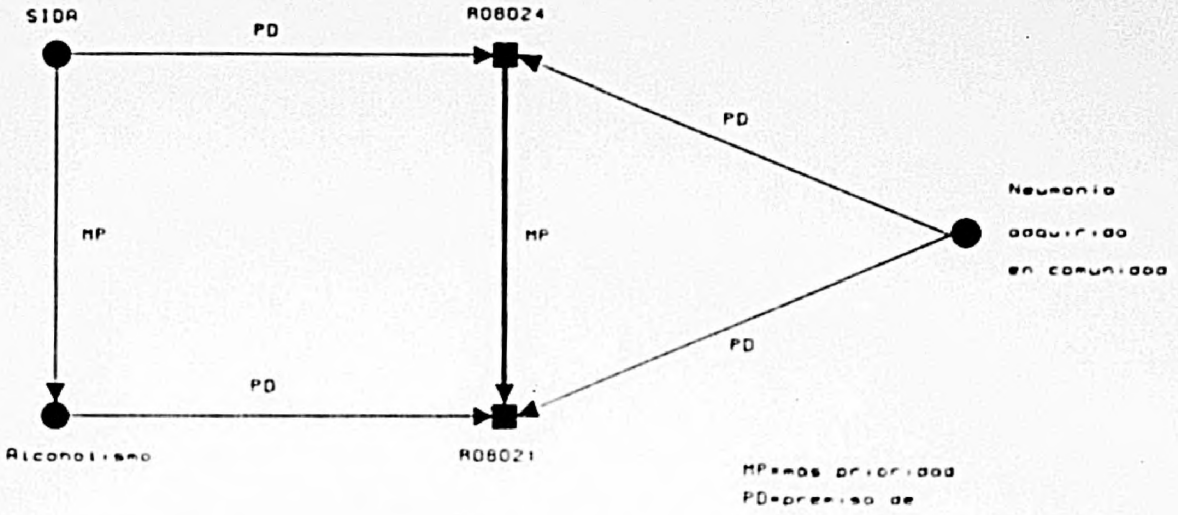


Figura 4.3.

Relación de Prioridades.

R02013 SI 1) neumonía adquirida en comunidad.
2) contactos frecuentes con animales.
3) enfermedad bacteriológica
ENTONCES (bastante probable)
NEUMOCOCO

R02014 SI 1) neumonía adquirida en comunidad.
2) contactos frecuentes con pájaros,
3) enfermedad bacteriológica
ENTONCES (muy probable)
NEUMOCOCO

Existe una relación 'un pájaro es una clase de animal', que hace la regla R02014 más especifica que la regla R02013.

A continuación se da un ejemplo de subsunción entre tres reglas (la regla R02004 subsume a las otras dos):

R02006 SI 1) neumonía adquirida en comunidad.
2) mayor de 70 años.
3) vive en un asilo.
4) enfermedad bacteriológica
ENTONCES (moderadamente posible)
NEUMOCOCO

R02005 SI 1) neumonía adquirida en comunidad.
2) mayor de 70 años.
3) vive en un asilo.
4) tratado con antibióticos,
5) enfermedad bacteriológica
ENTONCES (poco probable)
NEUMOCOCO

R02004 SI 1) neumonía adquirida en comunidad
2) mayor de 70 años
3) enfermedad bacteriológica
ENTONCES (posible)
NEUMOCOCO

La figura 4.3. muestra una relacion de prioridad entre las condiciones de las reglas.

4.2.4.- TRADUCCION DEL CONOCIMIENTO DE UNA REPRESENTACION EXTERNA A UNA INTERNA

Para mejorar el rendimiento del sistema, se hace una traducción de la representación externa. La representación interna del objeto se basa en una estructura de frame.

Cada objeto (hecho, regla o módulo), se representa por una estructura que soporta sus propiedades, relaciones y la información general. Esta representación hace explícitas las relaciones implícitas de la representación externa. Además, se pueden realizar algunos cálculos (máximo valor alcanzable para un hecho,...) y se pueden detectar errores sintácticos y semánticos. El examen del tipo se aplica también aquí para verificar la BC.

- El aspecto externo de una regla es:

```
R010014 SI 1) neumonía adquirida en comunidad.
           2) sin expectoración.
           3) sin deshidratación.
        ENTONCES (moderadamente probable)
        ENFERMEDAD ATIPICA.
```

La representación interna de una regla contiene información que no está explícita en la externa, tal como información de control y contextual para explicar

eficientemente las estrategias de razonamiento. La representación interna se implementa utilizando una primitiva **defstruct** de definición de estructura del **COMMON LISP**. Por ejemplo la representación interna de las reglas tiene la forma:

```
(defstruct regla
     (bc                nil        :type          atom)
     (nom               nil        :type          atom)
     (modul             nil        :type          atom)
     (si                nil        :type          list)
     (entonces          nil        :type          list)
     (documentación " "            :type          string)
     (mr-deducida       nil        :type          list)
     (inhibida          nil        :type          atom)
     (asumida           nil        :type          list)
     (valor             nil        :type          atom)
     (aplicada          nil)
     (explicación-tall-p nil       :type          list)
     (explicación-tall-o nil       :type          list))
```

La traducción del conocimiento, realizada off-line debe recalcularse cuando se introduzcan modificaciones en la BC. Este cálculo podría hacerse incrementalmente.

4.3.- ARQUITECTURA MULTINIVEL MILORD

Al definir una aplicación (una BC), existen dos aspectos fundamentales del problema a resolver: la representación del dominio del conocimiento y del control del conocimiento. El segundo aspecto es el principal en la mayoría de las aplicaciones de IA. En esta sección se señalan las relaciones entre ellas.

La arquitectura de MILORD presenta una estructura multinivel tanto en la representación del dominio como en la del control. Hay una relación entre estas dos representaciones como se puede ver en la figura 4.4.

La red semántica actúa sobre los hechos estructurando sus interdependencias y controlando mecanismos tales como suposiciones semánticas entre conceptos (por ejemplo, si el hecho SIDA está presente en un contexto de enfermedades bacteriológicas entonces el hecho alcoholismo no es relevante y no deberá ser considerado). La jerarquía de metarreglas controla la aplicación de su dominio de conocimiento correspondiente. Antes de bajar en los niveles jerárquicos del dominio (de las estrategias a las reglas y hechos) se consulta el nivel de control correspondiente.

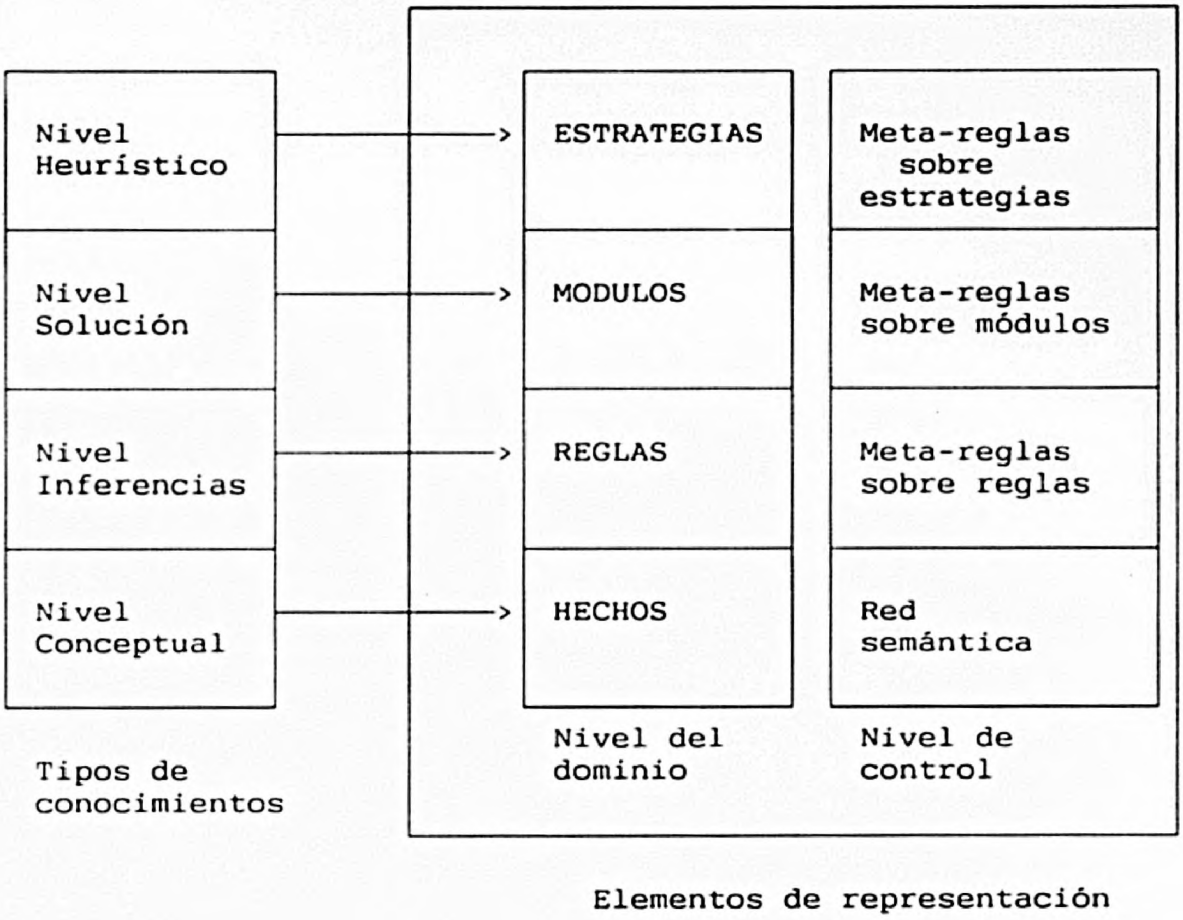


Figura 4.4.

Arquitectura multinivel de MILORD.

Esta separación entre los diferentes niveles de representación del control y del dominio encaja bastante bien con varios procesos de diseño en la definición de la BC:

a) Permite un diseño top-down de la BC, porque las decisiones sobre dominio del conocimiento y el control correspondiente de los elementos son independientes de refinamientos posteriores. Por ejemplo, en el caso de definiciones de estrategias, primero se pueden definir las estrategias basadas sólo en el dominio del conocimiento y posteriormente definir cómo solucionar los posibles conflictos entre ellos.

b) Permite definir diferentes métodos de resolución de problemas debido a la flexibilidad en el nivel operacional. Por ejemplo, se puede definir una simple clasificación o un proceso de clasificación heurístico.

c) Permite definir las diferentes estructuras jerárquicas: jerarquías mixtas o puras en el dominio del conocimiento y el nivel correspondiente de control que supervisa la utilización de las distintas estrategias.

Sin embargo esta arquitectura no es apropiada para definir sistemas generativos, sistemas que pueden generar o construir soluciones al problema. MILORD presupone que todas las soluciones se conocen a priori.

Esta definición multinivel tiene también algún aspecto de interés desde el punto de vista del ingeniero del conocimiento:

EXTENSIBILIDAD: Añadir nuevos dominios de conocimiento y nuevo control de conocimiento es fácil y seguro.

DEPURABILIDAD: Resulta fácil corregir los errores y validar la BC.

EXPLICABILIDAD: Explicar el comportamiento del sistema es más sencillo porque los niveles superiores en la jerarquía proporcionan las justificaciones correctas.

4.4.- RAZONAMIENTO CON INCERTIDUMBRE

Las aproximaciones numéricas a la representación de la incerteza implican la hipótesis de la independencia, exclusión mutua, etc., respecto de la información con la que se va a tratar. Por otro lado, esto obliga al experto y al usuario a ser irrealmente preciso y consistente en la asignación de tales valores numéricos a las reglas y hechos. Sin embargo, estas aproximaciones son computacionalmente caras.

En MILORD se han desarrollado dos aproximaciones basadas en la caracterización de la incerteza. Los valores de certeza son términos lingüísticos definidos por el experto. En la primera aproximación, la representación interna de cada término es un número difuso del intervalo [0,1], siguiendo el trabajo de Bonissone [BONI87] y permite tratar con valores de incerteza y con reglas cuya incerteza concierne a la solidez de la aplicación. La segunda aproximación es una extensión de la primera y permite tratar con reglas que contienen incerteza expresada lingüísticamente que modifica las condiciones y las conclusiones. Tal tipo de incerteza se representa internamente mediante etiquetas difusas. Por razones computacionales, ambas representaciones se parametrizan al principio de cada diseño de aplicación. En la aplicación de diagnóstico de neumonía (PNEUMONIA), se ha utilizado al principio la primera aproximación, y luego, se ha cambiado a la segunda aproximación de acuerdo con la preferencia de los expertos médicos.

MILORD se ha parametrizado también para ejecutar diferentes cálculos de incerteza operando con un conjunto definido de términos de valores lingüísticos de certeza.

Para modelizar la información que contiene incertidumbre, se han propuesto diversos enfoques basados en diferentes teorías entre las cuales destacan la teoría probabilista, la teoria de la evidencia, y la teoría de la posibilidad. En los tres métodos, la hipótesis elemental es

que la incertidumbre resultante de la inferencia de una regla se puede calcular a partir de la incertidumbre del antecedente y de la incertidumbre de la regla:

certeza(q) = h (certeza(p), certeza(p->q))

El significado de esta representación numérica y su obtención determinarán las propiedades de la función h, denominada ***modus ponens***.

4.4.1.- MANEJO DE LA INCERTIDUMBRE CON NUMEROS DIFUSOS

Cada valor lingüístico se representa internamente mediante un intervalo difuso (número difuso), es decir, mediante una función miembro de un conjunto difuso sobre la linea real, o más precisamente, sobre el espacio real representado mediante el intervalo [0,1]. Estas funciones miembro se pueden interpretar como los significados de los términos en el conjunto de términos. Los operadores de conjunción y disyunción aplicados a esas funciones pueden producir otra función miembro como resultado que deberá emparejarse con un término en el conjunto de términos, con el objeto de mantener cerrado el conjunto de términos. Esto puede hacerse mediante un proceso de aproximación lingüística [GODO88].

Aunque un SBC puede definir su propio conjunto de terminos junto con su representación interna, MILORD proporciona un conjunto de términos por omisión obtenido por medio de un estudio realizado entre varios de profesionales médicos en el área de Barcelona. El conjunto de términos por omisión es:

{IMPOSIBLE, CASI-IMPOSIBLE, LIGERAMENTE-POSIBLE, MODERADAMENTE-POSIBLE, POSIBLE, BASTANTE-POSIBLE MUY-POSIBLE, CASI-SEGURO, SEGURO}

Cada término Ti se representa mediante una función miembro $\mu i(x)$, para x en el intervalo [0,1].

Cada función miembro se representa mediante cuatro parámetros $Ti = (a_i, b_i, c_i, d_i)$ correspondientes a los intervalos con pesos de la Figura 4.5.

El conjunto de términos de nueve elementos tiene la siguiente representación resultante del estudio realizado y corresponde a las funciones de la Figura 4.6.

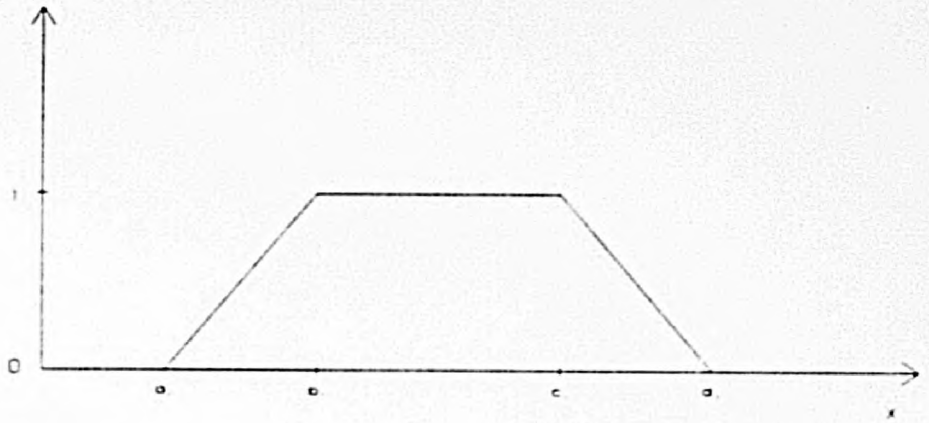


Figura 4.5.

Representación paramétrica de las funciones miembro.

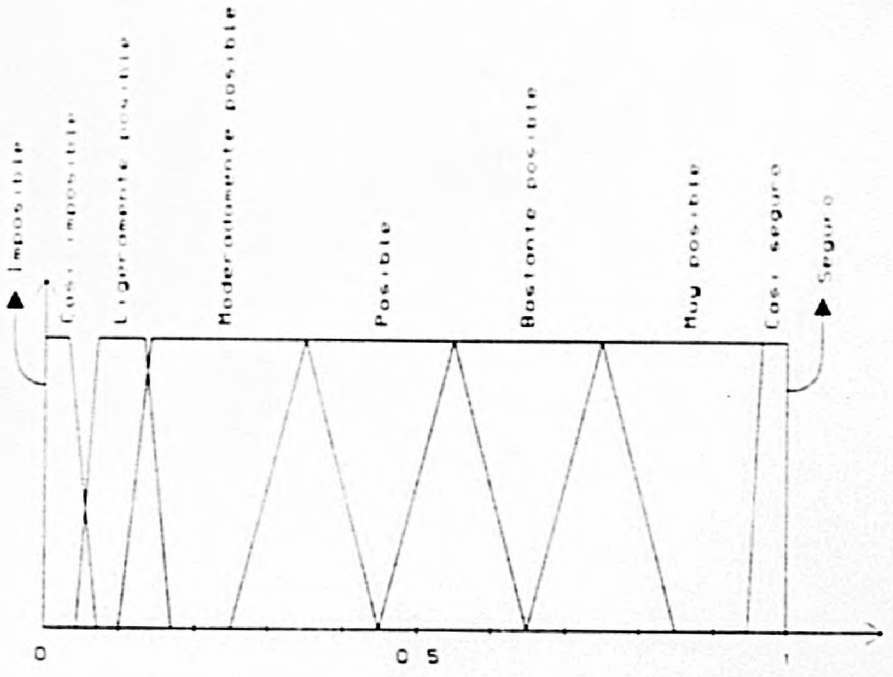


Figura 4.6.

Conjunto de términos de PNEUMONIA.

IMPOSIBLE = (0, 0, 0, 0)

CASI IMPOSIBLE = (0, 0, 0.05, 0.08)

LIGERAMENTE POSIBLE = (0.05, 0.07, 0.14, 0.17)

MODERADAMENTE POSIBLE = (0.10, 0.15, 0.35, 0.45)

POSIBLE = (0.25, 0.35, 0.55, 0.65)

BASTANTE POSIBLE = (0.45, 0.55, 0.75, 0.85)

MUY POSIBLE = (0.65, 0.75, 1, 1)

SEGURO = (1, 1, 1, 1)

Con el objeto de ser capaz de evaluar las normas-T T_1, T_2, T_3 y las conormas-T S_1, S_2, S_3 sobre los elementos de del conjunto de términos (que se utilizan para el cálculo de la incertidumbre en MILORD), se aplicará la siguiente fórmula de acuerdo con las reglas aritméticas de los números difusos: Dados dos intervalos difusos i = (a, b, c, d) e i' = (a', b', c', d') tenemos que:

i + i' = (a+a', b+b', c+c', d+d')

i - i' = (a-d', b-c', c-d', d-a')

i * i' = (aa', bb', cc', dd')

min(i,i') = (min(a,a'), min(b,b'), min(c,c'), min(d,d'))

max(i,i') = (max(a,a'), max(b,b'), max(c,c'), max(d,d'))

4.4.2.- MANEJO DE LA INCERTIDUMBRE UTILIZANDO ETIQUETAS LINGUISTICAS DIFUSAS

Como se ha mencionado anteriormente, esta es la aproximación que ha sido finalmente la preferida de los expertos. Es una aproximación alternativa que permite combinar la imprecisión y la incerteza. Se va a considerar una proposición tal como '**x es A**' donde **x** es una variable que toma sus valores de un subconjunto difuso **A** de un universo del discurso **U**. Entonces **x** induce a una distribución de posibilidades sobre **A**:

$$\pi(x) = \mu_A(x), \quad \text{para todo } x \in U$$

donde μ_A permanece para la función miembro del subconjunto difuso **A**, de donde se identificará **A** con la distribución de posibilidad π_x sobre **A**.

Se puede definir una etiqueta difusa por medio de la función t: [0, 1] —> [0, 1] de forma que la proposición '(**X es A**) es t' fuese equivalente a '**X es A**"' donde **A**" = t o **A**, en el sentido de las funciones de composición usuales.

Este conjunto de etiquetas se aplica actualmente sobre valores con un grado de posibilidad entre 0 y 1 (incrementándolo o disminuyéndolo), y no tiene ningún efecto sobre valores que son seguros o imposibles.

En cuanto a la selección del conjunto de términos, MILORD trabaja con dos conjuntos de términos de valores de certeza lingüísticos. Uno, proporcionado por el experto, se utiliza externamente, y el otro se genera automáticamente desde el principio y se utiliza de forma transparente para el usuario.

Cada valor de certeza lingüístico, se representa internamente mediante una etiqueta difusa, que puede entenderse como el significado de los términos en el conjunto de términos. Los operadores conectivos aplicados a esos valores de certeza, pueden producir otro valor de certeza, que estará emparejado con un término de uno de los dos conjuntos de términos, dependiendo del operador utilizado.

En el conjunto de términos expertos de PNEUMONIA existen nueve valores lingüísticos con la siguiente representación paramétrica:

t_1: IMPOSIBLE	(0,0)
t_2: CASI-IMPOSIBLE	(0,0.05)
t_3: LIGERAMENTE-POSIBLE	(0,0.1)
t_4: MODERADAMENTE-POSIBLE	(0,0.65)
t_5: POSIBLE	(0,1)
t_6: BASTANTE-POSIBLE	(0.35,1)
t_7: MUY-POSIBLE	(0.9,1)
t_8: CASI-SEGURO	(0.95,1)
t_9: SEGURO	(1,1)

Estas etiquetas se aplican a los hechos y a las reglas. Si no se especifica ninguna etiqueta, MILORD asume por defecto el valor CASI-SEGURO (t_8), en lugar de SEGURO (t_9), debido a que t_9 tiene un comportamiento diferente del resto: no puede modificarse por composición (por ejemplo, t o t_9 = t_9, para todo t), y es totalmente incompatible con cualquier otro (por ejemplo, la compatibilidad de t_9 con t es t_0 para todo t distinto de t_9). Por tanto, t_9 y, por dualidad, t_0 se reservan para razonamiento no difuso.

Como se ha mencionado anteriormente, después de toda operación lógica, se realiza un proceso de aproximación para mantener cerrado el conjunto de términos expertos.

A continuación se muestra un ejemplo del proceso de inferencia:

```
R08004    SI 1) Neumonia adquirida en comunidad es
                'casi seguro'
             2) Enfermedad bacteriana es 'posible'
             3) (No aspiración) es 'muy posible'
          ENTONCES [Posible]
          Enterobacteria es 'bastante posible'
```

Hechos observados:

(1') Neumonía adquirida en comunidad es 'muy posible'

(2') Enfermedad bacteriana es 'casi seguro'

(3') Aspiración es 'ligeramente posible'

Pasos de la inferencia:

(a) La compatibilidad entre (1) y (1') produce: 'moderadamente posible'.

(b) La compatibilidad entre (2) y (2') produce: 'casi seguro'.

(c) La negación de (3'): (no aspiración) es (no ligeramente posible) es decir, 'muy posible'.

(d) La compatibilidad entre (3) y (c) produce: 'posible'.

(e) [(a) y (b) y (d)] produce: 'moderadamente posible'.

(f) Inferencia: (e) y el valor de la regla (posible) produce: 'moderadamente posible'.

(g) La composición entre (f) y la etiqueta de la conclusión (bastante posible) produce: 'posible'.

4.5.- PNEUMONIA: UNA APLICACION EN MEDICINA

Esta aplicación concierne al diagnóstico y tratamiento de las enfermedades de neumonía. Como muchas otras en medicina, tiene muchas características interesantes y complejas que sirven como cota para los entornos de sistemas basados en el conocimiento. Entre ellos se pueden señalar: razonamieno no monótono, incertidumbre e información incompleta.

En el caso de una neumonía extrahospitalaria, se determina en primer lugar si es bacteriana, atípica o se da en un contexto más específico. Dentro de una gran hipótesis, se intenta llegar a conocer la enfermedad que padece el paciente planteándose subobjetivos más restringidos, como es por ejemplo, el tipo de germen en el caso de la neumonía. En este proceso cohexisten diversas hipótesis con diferente grado de certeza.

Aunque en la mayoría de los centros sanitarios que en nuestro país atienden a los enfermos afectados de neumonía, se pueden realizar exploraciones complementarias, éstas no informan generalmente de la etiología de las mismas. El médico que atiende al enfermo ha de hacer una aproximación etiológica para iniciar un tratamiento.

Estas aproximaciones se efectúan bajo cierta incertidumbre. La incertidumbre puede persistir durante todo el proceso terapeútico ya que en estas enfermedades no es frecuente el diagnóstico definitivo.

En resumen, el diagnóstico y tratamiento de las neumonías es un ejemplo paradigmático de como el médico en determinadas situaciones ha de decidir ciertas actuaciones con datos incompletos, y lo que es más importante, bajo incertidumbre. El objetivo de la aplicación PNEUMONIA es orientar los casos de neumonía en las primeras 24 horas, es decir, cuando el médico no dispone de la mayoría de pruebas que le permitan afirmar o descartar un determinado diagnóstico.

Ahora considérese un ejemplo de razonamiento no monótono. Supóngase el caso de un paciente bronquítico, fumador, de 62 años con neumonía adquirida en comunidad y cocos gram positivos en parejas.

Primeramente, se asume que el sistema deduce que es muy posible que la enfermedad tenga un origen bacteriológico y sólo poco probable que tenga un origen atípico. A continuación, una meta-regla de control deduce que, para un paciente con las características citadas, el sistema intentará primero validar la hipótesis: neumonía por estreptococos (un tipo de bacteria). Ahora, se supone que sólo hay dos reglas aplicables en este caso:

R02003 SI 1) neumonía adquirida en comunidad

2) enfermedad crónica respiratoria

3) enfermedad bacteriológica

ENTONCES (posible)

NEUMONIA POR ESTREPTOCOCOS

R02026 SI 1) neumonía adquirida en comunidad

2) pleuresía

3) cocos gram positivos en parejas o en cadenas

ENTONCES (muy posible)

NEUMONIA POR ESTREPTOCOCOS

Tras aplicar estas dos reglas decide que es muy posible que la bacteria sea neumonía por estreptococos. Dos días después, se realizan rayos X y se observa que existen cavidades lo cual es evidentemente contrario a neumonía por estreptococos. Esto implica que la certeza de neumonía por estreptococos era muy alta y se debe disminuir. En este caso, se aplica la siguiente regla no monótona:

R02034 SI 1) neumonía por estreptococos

es bastante posible

2) los rayos X demuestran que hay

cavidades.

ENTONCES (seguro)

la certeza de neumonía por estreptococos es

moderadamente posible.

Esta regla obliga a disminuir el grado de certeza de neumonía por estreptococos de acuerdo a un cierto grado propagado por la regla. El resultado final es que el valor de certeza de neumonía por estreptococos es ligeramente posible.

Cuando falla un diagnóstico o un tratamiento, hay que modificarlo. Dicha modificación debe tener en cuenta la información relacionada con el fallo y el diagnóstico y tratamiento previos. Son las metarreglas, cuyas condiciones son relevantes para el diagnóstico o tratamiento erróneo, las que dirigen dicha modificación. En cualquier momento, el usuario puede introducir nueva información relevante que, dependiendo del conjunto de metarreglas, puede provocar un cambio en el proceso de búsqueda que consiste en sumar módulos o quitar módulos de la lista de objetivos. La utilización de meta-reglas de control para reglas y módulos permite diseñar una estructura jerárquica de la base de conocimiento.

La BC de PNEUMONIA tiene tres contextos diferentes: diagnóstico, tratamiento y complicaciones, supervisadas por un módulo que contiene las metarreglas orientadas a módulos. Dicho módulo permite diferentes interacciones según lo que interese al usuario. Por ejemplo, un usuario que ya sabe el diagnóstico estará interesado en el tratamiento.

Actualmente, el contexto de diagnóstico (casi completo para neumonía adquirida extrahospitalariamente) contiene diez módulos: uno de metarreglas y nueve de reglas que agrupan a las reglas por germen o familias de gérmenes.

4.6.- SITUACION ACTUAL DE MILORD

Se han descrito algunos aspectos del sistema MILORD, en concreto su arquitectura y su gestión de la incertidumbre.

De entre las aplicaciones desarrolladas sobre MILORD, una de ellas, PNEUMONIA, ha producido resultados significativos en cuanto a su competencia; ha sido la primera utilizada habitualmente por los médicos del servicio de urgencias de un hospital español: L'Alianca Mataronina.

La aplicación al diagnóstico y tratamiento de la neumonía actualmente contiene unas mil reglas y 350 hechos cubriendo más del 95% de la neumonía adquirida extrahospitalariamente y las complicaciones postratamiento.

Otras aplicaciones utilizando MILORD incluyen diseño VLSI y control automático. Las futuras extensiones de MILORD conciernen a la gestión de información imprecisa y a un interface con un sistema de adquisición de conocimiento, basado en una construcción psicológica personal.

CAPITULO 5

MEDIDAS DE EVALUACION

5.1.- INTRODUCCION

El objetivo de la evaluación es medir la ejecución de un sistema basado en el conocimiento. Los SBCs se evalúan en primer lugar, para comprobar su precisión y utilidad.

Las evaluaciones realizadas *in situ* por los expertos, ayudan a determinar la precisión del conocimiento incorporado y de cualquier respuesta o conclusión que proporcione el sistema.

Las evaluaciones de los usuarios ayudan a determinar la utilidad del sistema: si producen resultados útiles, la extensión de sus capacidades, la facilidad de interacción, la inteligibilidad y credibilidad de sus resultados, su eficiencia y velocidad, y su fiabilidad.

El proceso de evaluación debería ser continuo, comenzando con el diseño del sistema, extendiéndose a lo largo de los diferentes niveles del desarrollo y convirtiéndose en metódico de forma creciente cuando se realice la implementación en el mundo real, mediante la realización de estadísticas anuales y retrospectivas.

A continuación se proporcionan algunas pautas generales sobre cómo evaluar SBC, concentrándose en cuestiones que complican la evaluación y en factores que deben considerarse durante el diseño. Es también deseable distinguir entre dos aspectos del término 'evaluación'. En Informática, el término 'evaluación del sistema' normalmente significa optimización en el sentido técnico - estudios de tiempo, por ejemplo, que miden el tiempo necesario de UCP (CPU) o el número de accesos a disco. Por otro lado, aquí se pone énfasis en la ejecución del sistema ante una tarea específica de consulta, para la que se diseñó. A diferencia de los programas convencionales, los SBC no trabajan con problemas para los que hay una clara respuesta correcta o errónea. Por tanto, rara vez es fácil demostrar rotundamente que las respuestas del sistema son correctas y concentrarse entonces en demostrar que se alcanza la solución del problema en la forma óptima. En el siguiente punto se van a subrayar algunas cuestiones clave que afectan a la decisión sobre cómo evaluar mejor un sistema basado en conocimientos.

5.2.- EL MARCO DE LA EVALUACION

Es necesario definir un conjunto de medidas para evaluar un SBC. Esto significa considerar los siguientes problemas:

- cómo definir un patrón realista,
- cómo controlar el sesgo,
- definir necesidades de tiempo realistas para la evaluación,
- controlar las variables,
- adecuar la evaluación al nivel de desarrollo,
- definir un protocolo.

5.2.1- DEFINIR UN PATRON

Las evaluaciones necesitan algún tipo de "patrón", es decir, una respuesta aceptada en general como correcta con la cual se pueda establecer una comparación. Pero ¿cuál es este estándar?: ¿uno que pueda darlo el experto humano?, ¿uno con el que estén de acuerdo un grupo de expertos?, ¿uno que represente una solución ideal?.

En general, existen dos perspectivas sobre cómo definir un patrón:

- lo que con el tiempo llega a ser la respuesta correcta para el problema (en sentido objetivo),

- lo que el experto humano (o un grupo de ellos) presentan como respuesta correcta, partiendo de la misma información para el programa.

Por ejemplo, MYCIN se ha evaluado comparándolo con el juicio de expertos humanos mientras que CADUCEUS se ha evaluado comparando sus diagnósticos con los publicados en la literatura médica sobre algunos casos seleccionados [BUCH85]. El método anterior es el que se utiliza más frecuentemente [VERD89]. Esta es la razón por la cual los problemas descritos en adelante se refieren a este tipo de patrón.

5.2.2.- EL SESGO

En el diseño de cualquier estudio de evaluación, es importante considerar las fuentes de sesgos. Si el patrón es aquel sobre el que un grupo de expertos están de acuerdo, es necesario tener en cuenta el sesgo entre los diagnósticos de los expertos humanos. Se debe considerar también la posibilidad de que los expertos humanos puedan valorar injustamente el trabajo realizado por un programa de

ordenador porque estén en cierta forma predispuestos negativamente hacia la ciencia informática. Este sesgo puede controlarse mediante evaluaciones a ciegas. Un ejemplo de ello es un test de Turing.

5.2.3.- NECESIDADES DE TIEMPO REALISTAS SOBRE LOS EVALUADORES

Una cuestión mundana que debe considerarse porque puede conducir a fallos en el diseño o a retrasos inaceptables en la terminación de la valoración del programa, es el tiempo necesario para que los evaluadores juzguen la ejecución del sistema y para que otros suministren los datos de comprobación. Si el patrón de la evaluación es el juicio de los expertos humanos, se presenta el problema de la selección del conjunto de casos a distribuir entre los expertos para que los juzguen. Un diseño que escoja los dos o tres problemas más oportunos a valorar y que se concentre en obtener las opiniones del experto de la forma más fácil posible, tendrá más oportunidades de éxito. Además es importante estimar con precisión la cantidad de trabajo realizada por los expertos y medir el tiempo necesario para obtener sus juicios o valoraciones.

5.2.4.- DEFINIR ESTANDARES REALISTAS DE LA EVALUACION

Antes de valorar la capacidad de un SBC es preciso definir los estándares mínimos aceptables que permitirán que se considere el éxito del sistema. Irónicamente, en muchos casos es difícil decidir el nivel de ejecución que lo califica como experto, puesto que cuando el sistema alcance un resultado que esté conforme con el patrón, el sistema no va a poder decir "correcto". Es necesario definir un estándar realista de la evaluación. Por ejemplo, éste podría ser un número de casos que el sistema pudiese resolver de acuerdo con el patrón.

5.2.5.- CONTROL DE LAS VARIABLES

Otros factores que se deben tener en cuenta en la evaluación, son las necesidades y especificaciones establecidas al principio de la construcción del sistema. La evaluación del SBC debe estar de acuerdo con ellas. Por ejemplo, si el enfoque del sistema es alcanzar las decisiones óptimas en un dominio de pericia, entonces puede aceptar un fallo que refleje una inadecuación del sistema en aspectos diferentes a la toma de decisiones [GASC83].

5.2.6.- IDONEIDAD DEL NIVEL DE DESARROLLO

Junto con el control de las variables, la evaluación debe adecuarse al nivel de desarrollo. El diseño inicial del desarrollo del sistema debe ir acompañado de sentencias explícitas sobre qué medidas habrá para determinar el éxito del sistema, y como se evaluará el éxito o fallo del mismo.

5.2.7.- DEFINICION DE UN PROTOCOLO

Como resumen de los problemas de evaluación expuestos anteriormente, se puede decir que es obligatorio definir un protocolo que establezca la evaluación del SBC. Este protocolo debe reflejar qué se va a evaluar, contra qué (patrón), cuándo, cómo controlar las predisposiciones, etc.

A continuación se da una breve descripción de las medidas de evaluación que han aparecido hasta ahora en la literatura. Es posible agruparlas en dos clases: medidas relativas a la estructura y contenido, y medidas referentes a la funcionalidad.

5.3.- MEDIDAS RELATIVAS A LA ESTRUCTURA Y CONTENIDO DE LA BASE DE CONOCIMIENTOS

Las medidas pertenecientes a este grupo se obtienen examinando la BC. Son: medidas sobre el uso de elementos estructurales, inteligibilidad, comprensibilidad y medidas de complejidad de tiempo y espacio.

5.3.1.- MEDIDAS SOBRE EL USO DE ELEMENTOS ESTRUCTURALES

Esta medida se refiere al uso de las estructuras proporcionadas por el formalismo estructural y al uso de parámetros. Esto se puede llevar a cabo mediante un análisis estadístico. Con esta medida se puede detectar que, por ejemplo, un SBC utilizando un formalismo borroso no necesita lógica borrosa porque la mayoría de las reglas utilizan sólo los límites más altos o más bajos del intervalo borroso, proporcionado por el formalismo, para asignar el valor de certeza de un hecho (por ejemplo absolutamente-cierto y absolutamente-falso), es decir, en realidad son booleanos.

5.3.2.- INTELIGIBILIDAD Y COMPRENSIBILIDAD

Puesto que la utilidad de un sistema depende de los usuarios, el conocimiento final representado en el SBC debe ser tan claro como sea posible para ellos (inteligibilidad).

La comprensibilidad se define como las medidas de facilidad para comprender descriptores y para explicarlos en términos de conceptos ya conocidos por el sistema [BERG88]. Esto está fuera del alcance de los SBCs ya construidos, pero no si el SBC que está siendo estudiado tiene capacidad de aprendizaje.

5.3.3.- MEDIDAS DE COMPLEJIDAD: TIEMPO Y ESPACIO

Esta medida se refiere a la profundidad y longitud del razonamiento. Otro aspecto importante es la medida de la eficiencia de la rutina de razonamiento. Esta tiene en cuenta el número de pasos necesarios para alcanzar una conclusión, la cantidad y relevancia de la información solicitada, y el orden en el que la información es requerida. Quizá sea la medida más estudiada debido a que existe una investigación anterior sobre medidas de tiempo y espacio en gráficos, árboles, etc. Por ejemplo, basado en un árbol de prueba del

proceso de refutación en el PROLOG, Shapiro [SHAP84] define la complejidad objetivo-tamaño, la complejidad profunda, y la complejidad de longitud. El problema principal es tal que resulta necesario seguir un proceso de simulación para obtener las medidas porque están definidas como límites altos de todas las posibles interpretaciones (soluciones) del problema y de sus pruebas. Esto hace que tales medidas sean poco prácticas cuando el SBC es grande. Tao [TAO87] ha estado trabajando también en medidas de complejidad pero en una dirección diferente. Ha desarrollado EVAL, un programa de evaluación que produce los resultados de la estimación de la evaluación para una estructura de inferencia. Se han definido varias medidas:

- una medida de búsqueda horizontal total para alcanzar el objetivo final mientras que se razona (factor horizontal),

- una medida del factor de ejecución medio de los caminos de inferencia parciales mientras que se razona (factor de profundidad), y

- el máximo del factor de ejecución entre tales caminos.

Estas medidas se basan en la función f que se deriva de la transformada z de la teoría de control de realimentación (adaptada con arreglo a algunas reglas de propagación). El

mayor problema de su actual trabajo radica en determinar el factor de ejecución de cada paso de la inferencia ya que tal factor juega un importante papel en el cálculo de la función f.

5.4.- MEDIDAS RELATIVAS A LA FUNCIONALIDAD

En este grupo están las medidas que necesitan información o los resultados del comportamiento del SBC. Es decir, es necesario tener al menos un conjunto de casos resueltos para realizar las mediciones. Son: eficiencia de la ejecución, eficiencia del razonamiento (discurso, ajuste de la solución del problema, poder de predicción), razonamiento correcto, precisión, análisis de sensibilidad, utilidad y transferibilidad.

5.4.1.- EFICIENCIA DE LA EJECUCION

El enfoque de esta medida estriba en la relación existente entre el proceso de toma de decisión/compromiso de tiempo de los usuarios, es decir, el tiempo de respuesta del sistema. Precisamente uno de los factores que distinguen los

SBC para aplicaciones de procesos de control es que tienen en cuenta las medidas de eficiencia. Puesto que este tipo de SBC dependen del tiempo, es importante conocer cuánto tiempo se necesita para tener un aviso del SBC antes de realizar la ejecución. Un sistema que necesita un compromiso de tiempo excesivo para los usuarios, por ejemplo, puede no ser aceptado incluso si les exime de otras tareas. Similarmente, deben garantizarse las técnicas de análisis sobre el comportamiento del sistema. Una potencia de UCP (CPU) infrautilizada o un comportamiento de búsqueda en disco diseñado pobremente, pueden introducir ineficiencias de recursos que limiten fuertemente el tiempo de respuesta del sistema o su costo-efectividad.

Sin embargo, estos SBC normalmente tienen en su BC un conjunto explícito de reglas heurísticas, que proporcionan esta información.

Volviendo a otros tipos de SBC, se puede considerar el trabajo de Hartman [HART87] que da una medida de la ejecución con el objeto de maximizar el recurso utilizado. Su medida se refiere al costo de la computación. La idea base consiste en calcular la utilidad esperada de la estrategia de resolución de un problema sobre unos problemas de ejemplo, y luego elegir uno que maximice el costo computacional esperado.

5.4.2.- EFICIENCIA DEL RAZONAMIENTO: DISCURSO

A pesar de que la confianza en los procesos de razonamiento de un SBC es crucial para su éxito, los ingenieros del conocimiento normalmente aceptan el hecho de que existen varios parámetros que influyen en la aceptación del sistema por los usuarios involucrados. La naturalidad del discurso entre el SBC y el usuario es esencialmente importante. Algunas cuestiones relevantes son [BUCH85]:

- la elección de las palabras utilizadas en las preguntas y respuestas generadas por el sistema,

- la habilidad del SBC para explicar la base de sus decisiones y para dar apropiadamente estas explicaciones según el nivel de pericia del usuario,

- la habilidad del SBC para ayudar a los usuarios cuando están confundidos por lo que se les pide o cuando necesitan ayuda por otra razón al utilizar el programa,

- la habilidad del SBC para aconsejar o educar al usuario en los propios términos del usuario de forma que se superen las barreras psicológicas hacia la utilización del ordenador,

Por causa de cuestiones tales como las anteriores, cualquier pequeñez es importante para ultimar el éxito práctico del SBC y la calidad de sus consejos.

5.4.3.- EFICIENCIA DEL RAZONAMIENTO: AJUSTE DE LA SOLUCION DEL PROBLEMA / PODER DE PREDICCION

El término poder de predicción [BERG88] refleja el grado en el cual un descriptor aprendido puede encajar en nuevos ejemplos. Tiene una fuerte conexión con la generalidad. Extendiendo este concepto a la evaluación de los SBC puede significar considerar el grado de generalidad de una solución. Por tanto, es necesario estimar el equilibrio entre lo especifico de una solución y su poder de predicción.

5.4.4.- RAZONAMIENTO CORRECTO

No todos los diseñadores de SBC se preocupan de si sus programas toman las decisiones de la misma forma que lo harían los humanos, ni tampoco de si el consejo que ofrecen es adecuado. Sin embargo, cada vez se presta más atención a

los mecanismos según los cuales los expertos humanos resuelven los problemas. En relación con este estudio, el interface entre la ingeniería del conocimiento y psicología es mayor, y dependiendo de la motivación de los diseñadores de sistemas y de los usuarios eventuales del programa experto, puede ser apropiado prestar atención a los mecanismos de razonamiento que el programa utiliza durante el proceso de evaluación.

Por tanto la medida del 'razonamiento correcto' expresa el grado en el cual la solución se ajusta al patrón. Esta medida debe contemplar los estándares mínimos aceptados así como el estado de desarrollo.

5.4.5.- PRECISION

Se debe también distinguir entre corrección y precisión. La medida de precisión consiste en ejecutar el SBC con información típica e información atípica, y comparar los resultados para ver si la respuesta del sistema es la precisa.

5.4.6.- ANALISIS DE SENSIBILIDAD

Mide el mínimo cambio significativo en la entrada que provoque un cambio en la salida. Por ejemplo, en un SBC que trabaje con incertidumbre, el análisis de sensibilidad podría mostrar los rangos de valores del factor de certeza correspondientes a las entradas que no producen cambios en los resultados.

Un análisis de sensibilidad se realiza cambiando sistemáticamente los valores variables de las entradas y los parámetros del SBC, sobre un rango de interés, y observando el efecto sobre la ejecución del sistema.

Considérese un sistema que produce un resultado satisfactorio (es decir, si los resultados intermedios, el resultado final, y el razonamiento se consideran suficientemente expertos) cuando trabaja con un caso C. Si C utiliza las entradas (tales como datos o valoraciones del usuario) i_1, i_2, ..., i_n, entonces la validación del análisis de sensibilidad implicaría alterar cada entrada y valorar los cambios producidos en la ejecución del sistema. Por ejemplo, si i_3 es la temperatura del paciente y el caso de consulta satisfactorio incluye una temperatura de 37º centígrados, y si además, se conoce que la temperatura podría no tener ningún efecto sobre el diagnóstico de dicho caso, entonces alterando i_3 con otro valor (dejando i_1, i_2, y i_4, i_5, ..., i_n,

como estaban anteriormente) podría no producirse ninguna alteración en los resultados intermedios, en el resultado final, o quizá incluso en el proceso de razonamiento.

Los encargados del desarrollo de sistemas basados en el conocimiento no han utilizado hasta ahora el análisis de sensibilidad - al menos no explícitamente. Sin embargo, el análisis de sensibilidad puede ser el método cualitativo a mano más potente: Es especialmente útil cuando se dispone de pocos o ningún caso de prueba. Es también muy apropiado para sistemas que utilizan medidas de certeza y necesitan que los usuarios proporcionen valoraciones de la carencia de certeza establecida, ya que éstos pueden alterarse como se desee y también puede examinarse el efecto sobre las medidas de certeza intermedias y final.

5.4.7.- UTILIDAD

La utilidad final del sistema dependerá de la completitud con respecto al área del problema y la conformidad con las características del dominio [BREN89]. De ahí que dicho sistema reúna todas las condiciones necesarias para poder utilizarse en casos reales.

5.4.8.- TRANSFERIBILIDAD

La calidad del SBC determinará su posterior uso, como por ejemplo un producto comercial. Los parámetros tales como la fuente de conocimiento utilizado en la construcción del SBC, estándares, metodologías, etc., serán los encargados de determinar este potencial.

5.5.- ESTADO ACTUAL Y PROBLEMAS ABIERTOS

El estado de la validación de SBC no está estructurado ni desarrollado. La mayoría del trabajo realizado en validación se basa en investigaciones ad hoc. En evaluación esta situación es aún peor. No hay ni especificaciones sobre como dirigir la evaluación de los problemas, ni especificaciones sobre las medidas necesarias. A pesar de esta situación, algunos autores han definido medidas para solventar sus necesidades. Por ejemplo Bergadano [BERG88] define un conjunto de medidas para su sistema de aprendizaje. Pero sus medidas están influenciadas por el objetivo de aprendizaje y dependen de datos particulares de la BC. Se puede concluir entonces, que se ha realizado un pequeño esfuerzo en la evaluación de SBC. No es posible aislar la evaluación de SBC de otros aspectos de la validación. Muchas

de las medidas de evaluación incluyen conceptos y técnicas fuertemente relacionadas con la estructura y funcionalidad de los SBC. Por tanto, el desarrollo de la evaluación de SBC requiere mayores estudios en aquellos aspectos de la validación de SBC sin olvidar la investigación sobre las medidas que proporcionan un mayor entendimiento del SBC.

CAPITULO 6

MODELOS DE EVALUACION PROPUESTOS

6.1.- INTRODUCCION

Para construir la base de conocimientos en los SBC actuales, se recurre a la competencia de los expertos humanos. Esta elaboración no se realiza de una sola vez. Durante el desarrollo del sistema, el experto ve la necesidad de aportar numerosos cambios en la BC, de modificar ciertas reglas, de cambiar de opinión sobre la forma de expresar ciertos conocimientos, etc., y todo ello a la vista de los resultados obtenidos en los test realizados. La mayoría de las BC se ponen a punto probando y corrigiendo la BC existente según los distintos casos sometidos al sistema. Este procedimiento es largo, y la puesta a punto es difícil incluso para los mejores expertos. Así en algunos casos se han utilizado programas que permiten aportar una ayuda al experto en el refinamiento de las reglas de conocimiento.

En esta Tesis se proponen tres modelos de medidas para evaluar una BC en ejecución, con el objeto de obtener pautas para refinarla posteriormente y mejorar el comportamiento del sistema. Dichos modelos son:

- Medida de la robustez

- Medida de la ruta

- Factor de enfoque

6.2.- MEDIDA DE LA ROBUSTEZ DE UNA BC

La fiabilidad de un sistema basado en el conocimiento no depende únicamente de su corrección, precisión y sensibilidad, sino también de su robustez, es decir, del grado hasta el cual el SBC produce resultados significativos aplicando diferentes entradas, cada vez con menos información.

Es interesante evaluar cuál es la amplitud de las variaciones de los datos introducidos al sistema que induce a un cambio de decisión. En el caso que se estudia, los expertos son médicos y, por tanto, responsables de la salud y de la vida de su paciente. Por ello, no pueden aceptar

fácilmente las propuestas de un sistema basadas en gran parte, en opiniones subjetivas de otras personas. Si se puede demostrar que el razonamiento del sistema es relativamente insensible a las variaciones de interpretación de los elementos subjetivos, el médico usuario se sentirá más seguro al saber que trabaja con un sistema sólido.

Por consiguiente, es importante la realización de un análisis de la robustez del sistema con el objeto de obtener información sobre su comportamiento ante la falta de ciertos datos de entrada. Sobre todo conviene conocer qué datos son imprescindibles porque influyen decisivamente en la obtención del diagnóstico final (de una neumonía, por ejemplo), qué datos pueden ser menos relevantes o meramente informativos, cómo responde el sistema ante la carencia de datos suficientes para emitir algún diagnóstico, etc..

Por todo lo expuesto, a continuación se presenta un método orientado a realizar un análisis para evaluar la robustez de un sistema basado en el conocimiento y los posibles resultados y conclusiones que pueden obtenerse tras su aplicación. Dicho método se ha desarrollado utilizando el sistema MILORD y la base de conocimientos PNEUMONIA, comentados en el capítulo 4.

El modelo propuesto para analizar la robustez de una BC consiste en ejecutar sucesivas veces un número de casos considerados completos C_1, C_2, ... , C_n. Cada uno de estos

casos contiene un número de datos variable d_1, d_2, ...,d_k, referentes a las características que presenta el enfermo, y hay que tener en cuenta que cada regla en la BC está expresada en forma normal conjuntiva ($d_1 \wedge d_2 \wedge \ldots \wedge d_n$). Cada caso C_i deberá ejecutarse varias veces eliminando datos sucesivamente.

En una primera ejecución de un caso, es decir cuando el caso está completo con todos los datos, el sistema puede inferir un solo diagnóstico o bien varios diagnósticos con diferentes grados de certeza. Los grados de certeza que utiliza MILORD son nueve:

- Imposible
- Muy poco posible
- Ligeramente posible
- Moderadamente posible
- Posible
- Bastante posible
- Muy posible
- Prácticamente seguro
- Seguro

Se pretende estudiar el comportamiento del sistema ante la sucesiva falta de información, es decir, analizar las diferentes situaciones que se pueden originar al carecer el sistema de los datos necesarios para emitir su diagnóstico, o para emitirlo con un grado de certeza suficiente.

En un principio, se pueden considerar varias posibilidades:

- El sistema cada vez puede emitir menos diagnósticos.

- El sistema cada vez emite sus diagnósticos con menor confianza.

- Puede llegar un momento en que el sistema se pare porque no tenga datos suficientes para dar un diagnóstico.

- Se emite un mensaje de posible error en los datos del caso.

Para la medida de la robustez de una BC se han considerado varias etapas:

a) Ejecución de los casos y cálculo

b) Representación gráfica

c) Análisis de resultados

6.2.1.- EJECUCION DE LOS CASOS Y CALCULO

La primera parte consiste en ejecutar un conjunto de n casos completos. Los datos de cada caso estarán almacenados en ficheros a los que se irá accediendo sucesivamente. El proceso consta de un bucle global que se repite tantas veces como casos se tengan que examinar.

Para cada caso concreto, se realizarán los siguientes pasos:

1.- Se ejecuta el caso completo, es decir, suministrando todos sus datos, y se obtiene el resultado consistente en uno o más diagnósticos con sus factores de certeza asociados.

2.- Se suprime un dato del caso anterior y se vuelve a ejecutar el caso, que en esta ocasión constará de n-1 datos.

3.- Se comparan los resultados obtenidos en los pasos 1 y 2, tomándose las siguientes acciones:

3.1.- Si los resultados de las dos ejecuciones son iguales, continuar en el punto 4.

3.2.- Si los resultados de las dos ejecuciones son diferentes, se deberá indicar por pantalla y además se registrará dicho suceso en un fichero de resultados,

señalando el dato que se ha eliminado y la variación observada - disminución del número de diagnósticos finales y sus valores de certeza asociados.

4.- Se elimina otro dato y se vuelve a ejecutar el caso. A continuación se repite el proceso de comparación respecto a la ejecución previa realizada.

La ejecución de un determinado caso puede repetirse tantas veces como datos se haya dispuesto eliminar, o bien puede imponerse una condición de finalización tal como:

- Terminar cuando los resultados no sean significativos, por ejemplo, cuando los diagnósticos finales no superen un grado de certeza establecido.

- Terminar cuando el sistema no pueda dar una solución al problema por no disponer de información suficiente.

Además de las diferencias, también se contabilizan los cambios registrados para la obtención de una estadística final:

- Qué datos producen variaciones
- Número total de variaciones observadas
- En qué momento el sistema no puede dar un resultado

Por ejemplo, supóngase un caso C_i para el cual los datos del enfermo son los indicados en la figura 6.1.

Nótese que la primera columna contiene el nombre de las variables, la segunda su significado, y la tercera el valor asignado para el caso C_i. El resultado obtenido por MILORD para dicho caso con todos los datos (m = 0, siendo m el número de datos suprimidos), consiste en los siguientes diagnósticos:

DIAGNOSTICO	CERTEZA
Anaerobio	Posible
Legionella	Moderadamente posible
Tuberculosis	Moderadamente posible
Virus	Moderadamente posible
Enterobacterias	Ligeramente posible
Haemophilus	Ligeramente posible
Micoplasma	Ligeramente posible
Chlamydia	Ligeramente posible

Si se supone que en el caso anterior se realiza un proceso iterativo, eliminando sucesivamente un dato, se observan las siguientes variaciones:

VARIABLE	SIGNIFICADO	VALOR
NE	Neumonía extrahospitalaria	S
ESTAT-MALALT	Estado del enfermo	G
COMA	Coma	N
DISPNEA	Dificultad respiratoria	S
HIPOTENSIO	Hipotensión	N
DESORIENTAT	Desorientación	N
DIES-INICI	Nº días desde primeros sint.	90
DIES	Tiempo desde diagnóstico	0
DIA	Día	4
MES	Mes	5
ANY	Año	89
SEXE	Sexo	H
EDAT	Edad	52
AP-INTERES	Existencia antec. patológicos	S
TIPUS-AP-INTERES	Antecedentes patológicos	OH
ABS	Antibióticos previos	N
INICI	Forma de inicio	SU
TOS	Tos	S
EXPECTORACIO	Expectoración	S
ESPUT	Tipo de esputo	MU
DOLOR-PLEURITIC	Dolor pleuritico	S
TEMPERATURA	Temperatura	38.09
FRE-CARD	Frecuencia cardiaca	108
TE-ASSOC	Existencia de datos clínicos	S
ASSOCIADES-GRAL	Tipo de datos clínicos	GI, RES
ASSOCIADES-GI	Datos cl. gastrointestinales	HD
RX-FET	Existencia de radiografías	S
DADES-RX	Datos radiológicos	INTER NTX VMALDEF CONFLUENT
RX-DISTR	Distribución infiltrados Rx	BRNEUM DF
BRNEUM-S	Certeza en inf. bronconeumón.	MOL-P
D-S	Certeza en inf. difuso	SEGUR
ANALAB	Existencia de analítica	S
LEUCOCITS	Nº leucocitos en sangre	11300
POLIS	Polinucleares totales	9040
PP	Porcentaje de polinucleares	80
BANDA	Polimorfonucleares en banda	1
GASOS	Existencia de Gasometría	S
PO2	Presión parcial de oxígeno	49
PCO2	Presión parcial de CO2	30
PH	Ph	7,4
BIC-ST	Bicarbonato estándar	21
B-E	Exceso de base	-3,8
EX-ESPUT	Examen de esputo	N
ALTREPN	Convivencia con otras neumon.	N
FET-Z-N	Efectuada tinción Ziehl-Nil.	N
EVOLUCIO	Evolución	PNCR
ADNOH	Adenopatías hiliares	N

Figura 6.1.

- Para m = 1, eliminando el dato COMA (es decir, la información sobre si el paciente está en coma), se observa que el resultado de la ejecución no varía.

- Para m = 2, es decir, si una vez eliminado el dato COMA se suprime también el dato ABS (es decir, la información sobre si el paciente ha tomado antibióticos previos), el resultado sigue sin variar.

- Para m = 3, eliminando también el dato TOS (es decir, la información sobre si el paciente tose o no) tampoco varía el resultado.

- Para m = 4. eliminando el dato DOLOR-PLEURITC (dolor pleurítico), se observa en siguiente resultado:

DIAGNOSTICO	CERTEZA
Anaerobio	Posible
Legionella	Moderadamente posible
Tuberculosis	Ligeramente posible
Virus	Moderadamente posible
Enterobacterias	Ligeramente posible
Micoplasma	Ligeramente posible
Chlamydia	Ligeramente posible

Es decir, hay un diagnóstico que se han suprimido de la solución (Haemophilus) y uno que continúa pero con un grado de certeza menor (Tuberculosis).

- Para m = 5, suprimiendo el dato INICI (forma de inicio de la neumonía) también se observan variaciones:

DIAGNOSTICO	CERTEZA
Anaerobio	Ligeramente posible
Legionella	Moderadamente posible
Tuberculosis	Ligeramente posible
Virus	Moderadamente posible
Enterobacterias	Ligeramente posible
Micoplasma	Ligeramente posible
Chlamydia	Ligeramente posible

Es decir, la certeza del diagnóstico Anaerobio ha disminuido de Posible a Ligeramente Posible.

- Para m = 6, eliminando el dato TEMPERATURA no se observan variaciones.

- Para m = 7, eliminando el dato ESTAT-MALALT (estado del enfermo), se obtiene:

DIAGNOSTICO	CERTEZA
Anaerobio	Ligeramente posible
Tuberculosis	Ligeramente posible
Virus	Moderadamente posible
Enterobacterias	Ligeramente posible
Micoplasma	Ligeramente posible
Chlamydia	Muy poco posible

Es decir, desaparece el diagnóstico Legionella y disminuye de certeza el diagnóstico Chlamydia (de Ligeramente posible a Muy poco posible).

- Para m = 8, eliminando el dato DISPNEA (dificultad respiratoria, el resultado es igual que en el caso anterior.

- Para m = 9, eliminando el dato DESORIENTAT (desorientación o confusión del enfermo), tampoco varía el resultado.

- Para m = 10, eliminando AP-INTERES (antecedentes patológicos de interés) el resultado es el siguiente:

DIAGNOSTICO	CERTEZA
Anaerobio	Ligeramente posible
Tuberculosis	Ligeramente posible

Es decir, han desaparecido cuatro diagnósticos de la solución, y los que perduran lo hacen con un nivel de confianza muy bajo. En este punto puede considerarse que dichos resultados no son ya significativos.

El proceso puede continuar hasta que el sistema se pare o bien hasta que el resultado no sea significativo. En este trabajo se ha considerado que los resultados no son significativos por debajo de un grado de certeza de '*moderadamente posible*'.

Una vez terminado este proceso se pasará a la segunda parte.

6.2.2.- REPRESENTACION GRAFICA

Esta segunda parte consiste en la representación gráfica de los resultados obtenidos en la etapa anterior, para una mayor facilidad visual en la interpretación. Las gráficas reflejan la relación existente entre la eliminación sucesiva de ciertos datos del caso y la variación experimentada en la emisión del diagnóstico por parte del SBC. De esa forma se puede apreciar claramente en qué momento los resultados dejan de ser satisfactorios, por no superar el mínimo de confianza establecido.

Se utilizan ejes coordenados en los cuales se representa:

- en el eje de abcisas el valor de m, es decir, el número de las datos que se han ido suprimiendo, partiendo de los n datos iniciales de que consta el caso completo.

- en el eje de ordenadas los factores de certeza.

- en el área comprendida entre ambos ejes, la evolución de cada diagnóstico emitido en la situación inicial al ir suprimiendo datos sucesivamente.

Por ejemplo, la figura 6.2. representa los resultados obtenidos en la ejecución del caso C_i de la sección 6.2.1. mediante un diagrama de barras, muy utilizado en el entorno médico. Puede apreciarse que para m=0, es decir, en la primera ejecución disponiendo de todos los datos de entrada, la respuesta del sistema consiste en ocho diagnósticos. representados cada uno mediante una barra de diferente tonalidad, de los cuales sólo cuatro (Anaerobio, Legionella, Enterobacterias y Tuberculosis) alcanzan el grado de certeza 'moderadamente posible'. A medida que se van eliminando datos, el número de barras (diagnósticos) tiende a desaparecer o a disminuir de altura (grado de certeza). Así, al llegar a m=10 solamente se obtienen dos diagnósticos como respuesta (Anaerobio y Tuberculosis), y ambos con un grado de

certeza insuficiente para considerarlos significativos. A continuación se muestran en la figura 6.3., los mismos resultados utilizando una representación por áreas donde también se observa la tendencia descendente de la respuesta del sistema.

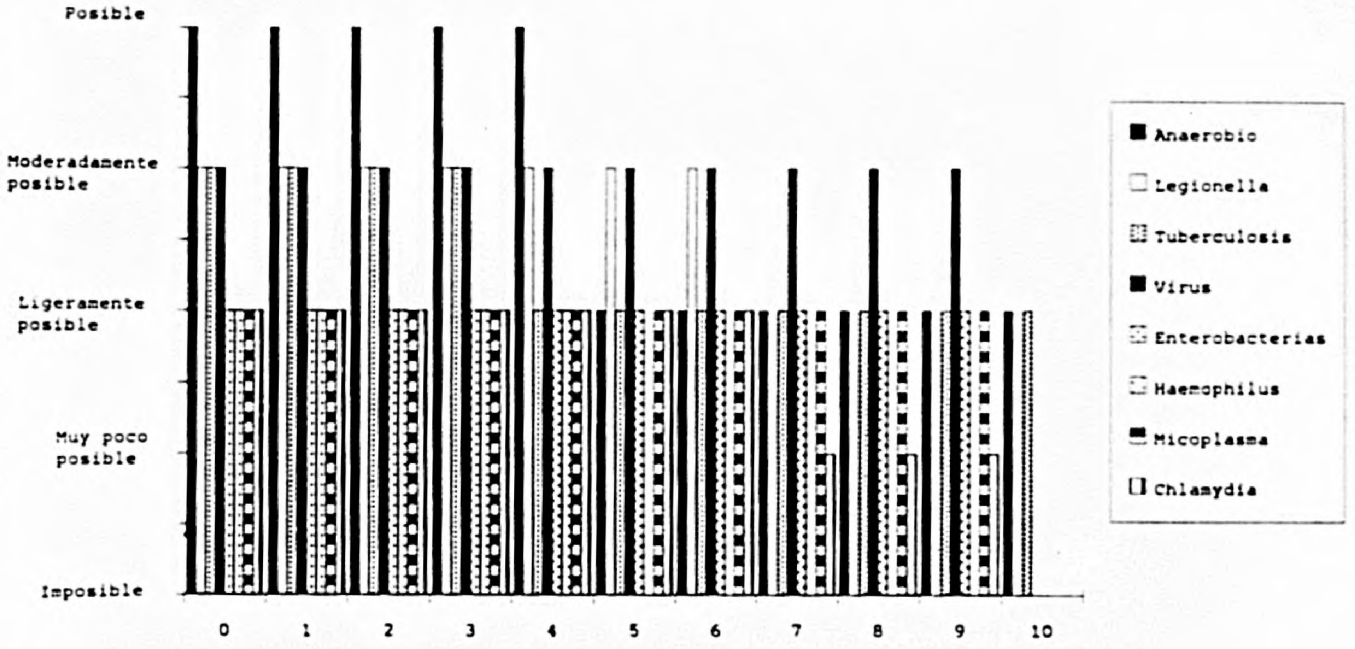


Figura 6.2.

Representación gráfica de los resultados obtenidos en la ejecución de un caso C_i mediante diagrama de barras.

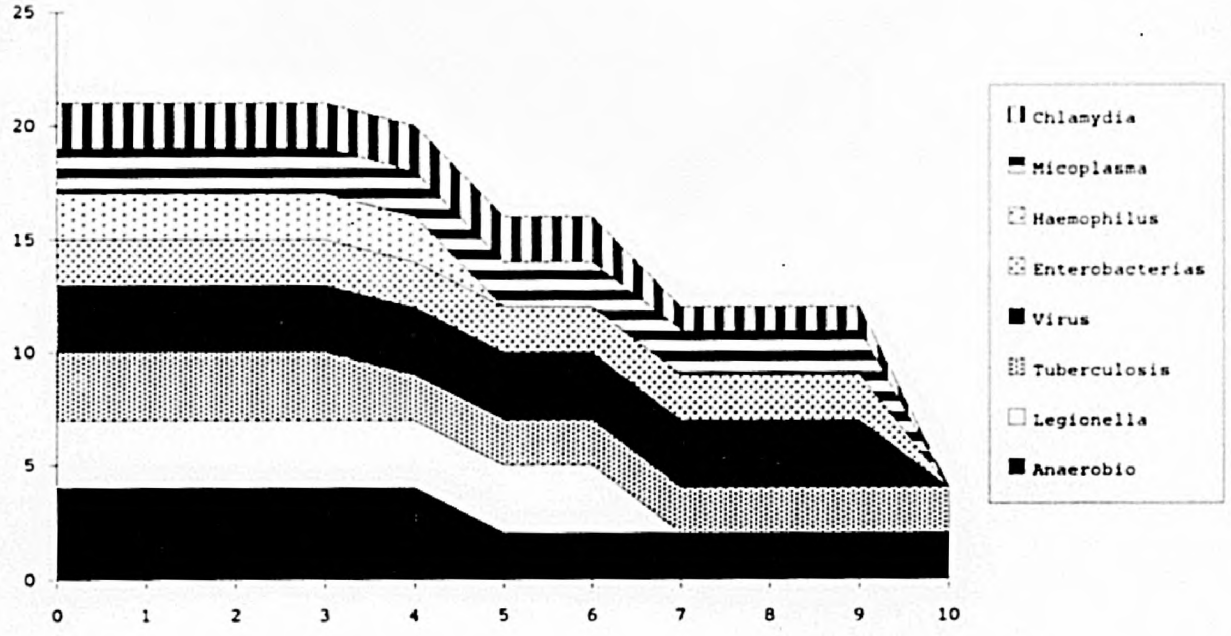


Figura 6.3.

Representación gráfica de los resultados obtenidos en la ejecución de un caso C_i mediante áreas.

6.2.3.- ANALISIS DE RESULTADOS

Esta tercera parte consiste en la observación de los resultados obtenidos en las sucesivas ejecuciones y en la obtención de conclusiones a partir de ellos.

En este trabajo se han procesado un total de 40 casos sobre diagnóstico de neumonía adquirida extrahospitalariamente. Dichos casos han sido proporcionados por el Centro de Estudios Avanzados de Blanes donde también se están realizando investigaciones en el área de validación de Bases de Conocimientos. El método propuesto se ha aplicado mediante un programa LISP (Apéndice 1), observándose los siguientes resultados:

a) La gráfica resultante del proceso aplicado muestra que la tendencia del comportamiento del sistema al suprimir información es disminuir el número de diagnósticos inferidos y los factores de certeza asociados a los mismos. Una vez decidido el grado de confianza a partir del cual los resultados no se consideran significativos, se puede apreciar cuántos datos de entrada pueden eliminarse manteniendo la robustez del sistema. En este trabajo se ha considerado que por debajo del grado de certeza 'moderadamente posible', los resultados no son significativos. La tabla de la figura 6.4. muestra los resultados en la ejecución de 40 casos por MILORD, indicando el número inicial de datos y el número

límite de eliminación de datos hasta el cual los resultados obtenidos son significativos. Por ejemplo, un LIMITE=5 indicaría que se podrían suprimir 5 datos de entrada y los resultados continuarían siendo significativos, pero que al eliminar el sexto ya no lo serían. Se puede observar que dicho sistema emite resultados significativos hasta una media de LIMITE=20, es decir, hasta la supresión de 20 datos, teniendo en cuenta que cada caso dispone originalmente de una media de 45 datos. Por tanto, se puede considerar que para cada caso son necesarios 25 datos por término medio.

b) Existen ciertos datos que al eliminarlos, producen variaciones más fuertes en el comportamiento del sistema. La figura 6.5. muestra una tabla que refleja las variaciones observadas en los diagnósticos finales referentes a la supresión sucesiva de los datos indicados, sobre un total de casos ejecutados que se especifica en la última columna. (Las variaciones se refieren tanto a variaciones en el número de diagnósticos como en los grados de certeza asociados a los mismos). Por ejemplo, el dato AP-INTERES (antecedentes patológicos de interés) produce variaciones en el diagnóstico final el 67% de las veces que se suprimió como dato de entrada. Esto implica que la información sobre los antecedentes patológicos del enfermo (diabetes, hábitos tóxicos como tabaco o alcohol, mieloma, etc.) son datos de primer interés para orientar una neumonía en las primeras horas de atención al enfermo.

c) Existen ciertos datos que apenas producen variación en el comportamiento del sistema, lo que puede inducir a pensar que son menos relevantes o incluso meramente informativos. Por ejemplo, observando la tabla de la figura 6.5., puede apreciarse que el dato ABS (antibióticos previos) sólo ha producido variaciones un 2,5% de las veces en que se ha eliminado como dato de entrada, lo que implica que no es un dato de los más decisivos a la hora de tenerlo en cuenta para el diagnóstico. También puede considerarse el dato SEXO que no ha producido ninguna variación, lo que supone que dicha información no es necesaria porque no interviene para la emisión del diagnóstico.

d) En algunos casos, la sucesiva falta de información provoca una disminución en el número de diagnósticos finales mientras que en otros llega un punto en que el número de diagnósticos empieza a aumentar, aunque todos con un grado de certeza muy bajo. Esto significa que en ese punto el sistema ha obtenido una respuesta formada por diagnósticos inconsistentes porque la información de la que dispone no es suficiente. Por tanto, dichos resultados no deberían considerarse muy fiables.

e) En los casos en los que no hay suficiente información para emitir un diagnóstico, el sistema no infiere nada.

Todo esto debería tenerse en cuenta para, por ejemplo, eliminar datos superfluos que no aporten gran cosa y hagan perder el tiempo al usuario, en el caso de una consulta

rápida, o bien para solicitar siempre primero aquellos datos más relevantes o decisivos de cara la diagnóstico final. En el caso de utilización de la misma BC con distintos fines, por ejemplo académicos, por un lado, y consulta real rápida, por otro, podría ser interesante incorporar al sistema dos modalidades de uso:

- una para orientación rápida en la que sólo se solicitasen los datos más imprescindibles,

- otra para un análisis más exhaustivo disponiendo de todas las pruebas.

CASO	TOTAL DATOS	LIMITE (MOD-P)	NºDATOS NECESARIOS
CASO 1	46	4	42
CASO 2	59	9	50
CASO 3	60	34	26
CASO 4	38	20	18
CASO 5	33	10	23
CASO 6	41	26	15
CASO 7	38	21	17
CASO 8	52	23	29
CASO 9	45	28	17
CASO 10	57	9	48
CASO 11	40	13	27
CASO 12	52	6	46
CASO 13	43	12	31
CASO 14	41	11	30
CASO 15	43	26	17
CASO 16	53	26	27
CASO 17	45	9	36
CASO 18	50	29	21
CASO 19	54	26	28
CASO 20	48	29	19
CASO 21	56	9	47
CASO 22	43	31	12
CASO 23	52	29	23
CASO 24	32	22	32
CASO 25	46	30	16
CASO 26	55	10	45
CASO 27	52	31	21
CASO 28	43	22	21
CASO 29	52	9	43
CASO 30	52	26	26
CASO 31	41	28	13
CASO 32	42	25	17
CASO 33	34	23	11
CASO 34	35	7	28
CASO 35	41	27	14
CASO 36	35	26	9
CASO 37	44	24	20
CASO 38	36	24	12
CASO 39	34	25	9
CASO 40	38	25	13

Figura 6.4.

DATO ELIMINADO	VARIACIONES PRODUCIDAS	TOTAL CASOS	PORCENTAJE VARIACIONES
ESTADO DEL ENFERMO	20	38	52,6%
COMA	7	40	17,5%
DIFICULTAD RESPIRAT.	5	27	18,5%
DESORIENTACION	0	40	0%
DIAS DESDE INICIO	0	40	0%
DIAS DESDE DIAG. NEUM	0	39	0%
SEXO	0	40	0%
EDAD	0	40	0%
ANTEDENTES PATOLOG.	25	37	67,5%
TIPO ANTECEDENTES PAT	9	22	40,9%
ANTIBIOTICOS PREVIOS	1	40	2,5%
FORMA DE INICIO	13	38	34,2%
TOS	7	40	17,5%
EXPECTORACION	4	36	11,1%
TIPO DE ESPUTO	5	22	22,7%
DOLOR PLEURITICO	17	38	44,7%
TEMPERATURA	3	39	7,6%
FRECUENCIA CARDIACA	0	38	0%
HIPOTENSION	0	10	0%
SIGNOS CONSOLIDACION	2	37	5,4%
DATOS CLINICOS ASOC.	8	38	21%
DATOS CLINICOS GRAL.	0	13	0%
EXISTENCIA RADIGRAF.	0	40	0%
DATOS RADIOLOGICOS	23	26	88,4%
DISTR. INFILTRAD. RX	10	21	47,6%
ANALISIS DE LABORAT.	0	40	0%
NºLEUCOCITOS	10	21	47,6%
NºPOLINUCLEARES	7	21	33,3%
PORCENTAJE POLINUC.	6	21	28,5%
EXISTENCIA GASOMETRIA	0	12	0%
CONVIVENCIA CON NEUM.	0	33	0%
NEUMONIAS MISMO EDIF.	0	29	0%
CONTACTO CON TUBER.	0	12	0%
EFECTUADA TINCION Z-N	0	9	0%
ADENOPATIAS HILIARES	1	8	1,25%

Figura 6.5.

6.3. MEDIDA DE LA RUTA

En toda base de conocimientos es importante considerar la disposición de los hechos y las reglas a la hora de encadenarse, para llegar a un objetivo. Con la medida de evaluación de la ruta se pretende estudiar el camino que sigue la traza de las reglas disparadas en la ejecución de un caso, para evaluar si esa ruta va más o menos directa hacia el objetivo o si realiza muchas desviaciones. En el caso de un SBC que tuviese que explorar muchas reglas inútiles cada vez, para poder dar una solución al problema, sería conveniente modificarla de tal forma que se optimizase dicho proceso.

Para representar la medida de la ruta se ha utilizado una función que representa el número de reglas disparadas en el proceso de inferencia. Obsérvese, por ejemplo, la figura 6.6., que representa la ruta que sigue la traza de un caso C_i que se dirige de forma clara hacia el objetivo, mientras que la ruta trazada en la figura 6.7. tiene muchas dificultades en alcanzarlo, pues explora muchas reglas que no necesita. Se pretende presentar un modelo que permita comparar las rutas originadas por las trazas de las reglas con el propósito de decidir cuáles se adaptan más al camino óptimo.

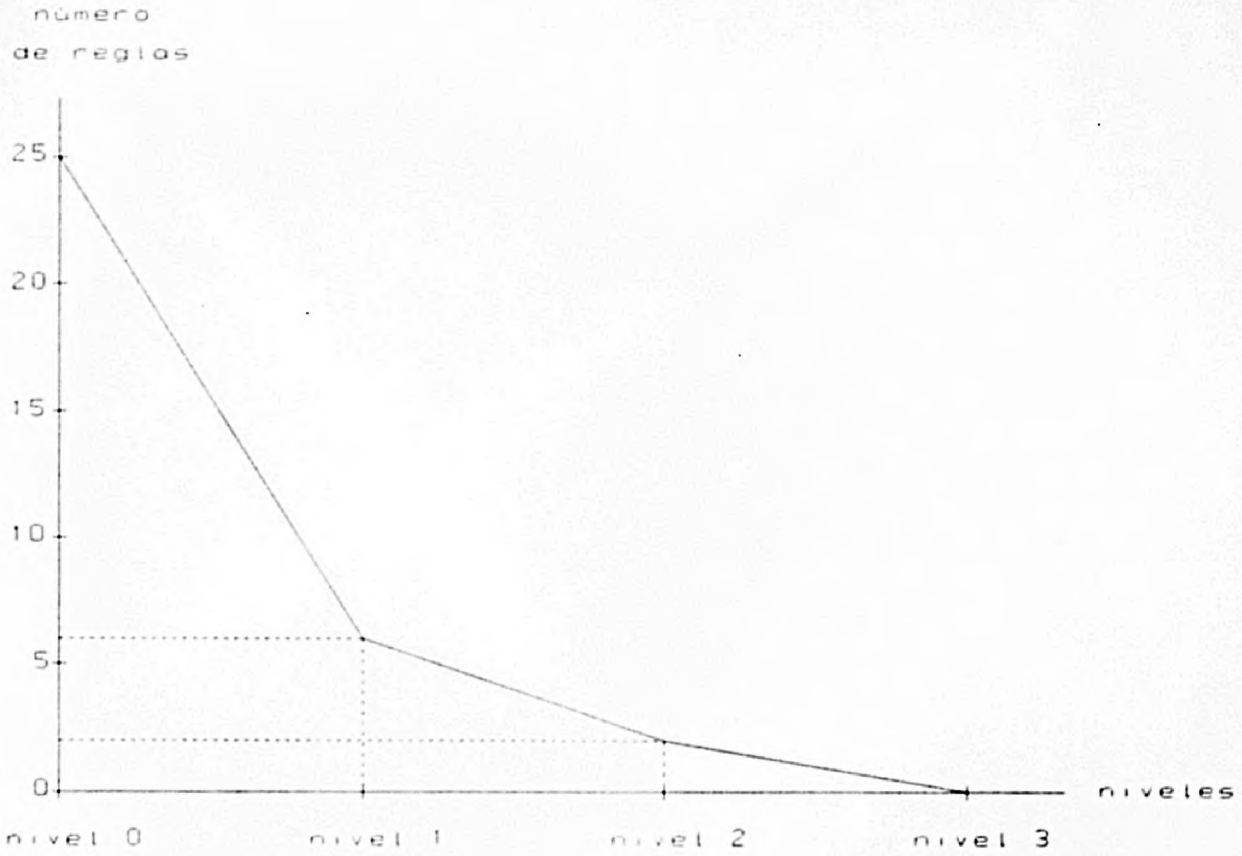


Figura 6.6.

Ruta óptima.

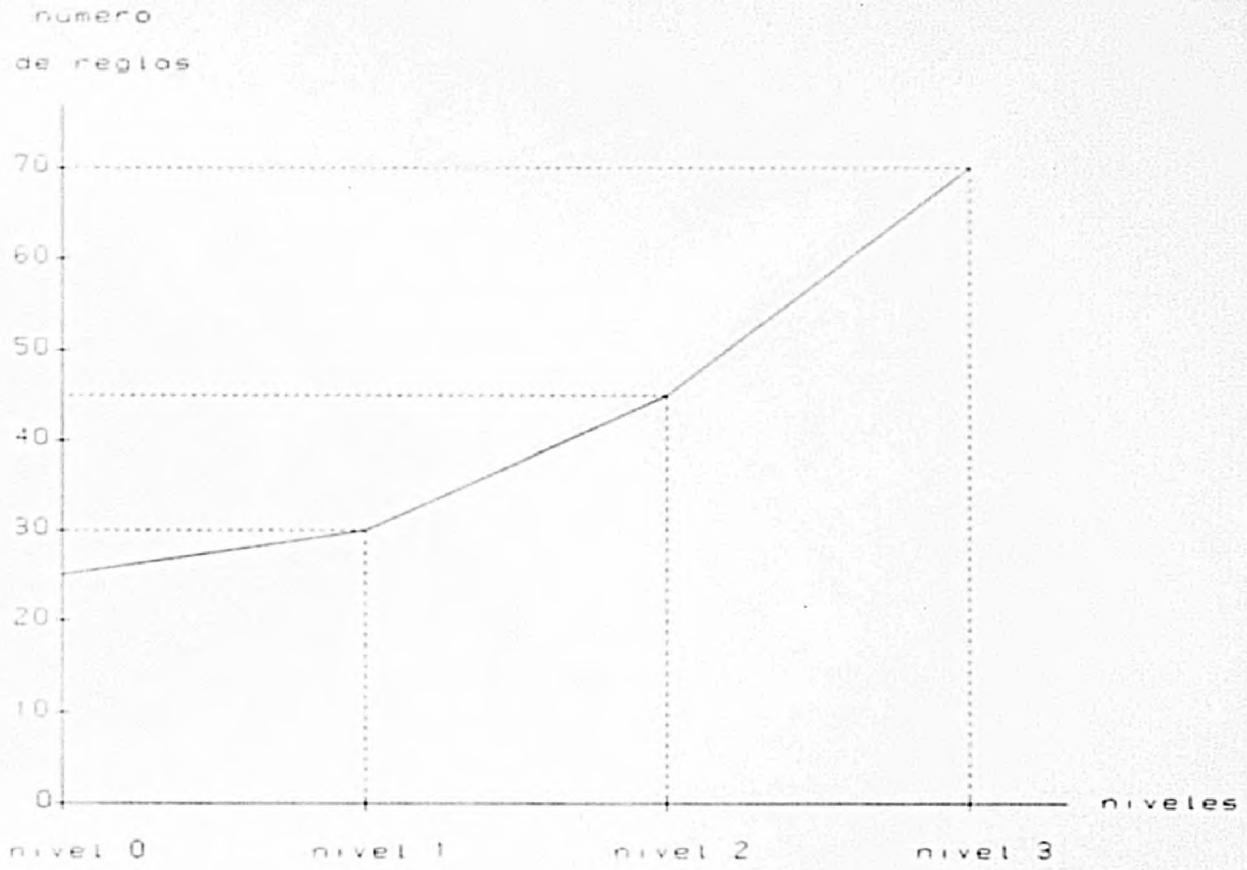


Figura 6.7.

Ruta deficiente.

6.3.1.- OBTENCION DE LA TRAZA

El primer paso para la aplicación de esta medida de evaluación es la obtención de la traza que siguen las reglas en cada caso que se ejecuta. Hay que tener en cuenta que las reglas son aserciones dadas en forma de implicaciones que se pueden expresar en la forma:

SI <condiciones>
ENTONCES <acciones>

La activación de las reglas constituye una cadena de acciones dirigida por Modus Ponens. La organización y el acceso a las reglas son importantes. El acceso puede variar desde un esquema muy simple, en el cual las reglas son utilizadas en un orden predeterminado, hasta esquemas más complejos que integran dispositivos que permiten resolución de conflictos, es decir, procesos dinámicos de selección de la regla activa.

Se partirá de un total de n casos de forma que pueda observarse el comportamiento general del sistema. En este trabajo, se ha realizado dicho análisis utilizando 40 casos de neumonía.

Por cada caso se realizará lo siguiente:

- Ejecución del caso registrando en un fichero las reglas que se van disparando y su secuencia.

- Obtención del encadenamiento de dichas reglas o traza de reglas. En este paso se obtendrá un árbol de reglas. Por ejemplo, dado un caso C_i, se puede obtener la traza representada en la figura 6.8.

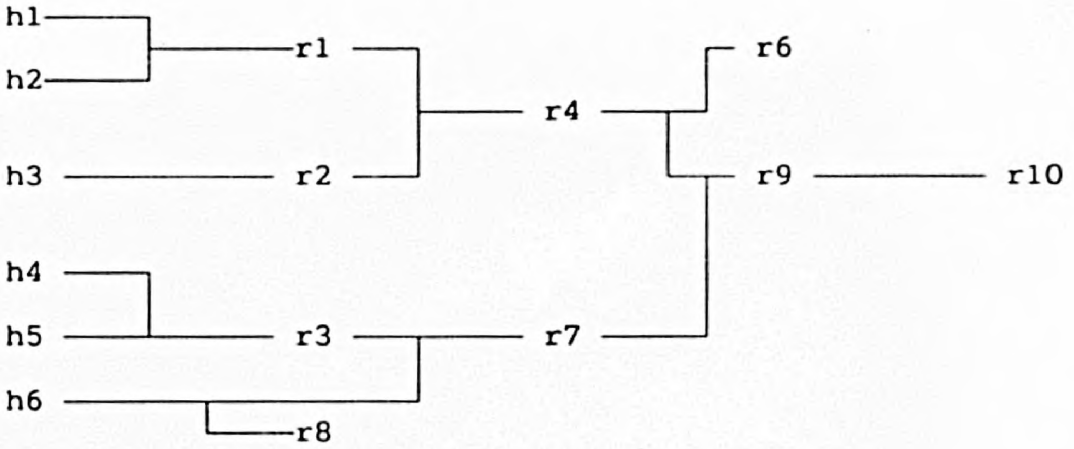


Figura 6.8.

Traza de reglas.

Se observa que se han disparado 10 reglas, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, siendo r10 el diagnóstico final inferido. De dichas reglas r6 y r8 no participan para la consecución del objetivo final.

6.3.2.- EVALUACION DE LA RUTA

Para evaluar la ruta que ha seguido la traza, habrá que tener en cuenta las reglas que se han disparado y cuáles de ellas no pertenecen al conjunto solución, es decir, cuales no se han utilizado para alcanzar el diagnóstico final. Por tanto, habrá que contabilizar en la ejecución dichos valores. Se propone establecer una relación entre el número de reglas utilizadas para alcanzar la solución y el número de reglas disparadas. El porcentaje de reglas utilizadas en el conjunto solución respecto del total de reglas disparadas en el proceso de inferencia puede ser significativo para evaluar la desviación general de la ruta que siguen los casos en una BC.

Además se ha utilizado una función para representar la ruta seguida por un caso en función de las reglas disparadas en su ejecución. Esta función se construye asignando las reglas a diferentes niveles dependiendo de su pertenencia o no al conjunto solución, y de su alejamiento al mismo. Es decir, una vez extraídas las reglas disparadas en la ejecución, se asignarán a los siguientes niveles:

NIVEL **VALOR**

Nivel 0: Reglas que pertenecen al conjunto solución.

Nivel 1: Reglas que no pertenecen al conjunto solución y su alejamiento es de primer grado.

Nivel 2: Reglas que no pertenecen al conjunto solución y su alejamiento es de segundo grado.

Nivel n: Reglas que no pertenecen al conjunto solución y su alejamiento es de grado n.

El *alejamiento del conjunto solución* se ha definido como sigue: Si para que se dispare la regla *r* se han disparado por delante de ella al menos n reglas a partir de la última que pertenecía al conjunto solución, el *grado de alejamiento de r es n*. En el caso de una regla que se obtenga directamente a partir de los datos de entrada, se considera su grado de alejamiento 1. En el caso de una regla que tenga como antecedentes a dos o más reglas, para calcular el alejamiento se tomará el camino más corto.

Una vez contabilizado el número de reglas de cada nivel, se obtiene la representación gráfica que indica la ruta que ha seguido la traza de las reglas.

Por ejemplo, supóngase que se ha obtenido la traza representada en la figura 6.9. al ejecutar el caso C_x.

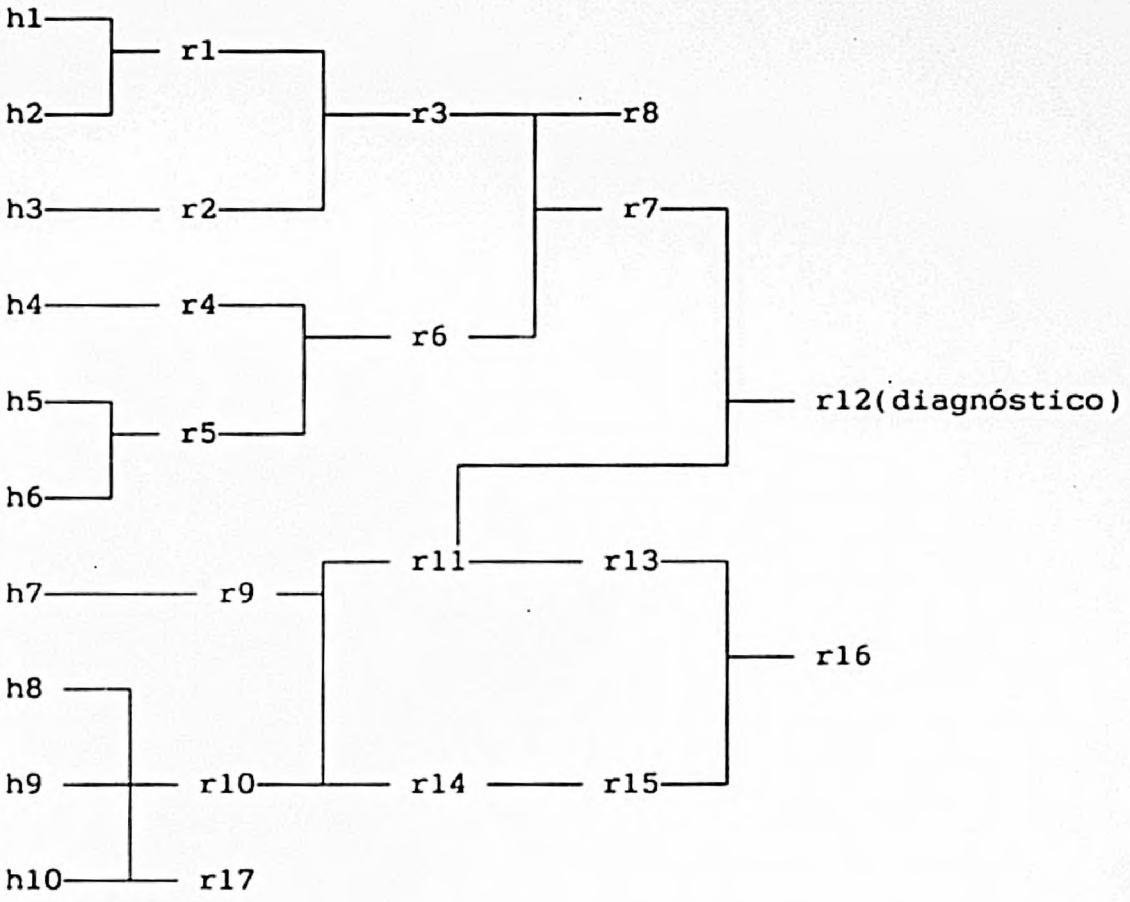


Figura 6.9.

Se tienen los siguientes niveles de reglas:

Nivel 0: r1, r2, r3, r4, r5, r6, r7, r9, r10, r11, r12

Nivel 1: r8, r13, r14, r17

Nivel 2: r15, r16

a los cuales les corresponden los valores:

Nivel 0: 11 reglas

Nivel 1: 4 reglas

Nivel 2: 2 reglas

La gráfica correspondiente se representa en la figura 6.10., teniendo en cuenta que en el eje de abcisas se representan los niveles y en el de ordenadas, el número de reglas correspondiente a cada uno de ellos. Esta gráfica se acompaña de un diagrama de barras (Figura 6.11) que representa claramente el número de reglas disparadas por cada nivel y su tendencia descendente. Ambas indican que dicha solución se adapta bastante a la solución óptima especificada en la figura 6.6.

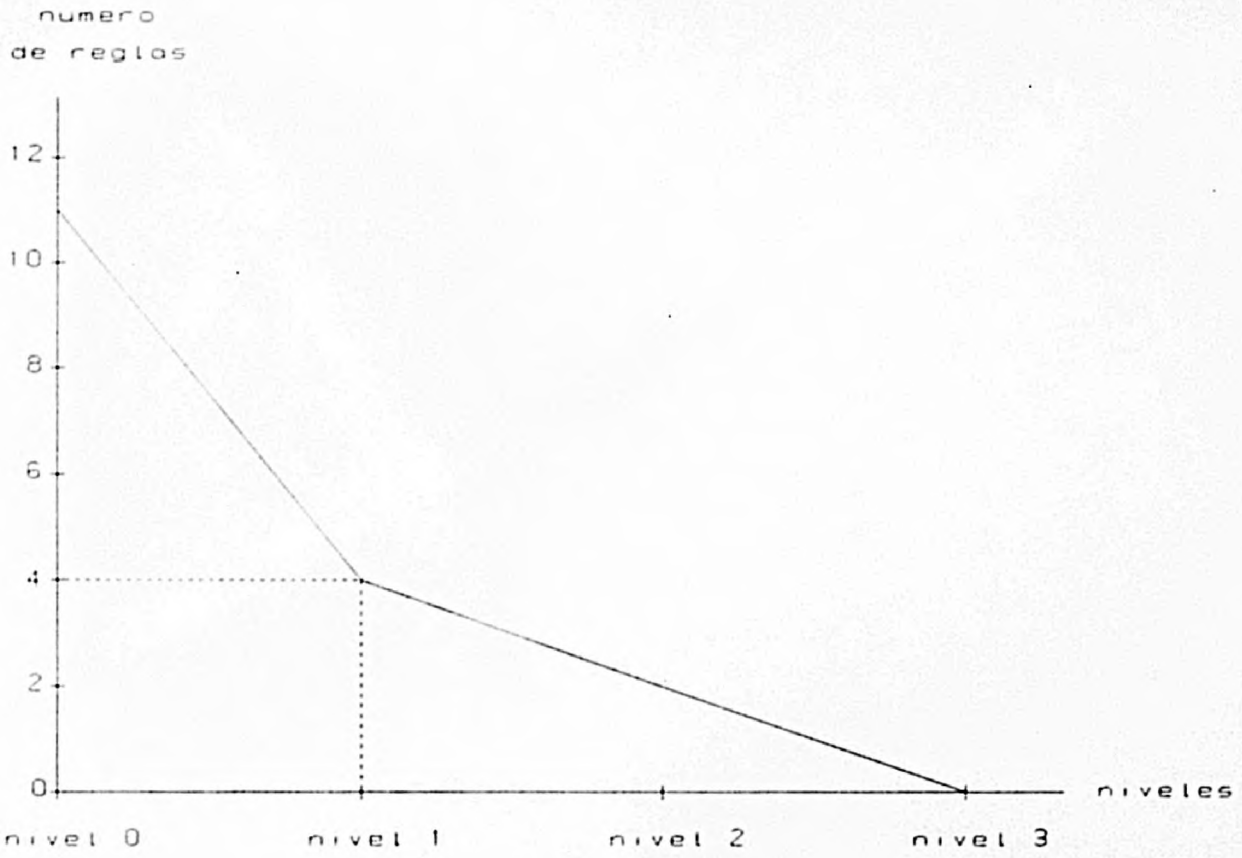

numero
de reglas
12
10
8
6
4
2
0
niveles
nivel 0
nivel 1
nivel 2
nivel 3

Figura 6.10.

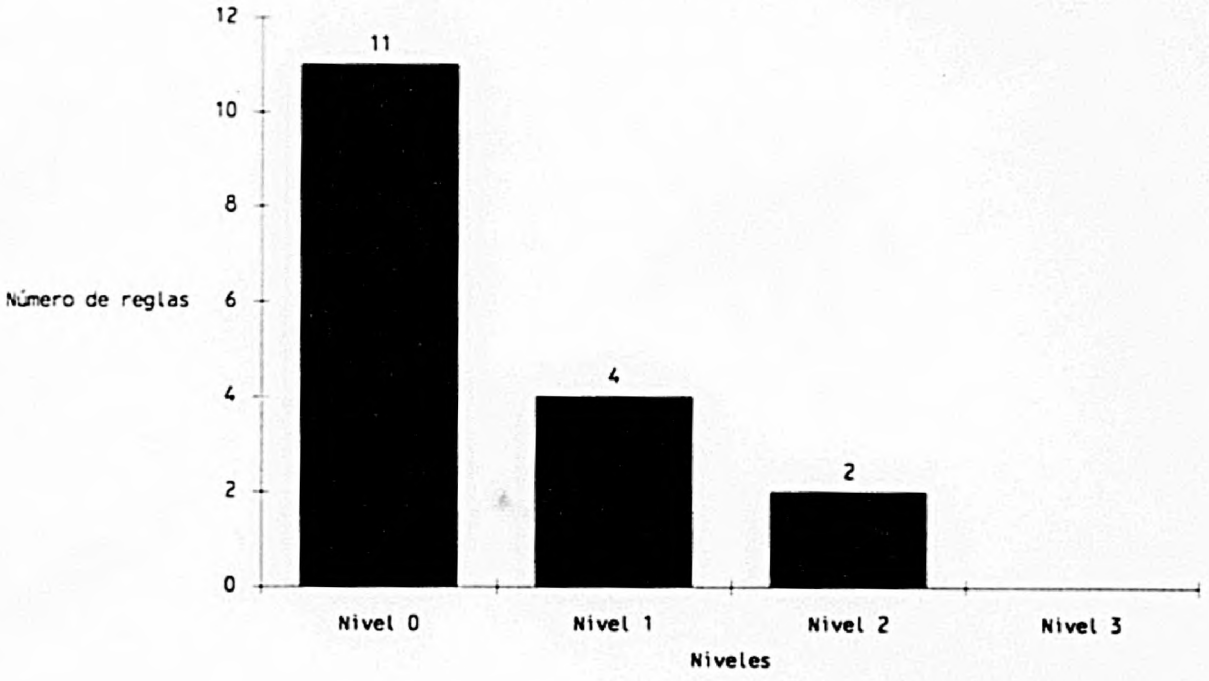

12
10
8
6
4
2
0
Número de reglas
11
4
2
Nivel 0
Nivel 1
Nivel 2
Nivel 3
Niveles

Figura 6.11.

Sin embargo, considérese ahora la traza de un caso C_y, representada en la figura 6.12., en la cual las reglas pertenecen a los siguientes niveles:

NIVEL 0: r2, r3, r4, r9, r13

NIVEL 1: r0, r1, r8, r5, r6, r14, r10, r11

NIVEL 2: r21, r12, r18, r15, r16, r20, r17, r22, r19

Cuyos valores correspondientes serán:

NIVEL 0: 5 reglas

NIVEL 1: 8 reglas

NIVEL 2: 9 reglas

Las gráficas correspondientes, se reflejan en las figuras 6.13. y 6.14., observándose en este caso que la ruta es deficiente debido al aumento de reglas en cada nivel y la tendencia creciente de la curva.

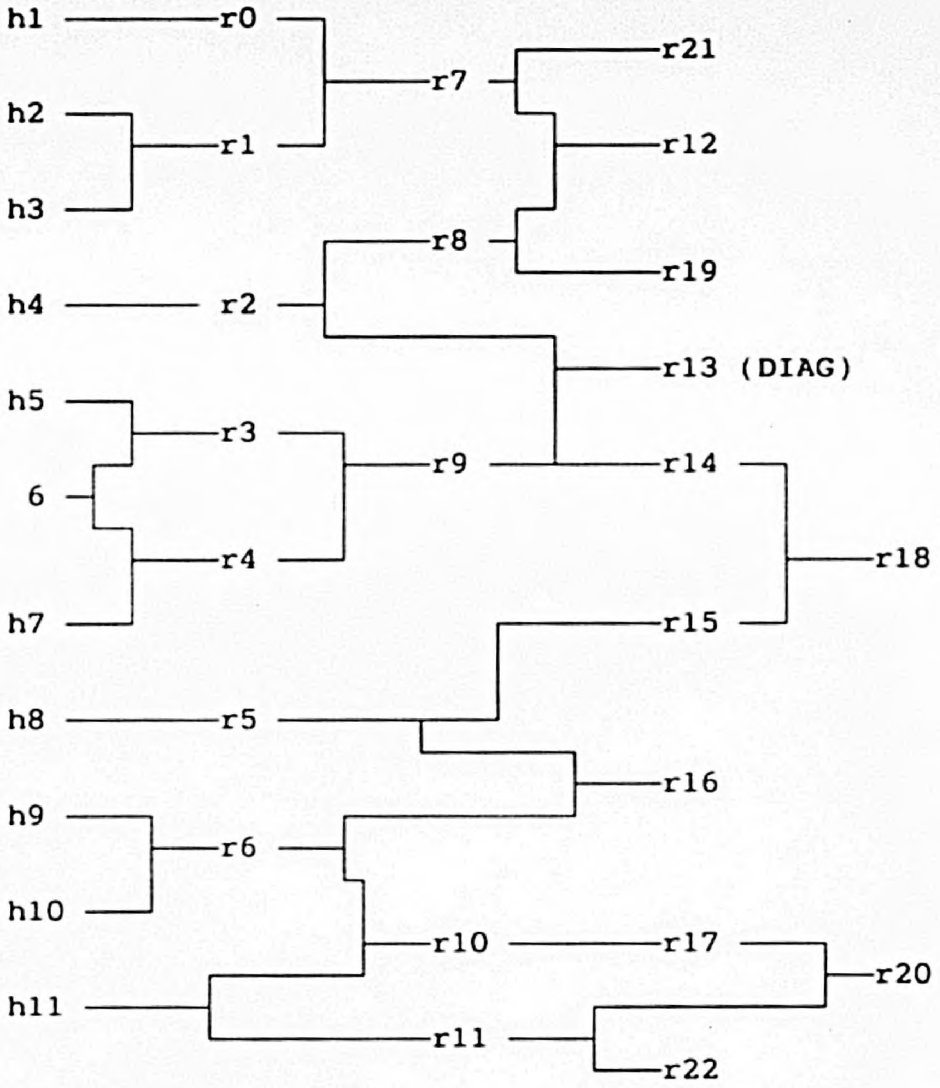


Figura 6.12.

Para evaluar el comportamiento general del sistema utilizando la medida de ruta, se obtendrá la media de los sumatorios de las reglas disparadas en el total de los casos ejecutados, correspondientes a cada nivel.

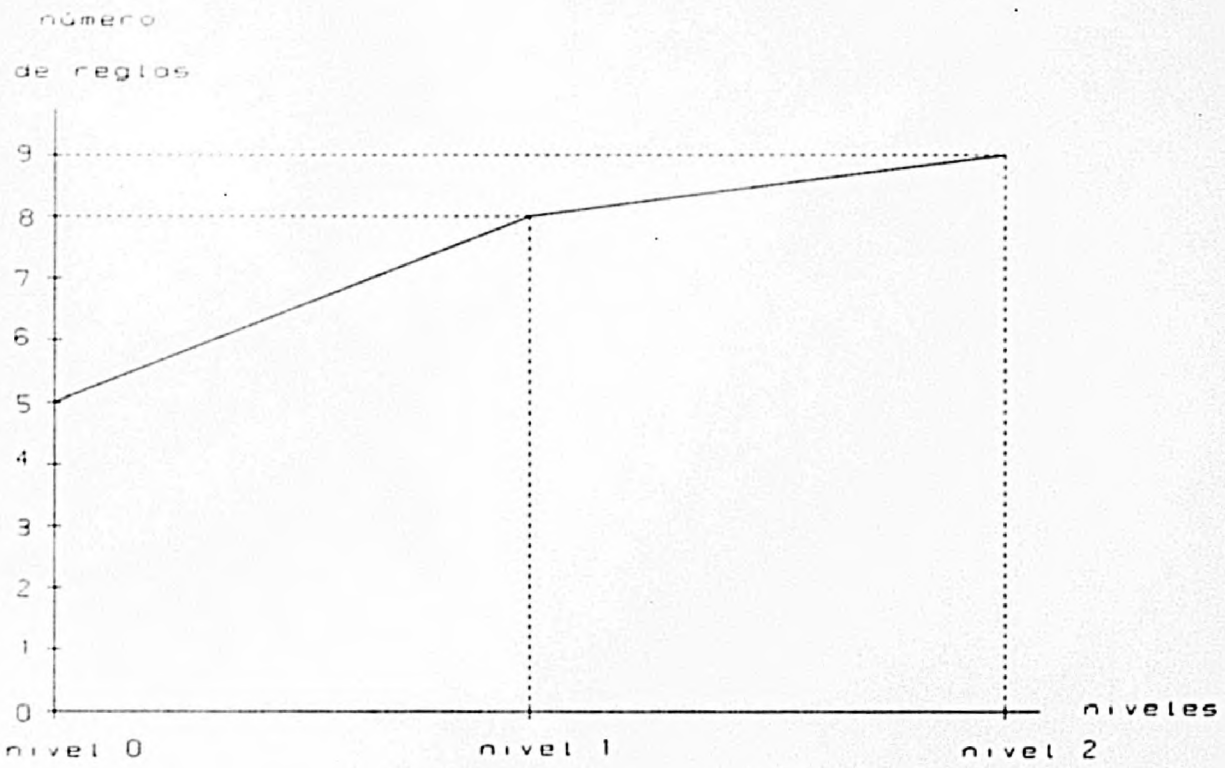


Figura 6.13.

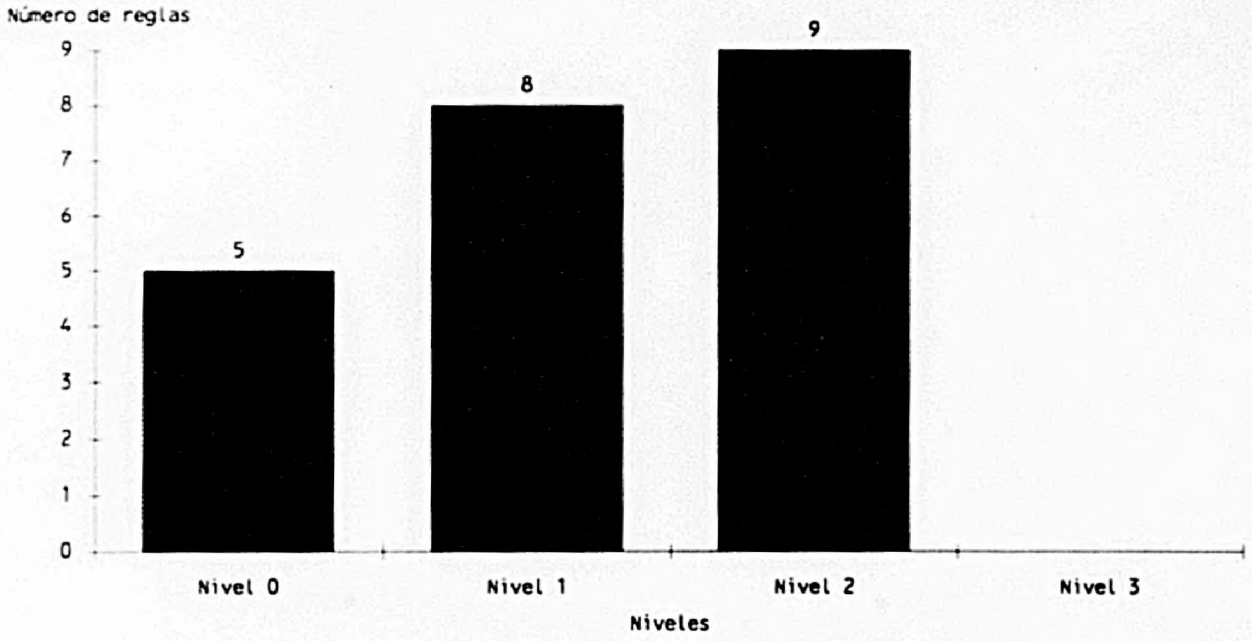

Número de reglas
9
8
7
6
5
4
3
2
1
0
5
8
9
Nivel 0
Nivel 1
Nivel 2
Nivel 3
Niveles

Figura 6.14.

6.3.3.- RESULTADOS

La medida expuesta se ha desarrollado mediante un programa LISP (Apéndice 2) y se ha aplicado al sistema MILORD con la BC PNEUMONIA obteniéndose las medidas de ruta correspondientes a los casos individuales, así como la global del sistema. En la figura 6.15. se muestra la ruta global que sigue el sistema, y en la figura 6.16. el diagrama de barras correspondiente, habiendo utilizado 40 casos para medirla. Los niveles de reglas obtenidos finalmente por término medio, son los siguientes:

NIVEL 0: 40,6

NIVEL 1: 208,3

NIVEL 2: 11

Esto indica, que si bien el sistema MILORD con la BC PNEUMONIA no sigue la ruta óptima, sí se enmarcaría en una situación media buena ya que aunque en el NIVEL 1 hay bastantes reglas, el número de nivel superior es 2, es decir, no se generan muchos niveles de reglas innecesarias.

Se puede comprobar que esta medida es de gran utilidad para aplicarla a cualquier sistema de diagnóstico, y da una visión simple y clara del comportamiento general del sistema en la consecución de los objetivos y por tanto, de si el mecanismo de inferencia se acerca o no a la solución óptima.

En el caso de que se disparasen muchas reglas innecesarias en el proceso de inferencia, aumentaría el número de niveles y su valor. En ese caso, sería recomendable modificar la BC de forma que la ruta de los casos fuese lo más directa posible hacia el objetivo. Esto podría hacerse, por ejemplo, añadiendo un orden de disparo en la parte SI de las reglas, de forma que aquellas reglas que contuviesen más atributos relevantes se disparasen antes que las otras.

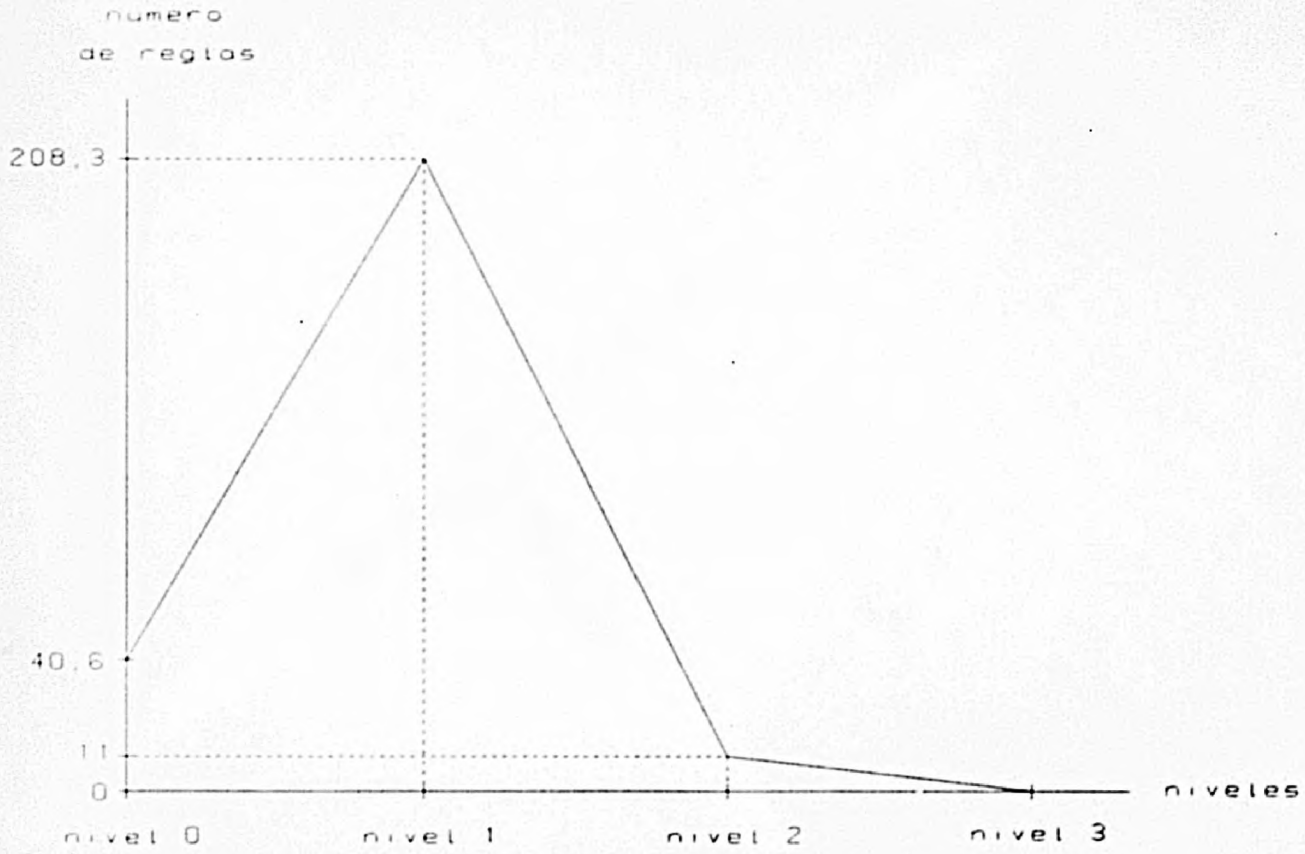

numero
de reglas
208,3
40,6
11
0
niveles
nivel 0
nivel 1
nivel 2
nivel 3

Figura 6.15.

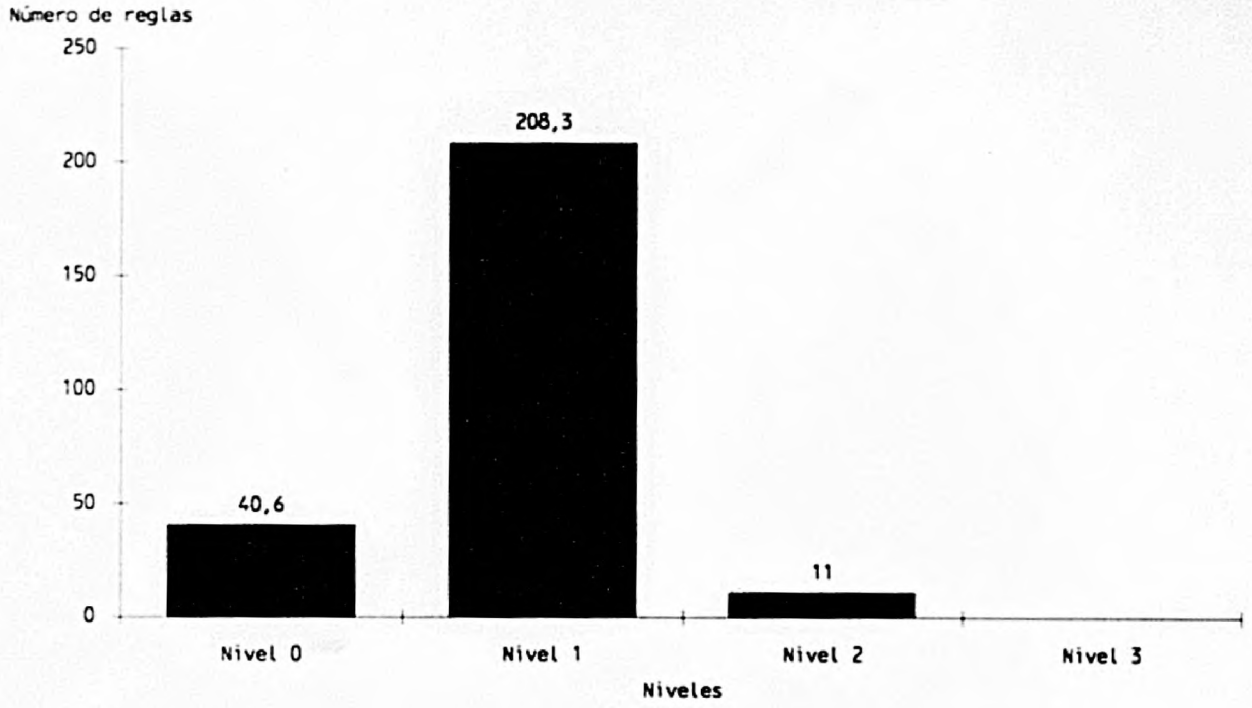

Número de reglas
250
200
150
100
50
0
208,3
40,6
11
Nivel 0
Nivel 1
Nivel 2
Nivel 3
Niveles

Figura 6.16.

6.4.- FACTOR DE ENFOQUE

Mediante el factor de enfoque se trata de medir el enfoque del sistema, es decir, si la deducción del/los diagnóstico/s se lleva a cabo de forma enfocada o no. Para ello es preciso estudiar la base de conocimientos dinámicamente, mediante la ejecución de un conjunto de casos.

Se ha definido el factor de enfoque como:

$$FE = \left(\sum_{1}^{n} M/(N-FC) \right) / n$$

siendo:

N = Número de objetivos estudiados o visitados en la ejecución, intentando verificarlos a través de los datos disponibles.

M = Número de objetivos alcanzados finalmente, es decir, número de diagnósticos que ofrece el sistema como solución final.

FC = Factor corrector

n = Número de casos estudiados

Su significado es el siguiente: cuanto más tienda a la unidad el factor de enfoque, tanto más enfocado estará el sistema. Cuanta más información incompleta exista, se explorarán más objetivos durante la ejecución, y en consecuencia el FE será menor y también el enfoque del sistema.

6.4.1- CALCULO DEL FACTOR DE ENFOQUE

Para su cálculo se propone el siguiente método:

1.- Se parte de n casos completos cuyos datos estarán contenidos en distintos ficheros.

Para cada caso se realizará el siguiente proceso:

1.1.- Ejecución del caso por el sistema, registrando los nombres de los módulos visitados en el proceso de inferencia así como los diagnósticos alcanzados.

1.2.- Obtención del número de módulos visitados durante la ejecución con el objeto de verificar el diagnóstico en ellos establecido (suponiendo que

cada módulo comprenda un diagnóstico; en caso contrario habría que contabilizar el número de diagnósticos visitados).

1.3.- Obtención del número de módulos utilizados en la respuesta del sistema, es decir, el número de diagnósticos emitidos finalmente.

1.4.- Cálculo del factor de enfoque individual para el caso C_i. En este trabajo, se ha considerado un FC = 2, ya que hay dos módulos, MENU y BACT-ATIP, que siempre se contabilizan como visitados pero que son módulos intermedios, es decir, no contienen diagnóstico final.

2.- Una vez ejecutados los n casos se calculará el enfoque global aplicando la fórmula.

6.4.2.- RESULTADOS PARA MILORD

Este proceso se ha desarrollado mediante un programa LISP (véase el apéndice 3) y se ha aplicado posteriormente a MILORD utilizando la base de conocimientos PNEUMONIA.

Para la evaluación se han utilizado 40 casos de prueba. En la tabla de la figura 6.16. se aprecian los resultados obtenidos en el cálculo de los FE parciales para dichos casos.

El factor de enfoque global resultante ha sido FE = 0,6, lo que indica que el enfoque del sistema MILORD con la BC PNEUMONIA es aceptable. En el supuesto de un sistema cuyo FE resultante fuese inferior a 0,5, debería pensarse en modificar la BC de forma que el sistema en su ejecución no necesitase visitar muchos más módulos de los estrictamente necesarios.

Este parámetro además de utilizarse para medir el enfoque de un sistema aislado, puede aplicarse también para comparar el enfoque de dos sistemas.

CASO ESTUDIADO	M	N-FC	FACTOR ENFOQUE
CASO 1	6	14	0,428
CASO 2	6	10	0,6
CASO 3	3	6	0,5
CASO 4	3	4	0,75
CASO 5	4	10	0,4
CASO 6	3	10	0,3
CASO 7	5	10	0,5
CASO 8	4	20	0,2
CASO 9	5	9	0,55
CASO 10	8	14	0,571
CASO 11	3	3	1
CASO 12	9	14	0,642
CASO 13	4	6	0,666
CASO 14	6	8	0,75
CASO 15	5	12	0,416
CASO 16	2	3	0,666
CASO 17	3	11	0,27
CASO 18	5	10	0,5
CASO 19	8	10	0,8
CASO 20	3	4	0,75
CASO 21	9	14	0,642
CASO 22	2	2	1
CASO 23	8	10	0,8
CASO 24	4	7	0,57
CASO 25	7	10	0,7
CASO 26	1	1	1
CASO 27	3	4	0,75
CASO 28	4	5	0,8
CASO 29	7	14	0,5
CASO 30	2	2	1
CASO 31	7	22	0,35
CASO 32	3	4	0,75
CASO 33	2	16	0,125
CASO 34	4	7	0,57
CASO 35	2	4	0,5
CASO 36	4	6	0,666
CASO 37	7	10	0,7
CASO 38	6	10	0,6
CASO 39	5	8	0,625
CASO 40	6	20	0,3

Figura 6.16.

CAPITULO 7

CONCLUSIONES Y

LINEAS FUTURAS

7.1.- CONCLUSIONES

Este trabajo ha consistido básicamente en el estudio de la validación de BC y en el diseño de tres medidas para evaluar las bases de conocimientos dinámicas. Seguidamente se han aplicado dichas medidas a la BC PNEUMONIA del SBC MILORD.

De acuerdo con los resultados obtenidos, se pueden extraer las siguientes conclusiones:

1.- La aplicación de las medidas de evaluación de BC propuestas en esta tesis: análisis de la robustez, medida de la ruta y factor de enfoque, proporcionan una mayor comprensión del comportamiento del SBC y aportan criterios

para decidir si es necesario modificar la BC con el objeto de mejorar su rendimiento. En otros tipos de medidas existentes, solamente se utilizan métodos estadísticos mientras que aquí se incorporan elementos simbólicos, tales como la selección de atributos.

2.- La medida de la robustez de una BC, sirve para evaluar el grado de solidez de una BC ante la posible falta de información en el tratamiento de un caso, acusando la necesidad de optimización si ésta es insuficiente, e indicando el número mínimo de datos necesarios para emitir un diagnóstico. Lo ideal para el médico, sería disponer siempre del mayor número de datos posibles (antecedentes personales, datos de exploración física, datos analíticos, exploraciones complementarias, etc.), que ayudasen en conjunto a enfocar un determinado diagnóstico. Sin embargo, esto no siempre es posible en un primer momento, y el médico dispone solamente de un número limitado de datos. Por lo tanto, puede ser muy útil para él saber con qué número mínimo de datos puede manejarse de cara a establecer un diagnóstico.

Además, mediante el proceso propuesto en la sección 6.2., se obtiene información sobre cuales son los datos más relevantes que se necesitan conocer para que el funcionamiento del SBC sea óptimo. Esto es de suma importancia, puesto que este tipo de sistema se dedica principalmente a orientar a los especialistas en las primeras horas de atención al enfermo, cuando aún no se conocen todos

los datos y pruebas, y por tanto, es ventajoso conocer qué datos deben ser obtenidos primero por ser más relevantes o significativos para el sistema a la hora de obtener un diagnóstico. Por ejemplo, de los resultados obtenidos con la BC PNEUMONIA se puede señalar que los datos 'antecedentes patológicos de interés' (alcoholismo, tabaquismo, diabetes, EPOC, etc.) y 'datos radiológicos del enfermo' (área del pulmón afectada, consolidación, infiltrados, etc.), son de primera importancia y es aconsejable su conocimiento para la decisión del diagnóstico. Por tanto, este método no sólo sirve para conocer el número mínimo de datos con los que puede trabajar el sistema, sino que también permite averiguar cuáles deberían ser esos datos para que la respuesta fuese la más adecuada.

3.- La medida de la ruta nos muestra si el camino que utiliza el sistema en el proceso de inferencia, se adapta al óptimo, o bien si los caminos que sigue para alcanzar la solución son muy dispersos, en cuyo caso debería ser perfeccionado para mejorar el rendimiento. Mediante el modelo propuesto en la sección 6.3. esto se puede interpretar fácilmente a través de la representación gráfica, comparando el resultado obtenido para el sistema en estudio, con el caso de ruta óptima. En el caso de un sistema que en su funcionamiento dispara muchas más reglas que las estrictamente necesarias, sería recomendable modificar la BC.

4.- La medida de enfoque nos da idea de la dificultad del sistema en su tarea de alcanzar una solución al problema planteado. Un sistema bien enfocado alcanza pronto el objetivo, ahorrando tiempo y recursos en su utilización. La aplicación de esta medida es sencilla utilizando el algoritmo propuesto en la sección 6.4., y además es de gran utilidad porque la interpretación matemática es muy simple y se puede aplicar a cualquier sistema de clasificación. No existe en ella problema de interpretación; puesto que el rango de valores entre los que puede variar el factor de enfoque es [0, 1], es claro que un buen FE será aquel que se aproxime a 1, es decir, el correspondiente a un sistema que no ha de explorar muchos más módulos que los estrictamente necesarios para alcanzar la solución.

5.- Los resultados obtenidos permiten proponer la utilización de las medidas mencionadas para comparar el funcionamiento de dos o más sistemas con el propósito de obtener una valoración de qué BC se ajusta más a un funcionamiento óptimo.

7.2.- LINEAS FUTURAS DE INVESTIGACION

A la vista de la situación actual en la evaluación de bases de conocimientos y de los resultados obtenidos en este estudio, se proponen las siguientes líneas de investigación:

1.- Adaptación directa de las medidas propuestas: medida de robustez, medida de la ruta y factor de enfoque, a un sistema como MILORD, de forma que éste las incorpore automáticamente en su funcionamiento.

2.- Realización de una comparación exhaustiva entre diferentes bases de conocimiento a través de la aplicación de dichas medidas a las mismas.

3.- Desarrollo de la medida de ruta utilizando otro tipo de estrategia para la medición, por ejemplo, considerando el camino más largo para calcular el alejamiento de una regla respecto del conjunto solución.

4.- A partir de los resultados obtenidos en la aplicación de la medida de ruta a una BC, modificación de dicha BC añadiendo un orden de disparo en la parte SI de las reglas, de forma que aquellas reglas con mayor número de atributos relevantes se disparen primero, y posterior comprobación de que con ello mejora la ruta.

5.- Puesto que pueden existir dos BC que utilicen diferentes términos para los mismos síntomas e incluso diferentes rangos de valores, se propone estudiar una posible unificación de vocabulario mediante técnicas de consenso. Y luego iterar el proceso de medición una vez conseguido el vocabulario (los atributos y objetos) y los rangos de los atributos.

6.- Valoración de la calidad de los diagnósticos obtenidos.

APENDICE 1

DESCRIPCION DEL METODO PROPUESTO PARA MEDIDA DE LA ROBUSTEZ DE UNA BC

A-1.1.- VARIABLES Y FICHEROS

El método propuesto en la sección 6.2. se ha desarrollado mediante un programa LISP que utiliza las siguientes variables y ficheros:

- **Variables:**

CASO: Contiene el nombre del caso que se está procesando.

LISTA: Es una lista que contiene los n casos que se van a procesar.

LISDAT: Lista que contiene los datos que se van a ir modificando, es decir, que se van a ir eliminando sucesivamente de cada caso como datos de entrada.

DATCOM: Lista resultado de leer el contenido del fichero ".MLT", que contiene los datos del enfermo.

VAAR: Contiene la lista de diagnósticos y certezas resultado de la ejecución *i* del caso en curso.

VAAR1: Contiene la lista de diagnósticos y certezas resultado de la ejecución *i-1* del caso en curso, es decir, de la ejecución anterior a la representada por VAAR.

PIL: Variable piloto que se inicializa a 0 y cambia su valor en el caso de que al recorrer la lista DATCOM se llegue a la sublista que contiene el dato que se quiere eliminar.

VALO: Contiene el nombre del dato que se ha eliminado para conservarlo a la hora de imprimir los resultados.

ENCONT: Variable piloto que se pone a 1 en caso de que encuentre el mismo diagnóstico en las dos listas VAAR y VAAR1. En este caso se compararán sus grados de certeza.

MLT: Contiene el nombre del caso en estudio con la extensión ".MLT" para acceder al fichero de datos del caso.

MLTB: Variable que contiene el nombre del caso con extensión ".MLT" y especifica la versión como ";1" para su posterior borrado.

TRC: Variable que contiene el nombre del caso con la extensión ".TRC" para acceder al fichero de diagnósticos y certezas.

TRCB: Variable que contiene el nombre del caso con extensión ".TRC" especificando la versión ";1".

CONVAR: Contador de los casos procesados.

LISVAR: Lista que contiene una posición por cada dato a eliminar, y en la que se guarda el número de variaciones producidas por ese dato en los distintos casos ejecutados.

PILAC: Variable booleana que se pone a 1 cuando el resultado de la ejecución es nulo.

PILIG: Variable booleana que se pone a 1 cuando en la lista resultado todos los diagósticos tienen la certeza menor a MOD-P.

PILIG2: Variable booleana que se activa cuando todos los diagnósticos resultado tienen certeza menor a POSSI.

- Ficheros:

RESCOM.DAT: Fichero de salida que contiene el resultado de la comparación de la lista de diagnósticos y certezas contenida en el fichero ".TRC" con la nueva lista resultado de haber suprimido un dato.

FICHMLT: Fichero que contiene la primera vez el caso de extensión ".MLT" y después de ser ejecutado éste contiene el mismo pero con las modificaciones efectuadas.

FICHTRC: Fichero que contiene el caso de extensión ".TRC", es decir, los resultados (dianósticos y certezas) del caso ejecutado.

A-1.2.- DESCRIPCION DEL PROCESO

El proceso que se ha seguido ha sido el siguiente:

1.- Inicialización:

- Se cargan en una lista (LISTA) los n casos que se van a ejecutar.

- Se cargan en una lista (LISDAT) los datos que se van a ir eliminando sucesivamente de cada caso.

- Se inicializa la lista de variaciones observadas (LISVA)

2.- Se ejecuta el caso contenido en el fichero XXX.MLT con todos sus datos y el resultado de la ejecución se deposita en la lista VAAR.

3.- Se elimina el siguiente dato de la lista LISDAT y se repite la ejecución.

Si VAAR resulta una lista vacía, se termina la ejecución con un mensaje indicativo; en caso contrario, se continúa.

4.- Se comprueba que los resultados obtenidos sean significativos. Para ello se han propuesto dos niveles de aceptación:

- En caso de que no haya ningún diagnóstico que supere el valor de certeza 'posible', se indicará por pantalla y en el fichero de resultados.

- En el caso de que no haya ningún diagnóstico que supere el valor de certeza 'moderadamente posible', igualmente se indicará por pantalla y en un fichero.

5.- Se comparan los resultados de las dos ejecuciones anteriores pudiéndose dar las siguientes posibilidades.

- Si el resultado es igual que en la ejecución anterior, se vuelve al punto 3.

- Si el resultado no es igual que en la ejecución anterior puede ser por varios motivos:

  a) Ha desaparecido algún diagnóstico de la solución.
  b) Ha variado la certeza de algún diagnóstico.
  c) Ha aparecido algún nuevo diágnostico en la solución.

En cualquiera de los supuestos, se indica por pantalla y en el fichero RESCOM.DAT con un mensaje. Además se suma 1 al elemento correspondiente al dato que ha originado la variación en la lista LISVA. A continuación se vuelve al punto 3 hasta que no queden más datos por

eliminar o hasta que el resultado sea nulo.

6.- Una vez finalizado el proceso de ejecución se registra por pantalla y en el fichero FVAR.DAT el total de variaciones observadas para cada dato eliminado.

Para ello se han utilizado las siguientes funciones:

CARCASOS: Esta función carga todos los nombres de los casos que se van a procesar en la variable LISTA.

PRAL: Función que realiza el bucle principal para procesar los n casos seleccionados y por cada caso llama a las funciones necesarias para realizar las pruebas que se aplican a cada caso particular.

EJEC: Función cuya tarea es realizar la ejecución de un caso para la obtención de los diagnósticos y valores de certeza asociados que se guardan en el fichero de extensión ".TRC".

COMPARAR: Función que compara los resultados obtenidos en dos ejecuciones sucesivas del mismo caso. Según el resultado de la comparación llama o no a dos funciones, ESCRIBE1 y ESCRIBE2 para registrar dichos resultados. Además puede llamar a la función ESCRIBE3 en el caso de que aparezca un nuevo diagnóstico en la última ejecución realizada.

ESCRIBE1: En el caso de que al comparar el resultado de dos ejecuciones sucesivas resulte que un determinado diagnóstico no ha variado pero sí el valor de certeza asociado al mismo, registra dicha variación en el fichero RESCOM.DAT así como también lo señala por pantalla con un mensaje tal como **"EN EL CASO xxx QUITANDO EL DATO yyy HA VARIADO LA CERTEZA DEL DIAGNOSTICO ddd DE mod-p A lle-p".**

ESCRIBE2: En el caso de que al comparar el resultado de dos ejecuciones sucesivas resulte que un determinado diagnóstico ha desaparecido en la segunda ejecución, registra dicha variación en el fichero RESCOM.DAT y también lo señala por pantalla con un mensaje tal como **"EN EL CASO xxx QUITANDO EL DATO yyy EL DIAGNOSTICO ddd YA NO APARECE".**

ESCRIBE3: Esta función es llamada desde COMPARAR en el caso de que aparezca un diagnóstico nuevo. El resultado lo escribe en pantalla y lo registra en el fichero RESCOM.DAT con un mensaje tal como **"EN EL CASO xxx QUITANDO EL DATO yyy HA APARECIDO EL DIAGNOSTICO ddd CON VALOR lle-p".**

RESNULL: Esta función se ejecuta si el resultado obtenido en el fichero de extensión ".TRC" es nulo, en cuyo caso termina la ejecución.

MIRCER: Función que comprueba si los valores de certeza de los diagnósticos resultado están dentro de un rango de valores (menores a POSSI o a MOD-P). Llama a las funciones ESC1, ESC2, ESC3 Y ESC4 para imprimir los resultados.

A-1.3.- PROGRAMA

A continuación se muestra el código fuente del programa:

```
;;; La función INIC pregunta si se desea inicializar la lista
;;; de variaciones. LLama a las funciones AUX y AUX2. Al
;;; final llama a la función PRAL.

(defun inic ()
     (declare (special lisva convar))
     (format t "QUIERE INICIALIZAR LA LISTA DE VARIACIONES?
     (SI/NO)")
     (setf varia (read))
     (format t "~%")
     (cond ((equal varia 'si) (aux2 condat))
           (t (aux)))
     (pral lista lisdat))


;;; La función AUX lee la lista de variaciones y el número
;;; de casos que se van ejecutando, del fichero FVAR.DAT

(defun aux ()
     (setf fvarp (open "fvar.dat" :direction :input))
     (setf lisva (read fvarp))
     (setf convar (read fvarp))
     (close fvarp))
```

```
;;; La función AUX2 inicializa la lista de variaciones y el
;;; contador de variaciones con valores nulos

(defun aux2 (condat)
     (setf convar '0)
     (setf lisva '())
     (do ((condat1 condat (1- condat1)))
         ((equal condat1 1))
         (setf lisva (cons '0 lisva))))


;;; La función CARCASOS va cargando los nombres de los
;;; casos que se deseen ejecutar en la lista LISTA

(defun carcasos (convar)
     (setf caso '9)
     (format t "~% ~%")
     (do ((lista '() (cons caso lista)) (cont 1 (1+ cont)))
         ((equal caso 'ff) (acabar lista))
         (format t "INTRODUZCA EL NOMBRE DEL CASO ~A ENTRE
                    COMILLAS FF PARA FINALIZAR:  " cont)
         (setf caso (read))))


;;; La función ACABAR es una función auxiliar de CARCASOS.
;;; Enlaza con la función INLISDAT

(defun acabar (lista)
     (setf lista (rest lista))
     (setf lista (reverse lista))
     (inlisdat lista))


;;; La función INLISDAT carga la lista de datos que se
;;; quieren eliminar durante el proceso

(defun inlisdat (lista)
     (setf lisda 'db)
     (do ((lisdat '() (cons lisda lisdat))
          (condat 0 (1+ condat)))
         ((equal lisda 'ff) (intdat lista lisdat condat))
         (format t "~%INTRODUZCA EL NOMBRE DEL DATO
                (FF PARA ACABAR)")
         (setf lisda (read))))
```

```
;;; La función PRAL es la función principal. Cada caso se
;;; va ejecutando, primero con todos los datos y después se
;;; repiten las ejecuciones eliminando un dato cada vez.
;;; Para ello se llama a la función EJEC. También compara
;;; los resultados de las ejecuciones por medio de la
;;; función COMPARAR. Si la lista de diagnósticos está vacía
;;; llama a la función RESNULL y termina la ejecución del
;;; caso en curso

(defun pral (lista lisdat)
     (declare (special pilva lisva lisdat convar pilac))
     (setf rescomp (open "rescom.dat" :direction :output :if-
                    exists :new-version))
     (do ((lis lista (cdr lis)))
         ((equal lis nil))
         (setf caso (car lis))
         (setf mlt (concatenate 'string caso ".mlt"))
         (setf fichmlt (open mlt))
         (setf datcom (read fichmlt))
         (close fichmlt)
         (setf valo '9)
         (format rescomp "~% ~% ~% CASO ~A ~% ~% ~%" caso)
         (ejec lis caso valo)
         (format rescomp " ~% EL RESULTADO DE LA EJECUCION
                           SIN QUITAR NINGUN DATO ES ")
         (format rescomp " ~% ~A ~%" vaar)
         (setf pilac '0)
         (do ((datos lisdat (cdr datos)) (contva 1
                   (1+ contva)))
             ((or (equal datos nil) (equal pilac '1)))
             (setf pilva '0)
             (setf vaar1 vaar)
             (setf pil '0)
             (do ((dat datcom (cdr dat))(cont '0 (1+ cont)))
               ((equal pil '1)(cambiar cont datcom))
               (cond ((equal (car dat) (car datos))
                      (setf pil '1))))
          (setf mltb (concatenate 'string mlt ";1"))
          (delete-file mltb)
          (setf fichmlt (open mlt :direction :output))
          (format fichmlt " ; DATOS DEL ENFERMO: ")
          (format fichmlt "\"")
          (format fichmlt "~A" caso)
          (format fichmlt "\" ~% ")
          (format fichmlt "\(")
          (do ((liscom datcom (cdr liscom)))
              ((null liscom))
              (setf cliscom (car liscom))
              (format fichmlt " \( ~A    " (car cliscom))
              (format fichmlt " \" ~A \" " (cadr cliscom))
              (format fichmlt " ~A  " (caddr cliscom))
              (format fichmlt " \" ~A \" \) ~% " (cadddr
                                  cliscom)))
          (format fichmlt "\)")
          (close fichmlt)
          (setf valo (car datos))
```

```
          (ejec lis caso valo)
          (cond ((null vaar) (renull caso valo))
                (t (comparar vaar1 vaar valo caso)))
          (cond ((equal pilva '1) (meter contva
                 (1+ (acceso contva lisva)))))))
     (close rescomp)
     (setf fvarp (open "fvar.dat" :direction :output
              :if-exists :new-version))
     (setf rconvar (+ convar (length lista)))
     (format fvarp " EL NUMERO DE CASOS PROCESADOS ES ~A ~%"
                   rconvar)
     (format t "NUMERO DE CASOS PROCESADOS ES ~A ~%" rconvar)
     (format  fvarp  "~% DATOS  ELIMINADOS
               VARIACIONES PRODUCIDAS ~%")
     (format t "~% DATOS ELIMINADOS
             VARIACIONES PRODUCIDAS ~%")
     (do ((lises lisdat (cdr lises))
          (lisv lisva (cdr lisv)))
         ((null lises))
         (setf clises (car lises))
         (setf clisv (car lisv))
         (format fvarp "   ~A                              ~A ~%"
                       clises clisv)
         (format t "    ~A                          ~A ~% clises
               clisv))
     (close fvarp))


;;; Función auxiliar de PRAL

(defun intdat (lista lisdat condat)
     (format t "~%")
     (setf lisdat (rest lisdat))
     (setf lidat (reverse lisdat))
     (inic lista lisdat condat))


;;; Función auxiliar de PRAL, que emite un mensaje
;;; en caso de que quitando un dato el resultado
;;; sea nulo

(defun resnull (caso valo)
     (declare (special pilac))
     (setf pilac '1)
     (format t "~%EN EL CASO ~A QUITANDO EL DATO ~A
             NO HAY RESULTADO ~%" caso valo)
     (format rescomp "~%EN EL CASO ~A QUITANDO EL DATO
               ~%NO HAY RESULTADO ~%" caso valo))
```

```
;;; La función COMPARAR compara dos listas, VAAR y VAAR1, que
;;; contienen los resultados de la ejecución en curso y de
;;; la ejecución anterior. Los resultados se registran a
;;; través de las funciones ESCRIBE1, ESCRIBE2 y ESCRIBE3

(defun comparar (vaar1 vaar valo caso)
     (declare (special pilva))
     (do ((lista1 vaar1 (cdr lista1)))
          ((null lista1))
          (setf nocam '0)
          (setf encont '0)
          (setf listacom (car lista1))
          (do ((lista2 vaar (cdr lista2)))
              ((or (null lista2) (equal encont 1)))
              (setf listacom2 (car lista2))
              (cond ((equal (car listacom)
                            (car listacom2))
                     (setf encont '1)))
              (cond ((equal encont 1)
                     (cond ((equal (cadr listacom)
                                   (cadr listacom2))
                            (return))
                           (t (escribe1 caso valo listacom
                                        listacom2 nocam))))))
          (cond ((equal encont 0) (escribe2 caso valo
                                            listacom))))
     (do ((lista3 vaar (cdr lista3)))
          ((null lista3))
          (setf enc '0)
          (setf liscom (car lista3))
          (do ((lista4 vaar1 (cdr lista4)))
              ((or (null lista4) (equal enc '1)))
              (setf liscom2 (car lista4))
              (cond ((equal (car liscom) (car liscom2))
                     (setf enc '1))))
          (cond ((equal enc '0) (escribe3 caso valo
                                          liscom)))))


;;; La función ESCRIBE1 escribe la variación observada por
;;; pantalla y en el fichero RESCOM.DAT, cuando varía la
;;; certeza de un diagnóstico.

(defun escribe1 (caso valo listacom listacom2 nocam)
     (declare (special pilva))
     (format t " ~% EN EL CASO ~A QUITANDO EL DATO ~A ~%"
             caso valo)
     (format t " HA VARIADO LA CERTEZA DEL DIAGNOSTICO ~A ~%"
             (car listacom))
     (format t " DEL VALOR ~A A ~A ~% " (cadr listacom)
             (cadr listacom2))
     (format rescomp " ~% EN EL CASO ~A QUITANDO EL DATO ~A
                 ~%" caso valo)
     (format rescomp " HA VARIADO LA CERTEZA DEL DIAGNOSTICO
                 ~A ~%" (car listacom)
```

```
     (format rescomp " DEL VALOR ~A A ~A ~%" (cadr listacom)
               (cadr listacom2))
     (setf pilva '1)
     (setf nocam '1))


;;; La función ESCRIBE2 escribe por pantalla y en el
;;; fichero RESCOM.DAT el resultado registrado al
;;; producirse una variación en el número de diagnósticos
;;; finales.

(defun escribe2 (caso valo listacom)
     (declare (special pilva))
     (format t " ~% EN EL CASO ~A QUITANDO EL DATO ~A ~%"
             caso valo)
     (format t " EL DIAGNOSTICO ~A YA NO APARECE ~% "
              (car listacom))
     (format rescomp " ~% EN EL CASO ~A QUITANDO EL DATO ~A
               ~%" caso valo)
     (format rescomp " EL DIAGNOSTICO ~A YA NO APARECE ~%"
               (car listacom))
     (setf pilva '1))


;;; La función ESCRIBE3 registra por pantalla y en el
;;; fichero RESCOM.DAT la aparición de un diagnóstico
;;; nuevo.

(defun escribe3 (caso valo liscom)
     (declare (special pilva))
     (format t " ~% EN EL CASO ~A QUITANDO EL DATO ~A ~%"
            caso valo)
     (format t " HA APARECIDO EL DIAGNOSTICO ~A CON VALOR ~A
              ~%" (car liscom) (cadr liscom))
     (format rescomp " ~% EN EL CASO ~A QUITANDO EL DATO ~A
               ~%" caso valo)
     (format rescomp " HA APARECIDO EL DIAGNOSTICO ~A CON
          VALOR ~A ~%" (car liscom) (cadr liscom))
     (setf pilva '1))


;;; La función CAMBIAR busca el dato en curso a eliminar en
;;; la lista DATCOM y pone su valor a desconocido

(defun cambiar (cont datcom)
     (setf (caddr (car (nth (1- cont) datcom))) 'desconegut))


;;; La función EJEC llama a la función de ejecución de
;;; casos en modo batch y a MIRTRC

(defun ejec (lis caso valo)
     (ejecutar-batch caso)
     (mirtrc caso valo))
```

```
;;; La función MIRTRC lee el resultado de la ejecucion del
;;; fichero ".TRC" del caso en curso y lo vuelca en VAAR.
;;; Llama a la función MIRCER

(defun mirtrc (caso valo)
     (setf trc (concatenate 'string caso ".trc"))
     (setf fichtrc (open trc))
     (setf vaar (read fichtrc))
     (mircer vaar caso valo)
     (close fichtrc)
     (setf trcb (concatenate 'string trc ";1"))
     (delete-file trcb))


;;; La función MIRCER mira si las certezas de todos los
;;; diagnósticos, son menores o iguales a un valor
;;; determinado

(defun mircer (vaar caso valo)
     (setf valores '(lle-p mpo-p gen-p))
     (setf pilig '0)
     (setf pilig2 '0)
     (do ((lismen vaar (cdr lismen)))
         ((or (null lismen)(equal pilig 1)(equal pilig2 1)))
         (setf clismen (car lismen))
         (do ((lisval valores (cdr lisval)))
          ((or (null lisval)(equal pilig 1)
                      (equal pilig2 1)))
          (setf clisval (car lisval))
          (cond ((not (equal (cadr clismen) clisval))
                 (setf pilig '1))
                (t (cond ((not (equal (cadr clismen)
                      'mod-p))
                   (setf pilig2 '1)))))))
     (cond ((not (equal valo '9))
            (cond ((equal pilig '0) (escl caso valo))
               (t (cond ((equal pilig2 '0)
                                (esc2 caso valo))))))
           (t (cond ((equal pilig '0) (esc3 caso))
                 (t (esc4 caso))))))


;;; Función auxiliar de MIRCER para imprimir el resultado

(defun escl (caso valo)
     (format t "~%EN EL CASO ~A QUITANDO EL DATO ~A
              ~%TODAS LAS CERTEZAS SON MENORES
              A MOD-P ~% ~%" caso valo)
     (format rescomp "~%EN EL CASO ~A QUITANDO EL DATO ~A
              ~%TODAS LAS CERTEZAS SON MENORES
              A MOD-P ~% ~%" caso valo))
```

```
;;; Función auxiliar de MIRCER para imprimir el resultado

(defun esc2 (caso valo)
     (format t "~%EN EL CASO ~A QUITANDO EL DATO ~A
             ~%TODAS LAS CERTEZAS SON MENORES
             A POSSI ~% ~%" caso valo)
     (format rescomp "~%EN EL CASO ~A QUITANDO EL DATO ~A
             ~%TODAS LAS CERTEZAS SON MENORES
             A POSSI ~% ~%" caso valo))


;;; Función auxiliar de MIRCER para imprimir el resultado

(defun esc3 (caso valo)
     (format t "~%EN EL CASO ~A SIN QUITAR NINGUN DATO
             ~%TODAS LAS CERTEZAS SON MENORES
             A MOD-P ~% ~%" caso valo)
     (format rescomp "~%EN EL CASO ~A SIN QUITAR NINGUN DATO
             ~%TODAS LAS CERTEZAS SON MENORES
             A MOD-P ~% ~%" caso valo))


;;; Función auxiliar de MIRCER para imprimir el resultado

(defun esc4 (caso valo)
     (format t "~%EN EL CASO ~A SIN QUITAR NINGUN DATO
             ~%TODAS LAS CERTEZAS SON MENORES
             A POSSI ~% ~%" caso valo)
     (format rescomp "~%EN EL CASO ~A SIN QUITAR NINGUN DATO
             ~%TODAS LAS CERTEZAS SON MENORES
             A POSSI ~% ~%" caso valo))


;;; La función METER actualiza la lista de variaciones

(defun meter (contva elem)
     (declare special lisva))
     (setf rdo '())
     (do ((cont 1 (1+ cont))
          (lista-ant lisva (cdr lista-ant)))
          ((and (null lista-ant) (>= cont (1+ contva))) rdo)
          (cond ((= cont contva) (setf rdo (append rdo
                                 (list elem))))
                (t (setf rdo (append rdo (list
                                 (car lista-ant))))))))
     (setf lisva rdo))


;;; Función auxiliar de METER.

(defun acceso (contva lisva)
     (declare (special lisva))
     (car (nthcdr (1- contva) lisva)))
```

APENDICE 2

DESCRIPCION DEL METODO PROPUESTO PARA LA MEDIDA DE LA RUTA

A-2.1.- VARIABLES Y FICHEROS

El método propuesto en la sección 6.3. para evaluar la ruta que sigue la traza de un caso en su ejecución se ha desarrollado mediante un programa LISP utilizando las siguientes variables y ficheros:

- **Variables:**

Del módulo NIVEL:

NOM: Variable que contiene el nombre del caso en curso de ejecución.

LISTA: Lista que contiene las reglas que se han disparado durante la ejecución del caso en curso.

TRC: Lista que contiene las reglas que se han utilizado para inferir los diagnósticos finales.

LISTA-REGLAS: Lista que contiene las reglas disparadas y el nivel al que pertenece cada una.

TABLA-FINAL: Lista que contiene el resultado del módulo ARBOL2, es decir, cada regla con sus condiciones y su conclusión.

NIVEL-MAYOR: Variable que contiene el mayor número de nivel alcanzado.

REGLA: Variable que contiene la primera regla de la lista LISTA.

NIVEL: Variable que contiene el nivel de la regla REGLA.

ANTECEDENTES: Contiene todas las reglas antecedentes de la regla contenida en REGLA. Es el resultado del módulo TRAMA2.

NIVEL-MENOR: Variable que contiene el nivel menor de todos los niveles de las reglas antecedentes de REGLA, es decir, el que se va a asignar a REGLA.

CONTADORES: Vector que contiene un elemento por cada nivel, indicando el número de reglas de cada uno. El número de elementos depende del mayor nivel alcanzado (NIVEL-MAYOR).

TOTAL: Variable que se utiliza para contabilizar el número de reglas de cada nivel.

Del módulo ARBOL2:

TABLA-MODULOS: Lista que contiene los nombres de los módulos diagnóstico con los dígitos indicativos de los mismos con el objeto de averiguar a qué módulo pertenece cada regla.

MODULO: Variable que incluye las reglas contenidas en el módulo al que pertenece una determinada regla.

Del módulo TRAMA2:

TABLA: Lista que contiene los antecedentes de la regla de la que se está averiguando el encadenamiento.

- **Ficheros:**

CASOXX.DAT: Fichero que contiene las reglas que se han disparado durante la ejecución del caso en curso.

CASOXX.TRC: Fichero que contiene las reglas que se han utilizado para inferir los diagnósticos finales.

CASOXX.NIV: Fichero que contiene el resultado de la ejecución, es decir, el número de niveles y el número de reglas en cada uno.

PNEUMONIA.LSP: Base de conocimientos, que contiene todas las reglas.

A-2.2.- DESCRIPCION DEL PROCESO

El proceso que se ha seguido consta de las siguientes etapas:

1.- Se ejecuta el caso del cual se quiere medir la ruta. Para ello se ha incorporado al programa principal que realiza la ejecución en batch, una función que registra cada regla que se activa durante dicha ejecución, en el fichero CASOXX.DAT.

Además, también se genera un fichero CASOXX.TRC con la traza de las reglas necesarias para alcanzar los diagnósticos finales.

2.- Se carga el contenido del fichero CASOXX.DAT en LISTA, y el contenido del fichero CASOXX.TRC en TRC para su posterior tratamiento.

3.- Se cede el control al módulo ARBOL2, el cual toma el contenido de LISTA y por cada regla que aparece en ella localiza el módulo al que pertenece y seguidamente crea una lista que contiene el nombre de la regla, sus condiciones y su conclusión. La información de todas las reglas la devuelve al módulo principal en TABLA-FINAL.

4.- Se va accediendo sucesivamente a cada regla contenida en LISTA y se realiza la siguiente comprobación:

- Si la regla pertenece al conjunto solución, es decir, existe en la lista TRC, se le asigna el nivel 0.

- En caso contrario, se calculará el alejamiento de dicha regla respecto del conjunto solución. Para ello se averiguan los antecedentes de la misma a través del módulo TRAMA2:

  - Si alguno de los antecedentes es un dato, se le asigna nivel 1.

- Si todos los antecedentes son reglas, se le asigna un nivel más que el menor de los niveles de las mismas.

El módulo TRAMA2 para averiguar los antecedentes de una regla compara las condiciones de la misma con las conclusiones de las demás reglas utilizando para ello TABLA-FINAL.

- Si no encuentra ninguna coincidencia para alguna de las condiciones, significa que dicha regla tiene como antecedente un dato. Por tanto se registra que el antecedente es DATO.

- Por cada coincidencia que encuentra, registra la regla a la que pertenece la conclusión como antecedente.

5.- Una vez averiguados los niveles de todas las reglas, se procederá a cargar el vector CONTADORES, con la información que aparece en la lista LISTA-REGLAS.

6.- A continuación se grabarán en el fichero CASOXX.NIV y se mostrarán por pantalla los niveles y el número de reglas de cada uno.

Para ello se han utilizado las siguientes funciones:

NIVEL: Función principal que crea una tabla (LISTA-REGLAS) con el nivel de alejamiento menor de cada regla y devuelve una tabla (CONTADORES) con el número de reglas de cada nivel, creando el fichero de salida CASOXX.NIV.

ARBOL2: Esta función averigua las condiciones y conclusión de las reglas del caso en curso, creando con esa información TABLA-FINAL.

LOCALIZAR: Localiza el módulo al que pertenece una regla dentro de la BC.

ARREGLAR: Obtiene las condiciones de una regla determinada.

TRAMA2: Esta función se aplica a una determinada regla y devuelve una lista con todas sus reglas antecedentes. Si algún antecedente es un dato, devuelve la palabra 'DATO' en lugar de una regla.

A-2.3.- PROGRAMA

```
;;;Esta función crea una tabla LISTA-REGLAS con el nivel de
;;;alejamiento menor de cada regla y devuelve una tabla
;;;CONTADORES, con el número de reglas de cada nivel.
;;;La lista de reglas se lee del fichero CASOXX.DAT
;;;Se crea un fichero de salida CASOXX.NIV con el resultado
;;;de CONTADORES.

(defun nivel ()
     (format t "~% INTRODUZCA EL NOMBRE DEL CASO ENTRE
               COMILLAS")
     (setf nom (read))
     (format t "~%")
     (setf fiche (open (concatenate 'string nom ".dat")
                       :direction :input))
     (read fiche)
     (setf lista (read fiche))
     (close fiche)
     (setf fiche3 (open (concatenate 'string nom ".niv")
                        :direction :output :if-exists
                        :new-version))
     (setf fiche2 (open (concatenate 'string nom ".trc"
                        :direction :input))
     (read fiche2)
     (setf trc (read fiche2))
     (close fiche2)
     (setf lista-reglas '())
     (setf tabla-final (arbol lista))
     (setf nivel-mayor 0)
     (setf s 'inicio)
     (setf lista2 '())
     (loop
        (cond ((null lista) (return))
              (t nil))
        (loop
           (cond ((null lista) (return))
                 (t nil))
           (loop
              (setf regla (car lista))
              (cond ((member regla trc) (setf nivel 0))
                    (t nil))
              (cond ((member regla trc)
                     (incluir regla nivel))
                    (t nil))
              (cond ((member regla trc) (return))
                    (t nil))
              (setf antecedentes (trama2 regla tabla-final))
              (cond ((member 'dato antecedentes)
                     (setf nivel 1))
                    (t nil))
              (cond ((> nivel nivel-mayor)
                     (setf nivel-mayor nivel))
                    (t nil))
```

```
        (cond ((member 'dato antecedentes)
               (incluir regla nivel))
              (t nil))
        (cond ((member 'dato antecedentes) (return))
               (t nil))
        (setf nivel-menor 100)
        (loop
           (cond ((null antecedentes) (return))
                 (t nil))
           (setf elemento (assoc (car antecedentes)
                                 lista-reglas))
           (cond ((null elemento)
                  (setf lista2 (cons regla lista2)))
                 (t nil))
           (cond ((null elemento) (setf s 'mal))
                 (t (setf s 'no-mal)))
           (cond ((null elemento) (return))
                 (t nil))
           (setf nivel (cadr elemento))
           (setf antecedentes (cdr antecedentes))
           (cond ((< nivel nivel-menor)
                  (setf nivel-menor nivel))
                 (t nil))
           (cond ((equal s 'mal)(return))
                 (t (incluir regla (1+ nivel-menor))))
           (cond ((> (1+ nivel-menor) nivel-mayor)
                  (setf nivel-mayor (1+ nivel-menor)))
                 (t nil))
           (cond ((equal s 'no-mal) (return))
                 (t nil)))
        (princ regla)
        (setf lista (cdr lista)))
     (setf lista lista2)
     (setf lista2 '()))
  (setf contadores (make-array (1+ nivel-mayor)
                               :initial-element 0))
  (loop
     (cond ((null lista-reglas) (return))
           (t nil))
     (setf elemento (car lista-reglas))
     (setf lista-reglas (cdr lista-reglas))
     (setf nivel (cadr elemento))
     (setf (aref contadores nivel)
           (1+ (contadores nivel))))
  (setf nivel 0)
  (loop
     (cond ((> nivel nivel-mayor) (return))
           (t nil))
     (setf total (aref contadores nivel))
     (format fiche3 "~% EN EL NIVEL ~S HAY ~A REGLAS"
                    nivel total)
     (setf nivel (1+ nivel)))
  (close fiche3)
  (format t "~%EL RESULTADO ESTA EN EL FICHERO"
            fiche3))
```

```
;;;Esta función crea una lista con DATO1 y DATO2 y la incluye
;;;en LISTA-REGLAS.

(defun incluir (dato1 dato2)
     (setf lista-reglas (cons (list dato1 dato2)
                              lista-reglas)))


;;;Esta función localiza el módulo a partir del nombre del
;;;módulo.

(defun localizar (nom-mod)
     (setf corriemte (open "pneumonia.lsp" :direction
                           :input))
     (loop
        (setf mod (read corriente))
        (cond ((equal nom-mod (cadr mod)) (return))
              (t nil)))
        (close corriente)
        (print nom-mod)
        (cadr (cadddr mod)))


;;;En esta función se obtienen las conclusiones de cada regla
;;;en forma de listas, devolviendo una lista que incluye a
;;;todas.

(defun arreglar (regla)
     (setf condiciones (cadadr regla))
     (setf resultado '())
     (serf prim-cond (car condiciones))
     (loop
        (cond ((null condiciones) (return)))
        (cond ((atom prim-cond) (setf prim-cond
                                      (list prim-cond)))
              (t nil))
        (setf resultado (cons prim-cond resulatdo))
        (setf condiciones (cdr condiciones))
        (setf prim_cond (car condiciones)))
     resultado)


;;;Función auxiliar (ASSOC)

(defun miassoc (indice mitabla)
     (loop
        (cond ((equal indice (caar mitabla)) (return))
              ((null mitabla) (return 'error))
              (t nil))
        (setf mitabla (cdr mitabla)))
     (cadadr mitabla))
```

```
;;;Esta función es el módulo principal del programa ARBOL2,
;;;y crea la tabla de las reglas que manda la función NIVELES
;;;a través del parametro LISTA. Para cada regla la tabla
;;;contiene el nombre de la regla, sus condiciones, y su
;;;conclusión. Coge las reglas del fichero CASO.DAT

(defun arbol2 (lista)
     (setf tabla-final '())
     (setf lista (reverse lista))
     (setf tabla-modulos '(((#\0 #\1) menu)
                           ((#\0 #\2) ab-grama)
                           ((#\0 #\3) bact-atip)
                           ((#\0 #\4) pneum)
                           ((#\0 #\5) legion)
                           ((#\0 #\6) enterobact)
                           ((#\0 #\7) hemoph)
                           ((#\0 #\8) pseudo)
                           ((#\0 #\9) anaerob)
                           ((#\1 #\0) staph)
                           ((#\1 #\1) meningo)
                           ((#\1 #\2) branha)
                           ((#\1 #\3) str-pio)
                           ((#\1 #\4) tbc)
                           ((#\1 #\5) nocardia)
                           ((#\1 #\6) criptococ)
                           ((#\1 #\7) myco)
                           ((#\1 #\8) clam)
                           ((#\1 #\9) rq)
                           ((#\2 #\0) viruses)
                           ((#\2 #\1) viruses-bact)
                           ((#\2 #\2) citomegalovirus)
                           ((#\2 #\3) asper)
                           ((#\2 #\4) p-car)
                           ((#\2 #\5) mr-inicial)
                           ((#\2 #\6) embassament)))
     (setf nom-mod-antiguo 'menu)
     (setf modulo1 (localizar 'menu))
     (setf modulo modulo1)
     (loop
        (cond ((null lista) (return)
               (t nil)))
        (setf elemento (car lista))
        (setf lista (cdr lista))
        (setf reg-stri (string elemento))
        (setf indice (list (aref reg-stri 1)
                           (aref reg-stri 2)))
        (setf nom-mod (miassoc indice tabla-modulos))
        (cond ((not (equal nom-mod nom-mod-antiguo))
               (setf modulo1 (localizar nom-mod)))
              (t nil))
        (cond ((not (equal nom-mod nom-mod-antiguo))
               (setf modulo modulo1))
              (t nil))
        (setf nom-mod-antiguo nom-mod)
```

```
        (loop
           (setf posible (caar modulo))
           (cond ((equal elemento posible) (return))
                 (t nil))
           (setf modulo (cdr modulo)))
        (princ elemento)
        (setf encontrado (car modulo))
        (setf modulo modulo1)
        (setf variable (cadr (cadddr encontrado)))
        (cond ((equal (car variable) 'concloure)
               (setf variable (cadr variable))))
        (setf tabla-final (cons (list elemento
              (arreglar encontrado) variable) tabla-final)))
              tabla-final)


;;;Esta función devuelve, para una REGLA concreta, una lista
;;;con sus reglas antecedentes. Si una condición es un dato
;;;inicial, incluye la palabra DATO en la lista de
;;;antecedentes. Los antecedentes los busca en la
;;;TABLA-FINAL, que ha sido creada por la función ARBOL2.

(defun trama2 (regla tabla-final)
     (setf tabla2 tabla-final)
     (loop
        (setf lista8 (car tabla2))
        (setf tabla2 (cdr tabla2))
        (setf nombre (car lista8))
        (cond ((equal nombre regla) (return lista8))))
     (setf tabla '())
     (setf condis2 (cdr condis2))
     (loop
        (setf tabla2 tabla-final)
        (setf ci (car condis2))
        (setf condis2 (cdr condis2))
        (setf provisional '())
        (loop
           (setf muestra (car tabla2))
           (cond ((equal ci (caddr muestra))
                  (setf tabla (cons (car muestra) tabla))))
           (cond ((equal ci (caddr muestra))
                  (setf provisional (cons (car muestra)
                                          provisional))))
           (setf tabla2 (cdr tabla2))
           (cond ((null tabla2) (return))))
        (cond ((null provisional) (setf tabla
                                  (cons 'dato tabla))))
        (cond ((null provisional) (return)))
        (cond ((null condis2) (return))))
     (cond ((null tabla) (setf tabla (cons 'dato tabla))))
     (print tabla))
```

APENDICE 3

DESCRIPCION DEL METODO PROPUESTO PARA MEDIDA DEL ENFOQUE

A-3.1.- VARIABLES Y FICHEROS

El método propuesto en la sección 6.4. para obtención del factor de enfoque, se ha desarrollado mediante un programa LISP que utiliza las siguientes variables y ficheros:

- **Variables:**

CASO: Contiene el nombre del caso que se está procesando.

LISTA: Es una lista que contiene los n casos que se van a procesar.

LISTRC: Contiene la lista de diagnósticos resultado de la ejecución del caso.

LISMOD: Contiene la lista de los módulos visitados en la ejecución del caso especificado.

ENFGLOB: Contiene la media aritmética de los factores de enfoque de cada caso.

- Ficheros:

CASOXX.TRC: Fichero de entrada que contiene la traza de un caso.

FICHCONT.DAT: Fichero de salida que contiene el nombre del caso ejecutado, y el factor de enfoque.

A-3.2.- DESCRIPCION DEL PROCESO

El proceso que se ha seguido para calcular el factor de enfoque a partir de una serie de casos es el siguiente:

1.- Se cargan en la lista LISTA los nombres de los ficheros que contienen los datos de cada caso que se va a procesar.

2.- Se ejecutan sucesivamente los casos considerados realizando los siguientes cálculos para cada uno:

2.1.- Se obtiene la lista de diagnósticos y certezas resultado de la ejecución, depositándolos en LISTRC.

2.2.- Se calcula la longitud de la lista LISTRC, con lo cual se obtiene el número de diagnósticos finales.

2.3.- Se obtiene la lista de módulos visitados a partir de una estructura que utiliza MILORD llamada ESTRA-MODULS-VISITATS, depositándola en LISMOD.

2.4.- Se calcula la longitud de LISMOD, con lo cual se sabe el número de módulos visitados durante la ejecución del caso en curso.

2.5.- Se calcula el factor de enfoque para el caso en curso como el cociente de el número de diagnósticos finales entre el número de módulos visitados menos 2 (factor corrector).

Dichos cálculos se graban en un fichero de salida (FICHCONT.DAT) y se notifican por pantalla.

3.- Se calcula el factor de enfoque global resultante de la ejecución como la media aritmética de los factores de enfoque parciales de cada caso.

Para ello se han utilizado las siguientes funciones:

ENFOQUE: Esta función carga los casos que se deseen ejecutar.

PRINCIPAL: Ejecuta cada caso, llama a la función CONTAR y escribe los resultados en el fichero FICHCONT.DAT.

CONTAR: Esta función calcula el factor de enfoque resultante de la ejecución de un caso.

A-3.3.- PROGRAMA

```
;;; Función que calcula el número de diagnósticos y el
;;; número de módulos visitados.

(defun contar (caso)
     (declare (special resul))
     (setf trcp (open (concatenate 'string caso ".trc")))
     (setf listrc (read trcp))
     (close trcp)
     (setf resul1 (length listrc))
     (setf lismod (estra-moduls-visitats
                   (car llistaestrategies)))
     (setf resul2 (- (length (estra-moduls-visitats
                         (car llistaestrategies))) 2)))
```

```
;;; Esta función pide el nombre del caso que se desea
;;; ejecutar para luego calcular el factor de enfoque.

(defun enfoque ( )
     (setf caso '9)
     (do ((lista '( ) (cons caso lista)))
         ((equal caso 'ff) (acabar lista caso)))
         (format  t  "INTRODUZCA  EL  NOMBRE  DEL  CASO  ENTRE
                     COMILLAS FF PARA ACABAR : ")
         (setf caso (read))))


;;; Función auxiliar de ENFOQUE, que conecta con la
;;; PRINCIPAL

(defun acabar (lista caso)
     (setf lista (rest lista))
     (setf lista (reverse lista))
     (principal lista caso))


;;; Función principal que ejecuta el caso y llama a la
;;; función CONTAR

(defun principal (lista caso)
     (declare (special resul))
     (setf    fichcont    (open    "fichcont.dat"    :direction
                    :output :if-exists :new-version))
     (format fichcont "~%    CASO        RESULTADO ~%")
     (format t "~%    CASO        RESULTADO ~%")
     (setf cont '0)
     (setf sum '0)
     (do ((listaf lista (cdr listaf)))
          (null listaf))
          (setf caso (car listaf))
          (ejecutar-batch caso)
          (contar caso)
          (setf enf (/ resul1 resul2))
          (setf cont (1+ cont))
          (setf sum (+ enf sum))
          (format fichcont "~%    ~A          ~A  / ~A"  caso
                          resul1 resul2)
          (format t "~%    ~A          ~A / ~A"  caso
                     resul1 resul2)
          (format fichcont "~%LA LISTA DE MODULOS VISITADOS
                                   ES ~A" lismod)
          (format t "~%LA LISTA DE MODULOS VISITADOS ES ~A"
                    lismod))
     (setf enfglob (/ sum cont))
     (format fichcont "~%  EL FACTOR DE ENFOQUE GLOBAL ES
                      ~A " enfglob)
     (format t "~% EL FACTOR DE ENFOQUE GLOBAL ES ~A "
                   enfglob)
     (close fichcont))
```

BIBLIOGRAFIA

[ANDE87] ANDERSON, J.; BANDLER, W.; KOHOUT, L.J.; TRAYNER, C.. "A Route-choosing Medical Diagnostic Technique". En *Fuzzy Sets and Systems 1987*, vol. 23, pp. 89-96.

[BERG88] BERGADANO, F.; MATWIN, S.; MICHALSKI, R.S.; ZHANG, J.. "Measuring Quality of Concept Descriptions". *Proc 3th European Working Session on Learning*, Turing Institute, Glasgow, 1988.

[BONI87] BONISSONE, P.P.; DECKER, K.S.. "Selecting uncertainty and granularity: An experiment in trading-off precision and complexity", *KBS Working Paper*, General Electric Corporate Research and Development Center, Schenectady, New York, 1987.

[BOBR75] BOBROW, D.G.; COLLINS A. (Eds). *Representation and Understanding: Studies in Cognitive Science*. Academic Press, 1975.

[BREN89] BRENDER, J.; LARSEN, H.L.; LOPEZ, B.; MESEGUER, P.; NONFJALL, H.; PLAZA, E.; VAUDET, J.P.; VESOUL, P.. *Valid Project: State of the Art*, 1989.

[BROW85] BROWNSTON, L.; FARREL, R.; KANT, E.; MARTIN, N.. *Programming Expert Systems in OPS5*. Reading, Mass: Addison Wesley, 1985.

[BUCH84] BUCHANAN, B.G.; SHORLIFFE, E.H.. *Rule-Based Expert Systems: the MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, Mass: Addison Wesley, 1984.

[BUCH85] BUCHANAN, B.G.; SHORTLIFFE, E.H.. "The Problem of Evaluation". En ***Rule-Based Expert Systems: The MYCIN experiments of the Stanford Heuristic Programming Project***. Reading, Mass: Addison Wesley, 1984.

[CHAR85] CHARNIAK, E.; McDERMOTT, D.. ***Introduction to Artificial Intelligence***. Addison Wesley 1985.

[CLAN85] CLANCEY, W.J. "Software Tools for Developing Expert Systems". En ***AAVV: Artificial Intelligence in Medicine, 1985***. Elsevier Science Publishers, pp. 155-178.

[CLAN86] CLANCEY, W.J. "From GUINDON to NEOMYCIN and HERACLES twenty short lessons: ONR final report 1979-1985". ***AI Mag***. 7(3): 40-60, 1986.

[COLM83] COLMERAUER, A. et al. *"Prolog, Theoretical Principles and Current Trends". **Technology and Science of Informatics***. Vol 2, nº 4, 1983.

[CUEN86] CUENA, J. ***Lógica Informática***, Alianza Editorial, 1985.

[FEIG71] FEIGENBAUM, E.A.; BUCHANAN, B.G.; LEDERBERG, J.. "On generality and problem solving: a case study using the DENDRAL program". En ***Machine Intelligence 6***, ed. B. Meltzer, D. Michie, pp 165 90. New York: American Elsevier, 1971.

[FIES87] FIESCHI, M.. ***Inteligencia Artificial en Medicina***, ed. Masson, 1987.

[GASC83] GASCHING, J.; KLAHR, P.; POPLE, H.; SHORTLIFFE, E.; TERRY, A.. "Evaluation of Expert Systems: Issues and Case Studies". En ***Building Expert Systems***. F. Hayes-Roth, D.A. Waterman and D.B. Lenat eds., Addison-Wesley, 1983.

[GINS85] GINSBERG, A.; WEISS, S.. "SEEK2: a generalized approach to automatic knowledge base refinement", ***Proc. IJCAI 1985***, pp. 367-374.

[GINS86] GINSBERG, A.. "A metalinguistic approach tothe construction of Knowledge base refinement systems". *Proc AAAI 1986*, pp. 436-441.

[GINS88a] GINSBERG, A.. *Automatic refinement of expert system knowledge bases Artificial Intelligence*, Pitman Publ., 1988.

[GINS88b] GINSBERG, A.; WEISS, S.M.; POLITAKIS, P. "Authomatic Knowledge Base Refinement for Clasification Systems", *Artificial Intelligence*, 35(1988), pp. 197-226.

[GODO87] GODO, LL.; LOPEZ DE MANTARAS, R.; SIERRA, C.; VERDAGUER, A.. "Managing Lingüistically Expressed Uncertainty in MILORD-application to Medical Diagnosis", *AAVV: 7th International Workshop Expert Systems and their Applications*, Avignon 1987, pp. 571-596.

[GODO88] GODO, L.; LOPEZ DE MANTARAS, R.; SIERRA, C.; VERDAGUER, A.. "MILORD: The Architecture and the management of Lingüistically Expresed Uncertainty", *Int. Journal of Intelligent Systems, 1988*.

[GOLD83] GOLDBERG, A.; ROBSON, D.. *Smalltalk-80: The language and its Implementation*. Menlo Park: Addison- Wesley, 1983.

[GORR70] GORRY, G.A.. "Modelling the diagnostic process". *J. Med. Educ.* 45: 293-302, 1970.

[HART87] HARTMAN, L.; TENENBERG, J.D.. "Performance in practical problem solving". *Proc. IJCAI, 1987.*

[HAYE83] HAYES-ROTH, F.; WATERMAN, D.A.; LENAT, D.B.; eds 1983. *Building Expert Systems. Reading*, Mass: Addison-Wesley.

[HEWI77] HEWITT, C.. "Viewing control structures as patterns of passing messages". *Artif. Intell.* 8: 323-64, 1977.

[HOLL84] HOLLAND, J.D.; HUTCHINS, E.L.; WEITZMAN, L.. "STEAMER: an interactive inspectable simulation-based training system". *AI Mag.* 5(2): 15-27, 1984.

[KEEN89] KEENE S.E.. *Object-Oriented Programming in Common Lisp*, Addison Wesley 1989.

[KERS86] KERSCHBERG, L. (Ed.). *Expert Database Systems: Proceedings of the First International Workshop.* Menlo Park: Benjamin Cumminngs, 1986.

[KERS89] KERSCHBERG, L. (Ed.). *Expert Database Systems: Proceedings of the 2nd International Workshop.* Benjamin, Cummings Pub. 1989.

[KING85] KINGSLAND III, L.C.. "The Evaluation of Medical Expert Systems: Experience with the AI/RHEUM knowlwdge-based Consultant System in Rheumatology". *Proc. Ninth Anual Symposium on Computer Applications in Medical Care 1985.* pp. 292-295. IEEE Computer Society Press, Washington D. C.

[KULI82] KULIKOWSKI, C.; WEISS, S.. "Representation of expert Knowledge for consultation: The CASNET and EXPERT projects". En *Artificial Intelligence in Medicine*, ed. P. Szolovits, pp 21-55. Boulder, Colo: Westview Press, 1982.

[LOPE89] LOPEZ, B.; MESEGUER, P.; PLAZA, E.. *Validation of Knowledge Based Systems: An State of the Art*, 1989.

[McCA58] McCARTHY, J.. "Programs with common sense". En *Proc. Symp. Mechanisation of Thought Processes*, pp. 77-84, Nat. Phis. Lab. Cambridge, Mass: MIT Press, 1958.

[McCA69] McCARTHY, J.; HAYES, P.. "Some philosophical problems from the standpoint of artificial intelligence". En *Machine Intelligence 4*, ed. B. Meltzer, D. Michie, pp. 463-502. Edinburgh: Edinburgh Univ. Press, 1969.

[McDE80] McDERMOTT, D.V.; DOYLE, J.. "Non-monotonic logic I", *Artificial Intelligence*, 13(1,2) pp. 41-72, 1980.

[MILL86] MILLER, P.L.. "The evaluation of artificial intelligence systems in medicine", *Computer Methods and Programs in Biomedicine*, 22 (1986), pp. 5-11.

[MINS75] MINSKY, M.. "A framework for representing knowledge". En *The Psychology of Computer Vision*, ed. Patrick H. Winston, pp. 211-77. New York: McGraw-Hill, 1975.

[NGUY85] NGUYEN, T.A.; PERKINGS W.A.; LAFFEY, T.J.; PECORA, D.. "Checking an expert system's knowledge base for consistency and completeness". *Proc. IJCAI 1985*, pp. 375-378.

[NGUY87] NGUYEN, T.A.; PERKINGS W.A.; LAFFEY, T.J.; PECORA, D.. "Knowledge Base Verification". En *AI Magazine*, summer 1987, pp 67-75.

[NII82] NII, H.P.; FEIGENBAUM, E.A.; ANTON, J.J.; ROCKMORE, A.J.. "Signal to symbol transformation: HASP/SIAP case study". *AI Mag.* 3(2): 23-35, 1982.

[NUÑE91] NUÑEZ, G.; ALVARADO, M.; CORTES, U.; BELANCHE, Ll.. *About the relevance's nature*, Departamento de Lenguajes y Sistemas Informáticos, Facultad de Informática de Barcelona, 1991.

[O'KEE87] O'KEEF, R.M.; BALCI, O.; SMITH, E.P.. "Validating expert system performance". *IEEE Expert*. 2(4) 81-89, 1987.

[PARS89] PARSAYE, K.; CHIGNELL, M.; KHOSHAFIAN, S.; WONG, H.. *Intelligent Databases: Object-Oriented Deductive Hipermedia Technologies*, John Wiley & Sons, Inc. 1989.

[PIPA88] PIPARD, E.. "Detection d'incoherences et d'incompletitudes dans les bases de regles: le systeme INDE". *Proc. AVIGNON 1988*, 15-33.

[POST43] POST, E.. "Formal reductions of the general combinatorial problem". *American Journal of Mathematics*, 65:197-215, 1943.

[RAUC86] RAUCH-HINDING, W.B.. *Artificial Intelligence In Business, Science, and Industry: Volume I - Fundamentals, VolumeII - Applications*. Englewood Cliffs, New Jersey: Prentice Hall, 1986.

[RICH85] RICHER, M.H.; CLANCEY, W.J.. "Guindonwatch: a graphic interface for viewing a knowledge-based system". *IEEE Comput. Graph. App.* 5(11): 51-64, 1985.

[ROUS87] ROUSSET, M.C.. "Sur la validate des bases des conaissances". *Proc 7 Journees Internationales Avignon 1987*, pp. 269-282.

[SHAP84] SHAPIRO, E.Y.. "Alternation and computational complexity of logic programs". *J. Logic Programming 1*, pp. 19-33, 1984.

[SIER89] SIERRA, C.. *Milord: Arquitectura multi-nivell per a sistemes experts en classificacio*. Tesis doctoral. Universidad Politécnica de Cataluña, 1989.

[SLAG90] SLAGUE, J.R.. "Paper on Expert System Shells". *4th International Symposium on Knowledge Engeneering, 1990*.

[SMIT84a] SMITH, R.G. "On the development of commercial expert systems". *AI Mag.* 5(3): 61-73, 1984.

[SMIT84b] SMITH, R.G.; YOUNG, R.L.. "The design of dipmeter advisor system". *Proc. ACM Annu. Conf. ACM*. New York. October pp. 15-23, 1984.

[SMIT85] SMITH, R.G.; WINSTON, H.A.; MITCHELL, T.M.; BUCHANAN, B.G.. "Representation and use of explicit justifications for knowledge base refinement". En *Proceedings of IJCAI 85*, pp. 673-80. Los Altos, Calif: Morgan Kauffmann.

[SMIT87] SMITH, R.G.; BARTH, P.S.; YOUNG, R.L.. "A substrate for object-oriented interface design". En *Research Directions in Object-Oriented Programming*, ed. B. Shriver, P. Wegner, pp. 253-315, Cambridge, Mass: MIT Press, 1987.

[STEE90] STEELS, L.. "Components of Expertise", en ***AAAI 1990***, pp. 29-49, 1990.

[STEF86] STEFIK, M.J.; BOBROW, D.G.. "Object-oriented programming: themes and variations". ***AI Mag.*** 6(4):40-62, 1986.

[SUWA82] SUWA, M.; SCOTT, A.C.; SHORTLIFFE, E.H.. "An approach to verify completeness and consistency in a rule-based expert system". ***AI Magazine***, pp. 16-21, 1982.

[SZOL88] SZOLOVITS, P.; PATIL, R.S.; SCHWARTZ, W.B.. "Artificial Intelligence in Medical Diagnosis", *Annals of Internal Medicine*, pp 80-87, 1988.

[TAO87] TAO, Y.; ZHIJUN, H.; RUIZHAO, Y.. "Performance evaluation of the inference structure in expert systems". ***Proc. IJCAI 1987***, pp. 945-950.

[VANM81] VAN MELLE, W.J.. ***System Aids in Constructing Consultation Programs***, UMI Research Press, Ann Arbor, MI(1981).

[VERD89a] VERDAGUER, A.. ***PNEUMON-IA: Desenvolupament d'un sistema expert d'ayuda al diagnòstic mèdic.*** Tesis doctoral en medicina. Universidad autónoma de Barcelona, 1989.

[VERD89b] VERDAGUER, A.; PATAK, A.; SANZ, F.; SIERRA, C.; LOPEZ DE MANTARAS, R.. "Validation du systeme expert Pneumon-IA Milord". ***Proc. Journees d'Informatique Medicale de Toulouse***, Mayo 1989.

[WALK86] WALKER, T.C.; MILLER, R.K.. ***Expert Systems.*** Madison, Ga: SEAI Tech. Publ. 1986.

[WINO72] WINOGRAD, T.. ***Understanding Natural Languaje.*** Academic Press 1972.

[ZADE65] ZADEH, L.A.. "Fuzzy Sets". ***Information and Control***, Vol. 8, 1965.

[ZADE79] ZADEH, L.A.. "A theory of appoximate reasoning". En ***Machine Intelligence 9***, ed. J. E. Hayes, D. Michie, L. I. Mikulich, pp. 149-95. Chichester: Ellis Horwood Ltd., 1979.

[ZADE83] ZADEH, L.A.. "The Role of Fuzzy Sets in the Management of Uncertainty in Expert Systems". ***Fuzzy Sets and Systems***. Vol. 11, pp. 199-227, 1983.