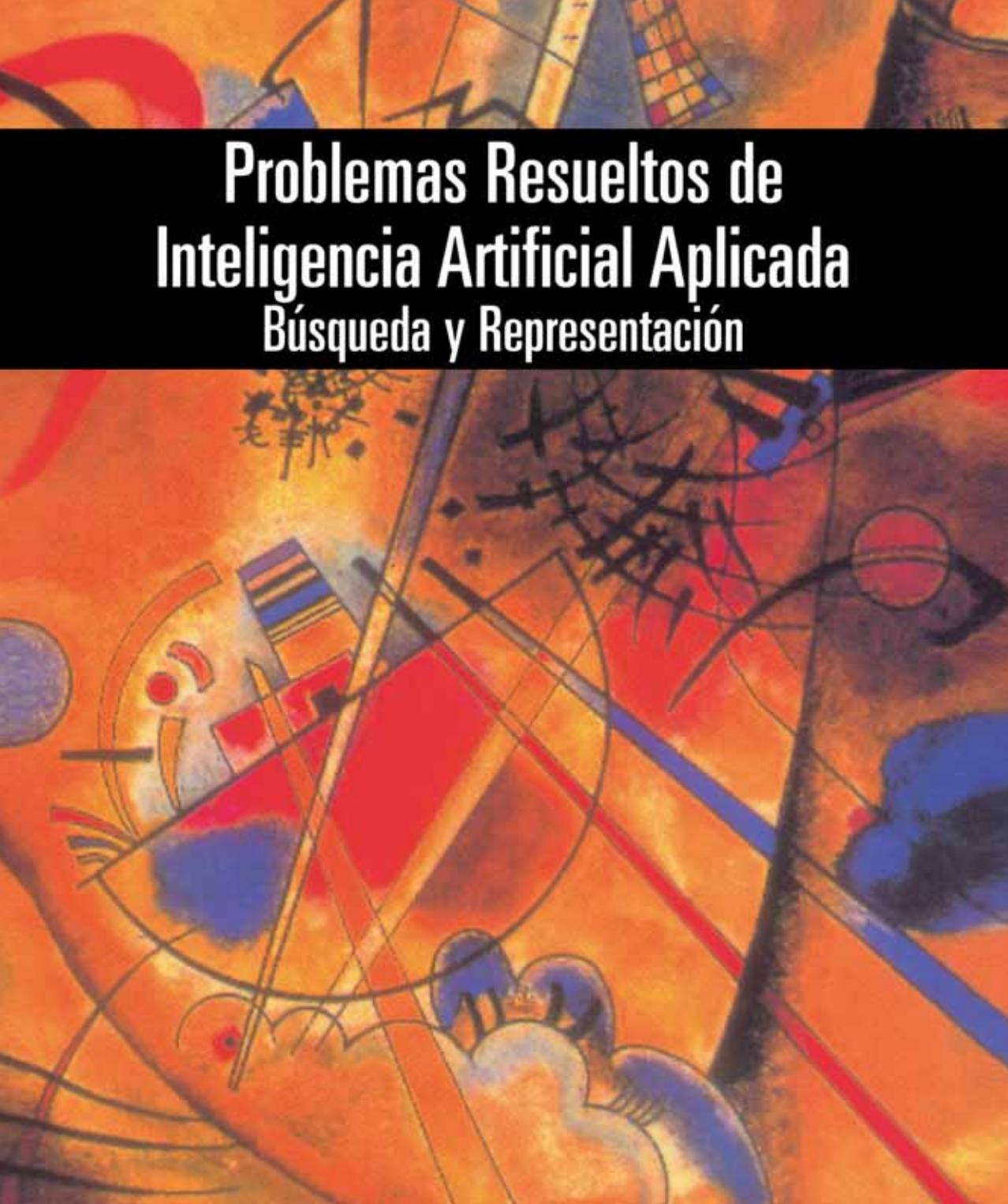


Problemas Resueltos de Inteligencia Artificial Aplicada

Búsqueda y Representación



PEARSON
Addison Wesley

Severiano Fernández Galán
Jesús González Boticario
José Mira Mira

PROBLEMAS RESUELTOS DE INTELIGENCIA ARTIFICIAL APLICADA

BÚSQUEDA Y REPRESENTACIÓN

Severino Fernández Galán
Jesús González Boticario
José Mira Mira

Universidad Nacional de Educación a Distancia
Madrid, España



Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima • Montevideo • San Juan
San José • Santiago • São Paulo • Reading, Massachusetts • Harlow, England

Severino Fernández Galán, Jesús González Boticario, José Mira Mira

**PROBLEMAS RESUELTOS DE INTELIGENCIA ARTIFICIAL APLICADA
Búsqueda y representación**

No está permitida la reproducción total o parcial de esta obra
ni su tratamiento o transmisión por cualquier medio o método,
sin autorización escrita de la Editorial.

DERECHOS RESERVADOS

© 1998 PEARSON EDUCACIÓN, S. A.
Ribera del Loira, 28
28042 Madrid (España)

ISBN: 978-84-782-9017-8

Depósito Legal: M. 12.827-2008

Última reimpresión, 2008

ISBN eBook: 978-84-7829-114-4

ADDISON WESLEY es un sello editorial autorizado de PEARSON EDUCACIÓN, S. A.

Ilustración de la cubierta: Wassily Kandinski 1925

Pequeño sueño en rojo

Composición e impresión: Publidisa

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

CONTENIDO

PRESENTACIÓN	xi
INTRODUCCIÓN	1
Motivación y guía didáctica	1
Contexto	5
Modelar conocimiento como tarea fundamental en IA	6
Contenido	13
CAPÍTULO 1. BÚSQUEDA SIN INFORMACIÓN DEL DOMINIO	19
1.1 Contexto previo	19
1.1.1 ¿Por qué está aquí este capítulo?	19
1.1.2 ¿Dónde hay conocimiento teórico del campo?	19
1.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	19
1.1.4 ¿Qué software de apoyo está disponible?	20
1.2 Contenido teórico	20
1.2.1 Introducción	20
1.2.2 Búsqueda en amplitud	20
1.2.3 Búsqueda en profundidad	22
1.2.4 Métodos derivados de los anteriores	24
1.2.5 Búsqueda general en grafos	25

1.3 Problemas resueltos	27
1.4 Contexto adicional	38
1.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	38
1.4.2 Pistas de autoevaluación	38
CAPÍTULO 2. BÚSQUEDA HEURÍSTICA	39
2.1 Contexto previo	39
2.1.1 ¿Por qué está aquí este capítulo?	39
2.1.2 ¿Dónde hay conocimiento teórico del campo?	39
2.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	40
2.1.4 ¿Qué software de apoyo está disponible?	40
2.2 Contenido teórico	40
2.2.1 Introducción	40
2.2.2 Una estrategia irrevocable: método del gradiente	40
2.2.3 Estrategias de exploración de alternativas	41
2.2.3.1 Búsqueda “primero el mejor”	41
2.2.3.2 Búsqueda en haz	42
2.2.3.3 Algoritmo A [*]	42
2.2.3.4 Exploración de grafos Y/O	44
2.2.4 Búsqueda con adversarios	48
2.2.4.1 Método MINIMAX	50
2.2.4.2 Método de poda α-β	52
2.3 Problemas resueltos	53
2.4 Contexto adicional	105
2.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	105
2.4.2 Pistas de autoevaluación	106
CAPÍTULO 3. LÓGICA	107
3.1 Contexto previo	107

Contenido	vii
3.1.1 ¿Por qué está aquí este capítulo?	107
3.1.2 ¿Dónde hay conocimiento teórico del campo?	107
3.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	108
3.1.4 ¿Qué software de apoyo está disponible?	108
3.2 Contenido teórico	109
3.2.1 Introducción	109
3.2.2 Lógica de predicados con identidad	109
3.2.3 Lógica modal	110
3.2.4 Lógica difusa	113
3.2.5 Lógicas no monótonas	119
3.3 Problemas resueltos	120
3.4 Contexto adicional	153
3.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	153
3.4.2 Pistas de autoevaluación	153
CAPÍTULO 4. REGLAS	155
4.1 Contexto previo	155
4.1.1 ¿Por qué está aquí este capítulo?	155
4.1.2 ¿Dónde hay conocimiento teórico del campo?	155
4.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	156
4.1.4 ¿Qué software de apoyo está disponible?	156
4.2 Contenido teórico	157
4.2.1 Introducción	157
4.2.2 Inferencia	158
4.2.3 Control del razonamiento o resolución de conflictos	159
4.2.4 Apéndice 4.A: Algoritmo RETE	159
4.2.5 Apéndice 4.B: Factores de certeza de MYCIN	166

viii	Contenido
4.3 Problemas resueltos	167
4.4 Contexto adicional	204
4.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	204
4.4.2 Pistas de autoevaluación	204
CAPÍTULO 5. REDES ASOCIATIVAS	207
5.1 Contexto previo	207
5.1.1 ¿Por qué está aquí este capítulo?	207
5.1.2 ¿Dónde hay conocimiento teórico del campo?	207
5.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	208
5.1.4 ¿Qué software de apoyo está disponible?	208
5.2 Contenido teórico	208
5.2.1 Introducción	208
5.2.2 Redes relacionales	209
5.2.2.1 Modelo de memoria semántica de Quillian	209
5.2.2.2 Grafos de dependencia conceptual de Schank	211
5.2.3 Redes proposicionales	213
5.2.3.1 Redes de Shapiro y grafos de Sowa	213
5.2.4 Redes de clasificación	215
5.2.5 Redes causales	215
5.2.5.1 Redes bayesianas	215
5.3 Problemas resueltos	218
5.4 Contexto adicional	244
5.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	244
5.4.2 Pistas de autoevaluación	244
CAPÍTULO 6. MARCOS Y GUIONES	245
6.1 Contexto previo	245

Contenido	ix
6.1.1 ¿Por qué está aquí este capítulo?	245
6.1.2 ¿Dónde hay conocimiento teórico del campo?	245
6.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	246
6.1.4 ¿Qué software de apoyo está disponible?	246
6.2 Contenido teórico	247
6.2.1 Marcos	247
6.2.2 Guiones	251
6.2.3 Apéndice 6.A: Ejemplo de un sistema completo: DIAVAL	253
6.3 Problemas resueltos	258
6.4 Contexto adicional	322
6.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	322
6.4.2 Pistas de autoevaluación	323
CAPÍTULO 7. PERSPECTIVA INTEGRADA	325
7.1 Contexto previo	325
7.1.1 ¿Por qué está aquí este capítulo?	325
7.1.2 ¿Dónde hay conocimiento teórico del campo?	326
7.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?	327
7.1.4 ¿Qué software de apoyo está disponible?	328
7.2 Motivación	329
7.3 Estructura y contenidos	331
7.4 Arquitecturas integradas	332
7.5 Esquema de una arquitectura	335
7.5.1 Características estructurales	337
7.5.2 Características funcionales	339
7.6 Ejemplo introductorio	346
7.7 Problemas asociados	353

x	Contenido
7.7.1 Apilar bloques	353
7.7.1.1 Representación	353
7.7.1.2 Inferencia	359
7.7.2 Revisión del problema “apilar bloques”	363
7.7.2.1 ¿Qué pretende este problema?	363
7.7.2.2 ¿De qué tipo de problemas es representativo?	365
7.7.2.3 ¿Qué dificultades puede haber encontrado?	365
7.7.3 Agenda personalizada: CAP	370
7.7.3.1 Planteamiento del problema	372
7.7.3.2 Elementos básicos	377
7.7.3.3 Duración de una reunión	383
7.7.3.4 Fecha de una reunión	388
7.7.3.5 Fecha de una reunión con aprendizaje	396
7.8 Contexto adicional	401
7.8.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?	401
7.8.2 Pistas de autoevaluación	401
REFERENCIAS	405

PRESENTACIÓN

El propósito de este libro es servir de camino de explicación o guía de navegación teórico-práctica en el campo de la **representación computacional del conocimiento científico-técnico**, tal como lo aborda la Inteligencia Artificial (IA). Su contenido está pensado para los alumnos de Informática de la **Universidad Nacional de Educación a Distancia (U.N.E.D.)** española, como complemento al libro de teoría “Aspectos Básicos de la Inteligencia Artificial” en las asignaturas de “Introducción a la Inteligencia Artificial”¹ y “Sistemas Basados en Conocimiento I”. Sin embargo, este libro puede ser también de utilidad para un amplio grupo de lectores interesados en los aspectos aplicados de la IA y sin posibilidades de una enseñanza-aprendizaje de naturaleza presencial, ya que hemos hecho un cierto esfuerzo en la estructuración de los contenidos, pensando qué problemas podría tener un alumno presencial y contestando a un conjunto amplio de esas posibles cuestiones. En la enseñanza presencial, cuando un alumno no entiende un concepto o el procedimiento de solución de un determinado problema, lo pregunta al profesor. En la enseñanza a distancia la falta del diálogo presencial debe compensarse con una mayor estructuración de los contenidos y con la previsión de esas preguntas potenciales y su contestación explícita o implícita.

Los distintos capítulos de este libro contienen problemas resueltos sobre los temas teóricos correspondientes a “búsqueda” y “representación del conocimiento” del citado texto de teoría, pero ni nosotros nos hemos limitado a este texto ni queremos tampoco que se limiten nuestros potenciales lectores. Siempre que ha sido necesario, hemos hecho referencia explícita al origen del problema, a sus fuentes de “inspiración” y a los sitios a los que puede acudir el lector para ampliar sus conocimientos o

¹ Los problemas en los que aparece un asterisco entre paréntesis al principio del enunciado fueron propuestos en exámenes correspondientes a esta asignatura.

simplemente para encontrar el conocimiento teórico que consideramos necesario y suficiente para la solución del problema.

No es sencillo encontrar “caminos de explicación” para cada una de las dificultades potenciales que pueda encontrar un alumno al intentar resolver un problema concreto. Sin embargo, los autores hemos hecho un esfuerzo en esta línea y en cada capítulo hemos seleccionado un problema y sobre él hemos sugerido los posibles sucesos (“no entiendo el enunciado”, “no entiendo una parte del enunciado”, “se hace referencia a un concepto que no conozco en profundidad”, etc.) y hemos señalado las **relaciones** y **enlaces** con otros hechos, otros problemas o contenidos teóricos con la esperanza de que el alumno sea capaz de generalizar y pasar la experiencia a los otros problemas del capítulo.

Siempre que nos ha parecido adecuado, hemos introducido “pistas” de **navegación conceptual** a varios niveles, relacionando el contenido de un problema con el contenido del capítulo y el de este capítulo en el contexto general de la obra.

El carácter distintivo del propósito que nos ha guiado a la hora de redactar este material de enseñanza-aprendizaje ha sido nuestro convencimiento de la necesidad de modelar y estructurar el conocimiento basado en ejemplos como componente esencial de la enseñanza a distancia.

Queremos agradecer a los coautores del libro de teoría (A.E. Delgado y F.J. Díez) su colaboración en todo momento. Además, a F.J. Díez le agradecemos la “cesión” de un conjunto de problemas sobre Redes Bayesianas y el resumen de su sistema DIAVAL como ejemplo de sistema implementado usando marcos.

Madrid, 8 de enero de 1998

Los autores.

INTRODUCCIÓN

Motivación y guía didáctica

El propósito de este libro es presentar un material didáctico sobre los temas de búsqueda y representación del conocimiento en el contexto de la enseñanza a distancia.

Junto a los problemas de cada capítulo aparece un material complementario cuyo propósito es situar al lector en el contexto adecuado y sustituir, al menos parcialmente, la tarea del profesor en la enseñanza presencial.

En todos los capítulos se introducen una serie de cuestiones previas que intentan predecir las que un alumno de la Universidad presencial podría formularse al comienzo de una clase de problemas o de ejercicios teórico-prácticos:

- ¿Por qué está aquí este capítulo dentro de la obra?
- ¿Dónde puedo encontrar el conocimiento teórico necesario para resolver estos problemas?
- ¿Qué conocimientos previos son necesarios para comprender este capítulo?
- ¿Qué software de apoyo a la enseñanza-aprendizaje está disponible y dónde puedo encontrarlo?, etc.

A continuación presentamos el material docente correspondiente a ese capítulo. Después, al final del capítulo, volvemos de nuevo a la función tutorial y proponemos una serie de pistas para la autoevaluación. El último apartado, usual en otras guías didácticas que aconsejan la mejor forma de navegación conceptual (*¿a dónde ir ahora?*), es de contenido obvio en este texto, dado el carácter secuencial de la organización de los temas. Primero se estudian los problemas de búsqueda y después los de representación, empezando por la lógica y terminando con los marcos. No obstante, hemos incluido bastantes referencias cruzadas cuando el contenido así lo exigía.

La figura 1 muestra el esquema del conocimiento incluido en cada capítulo distinguiendo las componentes tutoriales del material didáctico específico.

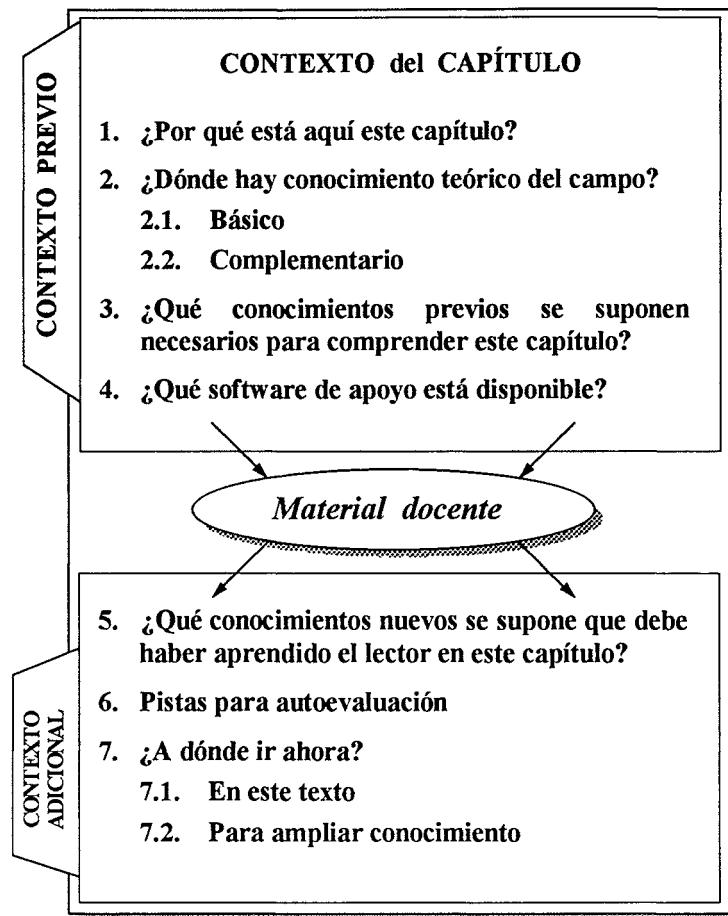


Figura 1 Esquema del conocimiento incluido en cada capítulo. Se distingue la componente tutorial (la guía didáctica pre y post-capítulo) y el material docente propiamente dicho.

Al igual que hemos intentado fijar el contexto de cada capítulo, repetimos el proceso para, al menos, un problema en cada capítulo intentando de nuevo facilitar el razonamiento del lector haciendo explícitas una serie de cuestiones que son frecuentes cuando nos enfrentamos a un problema concreto sin el apoyo presencial de un profesor. Parece razonable que nos preguntemos “¿de qué trata este problema?”. Es decir, ¿qué tipo de conocimientos ha pretendido el profesor que yo ponga de manifiesto al comprender el enunciado y aplicar un método de solución? La figura 2 intenta

esquematizar esta situación representando el proceso asociado a cada problema en términos de un conocimiento de entrada (el enunciado y los conceptos y métodos de que dispone el lector), un cuerpo del problema, consistente en un propósito y un conjunto de inferencias necesarias para la solución del problema y un conocimiento de salida que incluye lo aprendido durante el proceso de solución de ese problema. Este conocimiento de salida puede ordenarse, en general, en tres categorías: *abstracciones*, *generalizaciones* y *capacidad de predicción*. Es decir, qué conceptos y estrategias de solución usadas aquí con éxito pueden formularse a un nivel de abstracción superior para que puedan volver a ser usadas en otros problemas que consideremos análogos. Esto supone siempre un esfuerzo para sacar un mayor provecho al proceso de solución de un problema cambiando cantidad por calidad, haciendo menos problemas pero reflexionando más sobre el método y la estrategia.

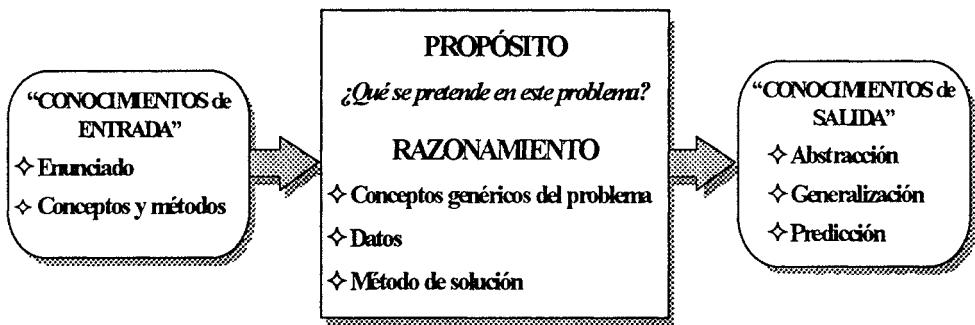


Figura 2 Esquema conceptual común a la mayoría de los problemas de inteligencia artificial aplicada.

Cada problema puede caracterizarse por un enunciado en lenguaje natural y un conjunto de conceptos y métodos de solución de problemas que, se supone, hemos aprendido en el estudio del material teórico. Lo que se pide del lector es que comprenda el enunciado, que identifique el tipo de problema genérico del que éste es un caso particular y que seleccione de sus conocimientos el método adecuado para su solución.

El resultado debe ser un incremento en el grado de comprensión de los conceptos y métodos generales, de sus casos particulares o de sus restricciones. Este incremento de conocimiento debería ponerse de manifiesto, por ejemplo, en la capacidad de generalización del "problema tipo" y/o en la predicción de resultados para nuevos datos. En muchas ocasiones este conocimiento se manifiesta a través de la capacidad de abstracción y la autoevaluación correspondiente consiste, de hecho, en la capacidad del

alumno para plantear otros problemas análogos, que mantengan la misma estructura y el mismo propósito, que usen el mismo método u otro análogo pero que operen sobre datos distintos.

En el intento de mimetizar en lo posible el diálogo profesor-alumno, propio de la enseñanza presencial, sugerimos que el lector analice sus propias dificultades en el proceso de solución. Primero, describiéndolas en lenguaje natural, después identificando las fuentes donde puede encontrarse el conocimiento necesario para “curar” esos fallos y, finalmente, intentando formular los aspectos genéricos (“reusables”) de las dificultades encontradas en la solución del problema y proponiendo otros fallos análogos. En la figura 3 se resumen estas cuestiones.

CONTEXTO DEL PROBLEMA

1. *¿Qué cree usted que pretende este problema?*

Es decir, ¿qué tipo de conocimiento cree usted que hemos pretendido que usted ponga de manifiesto al comprender el enunciado y saber resolver este problema?

2. *¿De qué tipo de problema genérico es representativo?*

Es decir, ¿qué aspectos encuentra usted en el enunciado, en los conocimientos teóricos a que hace referencia y en los procedimientos de solución que puedan ser “reutilizables” cambiando el conocimiento específico del dominio?

3. *¿Qué dificultades ha encontrado usted al resolverlo?*

Describalas en lenguaje natural (por ejemplo, no comprendo parcial o totalmente el enunciado, no conozco el significado de este o aquel concepto, no sé dónde se encuentra la teoría necesaria para su solución, no sé aplicarla, pasando de la descripción general a un ejemplo concreto, etc.).

4. *¿Sería usted capaz de proponer un enunciado de un problema análogo que le permitiera predecir un futuro examen?*

Inténtelo.

Figura 3 Cuestiones posibles para guiar la enseñanza-aprendizaje en el contexto de cada problema.

Complementando estas guías didácticas, al comienzo de cada capítulo se introduce una breve descripción del material teórico necesario y de su fuente. En otras ocasiones nos extendemos más en los aspectos teóricos porque, considerándolos necesarios, no se encuentran tratados con la extensión y/o profundidad adecuadas en los textos de teoría de nivel medio. En cualquier caso, en el apartado sobre el material bibliográfico correspondiente a los problemas de cada capítulo siempre se incluyen dos tipos de referencias: unas, de carácter básico y en número limitado y otras, más amplias, de material complementario específico para ese tema.

Contexto

Hemos hablado de la justificación de cada problema en el contexto del capítulo y de cada capítulo en el contexto del libro. Queremos ahora situar el libro en el contexto de los estudios de Informática. Este texto de problemas resueltos está pensado para complementar los estudios semestrales de una asignatura de Introducción a la Inteligencia Artificial de carácter básico. Por consiguiente, es lógico suponer que el lector de este texto posee ciertos conocimientos básicos de Lógica, Teoría de Autómatas, Estructura de Datos y Algoritmos y Programación. Análogamente, los conocimientos asociados a los temas de búsqueda y representación que aquí se tratan serán necesarios para estudios posteriores relacionados con la Ingeniería del Conocimiento, el desarrollo de Sistemas Expertos, la Visión Artificial, la Robótica Autónoma y el Aprendizaje Simbólico, por citar algunos ejemplos. Es decir, en todos aquellos estudios posteriores en los que sea necesario conocer las técnicas de búsqueda y representación del conocimiento, desde su primera descripción en el lenguaje natural del experto hasta la implementación.

La importancia de las técnicas de representación en el desarrollo de sistemas basados en conocimiento (SBC) parece evidente. Difícilmente vamos a ser capaces de desarrollar un programa que duplique alguna tarea científico-técnica si no somos capaces de representar el conocimiento que supuestamente usa el experto humano en su solución “con papel y lápiz”. Además, esta representación debe ser computable. Es decir, si un experto humano (un médico, por ejemplo) conoce la solución a un determinado problema (diagnóstico), puede contárselo a otro humano (un médico más joven y con menos experiencia, por ejemplo) usando el *lenguaje natural* y el metalenguaje propio del dominio (cardiología, por ejemplo). Sin embargo, para hacer computable la solución del mismo problema (para “contárselo” al computador), tenemos que usar un lenguaje de programación y esto nos obliga a representar las entidades y las relaciones del conocimiento intrínseco al problema y del otro conocimiento correspondiente a su procedimiento de solución usando sólo la lógica, las operaciones relacio-

nales, las reglas, las redes o los marcos y los mecanismos de inferencia que se pueden construir de forma natural sobre estas técnicas de representación. Por esto dedicamos un capítulo a cada una de estas técnicas haciendo menciones frecuentes a su uso posterior.

Hay opiniones distintas acerca de si los temas de búsqueda deben considerarse propios (específicos) de la IA, en el estado actual del conocimiento o, por el contrario, considerarse más generales y comunes a otras ramas de las Ciencias de la Computación y estudiarse, por ejemplo, en un contexto de programación y algoritmos. Sin embargo, en el contexto de un plan de estudios genérico, como el que se supone implícitamente en esta obra, no tenemos dudas acerca de la conveniencia de estudiar aquí estos temas. La búsqueda es un mecanismo general de solución de problemas cuando se dispone “a priori” de una parte importante del conocimiento necesario para resolver el problema y “sólo” tenemos que buscar hasta que lo encontramos. Los algoritmos y heurísticas de búsqueda en grafos aparecen desde en el dominio de los micromundos formales hasta en las tareas científico-técnicas más genéricas de lo humano, tales como el diagnóstico médico. Además, tienen un fuerte contenido metodológico al obligarnos a analizar un problema en términos de un conjunto de estados de conocimiento y otro conjunto de operadores de transformación, junto con ciertas funciones de coste que nos acotan el espacio de búsqueda. Finalmente, las estructuras subyacentes, los grafos, son probablemente las estructuras abstractas más comunes encontradas en computación.

Modelar conocimiento como tarea fundamental en IA

Ya hemos comentado los aspectos didácticos y el contexto en el que se encuentra el material de este libro dentro de los estudios de Informática en una Universidad a distancia, a cuyos alumnos va especialmente dirigido. Ahora queremos comentar la perspectiva global. Es decir, “con qué ojos” nos gustaría que nuestros lectores vieran estos temas. La respuesta es clara; nos gustaría que consideraran a la IA aplicada como un intento serio de modelar el conocimiento humano de la forma más análoga posible a cómo la Física y las Matemáticas han modelado el mundo no vivo. Estamos convencidos de que la tarea fundamental de la IA aplicada es desarrollar técnicas y métodos para modelar conocimiento, primero a nivel de conocimiento en el sentido de Newell, y después en el nivel de los símbolos. Por eso el estudio de las técnicas de búsqueda y de los distintos mecanismos de representación debe realizarse siempre desde la perspectiva de su uso posterior en las distintas fases del desarrollo de Sistemas Basados en Conocimiento. Empezando por la fase inicial de extracción del conocimiento (declaración explícita) a partir del diálogo con el experto humano, siguiendo con

el modelado conceptual y la selección de tareas, métodos, esquemas de control y modelos específicos de los distintos dominios de aplicación y terminando con la codificación, dentro del proceso de *reducción* de esos modelos de conocimiento hasta las fronteras de un compilador.

«Resulta algo curioso que este reconocimiento de la necesidad de modelar conocimiento como tarea básica de la IA se haya producido en la década de los 90, cerca de cuarenta años después del nacimiento oficial de la IA en la Conferencia del Dartmouth College en 1956. ¿Cómo podríamos construir un programa que resolviera ecuaciones diferenciales sin tener un modelo “claro, completo, preciso e inequívoco” del conocimiento que usa un matemático para su solución con “papel y lápiz”, junto con el conocimiento complementario necesario para reestructurar ese modelo en otro que finalmente sólo necesite las primitivas de un lenguaje de programación? Nos ha pasado con los modelos de conocimiento algo análogo a lo que Newell comentaba de su nivel de conocimiento. Todos los profesionales del campo los hemos estado usando de forma más o menos explícita pero sin reconocer de forma clara su necesidad.

Esto que es cierto para un conocimiento “fácil” de modelar, dada la naturaleza del problema (hace relación a la materia y la energía y sólo necesitamos productos, sumas, derivadas e integrales para describir la evolución temporal de las variables que caracterizan el proceso), lo es mucho más cuando no queda nada clara la fenomenología que queremos modelar. Es decir, cuando en vez de enfrentarnos a las ecuaciones de la dinámica o el electromagnetismo, nos tenemos que enfrentar al razonamiento que subyace a la conducta observable de un médico o un campeón de ajedrez, por ejemplo. Ahora, en vez de disponer de robustas variables físicas, tales como la posición, la velocidad o la temperatura y operadores analíticos suficientes para describir (predecir) su dinámica, nos encontramos con *observaciones* y *diálogos* con un experto que trata de contarnos cómo usa lo que sabe, pero que, lo diga o no, no siempre es capaz de hacer explícitos, de forma clara y precisa, los procedimientos de su sistema de pensamiento. La intuición, lo no consciente, lo pre-verbal, etc. está en la base de nuestro sistema de pensamiento». [Mira, 1997]

La diferencia fuerte de la IA con la Física y la Ingeniería es que ahora no modelamos las relaciones de la materia y la energía sino “algo” a lo que llamamos conocimiento, el cual, como la información, es pura forma. “En realidad sólo conocemos de su existencia a través de la *efectividad* en la solución de un problema científico-técnico de quien se supone que lo posee (el experto humano). Nosotros somos observadores del fenómeno del conocer y la adquisición de ese conocimiento consiste de hecho en la *construcción de un modelo* conceptual de las entidades y relaciones que supuestamente usa el experto humano, de su forma de razonar, combinando evidencia

qualitativa y numérica sobre esas relaciones con ciertos contenidos de su memoria que contienen *lo que el experto cree que sabe*” ([Mira et al., 1995]). Necesitamos además el esfuerzo adicional para “contarle a la máquina”, en su lenguaje, ese modelo que hemos construido sobre “lo que el experto cree que sabe”. Aquí está la dificultad y el desafío intelectual de la IA aplicada, lejos de otras perspectivas más antropomórficas.

Ya han pasado los tiempos en los que se concebía la adquisición del conocimiento como una tarea de transferencia de “algo” desde la cabeza del experto al computador. Ahora se acepta de forma prácticamente general que la adquisición de conocimiento es una tarea constructiva de modelado a varios niveles en la que se pretende analizar y comprender la naturaleza de la tarea, la ordenación del conjunto de entradas y salidas necesarias y el método más adecuado para la descomposición de la tarea hasta llegar al nivel de primitivas, donde ya se pueden codificar las inferencias usando sólo el conocimiento específico de cada dominio de aplicación.

Como alternativa a las arquitecturas “planas” a las que corresponden los sistemas expertos de primera generación y las formas unimodales de representación, (sólo reglas y encadenamiento como único mecanismo de inferencia o sólo marcos y herencias...), el estado actual del conocimiento en IA aplicada camina hacia la búsqueda de abstracciones recurrentes en varios dominios (tareas genéricas, o básicas) y bibliotecas de componentes reusables (estructuras de datos, métodos, esquemas inferenciales y procedimientos de control). Además, se hace cada vez más énfasis en la clasificación jerárquica del conocimiento distinguiendo esencialmente dos capas. Una capa con conocimiento del dominio, modelado mediante marcos que tienen asociados reglas y procedimientos en alguno de sus campos. La otra capa “superior”, incluye el esquema de inferencias, la estructura de la tarea (el método de descomposición en subtareas más elementales) y el conocimiento de control de las inferencias tal como se ilustra en el esquema de la figura 4. En ambas capas vamos a necesitar las técnicas de representación que se estudian aquí, en general de forma combinada. Por eso, en muchos de los ejemplos propuestos se hace referencia al supuesto uso posterior de esa representación o al mecanismo de inferencia asociado.

Conviene recordar, una vez más, que cada método de representación posee su propio mecanismo de inferencia asociado (el encadenamiento en la representación por reglas, la herencia en los marcos, etc.) y su correspondiente lenguaje de programación más idóneo, por lo que constituye, de hecho, un método para resolver problemas. Es decir, podríamos plantearnos modelar el conocimiento de una cierta tarea de monitorización o planificación usando sólo reglas pero esto nos obligaría a desarrollar un gran esfuerzo en la adquisición del conocimiento ya que tendríamos que “prepararlo” para conseguir

las funcionalidades de la tarea usando sólo el encadenamiento, que es un método muy específico de la forma de representación seleccionada. Por el contrario, si usamos métodos más generales y desligados de la representación (“establece y refina”, “propón-actúa-modifica”, “lanza una hipótesis y pruébala”, etc.) tendremos condiciones de contorno más estructuradas para ayudarnos en la adquisición del conocimiento, sin comprometerlos de momento sobre la forma de representación que usaremos más tarde. Es decir, podremos definir distintos niveles de abstracción por “encima” de la representación.

Así pues, al aceptar la perspectiva metodológica de distinguir entre *tarea, método y dominio*, los modelos a nivel de conocimiento para los que van a ser necesarios los métodos de búsqueda y representación que se estudian aquí pertenecerán a tres clases:

1. *Conocimiento específico del dominio*
2. *Conocimiento sobre el control de ejecución de la tarea*
3. *Esquema inferencial asociado al método usado para descomponer la tarea*

En la representación del conocimiento del dominio usaremos, en general, marcos. En el conocimiento inferencial usaremos un conjunto de subtareas correspondientes a los verbos de acción que aparecen en la descripción del experto. Por ejemplo, “propón-actúa-modifica” o “establece-refina”. Lo importante de estos métodos, ya lo hemos dicho, es que condicionan la forma de extraer el conocimiento del dominio. Para el control usaremos, en general, reglas.

De forma muy resumida conviene recordar aquí el proceso de desarrollo de un SBC. Este comienza con una *descripción en lenguaje natural* de la tarea que se pretende primero modelar y después codificar. En esta descripción realizada por el experto se eliminarán todos los elementos no causales y se hará énfasis en los verbos y sustantivos que apoyan el razonamiento. El siguiente paso es la *identificación* de la tarea básica que más se aproxima a las especificaciones funcionales subyacentes a la anterior descripción en lenguaje natural. Esta tarea puede ya existir en una biblioteca de “tareas genéricas”, como tareas de análisis, síntesis o modificación, o bien puede ser necesario construirla “a medida” de la aplicación. Después, para una misma tarea, pueden existir distintos *métodos* de solución. Es decir, distintos procedimientos de descomposición de la tarea básica en subtareas dando lugar a un esquema de inferencias que en el último nivel ya no necesita más que el conocimiento específico del dominio.

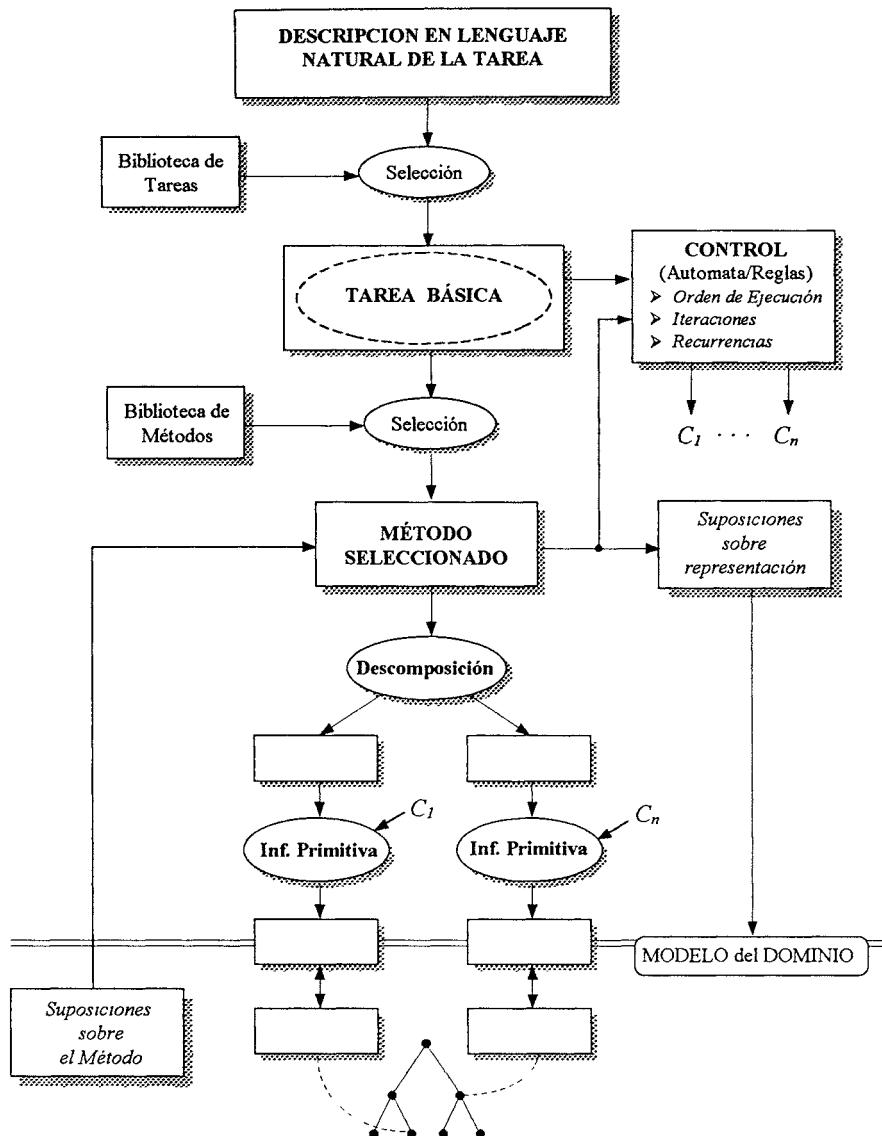


Figura 4 Esquema conceptual de las distintas fases del desarrollo de un Sistema Basado en Conocimiento. En el texto se hace referencia a los puntos en los que interviene la representación y al modo en el que lo hace (tomada de J. Mira , material docente de tercer ciclo en IA aplicada. UNED, 1997).

Un método, en un sentido no muy riguroso pero esclarecedor, es algo parecido a un algoritmo a nivel de conocimiento y determina cómo debe usarse el conocimiento del dominio para resolver un problema. Comenta Puppe que los métodos de solución de problemas y las técnicas de representación del conocimiento son como dos caras de la misma moneda. La forma en la que va a usarse un conocimiento impone restricciones sobre la representación más conveniente. Análogamente, la forma en la que se representa el conocimiento sólo alcanza su significado al considerar los procedimientos asociados. Recordemos finalmente que en el contexto de este libro sólo vamos a mencionar los métodos asociados de forma intrínseca con el mecanismo de representación (búsqueda en grafos, encadenamiento en reglas, herencia en marcos, etc.). Más adelante, en las asignaturas dedicadas de forma específica al desarrollo de SBCs se estudiarán otros métodos. Aquí sólo nos interesa recordar que ya desde las primeras fases en el desarrollo de un SBC se adquieren compromisos sobre las formas de representación que necesariamente tendrán que ser seleccionadas como fase previa a la implementación. Inversamente, el repertorio de formas de representación debe actuar siempre, de forma más o menos explícita, a la hora de dialogar con el experto humano en la fase de extracción del conocimiento.

Sigamos con el esquema de la figura 3 para detectar las interrelaciones entre la metodología usual para el desarrollo de SBCs y las formas de representación. Si ya hemos seleccionado (o construido “a medida”) la estructura de la tarea y el método de descomposición en subtareas hasta llegar a las primitivas, el siguiente paso es modelar el *conocimiento específico del dominio* teniendo en cuenta las restricciones impuestas por el método y la propia naturaleza del dominio de la aplicación.

En la construcción del modelo del dominio se hace uso de prácticamente todas las formas de representación. El proceso comienza construyendo un diccionario con todas las entidades físicas o conceptuales que hemos encontrado en la descripción en lenguaje de la tarea. En general, estos conceptos estarán asociados a sustantivos (prueba, signo, síntoma, medida, diagnóstico, temperatura...) y podrán representarse mediante marcos con campos del tipo “propiedad-valor”. Nuestra función será seleccionar de forma adecuada las propiedades de los marcos, buscar jerarquías y detectar las distintas relaciones existentes entre los marcos de distintas jerarquías. Un punto importante a la hora de especificar los marcos usados para construir el modelo del dominio es considerar el esquema inferencial y el método usado para descomponer la tarea. En la figura 4 hemos ilustrado esta dependencia con dos bloques, uno asociado al método que nos dice cómo tenemos que estructurar los objetos del dominio y otro asociado a las jerarquías de esos objetos, que nos dice cuál es el “papel” que deben representar esas estructuras de datos al actuar como “entradas” y/o “salidas” de las inferencias primiti-

vas. Es decir, al especificar su función cuando son usadas por los métodos. De nuevo es evidente la influencia de la representación en el modelado y viceversa.

Este proceso de modelar primero el nivel de “inferencia y tarea” y después el nivel de entidades específicas del dominio es recursivo. Es decir, será necesario pasar de inferencia a dominio y de aquí de nuevo a inferencia para garantizar que los datos que necesita el método los encuentra y que en el nivel del dominio no existen más que las entidades que necesita el método.

Finalmente, para que una estructura inferencial opere es necesario resolver el problema del control. Éste puede formalizarse usando reglas, autómatas finitos o algoritmos. Recuérdese que el control constituye de hecho la propia tarea. El control interno de las inferencias (orden de ejecución, iteraciones y recurrencias), que usan como entrada a las entidades del dominio, es la tercera componente básica del conocimiento que tenemos que representar.

Una vez terminada la fase de análisis, el proceso de desarrollo de un SBC continúa con la implementación del modelo resultante de esa fase de análisis (tarea, método, dominio). Si recordamos las propuestas de Newell sobre los tres niveles de la computación (de conocimiento, de los símbolos y físico), ahora estamos en la parte “mas baja” del nivel de conocimiento. Es decir, hemos obtenido un conjunto de entidades que describen el método (“propón”, “compara”, “refina”, “actúa”...) dentro de una tarea básica (diagnóstico, monitorización, planificación, diseño jerárquico, etc.) y, además, hemos construido un modelo de las entidades del dominio (diagnóstico en mecánica del automovil, planificación de terapias en oncología, etc.) usando un conjunto de marcos (“infección”, “virus”, “quimioterapia”, “sesión”, “droga”, “fiebre”...) y un conjunto de reglas (“si temperatura mayor que 37°C, fiebre”...) que tienen un significado relativamente claro y preciso en lenguaje natural pero que, evidentemente, no pasan la frontera de un compilador. Hay que asociarles las correspondientes tablas de semántica y reescribirlas usando entidades del nivel de los símbolos. Por ejemplo, las primitivas de un lenguaje como LISP o PROLOG o las facilidades de edición de un entorno como Nexpert o GoldWorks. Y de nuevo aquí es relevante el conocimiento de las distintas formas de representación que deben ser lo más próximas posibles a las que facilita el entorno. En la actualidad la mayoría de los entornos integran las distintas formas de representación y permiten trabajar con marcos, reglas y, por supuesto, con la lógica. A lo largo de los distintos capítulos del libro, en las secciones dedicadas a “guía didáctica”, siempre se incluye un apartado sobre software de apoyo en general de uso libre y haciendo referencia a las formas de representación y a los mecanismos de inferencia que incluye, junto con la dirección de la red en la que puede encontrarse.

Es difícil resumir el contenido de una asignatura dedicada al estudio del proceso de desarrollo de SBCs en sólo un apartado. De hecho, no es posible. Sin embargo, hemos introducido aquí esta sección para que el lector tenga bien claro que los capítulos que vienen a continuación no son “piezas” aisladas de conocimiento. Muy al contrario, todo su contenido queda dotado de significado al considerarlo en el contexto más general de la IA aplicada. Es decir, al considerarlo integrado en el proceso de modelar el conocimiento humano, analítico y no analítico, y hacer computables esos modelos.

Contenido

Los problemas resueltos incluidos en este texto hacen referencia a los contenidos teóricos de los temas que sobre “*búsqueda*” y “*representación del conocimiento*” incluyen la mayoría de los textos básicos (por ejemplo, temas 3 al 8 de [Mira *et al.*, 1995]. Estudiamos primero “*búsqueda*” porque en el panorama de métodos para resolver problemas, asociados en general a las distintas formas de descomponer tareas genéricas hasta llegar a las inferencias primitivas, es esencial conocer un conjunto de procedimientos de búsqueda en un espacio de “estados de conocimiento”. Estudiamos después la representación porque es evidente que sin representar y formalizar las entidades de un modelo de conocimiento, no podemos pasar a la producción de código. Además, nuestro conocimiento sobre el repertorio de métodos y técnicas de representación e inferencia disponibles nos va a condicionar a la hora de dialogar con un experto humano para extraer el conocimiento de cualquier tarea.

Las técnicas de representación y los métodos de solución de problemas limitan, establecen, las condiciones de contorno del conocimiento que seremos capaces de modelar de forma computable. Los líquidos toman la forma de la vasija que los contiene y el conocimiento toma la forma de la representación que “lo contiene”. El resto de los significados se quedan en forma de tablas semánticas y en el dominio del observador externo.

El esquema de la figura 5 muestra, en forma de grafo, la posición relativa de los distintos temas en el plan general de la obra. La forma usual de recorrer estos contenidos es la que indica la línea de trazo grueso que se corresponde con el orden de los capítulos en el texto de teoría. Sin embargo, “*búsqueda*” y “*representación*” son dos apartados que pueden estudiarse por separado. Dentro de “*búsqueda*” nos parece evidente que se estudien primero los ejemplos de búsqueda sin información del dominio y después la búsqueda con heurísticas que resumen todo lo que sabemos sobre el conocimiento que se modela en ese grafo de “estados de conocimiento” y que nos permite usar ciertas “pistas” para acotar el proceso y hacerlo más eficiente.

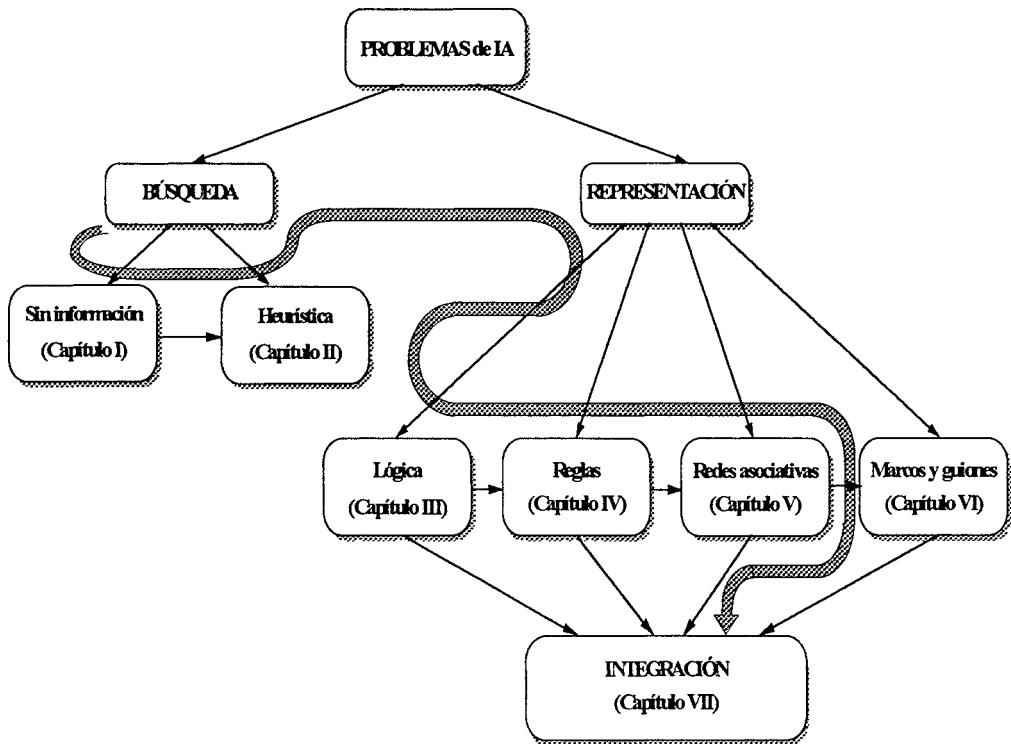


Figura 5 Contenido del libro.

En representación, empezamos con la lógica porque todas las otras formas de representación usan en mayor o menor grado la lógica de predicados. Siempre encontraremos operadores lógicos en el campo de condición de una regla o en un campo de un marco. Sin embargo, la lógica clásica es insuficiente para modelar de forma eficiente todo el conocimiento que quiere hacer computable la IA. Por eso se extiende en las lógicas modales, la lógica borrosa y las redes probabilísticas. Como existe un curso previo dedicado a la lógica, aquí sólo hacemos referencia a su uso en representación y con una extensión muy limitada.

Después de la Lógica estudiamos la representación del conocimiento mediante reglas. En algún momento histórico las reglas constituyeron el mecanismo de representación dominante. De hecho, la arquitectura de los sistemas basados en conoci-

miento de primera generación era una consecuencia de las exigencias de la representación por reglas. Ahora, tal como hemos comentado en el apartado anterior, se hace más énfasis en el modelado del conocimiento del dominio mediante jerarquías de “objetos estructurados”, en los que las reglas son una componente. Sin embargo, la representación por reglas sigue teniendo un valor indiscutible. Además, los mecanismos de inferencia asociados (encadenamiento hacia adelante y hacia atrás) siguen siendo métodos básicos y generales para la solución de problemas (métodos “débiles”). También se usarán las reglas al especificar el control de las inferencias y en ciertos campos de los marcos.

Cuando el lector se enfrente a los problemas de este capítulo debe darse cuenta que cada método de representación (las reglas, por ejemplo) se convierte en un paradigma de programación. Es decir, no podemos separar representación del conocimiento del uso posterior de “ese” conocimiento “así representado” en razonamiento e inferencia. Y lo mismo podemos decir de los mecanismos de adquisición del conocimiento que también están modulados por el método. Así, al elegir la representación por reglas, estamos aceptando que, si usamos sólo reglas, la inferencia básica será por encadenamiento, que la explicación será por “trazas”, que el conocimiento debe desmenuzarse en la fase de extracción de la forma que demanda la estructura y la sintaxis de las reglas, etc. Evidentemente, estas limitaciones desaparecen cuando las reglas se integran en otras estructuras.

Para el capítulo dedicado a las redes asociativas consideramos que sus demandas de teoría están razonablemente bien cubiertas en el texto base. Por eso, sólo incluimos un resumen de los distintos formalismos para las redes de Quillian, los grafos de dependencia conceptual de Schank, las redes proposicionales y las redes bayesianas. De todas ellas, son las redes bayesianas las que más desarrollo están experimentando en la actualidad, aunque su tratamiento aquí sea muy superficial. Hay otro tipo de red causal, las redes de neuronas artificiales, que no se menciona en este texto de problemas. La razón es que la representación neuronal (granular, de grano pequeño, distribuida por capas, con alta conectividad y autoprogammable por aprendizaje), constituye la base de la perspectiva conexionista de la IA y esta perspectiva no se considera dentro del alcance de este libro que pretende estudiar “sólo” las bases representacionales de la perspectiva simbólica de la IA aplicada. Es decir, los métodos de representación que se usan en la síntesis de sistemas basados en conocimiento de “grano grande”, cálculo localizado y programación total. Las asignaturas asociadas al desarrollo de SBCs en distintos dominios (visión, decisión, robótica, sistemas tutoriales, etc.) son el destino natural de los alumnos que hayan estudiado un curso de Introducción a la IA para el que son adecuados estos problemas, tal como comentamos al hablar del contexto.

El capítulo seis está dedicado a “Marcos y guiones”. Los marcos constituyen, de hecho el mecanismo de representación del conocimiento más general e integrador, ya que permiten usar todo el poder de los objetos estructurados e incluyen a los otros mecanismos de alguna forma. La *herencia* es, probablemente, el mecanismo de inferencia más potente en IA aplicada cuando se combina con representaciones jerárquicas. La organización del conocimiento en jerarquías busca el acople estructural con la forma natural en la que se almacena, modifica y recupera el conocimiento en los sistemas naturales. La jerarquía y la herencia están en la evolución, en la forma de crear y almacenar conceptos, en el propio conocimiento que queremos modelar y en las configuraciones subjetivas, que se derivan del modo humano de organizar su percepción del mundo que le rodea, clasificando los sucesos y situándolos en una estructura multicapa organizada en clases de conceptos. Estos conceptos (clases) se especifican en términos de su participación en un conjunto de propiedades de forma tal que los miembros de una misma clase poseen las mismas propiedades y las clases más específicas heredan las propiedades de las clases más generales.

La jerarquía y la herencia inherentes a la representación del conocimiento usando marcos no sólo va a contribuir a la formalización de los procedimientos de solución de problemas, sino que va a condicionar también la forma de extraer el conocimiento, haciendo uso amplio de la *abstracción* y ayudando a situar cada módulo de conocimiento en la *posición adecuada* dentro de la jerarquía mediante la selección de las propiedades relevantes para cada nivel, de forma tal que puedan ser compartidas en alto grado por las clases situadas en niveles inferiores. Merece la pena reflexionar sobre el posible carácter estructural de este método de representación. Además, con las actuales perspectivas metodológicas que separan el conocimiento del dominio (representado mediante jerarquía de objetos estructurados) y el conocimiento “superior” de tareas, inferencia y control, los marcos se convierten en la forma integradora y dominante de representación. La extensión de este capítulo es representativa de la importancia que, en nuestra opinión, tiene el tema.

El capítulo último es un poco diferente; su contenido no es tan básico como el de los temas previos y tampoco lo es su forma de redacción al contar desarrollos de sistemas concretos mezclando la teoría con la implementación y forzando un poco el formato “problema-solución” para contar la solución de problemas próximos al mundo real que poseen una complejidad superior a la de meros ejercicios. Además, el formalismo elegido está asociado a una opción de implementación específica y termina haciendo referencia a las técnicas de aprendizaje como un método más, muy adecuado en algunas ocasiones, para resolver problemas. Finalmente, queremos hacer notar que en este capítulo se involucra al lector en la implementación desde el primer modelo conceptual a nivel de conocimiento y, además, esto se hace de forma intencional para

que analice las ventajas e inconvenientes de esta alternativa, frente a la usual en las metodologías actuales que clasifican el conocimiento en varios niveles.

Como reflexión final de este capítulo queremos hacer explícito nuestro propósito de contribuir a la creación de material docente adecuado para la enseñanza a distancia, proponiendo mecanismos de suplencia de la actividad de un profesor presencial, al menos parcialmente. Para alcanzar este objetivo hemos dedicado casi más esfuerzo y extensión a “hablar sobre los problemas” que al propio desarrollo de su solución, la cual siempre se debe considerar como un “ejemplo bajo la ley”, como un problema tipo con un procedimiento protocolario (un “método”) de solución. Consideraremos que hemos alcanzado nuestros objetivos si el lector es capaz de identificar y clasificar los distintos problemas y sus procedimientos de solución y ser capaz de proponer otros análogos.

El propósito de la comprensión de estos problemas de búsqueda y representación es dotar al lector del conocimiento básico sobre los métodos y técnicas que, necesariamente, tendrá que usar más tarde al enfrentarse con el problema real de desarrollar un SBC para una tarea técnica concreta, donde los conocimientos sobre búsqueda y representación estarán siempre subyacentes, empezando por la etapa de extracción del conocimiento y de construcción de modelos a nivel de conocimiento y terminando con la implementación y posterior evaluación y refinamiento.

1 BÚSQUEDA SIN INFORMACIÓN DEL DOMINIO

1.1 Contexto previo

1.1.1 ¿Por qué está aquí este capítulo?

Gran parte de los primeros trabajos desarrollados en el campo de la inteligencia artificial (décadas de los cincuenta y sesenta) abordaban problemas que eran idealizaciones o simplificaciones muy fuertes del mundo real: demostración de teoremas, planificación en el mundo de los bloques, problemas de juegos, etc. La metodología propia de esta época consistía en la realización de procesos de búsqueda en espacios de estados, dando poco énfasis al empleo de conocimiento específico sobre el dominio. Las estrategias de control que guían la búsqueda marcaron uno de los principales temas de interés en este periodo y todavía hoy constituyen una parte importante de la inteligencia artificial aplicada.

1.1.2 ¿Dónde hay conocimiento teórico del campo?

Información adicional referente a los apartados teóricos de este capítulo puede encontrarse en:

Referencias básicas:

- [Mira *et al.*, 1995], capítulo 3.
- [Nilsson, 1987], capítulos 1 y 2.
- [Rich & Knight, 1994], capítulo 2.

Referencias complementarias:

- [Borrajo *et al.*, 1993], capítulos 2 y 4.
- [Winston, 1994], capítulo 4.

1.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

- Conceptos básicos sobre árboles y grafos.

- Nociones elementales sobre estructuras de datos como pilas, colas y listas.
- Conocimientos sobre cálculo de la complejidad de un algoritmo.
- Nociones sobre conceptos propios de programación tales como recursividad e iteratividad.

1.1.4 ¿Qué software de apoyo está disponible?

Acudir al mismo apartado del capítulo 2.

1.2 Contenido teórico

1.2.1 Introducción

Los procesos de *búsqueda* tienen sentido en problemas que reúnen una serie de características:

- Cabe la posibilidad de asociar un conjunto de *estados* a las diferentes situaciones en que se puede encontrar el objeto del *dominio* sobre el que se define el problema.
- Hay una serie de *estados iniciales* desde los que empezaría el proceso de búsqueda.
- Existen ciertos *operadores*, tal que un operador aplicado sobre un estado producirá otro estado.
- Existe al menos un *estado meta* o *estado solución*.

Cualquier proceso de búsqueda persigue, asociando nodos con estados y arcos con operadores, encontrar un camino que conduzca de un nodo inicial a otro meta. Se define el *espacio de estados* como el conjunto de los mismos que podrían obtenerse si se aplicaran todos los operadores posibles a todos los estados que se fueran generando. La búsqueda sin información del dominio pretende realizar una *exploración exhaustiva* del espacio de estados —dado que no hay *conocimiento* que pueda guiar la misma— que, en principio, no deje ningún *nodo* sin ser examinado. Existen diversas formas de llevar a cabo el proceso anterior:

1.2.2 Búsqueda en amplitud

La búsqueda puede estar orientada a recorrer el *árbol* o *grafo* correspondiente por niveles. Para conseguirlo se utiliza una estructura tipo cola (FIFO) en la que se van introduciendo los nodos que son generados. Este tipo de exploración recibe el nombre

de *búsqueda en amplitud* y garantiza la obtención de la solución de menor coste (óptima), si es que ésta existe.

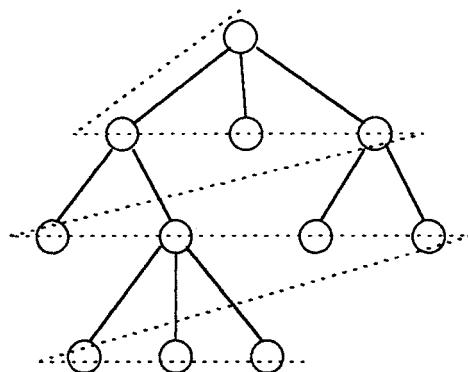


Figura 1-1 Recorrido en amplitud de izquierda a derecha de un árbol.

Algoritmo de *búsqueda en amplitud*:

1. Crear una lista de nodos llamada ABIERTA y asignarle el nodo raíz, que representa el estado inicial del problema planteado.
2. Hasta que ABIERTA esté vacía o se encuentre una meta, realizar las siguientes acciones:

2.1 Extraer el primer nodo de ABIERTA y llamarlo m .

2.2 Expandir m (generar todos sus sucesores). Para cada operador aplicable y cada forma de aplicación:

- (1) Aplicar el operador a m , obtener un nuevo estado y crear un puntero que permita saber que su predecesor es m .
- (2) Si el nuevo estado generado es meta, salir del proceso iterativo iniciado en **2.2** y devolver dicho estado.
- (3) Incluir el nuevo estado al final de ABIERTA. (Una vez completado este proceso para todos los sucesores de m —cuando no se haya encontrado antes una meta— se continúa el proceso iterativo en el paso 2.)

Observación: si el algoritmo termina con una meta, el camino de la solución puede obtenerse recorriendo los punteros creados desde el nodo meta al nodo raíz. En caso

contrario, el proceso terminaría sin haber encontrado la solución. Más adelante, en la tabla 1-1 aparece la complejidad de este algoritmo.

PROBLEMAS RELACIONADOS: 1.1, 1.2.

1.2.3 Búsqueda en profundidad

La *búsqueda en profundidad* se centra en expandir un único camino desde la raíz; en el caso de llegar a un “callejón sin salida” se retrocede hasta el nodo más cercano desde donde se pueda tomar una rama alternativa para poder seguir avanzando. Para llevar a cabo este tipo de búsqueda debe utilizarse una estructura tipo pila (LIFO) que vaya almacenando los nodos generados. Suele establecerse, por otra parte, el llamado *límite de exploración*, que marca la máxima longitud que puede alcanzar cualquier camino desde la raíz durante el proceso de búsqueda.

Una ligera variante de la búsqueda en profundidad es la *búsqueda con retroceso*. En ésta última, en lugar de generar todos los sucesores al expandir un nodo, se genera un único sucesor en cada paso.

En la figura 1-2 aparece un ejemplo de recorrido en profundidad de un árbol mediante este tipo de búsqueda.

Algoritmo de *búsqueda en profundidad*:

1. Crear una lista de nodos llamada ABIERTA y asignarle el nodo raíz, que representa el estado inicial del problema planteado.
2. Hasta que se encuentre una meta o se devuelva fallo, realizar las siguientes acciones:
 - (1) Si ABIERTA está vacía, terminar con fallo; en caso contrario, continuar.
 - (2) Extraer el primer nodo de ABIERTA y denominarlo m .
 - (3) Si la profundidad de m es igual a l_p (límite de profundidad), regresar a 2; en caso contrario, continuar.
 - (4) Expandir m creando punteros hacia este nodo desde todos sus sucesores, de forma que pueda saberse cuál es su predecesor. Introducir dichos sucesores al principio de ABIERTA siguiendo un orden arbitrario. (La “falta de orden” refleja el carácter no informado de este procedimiento.)

- (4.1) Si algún sucesor de m es meta, abandonar el proceso iterativo señalado en 2, devolviendo el camino de la solución, que se obtiene recorriendo los punteros de sus antepasados.
- (4.2) Si algún sucesor de m se encuentra en un “callejón sin salida”, eliminarlo de ABIERTA. (Se continúa el proceso iterativo en el paso 2.)

Observación: en el paso (4), en lugar de generar todos los sucesores de m , se podría haber generado sólo uno. En este caso, (4.1) y (4.2) se particularizarían para ese único nodo generado. El procedimiento se modificaría para convertirse en *búsqueda en retroceso* (*backtracking* en inglés), tal como se indica más adelante. Ver la complejidad de este algoritmo en la tabla 1-1.

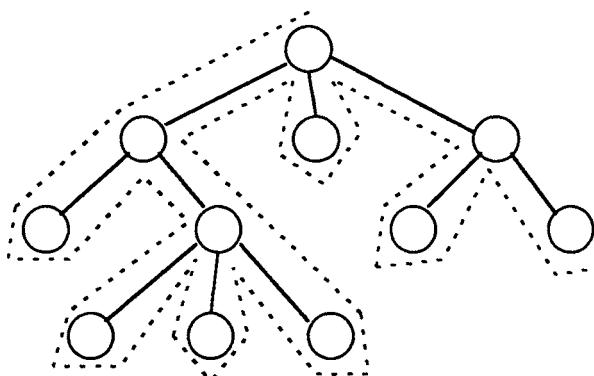


Figura 1-2 Recorrido en profundidad de izquierda a derecha de un árbol.

PROBLEMAS RELACIONADOS: 1.1.

Algoritmo de *búsqueda con retroceso*:

1. Crear una lista de nodos llamada ABIERTA y asignarle el nodo raíz, que representa el estado inicial del problema planteado.
2. Hasta que se encuentre una meta o se devuelva fallo, realizar las siguientes acciones:
 - (1) Si ABIERTA está vacía, terminar con fallo; en caso contrario, continuar.
 - (2) Seleccionar el primer nodo de ABIERTA y denominarlo m .

- (3) Si la profundidad de m es igual a lp o si m ya no tiene más sucesores posibles (que no hayan sido examinados anteriormente), eliminarlo de ABIERTA y regresar a 2; en caso contrario, continuar.
- (4) Generar un nuevo sucesor m' de m e introducirlo al principio de ABIERTA, creando un puntero a m , y señalar que dicha rama ya ha sido considerada.
- (4.1) Si m' es meta, abandonar el proceso iterativo iniciado en el paso 2, devolviendo el camino de la solución, que se obtiene recorriendo los punteros de sus antepasados.
- (4.2) Si m' se encuentra en un callejón sin salida, eliminarlo de ABIERTA. (Se continúa el proceso iterativo en el paso 2.)

(Ver complejidad más adelante en la tabla 1-1)

PROBLEMAS RELACIONADOS: 1.1, 1.2.

1.2.4 Métodos derivados de los anteriores

Una variante de los métodos anteriores es la *búsqueda en profundidad progresiva*, que no es más que una búsqueda en profundidad por niveles donde el límite de exploración va aumentando progresivamente. La gran ventaja de este método es que es capaz de encontrar la solución de menor coste existente —al igual que en el caso de búsqueda en amplitud— con una complejidad espacial únicamente del orden de $O(p)$ —al igual que en el caso de la búsqueda en retroceso—. Por tanto, reúne las ventajas de los métodos vistos hasta ahora.

Otra variante es la *búsqueda bidireccional*, donde se parte de un estado inicial desde el que se busca la meta y de un estado meta desde donde se busca el estado inicial; con la única condición de que una de las dos búsquedas anteriores sea en amplitud, se garantiza que si existe algún *camino solución*, en algún momento las dos búsquedas pasarán por un estado común desde el que se podrá reconstruir dicho camino.

En la tabla 1-1 aparecen las complejidades de estos algoritmos.

PROBLEMAS RELACIONADOS: 1.3.

A continuación figuran las complejidades temporal y espacial de todos los algoritmos vistos:

COMPLEJIDADES	TEMPORAL	ESPACIAL
Búsqueda en amplitud	$O(n^p)$	$O(n^p)$
Búsqueda en profundidad	$O(n^p)$	$O(n \cdot p)$
Búsqueda con retroceso	$O(n^p)$	$O(p)$
Búsqueda en profundidad progresiva	$O(n^p)$	$O(p)$
Búsqueda bidireccional en amplitud	$O(n^{p/2})$	$O(n^{p/2})$

Tabla 1-1 Complejidades de los algoritmos.

n : “factor de ramificación” (número medio de sucesores de todos los nodos)

p : “profundidad de la solución”

1.2.5 Búsqueda general en grafos

En cualquier proceso de búsqueda es importante tener la posibilidad de determinar si un nuevo estado ya había sido generado y expandido previamente, pues con ello se evitaría repetir la exploración de determinados caminos. El algoritmo general de búsqueda en grafos se basa en la idea anterior. Este algoritmo general maneja dos listas denominadas ABIERTA y CERRADA. En ABIERTA figuran los nodos del grafo que han sido generados, pero que todavía no han sido expandidos. Por otra parte, en CERRADA se van almacenando aquellos nodos de ABIERTA que son seleccionados para su expansión; esta selección puede ser informada o no. Si en algún momento durante el proceso de ejecución del algoritmo se quiere reanudar un camino abandonado, no hay más que acudir a la lista ABIERTA para poder hacerlo. Cualquier algoritmo particular basado en éste de búsqueda general en grafos, podrá seleccionar los nodos de ABIERTA según un criterio propio que consista en calcular el valor de una función de evaluación heurística, como pudiera ser “menor distancia estimada a la meta”. Además del grafo parcial que se va generando durante la ejecución del algoritmo, también se gestiona un árbol cuyos enlaces indican los caminos de menor coste encontrados en cualquier momento desde cada nodo del grafo parcial al nodo raíz. Este árbol va cambiando progresivamente, ya que en el proceso de generación del grafo parcial mencionado puede aparecer un nuevo camino más corto desde la raíz a un determinado nodo. Esto produce una reorientación del puntero del árbol que parte de dicho nodo. Hay que considerar tres situaciones diferentes a la hora de realizar la expansión de un nodo:

- a) Que el nodo generado no esté ni en ABIERTA ni en CERRADA. En este caso basta con enlazar el nodo generado con su padre.

b) Que el nodo generado esté en ABIERTA. Ahora habrá un nuevo camino desde el nodo raíz hasta el nodo generado. Por tanto, debe determinarse si este nuevo camino es menos costoso que el anterior. En caso afirmativo hay que redirigir el puntero del árbol que parte del nodo generado.

c) Que el nodo generado ya estuviera en CERRADA. Si el nuevo camino es menos costoso, además de llevar a cabo los pasos del caso anterior, hay que hacer un estudio de los descendientes del nodo generado por si hubiera que redirigir sus enlaces actuales.

Algoritmo de búsqueda general en grafos:

1. Crear un grafo de exploración G que consista en un único nodo que contiene la descripción del problema. Crear una lista de nodos llamada ABIERTA y asignarle dicho nodo.

2. Crear una lista de nodos llamada CERRADA que inicialmente estará vacía.

3. Hasta que se encuentre una meta o se devuelva fallo, realizar las siguientes acciones:

(1) Si ABIERTA está vacía, terminar con fallo; en caso contrario, continuar.

(2) Eliminar el primer nodo de ABIERTA, llamarlo m e introducirlo en la lista CERRADA.

(3) Expandir m :

(3.1) Generar el conjunto M de todos sus sucesores que no sean antecesores e introducirlos como sucesores de m en G .

(3.2) Si algún miembro de M es meta, abandonar el proceso iterativo señalado en 3, devolviendo el camino de la solución, que se obtiene recorriendo los punteros de sus antepasados.

(3.3) Poner un puntero a m desde los nuevos nodos generados (aquellos que no pertenezcan a G). Incluir dichos elementos en ABIERTA.

(3.4) Para cada nodo de M que ya estuviese en ABIERTA o en CERRADA, decidir si se redirige o no su puntero a m .

(3.5) Para cada nodo de M que ya estuviese en CERRADA, decidir para cada uno de sus descendientes si se redirigen o no sus punteros.

(4) Reordenar la lista ABIERTA aplicando algún criterio previamente establecido.

En el próximo capítulo de búsqueda heurística aparecen varios problemas de aplicación de algoritmos basados en éste más general.

1.3 Problemas resueltos

- **PROBLEMA 1.1: (*)¹**

Dado el árbol de la figura donde B y L son los dos únicos nodos meta y A es el nodo inicial:

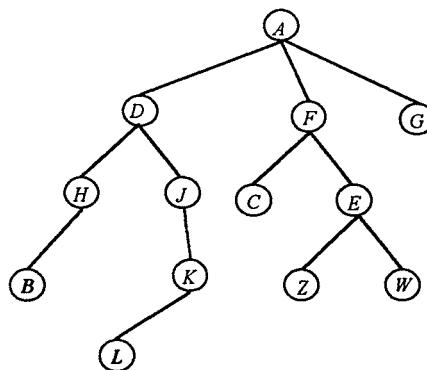


Figura 1-3

indique el orden en que se visitan los nodos, distinguiendo los que sólo se han generado de aquéllos que se han elegido en el proceso de búsqueda de la solución, para cada uno de los procedimientos siguientes:

- Búsqueda en amplitud
- Búsqueda en profundidad
- Búsqueda con retroceso

Se supone que en cada nivel del árbol los nodos se recorren de izquierda a derecha y que el límite de profundidad es 5.

¹ Los problemas en los que aparece un asterisco fueron propuestos en exámenes correspondientes a la asignatura “Introducción a la inteligencia artificial” perteneciente al plan de estudios de la Ingeniería Técnica en Informática de Sistemas de la U.N.E.D.

SOLUCIÓN:

¿Qué pretende este problema? En este problema, así como en el 1.2, se aplican algoritmos básicos de búsqueda no informada en un espacio de estados.

a) Búsqueda en amplitud

Se va a describir cuál es el contenido de la lista ABIERTA, que en este algoritmo va a actuar como cola (primero en llegar, primero en salir; FIFO). Los nodos que haya en ABIERTA serán aquéllos que hayan sido generados, pero que todavía no hayan sido expandidos. Siguiendo los pasos del algoritmo que aparece en el apartado teórico:

Punto 1) ABIERTA: A

2.1) $m = A$

2.2) ABIERTA: D, F, G

2.1) $m = D$

2.2) ABIERTA: F, G, H, J

2.1) $m = F$

2.2) ABIERTA: G, H, J, C, E

2.1) $m = G$

2.2) ABIERTA: H, J, C, E

2.1) $m = H$

Por ser **B** un nodo meta, el paso 2.2.(2) pone fin a la ejecución del algoritmo. Por tanto, el orden en que han sido visitados los nodos es: *A, D, F, G, H, J, C, E, B*.

Los elementos que van a ser expandidos se toman del comienzo de la lista ABIERTA. Sus sucesores se añaden al final. De esta forma siempre se expanden primero los nodos más antiguos.

b) Búsqueda en profundidad

Es similar al caso de búsqueda en amplitud, pero ahora ABIERTA va a actuar como una pila (último en entrar, primero en salir; LIFO). Se sigue extrayendo el primer nodo de ABIERTA y se añaden al principio de la misma todos sus sucesores, siendo el orden de colocación irrelevante (es un algoritmo no informado).

A partir del algoritmo del apartado teórico:

Punto 1) ABIERTA: A

2.(2) $m = A$

2.(4) ABIERTA: D, F, G

2.(2) $m = D$

2.(4) ABIERTA: H, J, F, G

2.(2) $m = H$

2.(4) ABIERTA: B, J, F, G

Por ser B un nodo meta, el algoritmo finaliza su ejecución en el punto **2.(4.1)**. Ahora el orden para llegar al estado meta ha sido: A, D, H, B

Hay que reseñar que en el proceso de ejecución del presente algoritmo se podría haber introducido en otro orden los nodos en la pila y como consecuencia de ello el camino no habría resultado tan directo.

c) Búsqueda con retroceso

Es similar a la búsqueda en profundidad, aunque ahora no se generan todos los sucesores de un nodo en cada paso del algoritmo, sino sólo uno cualquiera de ellos. La eficiencia del algoritmo depende del orden en que se expanden los nodos (al ser no informado no se aplica ningún criterio para elegirlos). A partir del algoritmo del apartado teórico:

Punto 1) ABIERTA: A

2.(2) $m = A$

2.(4) $m' = D$; ABIERTA: D, A

2.(2) $m = D$

2.(4) $m' = H$; ABIERTA: H, D, A

2.(2) $m = H$

2.(4) $m' = B$; ABIERTA: B, H, D, A

2.(4.1) Fin, por ser B un nodo meta.

En este caso la selección de un nodo de ABIERTA no implica que sea sacado de dicha lista. Esto último ocurre cuando el nodo seleccionado ha llegado a un callejón sin salida, es decir, no quedan más sucesores suyos por seleccionar o cuando dicho nodo no

tiene más sucesores posibles. La razón de esto es que puede ser necesario, en caso de retroceso, retomar un camino distinto no escogido previamente. El camino encontrado es: A, D, H, B .

• **PROBLEMA 1.2:**

Dado el siguiente grafo, donde A es el nodo inicial y H el nodo meta, explorarlo mediante los siguientes métodos:

- a) Búsqueda en amplitud
- b) Búsqueda con retroceso

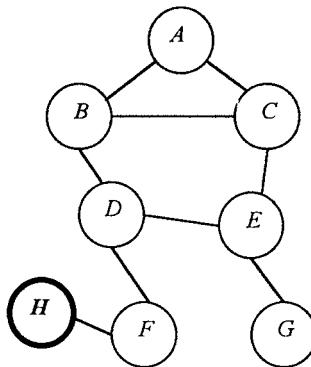


Figura 1-4

SOLUCIÓN:

a) Búsqueda en amplitud

En el proceso de generación del árbol de búsqueda, al expandir un nodo se van a generar todos sus sucesores que no sean antecesores del mismo en dicho árbol. Los números que acompañan a los nodos de la figura 1-5 muestran el orden de expansión de los mismos.

b) Búsqueda con retroceso

Aquellos enlaces que llevan a “callejones sin salida” se representarán mediante líneas a trazos. Etiquetando los arcos por el orden de generación de los mismos, se obtiene el resultado que aparece en la figura 1-6. Obsérvese cómo en esta estrategia de búsqueda sólo es necesario almacenar en cada momento un camino hasta el nodo raíz, al contrario que en el método de búsqueda en amplitud.

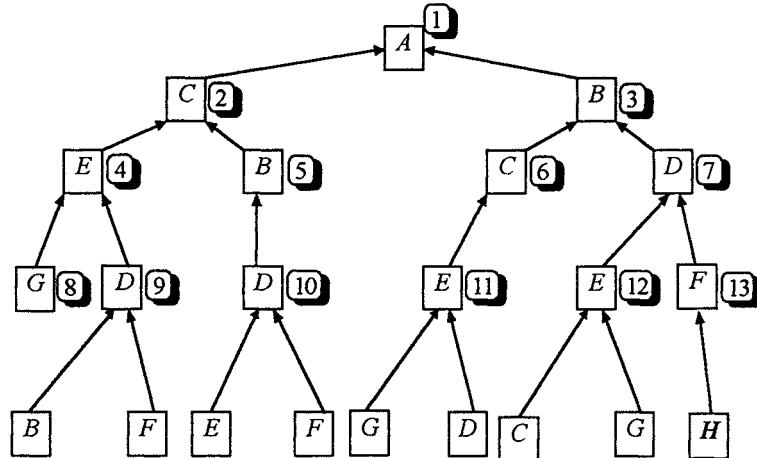


Figura 1-5

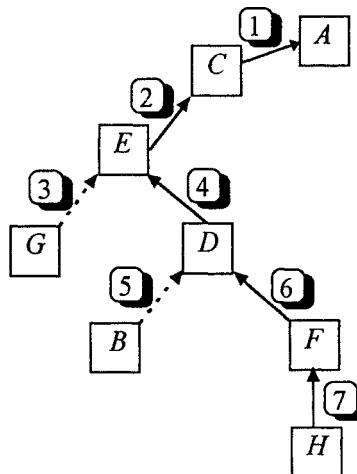


Figura 1-6

• PROBLEMA 1.3:

Se dispone de dos cántaros de agua, uno de 4 litros y otro de 3 litros de capacidad, siendo ésta la única información que se tiene de los mismos. Existe una bomba de agua con la que se pueden llenar los cántaros. Se desea que el cántaro de 4 ls. de capacidad quede lleno por la mitad y el de 3 ls. vacío. Abordar esta cuestión como un problema de búsqueda en un espacio de estados.

SOLUCIÓN:

¿Qué pretende este problema? Se presenta un ejemplo donde es necesario definir un espacio de estados a partir de un problema concreto real. Además, se muestra la conveniencia de realizar en dicho espacio una búsqueda derivada de los tipos básicos.

El espacio de estados para este problema consistirá en el conjunto de pares de enteros (x, y) , tal que $x = 0, 1, 2, 3$ o 4 e $y = 0, 1, 2$ o 3 , donde x e y representan el número de litros de agua que hay en los cántaros de 4 y 3 litros respectivamente. Se considerará que el estado inicial es $(0, 0)$ y el estado meta $(2, 0)$. En cuanto a los operadores que se pueden aplicar a los estados descritos con anterioridad, se tendrán los siguientes:

1. $(x, y) \longrightarrow (4, y)$ *Llenar el cántaro de 4 ls.*
si $x < 4$
2. $(x, y) \longrightarrow (x, 3)$ *Llenar el cántaro de 3 ls.*
si $y < 3$
3. $(x, y) \longrightarrow (0, y)$ *Vaciar en el suelo el cántaro de 4 ls.*
si $x > 0$
4. $(x, y) \longrightarrow (x, 0)$ *Vaciar en el suelo el cántaro de 3 ls.*
si $y > 0$
5. $(x, y) \longrightarrow (4, y - (4 - x))$ *Verter agua del cántaro de 3 ls. al de 4 hasta llenarlo.*
si $x + y \geq 4$,
 $y > 0$ y
 $x < 4$
6. $(x, y) \longrightarrow (x - (3 - y), 3)$ *Verter agua del cántaro de 4 ls. al de 3 hasta llenarlo.*
si $x + y \geq 3$,
 $x > 0$ e
 $y < 3$

7. $(x, y) \longrightarrow (x + y, 0)$ Verter todo el agua del cántaro de 3 ls. al de 4.

si $x + y \leq 4$ e

$y > 0$

8. $(x, y) \longrightarrow (0, x + y)$ Verter todo el agua del cántaro de 4 ls. al de 3.

si $x + y \leq 3$ y

$x > 0$

Respecto al proceso de búsqueda en sí que se va a emplear, se puede aprovechar el hecho de que se conoce con exactitud cuáles son el estado inicial y meta. Como consecuencia de ello, el método de *búsqueda bidireccional en amplitud* (ver apartado teórico) puede ser apropiado para abordar este problema.

El proceso de búsqueda tendrá las siguientes características:

a) Por un lado, se parte del estado inicial y se realiza un encadenamiento hacia adelante de los operadores. En cada paso, para cada estado se seleccionan aquellos operadores que puedan ser aplicables al mismo (comparando el estado con el antecedente de cada operador), generándose una serie de estados nuevos y un árbol de búsqueda que será recorrido en amplitud.

b) Por otro lado, se parte del estado final y se realiza un encadenamiento hacia atrás de los operadores. Ahora, en cada paso, para cada estado se seleccionan aquellos operadores que, una vez ejecutados, pueden llegar a producir el estado mencionado (comparando dicho estado con el consecuente de cada operador), generándose en este caso también una serie de estados nuevos y un árbol de búsqueda que será recorrido en amplitud. A diferencia del apartado a), en éste hay que ir propagando una serie de ligaduras o restricciones entre las variables que definen cada estado a lo largo del árbol (ver figura 1-7).

c) En cuanto a cada una de las dos búsquedas en amplitud y los dos árboles que se van creando, reseñar lo siguiente:

1) Puede ocurrir que la aplicación de un operador conduzca a un estado que coincida con un antecesor suyo en el árbol (ciclo). En este caso no se aplicará este operador.

2) Si no ocurre lo mencionado en el apartado c.1), pero el estado generado ya figuraba en el árbol de búsqueda, no tiene sentido seguir la búsqueda por este estado, puesto que la misma ya se está realizando en otra parte del árbol. Cuando esto ocurra se subrayará el estado donde se pare la búsqueda y no se colgará ningún subárbol de él.

3) Puede ocurrir que no se pueda aplicar ningún operador a un determinado estado. En este caso se subrayaría doblemente dicho estado (ver figura 1-7).

4) La generación de niveles será alternativa en los dos árboles existentes. Primero se expandirá un nivel de un árbol, luego otro nivel del otro... En cada generación de un nuevo nodo o estado se comprobará si éste figura en el otro árbol de búsqueda. En caso afirmativo, la búsqueda finaliza, ya que se ha encontrado un punto de unión entre los estados inicial y meta y, por tanto, se ha hallado un camino solución tal como se puede apreciar en la figura 1-7.

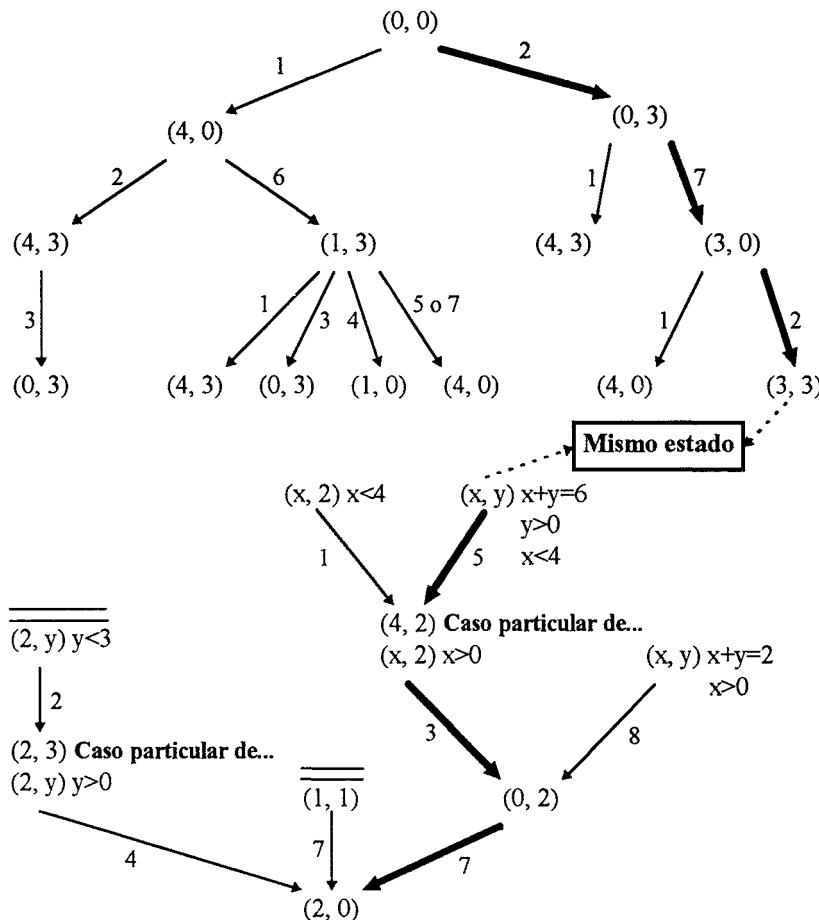


Figura 1-7

Por tanto, el camino solución encontrado es:

$$(0,0) \xrightarrow[2]{ } (0,3) \xrightarrow[7]{ } (3,0) \xrightarrow[2]{ } (3,3) \xrightarrow[5]{ } (4,2) \xrightarrow[3]{ } (0,2) \xrightarrow[7]{ } (2,0)$$

¿Qué dificultades puede haber encontrado en este problema?

El proceso de lectura de un texto general, o de un libro de problemas en particular, puede convertirse en un tarea nada fácil. El lector se encontrará, sin duda, con partes de la obra que no entiende, lo cual impedirá un correcto aprovechamiento de los contenidos de la misma. Sería conveniente que las explicaciones fueran tan minuciosas y claras que no existiera la posibilidad de que cualquier lector no fuera capaz de asimilar los contenidos. Sin embargo, lo que se habría ganado en hacer más didáctico un texto, se habría perdido en precisión y concisión; es decir, para un lector ya familiarizado con los temas desarrollados, la lectura de un libro de tal naturaleza resultaría tediosa.

Quizá una solución viable en el proceso de facilitar la asimilación de los contenidos de un texto a un lector indeterminado consistiría en el diseño de un sistema tutorial inteligente paralelo al propio libro. Este sistema se encargaría de ayudar al lector sólo cuando éste lo solicitara y debería poseer conocimiento sobre qué posibles dudas podrían aparecer en la consulta del texto por un lector y cómo solucionarlas. Sería, por tanto, una especie de sistema experto que imitara la labor desempeñada por el autor del libro si éste tuviera que responder a las cuestiones que le pudieran ser planteadas sobre el texto por un lector.

Parece claro, después de lo comentado en párrafos previos, que el campo de la inteligencia artificial puede resultar adecuado para la implementación de estos sistemas tutoriales, tanto para estructurar y modelar el conocimiento explícito que aparece en el texto como para ayudar a tomar la decisión de qué acción llevar a cabo ante cada posible duda que se le pueda presentar al lector.

Lógicamente, la intención de los autores de este libro habría sido dotarlo de un completo sistema tutorial de apoyo, pero la magnitud de tal labor (es comparable a la elaboración del propio libro) hace que finalmente sólo se den unas cuantas pinceladas de lo que habría sido tal sistema. Estas pinceladas se basan en suponer que la lectura del libro se realiza de forma secuencial, intentando asimilar los conceptos teóricos que aparecen al principio de cada capítulo antes de intentar resolver los problemas correspondientes al mismo. A continuación queda reflejado, en forma de reglas de producción, parte del conocimiento correspondiente al sistema tutorial mencionado.

Sería conveniente que el lector aplicara este conocimiento general siempre que se le presentara una duda al leer el texto. Se han aplicado estas reglas a los problemas 1.3, 2.11, 3.15, 4.3, 5.3, 6.4 y 7.7.1 donde, al final de los mismos, en un apartado que ha sido llamado “Revisión del problema”, figuran las acciones a efectuar para cada posible duda que pudiera aparecer. Algunas de las reglas que reflejan el conocimiento referente a las dificultades que puede encontrar el lector a la hora de comprender un problema son:

Regla A (antecedente):

El lector no entiende una parte del enunciado de un problema.

Regla A (consecuente):

El lector no ha asimilado la parte de contenido teórico que dio pie a la pregunta que aparece en el enunciado. Por tanto, debería regresar a dicho apartado teórico y releerlo. Si ello no es suficiente, debería acudir a otros textos.

Regla B (antecedente):

El lector no sabe interpretar una figura que aparece en un problema.

Regla B (consecuente):

Probablemente exista otra figura más explicativa que pueda ayudar en la interpretación de la presente. Para buscar tal figura, acudir, por este orden, a:

- a) Otros problemas previos del libro similares al actual.
- b) Apartado teórico del libro relacionado con el enunciado del problema.
- c) Otros libros.

Regla C (antecedente):

El lector no sabe cómo resolver un problema, aunque haya entendido el enunciado.

Regla C (consecuente):

Seguramente el lector haya asimilado los conceptos teóricos básicos, pero no haya captado la esencia del método o algoritmo que permite resolver el problema. Por tanto, debería regresar al apartado teórico del libro e intentar realizar una traza del algoritmo correspondiente con un problema inventado, más simple que el del enunciado.

Regla D (antecedente):

El lector no entiende un paso concreto de la solución de un problema.

Regla D (consecuente):

Buscar si en un problema previo aparece también este paso y está explicado con más detalle. En caso negativo, se propondrá al lector la consulta de un ejemplo ilustrativo de fácil comprensión.

Revisión del problema 1.3:**Aplicación 1 de la regla A:**

El lector no entiende el término “búsqueda en un espacio de estados”.

Acciones:

Leer la introducción correspondiente al contenido teórico del capítulo 1 (sección 1.2.1). Si esto no es suficiente, acudir a: [Mira *et al.*, 95], sección 3.2.2 o a [Rich & Knight, 1991], sección 2.1.

Aplicación 1 de la regla D:

El lector ha decidido que el método de búsqueda bidireccional en amplitud es el más adecuado para la resolución del problema, pero no es capaz de aplicarlo.

Acciones:

Realizar una traza del algoritmo correspondiente a búsqueda en amplitud (sección 1.2.2 del apartado teórico) para el siguiente árbol:

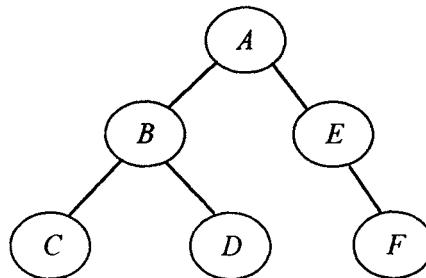


Figura 1-8

Aplicación 1 de la regla B:

El lector no entiende la figura 1-7.

Acciones:

Analizar con detenimiento las 8 reglas u operadores de transformación dados al principio de la solución del problema.

1.4 Contexto adicional

1.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

- Saber determinar en qué problemas se pueden aplicar técnicas de búsqueda en un espacio de estados.
- Adquirir una comprensión clara de los pasos que se siguen en las estrategias de búsqueda en amplitud, en profundidad, en retroceso y métodos derivados.
- Apreciar los cambios que hay que efectuar respecto a las estrategias anteriores a la hora de conseguir realizar una búsqueda general en grafos:
 - a) Introducción de las listas ABIERTA y CERRADA.
 - b) Existencia de posibles reorientaciones de enlaces en el subgrafo parcial.

1.4.2 Pistas de autoevaluación

- ¿Cómo quedaría modificada la solución del problema 1.1 si el límite de profundidad fuera 2 en lugar de 5?
- Hallar la solución que resulta al aplicar en el problema 1.3 una búsqueda bidireccional formada por una búsqueda en amplitud y otra en profundidad con límite de profundidad de valor 3.

2 BÚSQUEDA HEURÍSTICA

2.1 Contexto previo

2.1.1 ¿Por qué está aquí este capítulo?

Aunque en un principio se pensó que toda tarea de búsqueda podía ser completada por un computador sin más que realizar una exploración de todos los caminos que llevan a una solución y una posterior selección del mejor de tales caminos, más tarde se comprobó que, aunque esta suposición era cierta, no era eficaz debido a la *explosión combinatoria* que aparece en este tipo de problemas.

El conocimiento dependiente del dominio puede ayudar a dirigir el proceso de búsqueda de manera que sean exploradas en primer lugar aquellas trayectorias que parecen más prometedoras a la hora de conducir a un estado solución; la búsqueda inspirada en el razonamiento anterior se denomina heurística.

Los métodos heurísticos no garantizan hallar la solución óptima a un problema, pero permiten, de una manera más eficiente desde el punto de vista computacional, aproximarse a tal solución.

2.1.2 ¿Dónde hay conocimiento teórico del campo?

Algunas referencias bibliográficas donde el lector puede ampliar sus conocimientos sobre búsqueda heurística son:

Referencias básicas:

- [Mira *et al.*, 1995], capítulo 4.
- [Nilsson, 1987], capítulos 1, 2 y 3.
- [Rich & Knight, 1994], capítulos 3 y 12.

Referencias complementarias:

- [Borrajo *et al.*, 1993], capítulos 2 y 5.
- [Norvig, 1992], capítulo 18.
- [Winston, 1994], capítulos 4, 5 y 6.

2.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

- Todos los mencionados en el capítulo 1.

2.1.4 ¿Qué software de apoyo está disponible?

- **AI Search (AI C++ Search Class Library)**

Esta librería ofrece al programador un conjunto de algoritmos de búsqueda que pueden ser usados en la resolución de cualquier tipo de problemas. Entre los algoritmos implementados figuran los de búsqueda en profundidad en árboles y grafos, búsqueda “primero el mejor”, búsqueda bidireccional, búsqueda en árboles Y/O... Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/search/aisearch/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/search/aisearch/0.html)

2.2 Contenido teórico

2.2.1 Introducción

La principal diferencia de este tipo de búsqueda respecto a la que no empleaba información del dominio (búsqueda exhaustiva) es que ahora a cada nodo se le va a poder asociar un valor que dará idea de lo cerca que se encuentra de un nodo meta. Evidentemente, dicho valor no será más que una estimación de la distancia real a la meta. La incorporación de este tipo de información, normalmente a través de las llamadas *funciones de evaluación heurística*, permitirá guiar la búsqueda hacia aquellos caminos que se supone son más prometedores. A continuación se describen diferentes tipos de algoritmos basados en este tipo de búsqueda.

2.2.2 Una estrategia irrevocable: método del gradiente

Consiste, para cada expansión de un nodo, en seguir el recorrido a través de aquel nodo hijo cuyo valor de la función de evaluación heurística (f) sea el mayor, suponiendo que en los nodos meta f alcanza su valor máximo. No existe la posibilidad de retomar caminos abandonados, es decir, no se pueden reconsiderar las decisiones tomadas. Para que este algoritmo pueda ser aplicado, f debe cumplir la condición de que, para todo nodo visitado, debe existir al menos un sucesor suyo con un valor mayor de f —o menor si f alcanza su valor mínimo en los nodos meta—.

Procedimiento de *método del gradiente o búsqueda en escalada*:

1. Denominar m al estado inicial del problema planteado y asignar m a una variable llamada *elegido*.
2. Hasta que se encuentre una meta o se devuelva fallo, realizar las siguientes acciones:

2.1. Expandir m creando el conjunto de todos sus sucesores.

Para cada operador aplicable y cada forma de aplicación:

- (1) Aplicar el operador a m , generando un estado *nuevo*.
- (2) Si *nuevo* es meta, salir del proceso iterativo iniciado en el paso 2 y devolver dicho estado.
- (3) Si $f(nuevo)$ es mejor que $f(elegido)$, cambiar el valor de la variable *elegido*, asignándole el valor *nuevo*.

2.2. Si $f(elegido) \neq f(m)$, asignar $m = elegido$; en caso contrario, devolver fallo.

PROBLEMAS RELACIONADOS: 2.1, 2.3.

2.2.3 Estrategias de exploración de alternativas

2.2.3.1 Búsqueda “primero el mejor”

Este tipo de búsqueda no es más que un procedimiento general de búsqueda en grafos (ver capítulo de búsqueda sin información del dominio), donde se realiza en cada paso del algoritmo una ordenación de la lista ABIERTA. Dicha ordenación está basada en la asociación a cualquier nodo del grafo del valor que toma una función de evaluación heurística sobre el mismo. Esta función puede tener en cuenta o no el coste del mejor camino parcial encontrado en cada momento desde la raíz hasta el nodo considerado.

Normalmente se reserva el nombre de *búsqueda primero el mejor* para aquel procedimiento que no considera el coste mencionado, de manera que la función heurística utilizada consiste únicamente en una estimación del menor coste desde cada nodo a un nodo meta. En otros procedimientos, por ejemplo el A*, sí se tiene en cuenta dicho valor.

PROBLEMAS RELACIONADOS: 2.2.

2.2.3.2 Búsqueda en haz

La *búsqueda en haz* (*beam search* en inglés) es una variación del método *primero el mejor*. Esta estrategia pretende acelerar el proceso de búsqueda reduciendo en cada paso del algoritmo el número de nodos generados que van a poder ser expandidos posteriormente. La selección mencionada se puede llevar a cabo de diferentes formas:

- Permitiendo sólo que un número fijo de los nodos más prometedores generados en cada paso sean expandidos más adelante.
- Estableciendo un valor umbral f_0 de la función de evaluación heurística por debajo del cual (o por encima, según el criterio que se esté empleando) ningún nodo generado podrá ser expandido.

Por lo general, este método requiere menos recursos, por considerarse menos nodos en el espacio de búsqueda.

PROBLEMAS RELACIONADOS: 2.3.

2.2.3.3 Algoritmo A*

Como ya quedó reseñado previamente, es un caso particular de búsqueda *primero el mejor*. La función de evaluación heurística utilizada en este método es:

$$f(n) = g(n) + h(n)$$

donde

$g(n)$: coste real del mejor camino encontrado en un determinado momento desde la raíz hasta n .

$h(n)$: estimación del coste del camino óptimo desde n a una meta.

Procedimiento A*:

1. Crear una lista de nodos llamada ABIERTA y asignarle el nodo raíz, que representa el estado inicial del problema planteado. Llamar a este elemento r y asignarle $g(r) = 0$.
2. Crear una lista de nodos llamada CERRADA que inicialmente estará vacía.
3. Hasta que se encuentre una meta o se devuelva fallo, realizar las siguientes acciones:
 - 3.1. Si ABIERTA está vacía, terminar con fallo; en caso contrario, continuar.

3.2. Eliminar el nodo de ABIERTA que tenga un valor mínimo de f , llamar a este nodo m e introducirlo en la lista CERRADA.

3.3. Si m es meta, abandonar el proceso iterativo señalado en 3, devolviendo el camino de la solución que se obtiene recorriendo los punteros de sus antepasados (creados en 3.5.).

3.4. En caso contrario, expandir m generando todos sus sucesores.

3.5. Para cada sucesor n' de m :

(1) Crear un puntero de n' a m .

(2) Calcular $g(n') = g(m) + c(m,n')$, tal que $c(a,b)$: coste de pasar de a a b .

(3) Si n' está en ABIERTA, llamar n al nodo encontrado en dicha lista, añadirlo a los sucesores de m y realizar (3.1.).

(3.1.) Si $g(n') < g(n)$, entonces redirigir el puntero de n a m y cambiar el camino de menor coste encontrado a n desde la raíz; $g(n) = g(n')$ y $f(n) = g(n') + h(n)$.

(4) Si n' no cumple (3), comprobar si está en CERRADA; llamar n al nodo encontrado en dicha lista y realizar las siguientes acciones:

Si (3.1.) no se cumple, abandonar (4); en caso contrario, propagar el nuevo menor coste $g(n')$ (por lo que también se actualizarán los valores de f correspondientes) a sus descendientes (que llamaremos n_i , tal que $i = 1, 2, \dots$, siendo sus costes anteriores $g(n_i)$), realizando un *recorrido en profundidad* de éstos, empezando en n' y teniendo en cuenta las siguientes consideraciones:

(4.1.) Para los nodos descendientes n_i cuyo puntero (que debe apuntar siempre al mejor predecesor hasta ese momento) conduzca hacia el nodo n' , actualizar $g(n_i) = g(n')$ y $f(n_i) = g(n') + h(n_i)$ y seguir el recorrido hasta que se encuentre un n_i que no tenga más sucesores calculados o se llegue a un nodo en que ya ocurra que $g(n_i) = g(n'_i)$, en cuyo caso se habría producido un ciclo y también habría que terminar la propagación.

(4.2.) Para los nodos descendientes n_i , cuyo puntero no conduzca hacia el nodo n' , comprobar si $g(n_i') < g(n_i)$, en cuyo caso se debe actualizar el puntero para que conduzca hacia el nodo n' (mejor camino desde la raíz encontrado hasta ese momento) y se continúa el proceso de propagación.

(5) Si n' no está en ABIERTA o en CERRADA, calcular $h(n')$ y $f(n') = g(n') + h(n')$, introducirlo en ABIERTA y añadirlo a la lista de sucesores de m .

Propiedades del método A^{*}:

- a) Siempre acaba encontrando un camino solución, si éste existe (A^* es *completo*).
- b) Si h cumple que

$$\forall n \quad h(n) \leq h^*(n) \quad (h \text{ es } \textit{minorante} \text{ de } h^*)$$

donde $h^*(n)$ representa el coste del camino real óptimo que une n con un nodo meta, entonces A^* encuentra el camino solución óptimo o, dicho de otra forma, en las condiciones mencionadas para h , A^* es *admissible*.

- c) Se dice que una función heurística h_1 está *más informada* que h_2 si:
 - 1) h_1 y h_2 son minorantes de h^* .
 - 2) $\forall n \quad h_1(n) \geq h_2(n)$.

En las condiciones anteriores, todo nodo generado por A^* con h_1 , también lo será con h_2 (A^* será más eficiente con h_1).

- d) Si h es *monótona*:

$$\forall (n, n') \quad h(n) \leq \text{coste}(n, n') + h(n')$$

donde n' es un nodo sucesor de n y $\text{coste}(n, n')$ representa el coste del enlace entre n y n' , entonces el algoritmo A^* encuentra un camino óptimo para todos los nodos expandidos. Como consecuencia de ello, deja de ser necesario revisar la existencia de posibles reorientaciones de enlaces.

PROBLEMAS RELACIONADOS: 2.2, 2.4 a 2.7.

2.2.3.4 Exploración de grafos Y/O

Los nodos de un grafo Y/O vienen a representar subproblemas a resolver, originados a partir de un problema inicial que se correspondería con el nodo superior del grafo. Se pueden dar dos tipos de enlaces en estos grafos:

a) **Enlaces O:** indican las diferentes alternativas existentes para la solución de un problema o nodo del grafo.

b) **Enlaces Y:** conectan un nodo padre con los subproblemas que hay que solucionar para que ese nodo padre se considere resuelto. Estos enlaces se representan uniendo varios arcos mediante una línea curva tal como ocurre con los nodos n_0 , n_4 y n_5 de la figura 2-1.

Un *nodo hoja* será aquél que no posea sucesores en el grafo Y/O, debido a que no se conozca la forma de dividir el problema que representa en subproblemas. Si un nodo hoja se corresponde con un problema del que se conoce la solución, se le llamará *nodo terminal*. En la exploración de grafos Y/O se pretende obtener un grafo solución desde el nodo inicial hasta los nodos terminales. En la figura 2-1 aparece un grafo Y/O y más adelante en las figuras 2-2 y 2-3 se han trazado dos subgrafos solución del mismo.

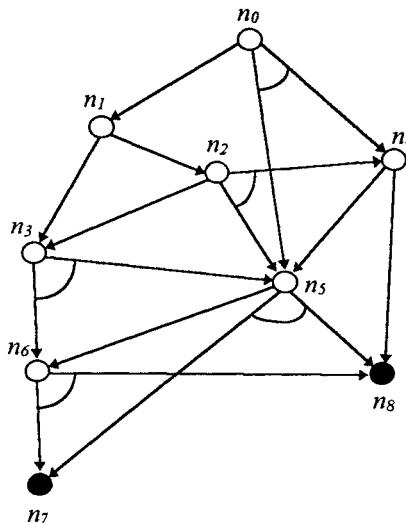


Figura 2-1 Ejemplo de grafo Y/O. Nodo inicial: n_0 . Nodos terminales: n_7 y n_8 .

Se considerará que el coste de un grafo solución es la suma del coste de los arcos de dicho grafo que parten del nodo inicial más la suma de los costes de los subgrafos soluciones cuyo nodo origen son los sucesores de ese nodo inicial. Para que esta definición recursiva sea satisfactoria, se supondrá que los grafos considerados no poseen ciclos. De cualquier forma, hay que tener en cuenta que en la definición anterior pueden contarse más de una vez los costes de algunos de los arcos. Los costes de los grafos solución de las figuras 2-2 y 2-3 son 8 y 7, respectivamente. Téngase en cuenta,

por ejemplo, que en el segundo grafo solución los arcos (n_5, n_7) y (n_5, n_8) se cuentan dos veces a la hora de hallar el coste total del grafo solución. De todos los posibles grafos solución, sería interesante poder hallar el de coste mínimo. Siendo n el nodo inicial, se denominará $h^*(n)$ al coste del grafo solución óptimo.

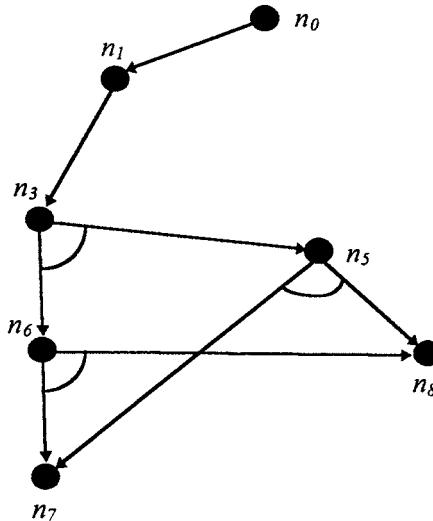


Figura 2-2 Subgrafo solución con los nodos resueltos en negro.

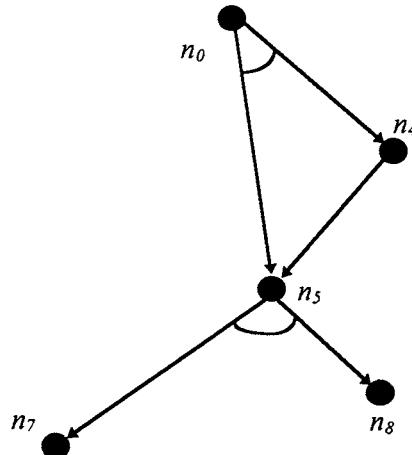


Figura 2-3 Otro subgrafo solución.

Se va a describir un procedimiento de exploración (YO^*) que hace uso de una función heurística $h(n)$, que es una estimación de $h^*(n)$. Se puede demostrar que si existe al menos un grafo solución y se cumple:

- $\forall \text{nodo } h(\text{nodo}) \leq h^*(\text{nodo})$
- h satisface la restricción de monotonía

entonces el algoritmo YO^* halla el grafo solución óptimo. Recuérdese que la restricción de monotonía consistía en imponer para toda conexión entre un *nodo* y sus sucesores ($\text{nodo}_1, \dots, \text{nodo}_k$) que

$$h(\text{nodo}) \leq c + h(\text{nodo}_1) + \dots + h(\text{nodo}_k)$$

donde c es el coste de la conexión. Si n es cualquier nodo terminal y $h(n) = 0$, la restricción de monotonía implica que $h(\text{nodo}) \leq h^*(\text{nodo})$ para cualquier *nodo*.

El procedimiento YO^* sería el siguiente:

1. Crear un grafo G que en principio está formado exclusivamente por el nodo raíz m . Estimar la distancia heurística a la meta de m mediante $h(m)$.
2. Realizar hasta que m se considere resuelto o hasta que su coste supere un cierto valor *coste-máximo* (incluye el caso en que no queden más sucesores por obtener; ver paso (2)):

(1) Seguir los enlaces marcados que señalan el mejor grafo parcial de la solución obtenido hasta el momento, hasta alcanzar los extremos de dicho grafo. Escoger alguno de dichos extremos, que llamaremos n .

(2) Expandir n generando todos sus sucesores s . Si no tiene ninguno, asignarle el valor *coste-máximo*.

(2.1.) $\forall s$: si s es un nodo terminal, marcarlo como resuelto y asignarle $h(s) = 0$; en caso contrario, asignarle $h(s)$.

(3) Crear un conjunto S sólo con n .

(4) Realizar hasta que S se vacíe:

(4.1.) Sacar de S algún nodo s' que no tenga descendientes en S .

(4.2.) Actualizar el coste de s' : calcular el coste de todos los enlaces que parten de s' y asignarle como $h(s')$ el menor de éstos, marcando dicho enlace (y borrando, si existiera, una marca de otro enlace,

previamente elegido, saliente de s'). Si todos los nodos del enlace marcado son terminales, marcar también s' como resuelto.

(4.3.) Si (4.2.) ha hecho que s' sea resuelto o si su $h(s')$ ha cambiado, añadir todos sus predecesores a S y continuar el proceso recursivo de actualización de valores de los nodos (que pueden llegar hasta la raíz del grafo G).

El algoritmo YO* repite dos operaciones: una hacia abajo en la que encuentra el grafo solución parcial y otra hacia arriba de revisión de costes, con establecimiento de nuevos subgrafos parciales menos costosos desde cada nodo, si procede.

PROBLEMAS RELACIONADOS: 2.8, 2.9.

2.2.4 Búsqueda con adversarios

Existen estrategias diseñadas para el tratamiento de problemas de búsqueda con dos adversarios cuyo objetivo es ganar una partida en la que realizan movimientos alternativos. En cada momento, cada jugador conoce tanto las posibilidades propias como las de su contrincante. Normalmente, cuando cualquiera de los adversarios tiene que efectuar una jugada, está interesado en saber cómo evolucionaría el juego a partir de cada posible elección que pudiera tomar. Debido a la imposibilidad práctica de tener en cuenta todos los caminos por los que podría pasar el juego hasta la resolución de la partida (problema de la *explosión combinatoria*), lo que normalmente se hace es desarrollar un árbol de búsqueda con una profundidad limitada a partir de la situación o estado actual de la partida. Cada nivel del árbol corresponde a un turno de movimiento para uno de los dos jugadores, suponiéndose que siempre elegirán aquel movimiento que más ventaja les dé en la partida. Para decidir qué movimiento es el mejor, cada jugador analiza sus posibles jugadas, las reacciones del contrario, sus reacciones a las de éste y así sucesivamente hasta que, llegado un nivel (límite de profundidad), se estima por medio de una función heurística cuál es la posibilidad de que dichos nodos sean ganadores, empate o perdedores. Esta función es una estimación estática del tipo de juego más probable que “cuelga” de cada nodo en dicho límite de profundidad, también llamado “frontera de exploración”.

Existen dos formas de etiquetar los nodos del árbol de búsqueda o, dicho de otro modo, de evaluarlos (ver figura 2-4):

a) **Etiquetado según MAX:** a cada *nodo* se le asocia un valor que representa lo prometedora que es dicha situación de la partida para el jugador **MAX**, independientemente de si dicho nodo se corresponde con una situación de la partida en la que le

toca “mover” a MAX o a MIN. Como consecuencia de ello, desde un nodo padre MAX se elegirá aquella jugada que conduzca al hijo con mayor valor asociado. Por otra parte, desde un nodo padre MIN se elegirá la jugada que lleve al hijo con menor valor asociado. Por tanto,

$$\text{MAX}(m) = \begin{cases} \bullet fev(m) \text{ si } m \text{ no tiene sucesores.} \\ \bullet \max_n\{\text{MAX}(n)\}, \text{ siendo } n \text{ sucesor de } m \text{ y } m \text{ un nodo MAX.} \\ \bullet \min_n\{\text{MAX}(n)\}, \text{ siendo } n \text{ sucesor de } m \text{ y } m \text{ un nodo MIN.} \end{cases}$$

b) **Método *MMvalor*:** este tipo de etiquetado es menos intuitivo que el anterior y se explica aquí únicamente debido a que los algoritmos que lo emplean suelen ser más eficientes. Se intenta que desaparezca la distinción entre nodos MAX y MIN. Asigna un valor a cada nodo que representa lo prometedora que es dicha situación de la partida para el jugador que posee el turno de movimiento en la misma. Se sigue el convenio de que si una situación de la partida es prometedora para el jugador MAX en un valor f_0 , lo será para MIN con un valor $-f_0$. En estas condiciones, independientemente de si nos encontramos en un nodo MAX o MIN, la labor que hay que realizar es siempre la misma: desde cualquier nodo padre se elegirá aquella jugada que conduzca al hijo con un valor asociado menor, al que llamaremos v_0 . Como desde el nodo padre se ha elegido acceder a una situación de la partida que es prometedora para su contrincante en un valor v_0 , a dicho nodo padre se le asociará un valor $-v_0$, ya que se ha supuesto en este método que el criterio para asignar valores a nodos depende del tipo de nodo considerado. Como consecuencia de todo lo anterior, en cada actualización del valor de un nodo hay que realizar primeramente una operación de cálculo de un mínimo y posteriormente un cambio de signo. En la literatura de búsqueda, sin embargo, normalmente aparece otro conjunto de operaciones equivalentes a la anterior: primero se cambian de signo los valores asociados a los nodos hijo y a continuación se elige el máximo de los mismos como valor asociado al nodo padre. Por tanto,

$$\text{MMvalor}(m) = \begin{cases} \bullet fev(m) \text{ si } m \text{ no tiene sucesores y es un nodo MAX.} \\ \bullet -fev(m) \text{ si } m \text{ no tiene sucesores y es un nodo MIN.} \\ \bullet \max_n\{-\text{MMvalor}(n)\}, \text{ siendo } n \text{ sucesor de } m. \end{cases}$$

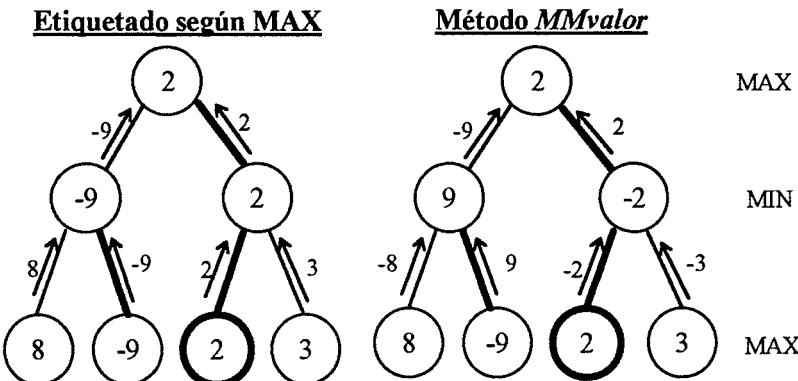


Figura 2-4 Evaluación de nodos en la estrategia MINIMAX.

2.2.4.1 Método MINIMAX

El método MINIMAX realiza una exploración exhaustiva del árbol de búsqueda, de manera que el jugador que debe mover en primer lugar (MAX) tiende a elegir en cada uno de sus movimientos aquel camino que le conduzca a un nodo frontera con el mayor valor posible de la función de evaluación heurística (si se aplica el método MAX). La estrategia del otro jugador (MIN) es la contraria, ya que tratará de contrarrestar las decisiones de MAX, eligiendo el peor valor para aquélla, es decir, el más pequeño. En el algoritmo que figura a continuación, se utiliza para el etiquetado de los nodos el método *MMvalor* visto con anterioridad.

Procedimiento MINIMAX:

MINIMAX (m, profundidad, jugador)

1. Si m no tiene sucesores o si se considera que m ha alcanzado el límite de profundidad en *profundidad*, devolver $fev(m)$ si m es un nodo MAX. En las mismas condiciones anteriores, devolver $-fev(m)$ si m es un nodo MIN (Recordamos aquí que $fev(m)$ representa lo prometedor que es el nodo m para MAX, independientemente de si m es un nodo MAX o MIN.).

2. Generar los sucesores de m :

- 2.1. Asignar a la variable *mejor* el valor mínimo que *fev* pueda tener (puede ser un valor de referencia previamente establecido).

$$mejor = \min_j \{fev(j) \ \forall j\}$$

2.2. Para cada sucesor n de m :

$$(1) M(n) = \text{MINIMAX}(n, \text{profundidad} + 1, C(\text{jugador}))$$

$C(\text{jugador})$ es una función que cambia de jugador.

$$(2) \text{mejor} = \max(-M(n), \text{mejor})$$

3. Una vez que se han analizado recursivamente todos los sucesores de un determinado nivel (indicado por la variable profundidad), se devuelve el valor *mejor*.

Observación: la primera llamada al procedimiento sería $\text{MINIMAX}(\text{nodo-inicial}, 0, \text{MAX})$.

El método MINIMAX no se aprovecha de situaciones especiales en las que no es necesario explorar ciertos caminos. Considérese el siguiente ejemplo:

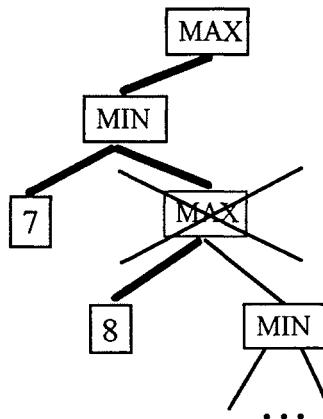


Figura 2-5

Desde el nodo MIN superior se puede llegar a un nodo con valor 7. El otro camino posible sólo es conveniente explorarlo mientras se sepa que se puede llegar por él a un nodo cuyo valor sea menor que 7. Esto último se convierte en imposible desde el momento en que desde el nodo MAX inferior se tiene acceso a un nodo de valor 8, lo que implica que desde este nodo MAX se accederá únicamente a nodos terminales con valores mayores o iguales que 8. Como consecuencia de todo ello, no es necesario explorar el subárbol derecho que cuelga del nodo tachado o cualquier otro subárbol que pudiera colgar del mismo.

PROBLEMAS RELACIONADOS: 2.10, 2.11.

2.2.4.2 Método de poda α - β

El *método de poda alfa-beta* se encarga de llevar una anotación para saber cuándo se puede realizar un “poda” o corte en el árbol de búsqueda. Lo que hace es pasar, en cada llamada recursiva a un nodo hijo, dos valores (α y β) de manera que α marque la cota inferior de los valores que se van a ir buscando en la parte del árbol que queda por explorar, siendo β la cota superior de esos mismos valores. Si en algún momento α llega a ser mayor o igual que β , como es lógico no tendrá sentido seguir con la búsqueda, realizándose una poda α si estamos en un nodo MAX o β si estamos en un nodo MIN.

El algoritmo para la estrategia de poda α - β pasaría a tener la forma siguiente:

J : nodo actual

J_k ($k = 1..b$): nodos hijos del nodo actual

$fev(J)$: valor de la función de evaluación heurística en el nodo J .

$\alpha\beta(J, \alpha, \beta)$:

Si J es terminal, devolver $fev(J)$.

En caso contrario, si J es un nodo MAX, ir a 1); si es MIN, ir a 2).

1) Realizar ($k = 1$):

1. $\alpha \leftarrow \max[\alpha, \alpha\beta(J_k, \alpha, \beta)]$

2. Si $\alpha \geq \beta$ devolver β ; si no, continuar.

3. Si $k = b$ (nº de hijos), devolver α ; si no: $k = k + 1$ y volver a 1.

2) Realizar ($k = 1$):

1'. $\beta \leftarrow \min[\beta, \alpha\beta(J_k, \alpha, \beta)]$

2'. Si $\alpha \geq \beta$ devolver α ; si no, continuar.

3'. Si $k = b$ devolver β ; si no, $k = k + 1$ y volver a 1'.

La primera llamada al procedimiento se hace con $\alpha = -\infty$ y $\beta = +\infty$. Otra alternativa es hacerla con $\alpha = fev_{MIN}$ y $\beta = fev_{MAX}$, donde fev_{MIN} y fev_{MAX} representan respectivamente los valores mínimo y máximo que puede alcanzar la función de evaluación heurística. Por último, reseñar que este algoritmo produce los mismos resultados que el procedimiento de poda α - β que aparece más adelante en el problema 2.13, siendo aquél más eficiente desde el punto de vista computacional, ya que utiliza menos variables.

PROBLEMAS RELACIONADOS: 2.10 a 2.14.

2.3 Problemas resueltos

- **PROBLEMA 2.1: (*)**

Dado el árbol que aparece en la figura 2-6 en el que el valor adjunto a cada nodo es el de la función heurística f , que señala el grado en que cada nodo satisface las condiciones del objetivo:

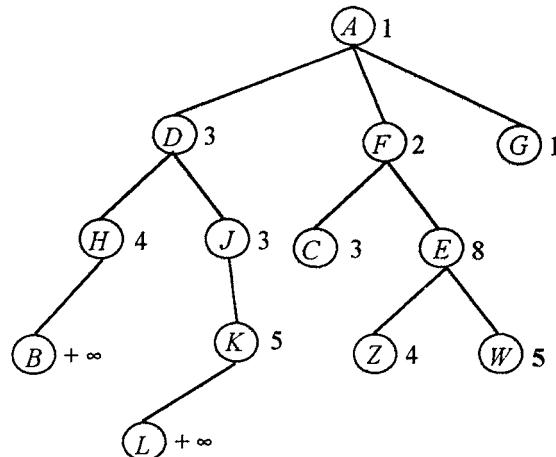


Figura 2-6

indique el orden en que se visitan los nodos, distinguiendo los que sólo se han generado de aquéllos que se han elegido en el proceso de búsqueda de la solución, para el procedimiento de *búsqueda en escalada*.

SOLUCIÓN :

¿Qué pretende este problema? En el presente problema se muestra la aplicación de una estrategia irrevocable, en concreto el método del gradiente, en la resolución de un problema de búsqueda heurística.

Se trata de realizar una búsqueda en profundidad sin posibilidad de retroceso, donde la elección del nodo a expandir depende de una función heurística; por tanto, este método no pertenece al conjunto de estrategias que no emplean información del dominio. La variable *elegido* almacenará aquel nodo por el que f aumente de forma más abrupta. Por tanto, se recorre el camino de máxima pendiente con respecto al valor de la función heurística utilizada. Teniendo en cuenta el algoritmo que aparece en el apartado teórico:

Punto 1) $m = A$, elegido = A

2.1) Se expande A

2.1.(1) nuevo = D

2.1.(3) Como $f(D) > f(A)$, elegido = D

2.1.(1) nuevo = F

2.1.(1) nuevo = G

Al final, elegido = D y $f(\text{elegido}) = 3$.

2.2) $m = D$

2.1) Se expande D

2.1.(1) nuevo = H

2.1.(3) Como $f(H) > f(D)$, elegido = H

2.1.(1) nuevo = J

Al final, elegido = H y $f(\text{elegido}) = 4$

2.2) $m = H$

2.1) Se expande H

2.1.(2) Fin, ya que B es un estado meta.

Por tanto, el camino para llegar a la meta ha sido:

A, D, H, B

Consideraciones finales :

El método más adecuado para este problema es el de *escalada*, debido a que en aquellos caminos que conducen a un nodo meta, f es una función creciente. Es decir, la infalibilidad del método aplicado se ajusta al problema planteado. Además, no hay garantía en otros algoritmos no informados de que el camino recorrido sea tan corto, ya que éste dependerá del orden arbitrario en que se hayan elegido los nodos.

- **PROBLEMA 2.2: (*)**

Recorra el grafo de la figura 2-7 según el procedimiento *primero el mejor*, suponiendo que los nodos están etiquetados según el valor de la función de evaluación heurística “distancia estimada a la meta” en cada uno de ellos. Para ello, opte por

indicar los valores de las listas ABIERTA y CERRADA a lo largo de cada ciclo del procedimiento, o bien represente los árboles de búsqueda considerados sucesivamente hasta alcanzar la solución.

Considere que el coste en la generación de cada sucesor en el grafo anterior es un coste uniforme de valor 1 y realice el mismo estudio siguiendo el procedimiento A*. Analice las diferencias, en caso de existir, entre las dos soluciones obtenidas, indicando cuál de los dos métodos encuentra una solución más eficiente. ¿Qué condiciones debería cumplir el procedimiento A* para que la solución obtenida tuviera la garantía de ser óptima?

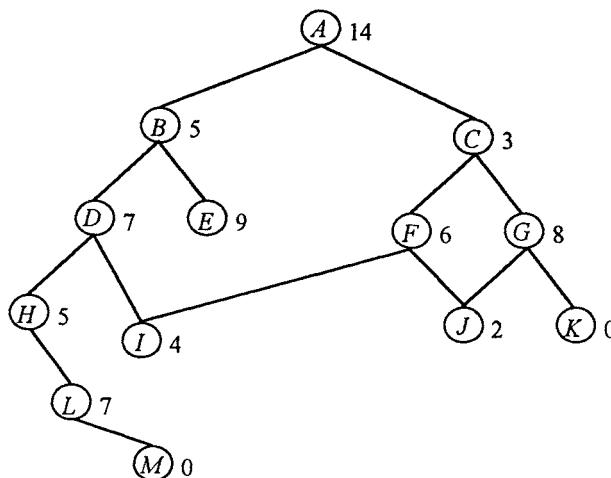


Figura 2-7

SOLUCIÓN:

¿Qué pretende este problema? En este ejercicio se realizará una comparación entre los algoritmos de búsqueda heurística *primero el mejor* y A*.

a) Procedimiento *primero el mejor*

Se van a dibujar los arcos del árbol que une el nodo raíz con cada nodo a través de los mejores caminos encontrados en el grafo (ver figura 2-8). También se indicará el orden en que los nodos son seleccionados para su expansión. En cada momento se seleccionará de ABIERTA aquel nodo cuyo valor de distancia estimada a la meta sea el menor. Se describirá cómo evolucionan ABIERTA y CERRADA tras cada paso del algoritmo. Según el algoritmo del apartado teórico:

	<u>ABIERTA</u>	<u>CERRADA</u>
Paso 1)	$A(14)$	\emptyset
Paso 2)	$C(3), B(5)$	A
Paso 3)	$B(5), F(6), G(8)$	A, C
Paso 4)	$F(6), D(7), G(8), E(9)$	A, C, B
Paso 5)	$J(2), I(4), D(7), G(8), E(9)$	A, C, B, F
Paso 6)	$I(4), D(7), G(8), E(9)$	A, C, B, F, J

En este ciclo del algoritmo se encuentra un descendiente (G) del nodo que se está expandiendo (J), que ya estaba en ABIERTA. A pesar de ello no se produce reorientación por ser el anterior camino desde G al nodo raíz de menor coste que el encontrado ahora.

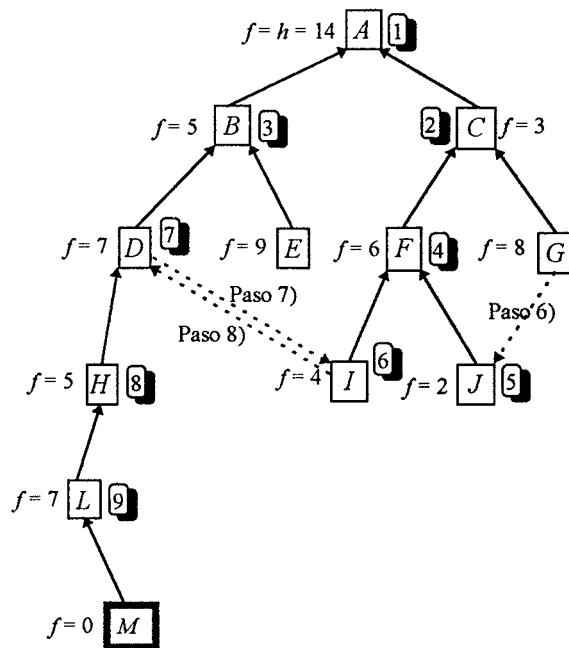


Figura 2-8 Método *primero el mejor*.

Paso 7)	$D(7), G(8), E(9)$	A, C, B, F, J, I
Tenemos un caso análogo al del ciclo anterior.		

Paso 8) $H(5), G(8), E(9)$ A, C, B, F, J, I, D

Al expandir D se genera I , que ya estaba en CERRADA. El nuevo camino encontrado hasta I no es menos costoso que el anterior; por lo tanto, no hay que redirigir arcos.

Paso 9) $L(7), G(8), E(9)$ A, C, B, F, J, I, D, H

Paso 10) $G(8), E(9)$ $A, C, B, F, J, I, D, H, L$

Con la generación de M se llega al final del algoritmo. La lista ABIERTA debe mantenerse ordenada a lo largo de todo el proceso, ya que el “siguiente nodo a expandir” es el primero de dicha lista.

El camino encontrado siguiendo los punteros, que siempre deben señalar al mejor antecesor de un nodo en el grafo, es:

$$A, B, D, H, L, M$$

b) Procedimiento A^*

Ahora habrá que sumar a la distancia estimada a la meta el coste del camino de menor coste encontrado hasta cada nodo, para formar una nueva función heurística f (ver figura 2-9). La ventaja, por tanto, es que este algoritmo tiene en cuenta el camino recorrido hasta un nodo dado, lo que evita ciertas deficiencias que aparecían en el caso del algoritmo *primero el mejor*. En la lista ABIERTA y entre paréntesis, al lado de cada nodo, figurará el valor que la función heurística toma en ese nodo. El proceso de finalización del algoritmo A^* va a ser algo diferente al de *primero el mejor*. En este último la búsqueda finaliza cuando se genera un nodo meta desde su nodo padre; en cambio, en el A^* , hasta que el nodo meta no es seleccionado para ser expandido, el algoritmo no acaba. Esto garantiza que el camino encontrado sea óptimo si se cumple la siguiente condición:

$$\forall n \quad h(n) \leq h^*(n) \quad (h^*(n): \text{coste real óptimo desde el nodo } n \text{ hasta una meta})$$

Como en el caso del algoritmo anterior, no se va a tener que producir ninguna reorientación, ya que el coste de los nuevos caminos encontrados hasta un nodo en ningún caso va a ser menor que el del camino que ya se había encontrado con anterioridad hasta ese nodo. A partir del algoritmo que aparece en el apartado teórico:

	<u>ABIERTA</u>	<u>CERRADA</u>
Paso 1)	$A(0+14)$	\emptyset
Paso 2)	$C(1+3), B(1+5)$	A
Paso 3)	$B(1+5), F(2+6), G(2+8)$	A, C
Paso 4)	$F(2+6), D(2+7), G(2+8), E(2+9)$	A, C, B

Paso 5) $J(5), I(7), D(9), G(10), E(11)$

A, C, B, F

Pasos 6) y 7) $D(9), G(10), E(11)$

A, C, B, F, J, I

No es necesario redirigir G desde J ni D desde I .

Paso 8) $H(3+5), G(10), E(11)$

A, C, B, F, J, I, D

En este caso, se ha encontrado un nuevo camino desde el nodo I hasta la raíz, que pasa por D ; pero este camino es del mismo coste que el existente con anterioridad; por tanto, no habrá reorientación del enlace que parte de I .

Paso 9) $G(10), E(11), L(4+7)$

A, C, B, F, J, I, D, H

Paso 10) $K(3), E(11), L(11)$

$A, C, B, F, J, I, D, H, G$

No hay reorientación desde J .

Paso 11) $E(11), L(11)$

$A, C, B, F, J, I, D, H, G, K$

El proceso finaliza cuando se expande el nodo K que es un nodo meta.

El camino encontrado es A, C, G, K que es mejor que el hallado mediante el método *primero el mejor*.

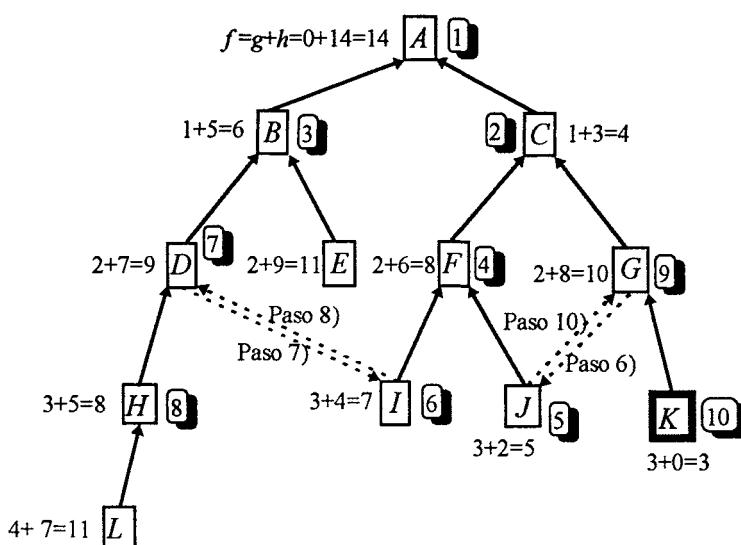


Figura 2-9 Método A^{*}.

Mientras que el algoritmo *primero el mejor* no garantiza que la solución encontrada sea óptima, A^{*} sí lo hace, siempre que la función h de distancia estimada a la meta cumpla: $\forall n \ h(n) \leq h^*(n)$, donde h^* representa distancias reales óptimas de cada nodo a

una meta. En este problema no se cumple la condición anterior. Por ejemplo, $h(G) = 8$ y en realidad estaba a una distancia de 1 de un nodo meta. Esto quiere decir que en este caso no se puede garantizar que la solución encontrada sea óptima.

• **PROBLEMA 2.3:** (adaptado de [Winston, 1992])

Dado el grafo de la figura 2-10 donde n_0 es el nodo inicial y n_7 el nodo meta, explorarlo mediante los siguientes métodos:

a) *Búsqueda en escalada*

b) *Búsqueda en haz*

Al lado de cada nodo aparece el valor de la función de evaluación heurística h , que es una estimación del coste del camino óptimo desde ese nodo a una meta.

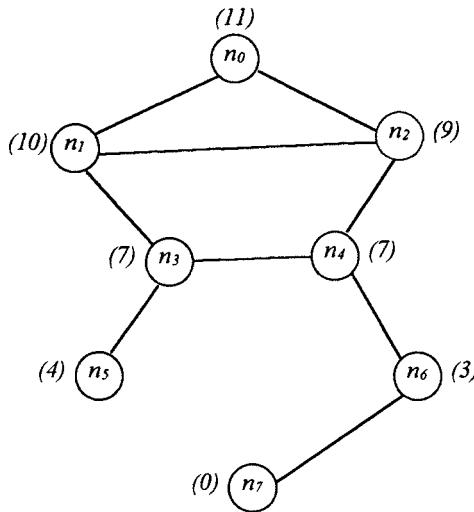


Figura 2-10

SOLUCIÓN:

a) *Búsqueda en escalada o método del gradiente*

Este tipo de búsqueda se puede aplicar sobre este grafo, ya que siempre el valor de la función h en alguno de los sucesores de cualquier nodo es menor que el del propio nodo. En el árbol que aparece en la figura 2-11, los números entre paréntesis que figuran al lado de cada nodo indican el valor de la función de evaluación heurística en los mismos. Evidentemente, para este caso particular se intentará tomar aquel camino

con un menor valor de esta función, de manera que en cada paso del algoritmo el coste estimado del camino que resta hasta la meta vaya decreciendo.

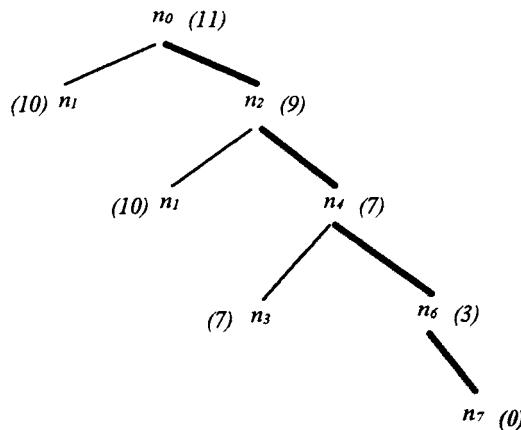


Figura 2-11

Por tanto, el camino encontrado es:

$$n_0, n_2, n_4, n_6, n_7.$$

- b) *Búsqueda en haz* (Se supondrá que en cada paso se expanden los 2 nodos más prometedores.)

En este tipo de búsqueda se van expandiendo en cada paso un número fijo de los nodos más prometedores que pueden ser expandidos en cada nivel. Por lo demás, no es más que una variedad de la búsqueda *primero el mejor*. Las cruces (x) indicarán los caminos que dejan de ser explorados. Los guiones (-) señalarán los caminos por los que no se puede seguir avanzando, por llegarse a un “callejón sin salida”.

Paso 1)

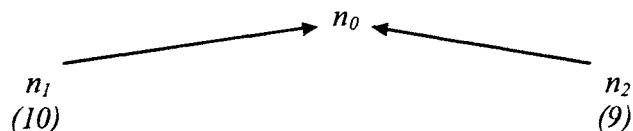


Figura 2-12

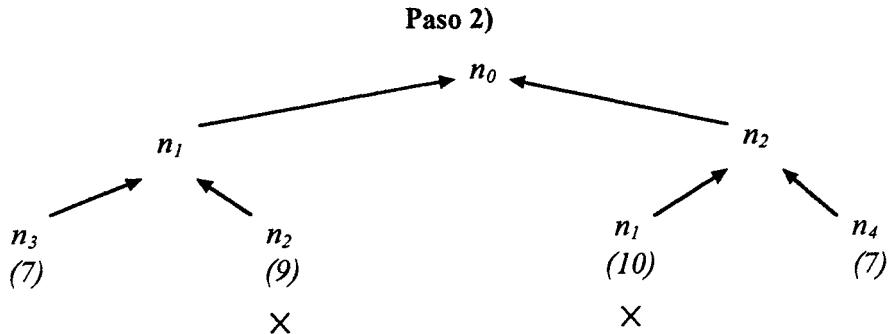


Figura 2-13

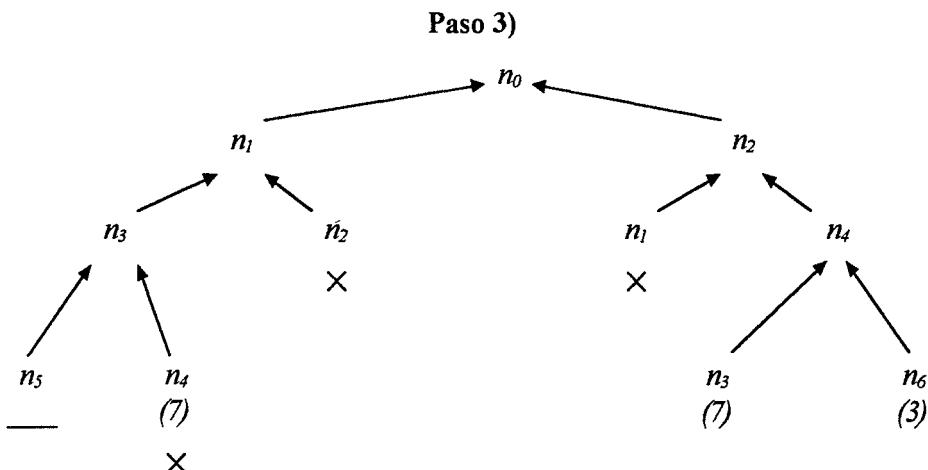


Figura 2-14

En este algoritmo, *al generarse un nodo*, se comprueban una serie de condiciones, como pueden ser si ese nodo es meta o un callejón sin salida. En este **Paso 3**), al generarse el nodo n_5 , se comprueba que no tiene sucesores, con lo cual es desecharlo, pasando ahora n_3 y n_6 a ser los dos caminos por donde sigue la exploración.

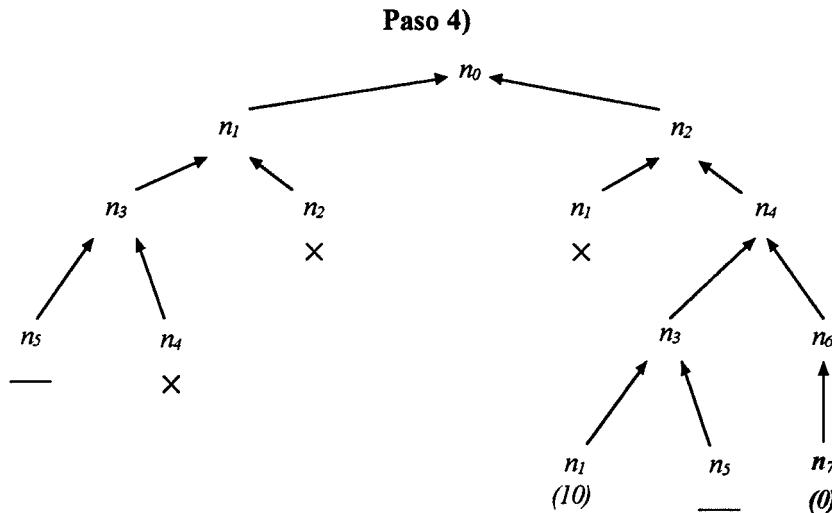


Figura 2-15

Por tanto, de nuevo el camino encontrado es: n_0, n_2, n_4, n_6, n_7 .

• **PROBLEMA 2.4:**

Aplicar el algoritmo A* al siguiente grafo. El nodo inicial es I y hay un solo nodo meta que en este caso es Z. A cada arco se le ha asociado su coste y a cada nodo la estimación de la menor distancia desde ese nodo al nodo meta.

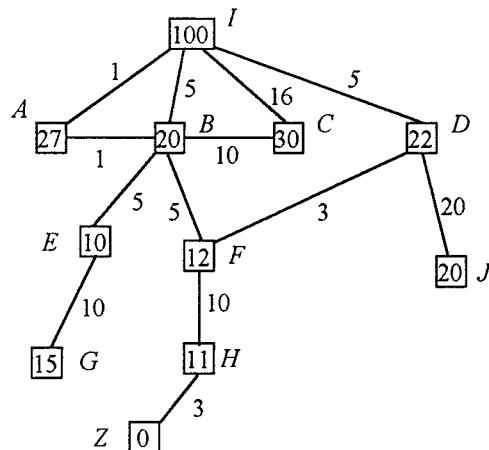


Figura 2-16

SOLUCIÓN :

¿Qué pretende este problema? Este ejercicio, junto con el 2.5, hace hincapié en los procesos de reorientación de enlaces que pueden ocurrir durante la ejecución del algoritmo A*.

Junto con la explicación de lo que se hace en cada etapa del algoritmo, figura cómo quedarían ABIERTA, CERRADA y el grafo correspondiente después de completarse esa etapa. Al lado de cada nodo en la lista ABIERTA figura el valor de la función heurística en ese nodo.

- Inicialmente:

ABIERTA: $I(0+100)$

CERRADA: \emptyset

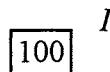


Figura 2-17

- La única posibilidad ahora es expandir I :

ABIERTA: $A(1+27=28), B(5+20=25), C(16+30=46), D(5+22=27)$

CERRADA: I

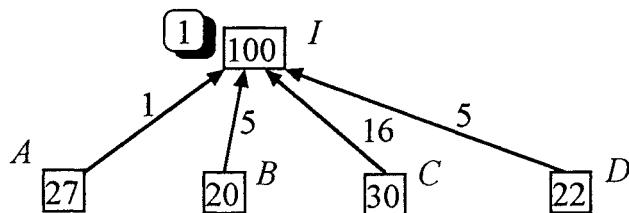


Figura 2-18

• Ahora el nodo más prometedor de ABIERTA es B , que es seleccionado para su expansión. C y A , dos de los sucesores de B , ya estaban en ABIERTA. El nuevo camino encontrado hasta C es menos costoso que el anterior, con lo que hay que efectuar una reorientación. Esto no ocurre con A , ya que el coste del nuevo camino encontrado desde este nodo hasta la raíz es 6, mientras que el coste inicial era 1.

ABIERTA: $A(28), C(45), D(27), E(20), F(22)$

CERRADA: I, B

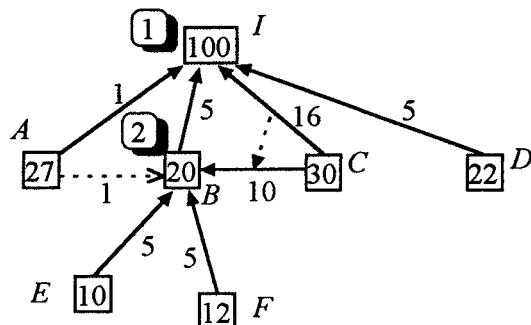


Figura 2-19

Como consecuencia de la reorientación hecha, se pasa de $C(46)$ a $C(45)$. Este cambio queda resaltado el nodo C en la lista ABIERTA.

- El siguiente nodo a expandir de entre los que están en ABIERTA es E :

ABIERTA: A(28), C(45), D(27), F(22), G(35)

CERRADA: I, B, E

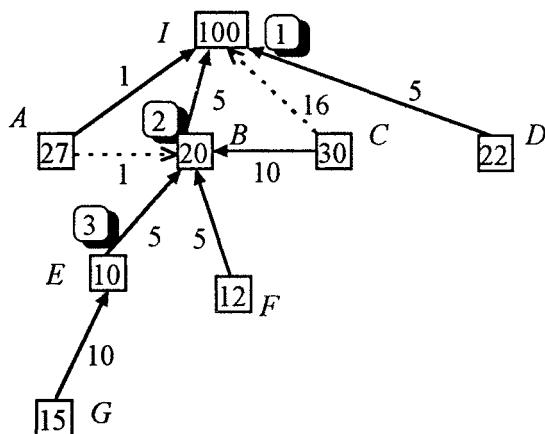


Figura 2-20

- Expandiendo ahora F :

ABIERTA: A(28), C(45), D(27), G(35), H(31)

CERRADA: I, B, E, F

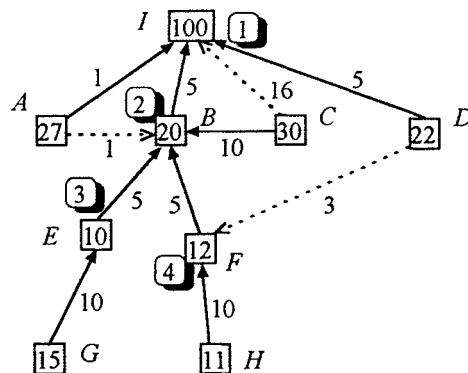


Figura 2-21

Obsérvese que no hay reorientación desde el nodo D .

- Al expandir en esta ocasión el nodo D , su sucesor F ya estaba en CERRADA. Como el nuevo camino encontrado hasta F es menos costoso que el anterior, hay que redirigir el enlace del árbol que parte de F . Como consecuencia de ello, habrá que estudiar si los caminos de sus descendientes también pueden verse afectados. La situación final es la siguiente :

ABIERTA: $A(28)$, $C(45)$, $G(35)$, $H(29)$, $J(45)$

CERRADA: I, B, E, F, D

H pasa de tener un coste estimado de 31 a 29 (ver figura 2-22) como consecuencia de que el camino de menos coste entre la raíz y H ya no pasa por el nodo B , sino por D .

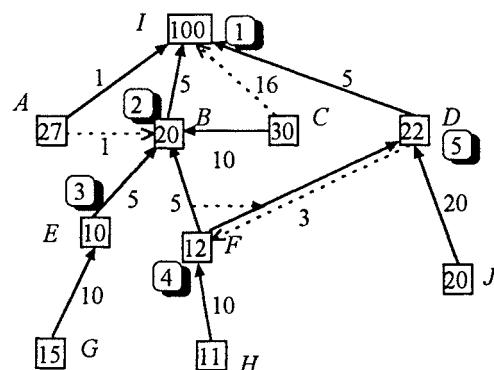


Figura 2-22

- El nodo a expandir ahora es *A*. Al generarse un nuevo camino desde el nodo raíz hasta *B*, estamos en el mismo caso de la última expansión. Ahora habrá que redirigir el enlace que partía de *B* y, además, habrá que estudiar lo que pasa con sus descendientes: *E*, *C* y *G* verán modificado el valor que la función de evaluación heurística toma sobre ellos; por otra parte hay que redirigir nuevamente el enlace que parte de *F* y recalcular, como consecuencia de ello, el valor que la función heurística asocia al nodo *H*.

ABIERTA: *C*(42), *G*(32), *H*(28), *J*(45)

CERRADA: *I*, *B*, *E*, *F*, *D*, *A*

Obsérvese en la figura 2-23 la segunda reorientación que hay que efectuar como consecuencia de la primera ($1^{\circ} \Rightarrow 2^{\circ}$).

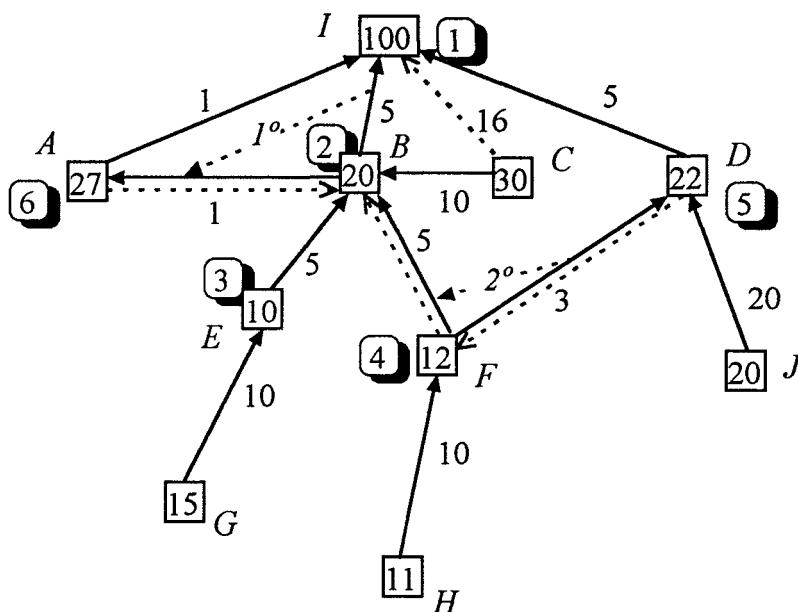


Figura 2-23

- Finalmente, el algoritmo expandirá el nodo *H* y posteriormente el nodo meta *Z*, con lo que termina todo el proceso.

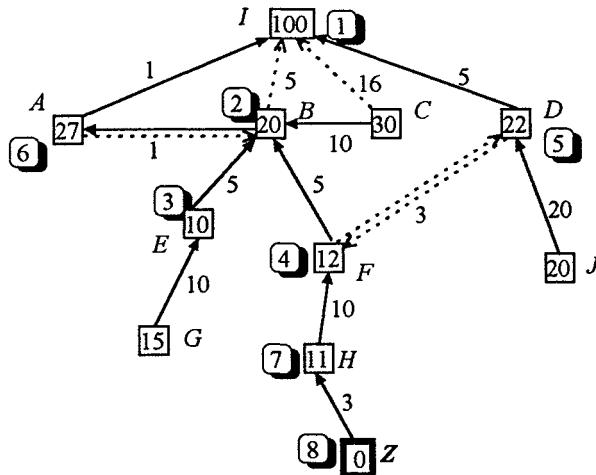


Figura 2-24

El camino solución será (ver figura 2-24):

$$I, A, B, F, H, Z$$

- **PROBLEMA 2.5: (*)**

Aplicar el algoritmo A* al siguiente grafo. A es el nodo inicial y Z el único nodo meta. Cada arco lleva asociado su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo a la meta. Dibujar en cada etapa del algoritmo el subgrafo parcial creado y la situación de las listas ABIERTA y CERRADA.

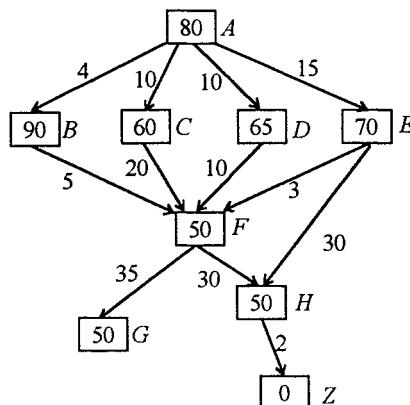


Figura 2-25

SOLUCIÓN:

Se explicará detalladamente cuáles son los procesos que se llevan a cabo en cada etapa del algoritmo. En los nodos que se encuentran en ABIERTA se indicará el valor de la función de evaluación heurística de los mismos.

- Inicialmente:

ABIERTA: $A(0+80=80)$

CERRADA: \emptyset

- A continuación la única posibilidad es expandir A :

ABIERTA: $B(4+90=94), C(10+60=70), D(10+65=75), E(15+70=85)$

CERRADA: A

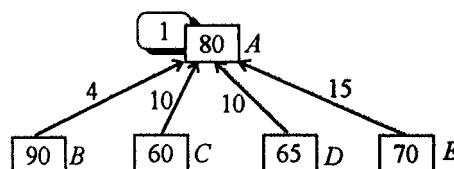


Figura 2-26

- El nodo más prometedor de ABIERTA es ahora C que es seleccionado para su expansión:

ABIERTA: $B(94), D(75), E(85), F(80)$

CERRADA: A, C

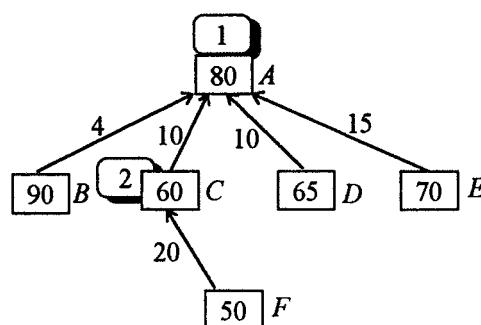


Figura 2-27

- El siguiente nodo a expandir es ahora **D**. Se va a crear un nuevo camino, menos costoso que el ya existente, entre **F** y **A**, con lo que será necesario redirigir el enlace que parte de **F**.

ABIERTA: **B(94), E(85), F(70)**

CERRADA: **A, C, D**

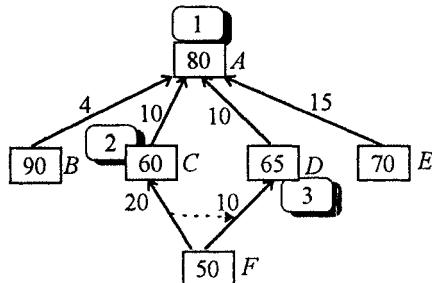


Figura 2-28

- Expandiendo **F**:

ABIERTA: **B(94), E(85), G(105), H(100)**

CERRADA: **A, C, D, F**

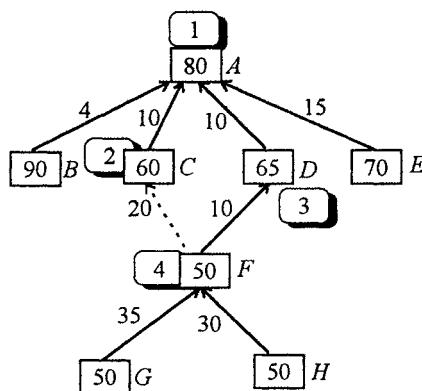


Figura 2-29

- La expansión de **E** va a provocar la reorientación del enlace que parte de **F**. Como **F** estaba en CERRADA, hay que estudiar lo que pasa con sus sucesores, en este caso **G** y **H**. La función de evaluación heurística para **G** va a pasar de 105 a 103 y para **H** de 100 a 98, aunque para este último nodo el camino creado desde **E** provoca una

nueva reorientación que deja el valor de su función de evaluación heurística finalmente en 95.

ABIERTA: B(94), G(103), H(95)

CERRADA: A, C, D, F, E

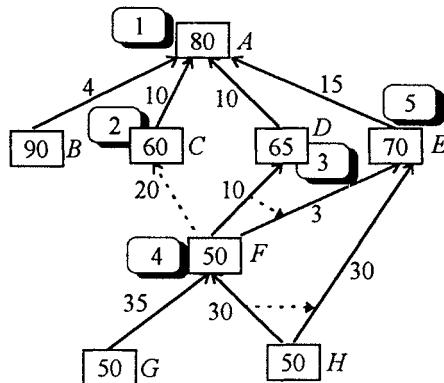


Figura 2-30

- Como consecuencia de la expansión de **B** se va a redirigir el nodo que parte de **F**, al ser el nuevo camino encontrado desde **A** hasta **F** de menos coste que el anterior. También habrá que deshacer la reorientación hecha en la etapa anterior desde el nodo **H**.

ABIERTA: G(94), H(89)

CERRADA: A, C, D, F, E, B

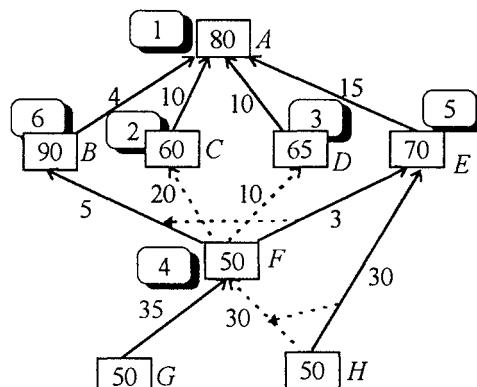


Figura 2-31

• Expandiendo H :

ABIERTA: $G(94), Z(41)$
 CERRADA: A, C, D, F, E, B, H

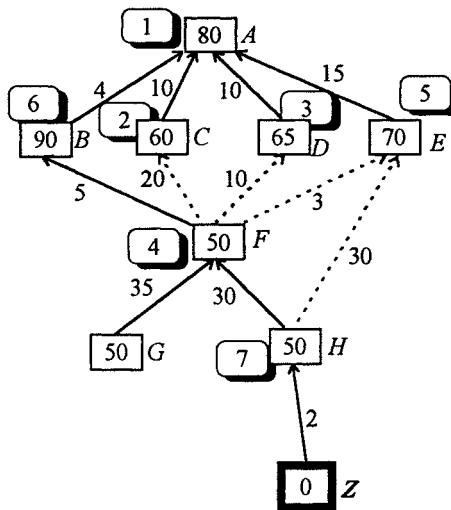


Figura 2-32

- Finalmente, con la expansión de Z se llega al estado meta, habiéndose encontrado el siguiente camino solución:

Z, H, F, B, A

que puede seguirse a través de los enlaces que se han ido trazando etapa a etapa del algoritmo.

• PROBLEMA 2.6: (*)

Dado el estado inicial $E_i = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 4 & \\ 8 & 7 & 5 \end{bmatrix}$ y el estado meta $E_m = \begin{bmatrix} 1 & 2 & 3 \\ 8 & & 4 \\ 7 & 6 & 5 \end{bmatrix}$ en el

problema del 8-Puzzle, describa los dos grafos de búsqueda de la solución resultantes de aplicar el procedimiento A*, suponiendo un coste uniforme 1 de aplicación de cada uno de los operadores disponibles y considerando las dos heurísticas siguientes:

- h_1 = suma de las *distancias de Manhattan* de todas las fichas que forman un estado concreto del tablero.

b) h_2 = número de casillas mal colocadas.

Analice las diferencias existentes entre ambas soluciones. Para realizar este ejercicio tenga en cuenta los siguientes datos:

reglas aplicables en el problema del 8-puzzle

- R₁: si $b \neq \text{Fila}_1 \Rightarrow$ Mover el blanco hacia arriba
- R₂: si $b \neq \text{Fila}_3 \Rightarrow$ Mover el blanco hacia abajo
- R₃: si $b \neq \text{Columna}_3 \Rightarrow$ Mover el blanco a la derecha
- R₄: si $b \neq \text{Columna}_1 \Rightarrow$ Mover el blanco a la izquierda
(b señala la posición del blanco en el tablero)

SOLUCIÓN:

¿Qué pretende este problema? La introducción del problema del 8-puzzle supone la aplicación del algoritmo A* a un dominio real, lo cual representa una novedad respecto a los problemas vistos hasta ahora. Por otra parte, se realizará una comparación de diferentes heurísticas para mostrar la importancia de que las mismas estén bien informadas a la hora de mejorar la eficiencia del proceso de búsqueda.

a) h_1 : suma de las distancias de Manhattan de todas las fichas que forman un estado concreto del tablero.

En cuanto a la heurística h_1 , no es más que la suma para todas las casillas mal colocadas de las distancias verticales y horizontales a las posiciones de esas mismas casillas en el estado meta. En la figura 2-34 aparece el grafo de búsqueda correspondiente a este apartado. El orden de expansión de los nodos se indica por medio de números encuadrados que figuran al lado del correspondiente nodo. La función h_1 es monótona, por lo que no habrá que tener en cuenta posibles reorientaciones (ver apartado teórico del algoritmo A*) y, además, cumple la condición mínima de optimicidad:

$$\forall n \ h_1(n) \leq h_1^*(n).$$

Por tanto, el camino solución es :

R₄, R₄, R₂, R₃, R₁

b) h_2 : número de casillas mal colocadas

En la figura 2-35 se puede ver el grafo de búsqueda resultante con esta función heurística. Al hacer la expansión etiquetada con el número 2, existían dos nodos que podían haber sido elegidos como el siguiente a expandir, ya que su valores de f_2 coincidían (2+3). La elección tomada finalmente es totalmente arbitraria y no responde a ningún criterio prefijado. El camino solución obtenido es idéntico al del apartado

anterior. Con esta segunda heurística se expanden más nodos debido a que está menos informada; es decir, se cumple que :

- $\forall n \ h_1(n) \leq h^*(n)$ y $h_2(n) \leq h^*(n)$
- $\forall n \ h_1(n) \geq h_2(n)$

Además, siempre se cumplirá que con h_2 se expanden, por lo menos, los mismos nodos que con h_1 .

• **PROBLEMA 2.7: (*)**

Dado el siguiente mapa de carreteras en el que los caminos entre cada dos ciudades están etiquetados con sus distancias en kilómetros:

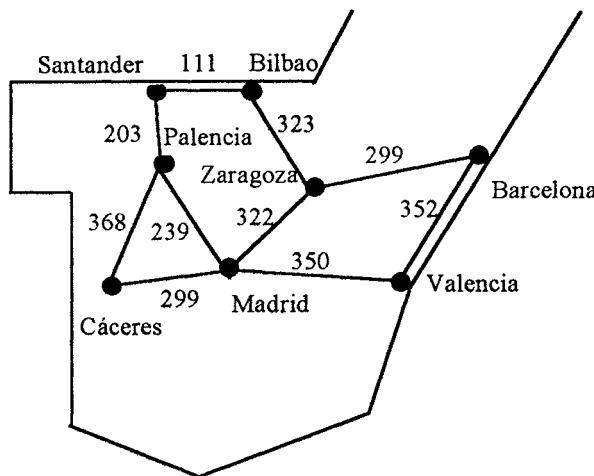
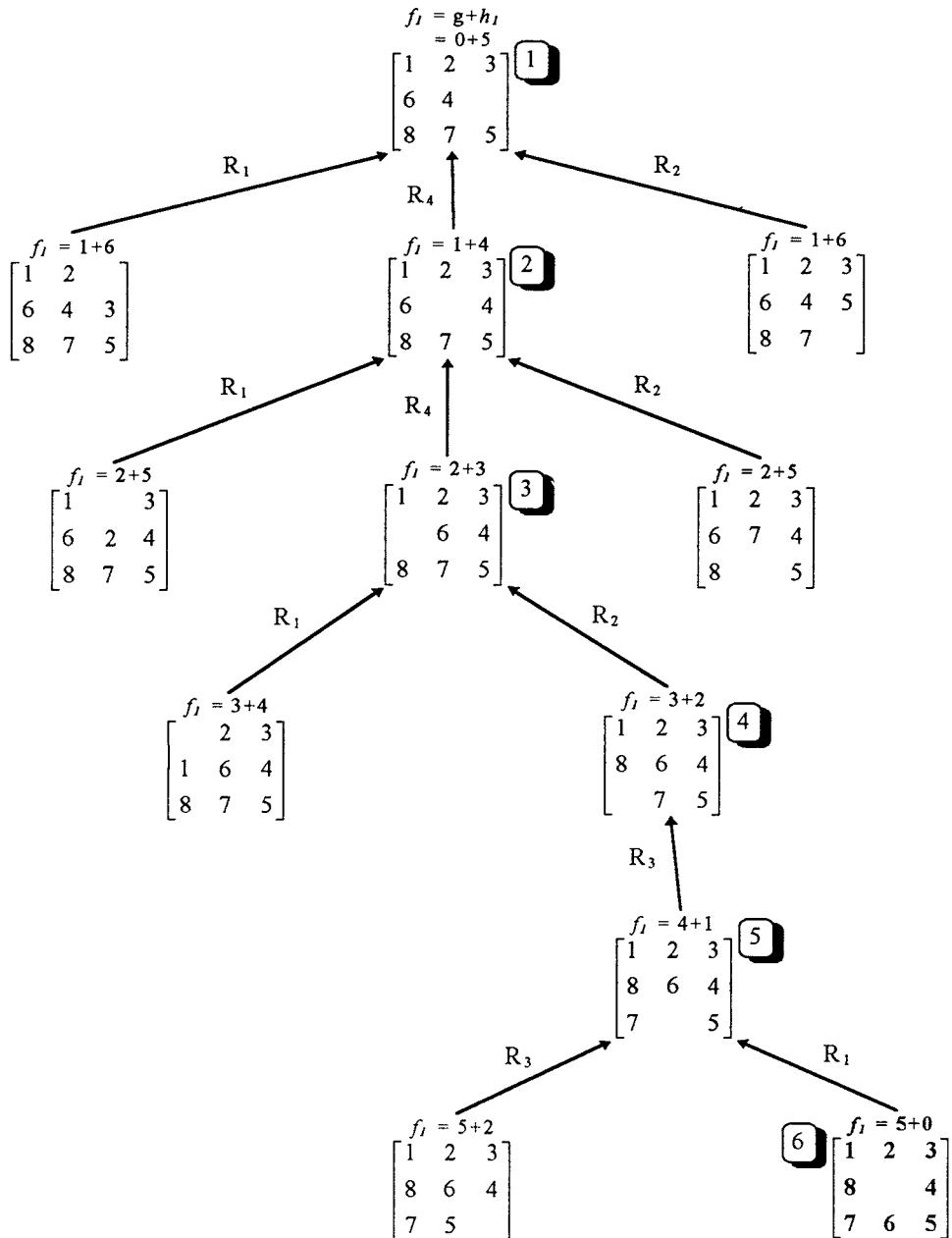


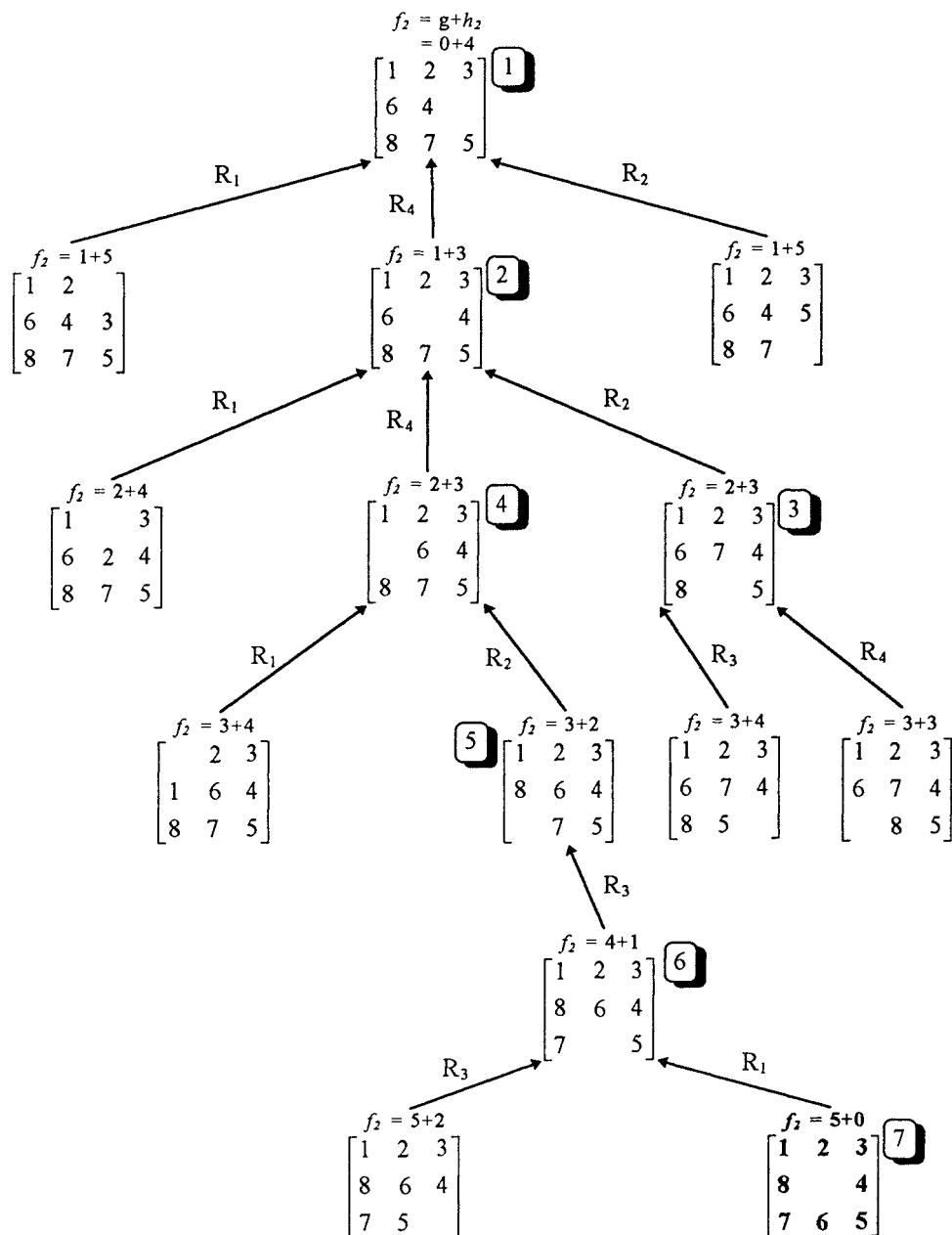
Figura 2-33

1) Describa el grafo correspondiente a la búsqueda del camino más corto entre Palencia y Barcelona. Para ello señale cuál es el algoritmo adecuado y aplíquelo. Indique y describa la utilización del método que le permite encontrar el camino que recorra el menor número de ciudades para trasladarse entre las capitales anteriormente señaladas.

2) Téngase en cuenta el siguiente cuadro de distancias aéreas estimadas desde cada ciudad a Barcelona:

	Bilbao	Cáceres	Madrid	Palencia	Santander	Valencia	Zaragoza
Barcelona	502	850	550	580	605	303	275

Figura 2-34 Grafo de búsqueda con h_1 .

Figura 2-35 Grafo de búsqueda con h_2 .

Utilizando como función heurística la distancia aérea estimada a la meta, que llamaremos “ d ”, describa el grafo de búsqueda resultante de la aplicación del método heurístico que garantice encontrar la solución óptima. Justifique su elección frente a otros algoritmos de “búsqueda informada”.

SOLUCIÓN:

¿Qué pretende este problema? Dependiendo del tipo de problema que se esté intentando resolver y de la información disponible sobre el mismo, un algoritmo de búsqueda será más adecuado que otro.

1a) El algoritmo apropiado para encontrar el camino más corto entre Palencia y Barcelona es el A*, siempre que la función heurística que estima la distancia a la meta desde cada nodo nunca supere la distancia real existente. En estas condiciones se garantiza que el algoritmo A* encontrará la solución óptima al problema. Una forma de asegurar la condición mencionada es suponer que la función de estimación vale 0 (este procedimiento se denomina *de coste uniforme* y es una variación de la búsqueda en amplitud, pero en lugar de obtener soluciones de menor longitud genera soluciones de menor coste).

El grafo de búsqueda con líneas a trazos representando caminos alternativos que no son los mejores en cuanto a coste, aparece más adelante en la figura 2-36. Los números al lado de cada nodo indican el orden de expansión de los mismos. En ningún momento hay que redirigir ningún enlace, ya que en todos los casos los nuevos caminos que se abren hasta la raíz son de mayor coste que los anteriores.

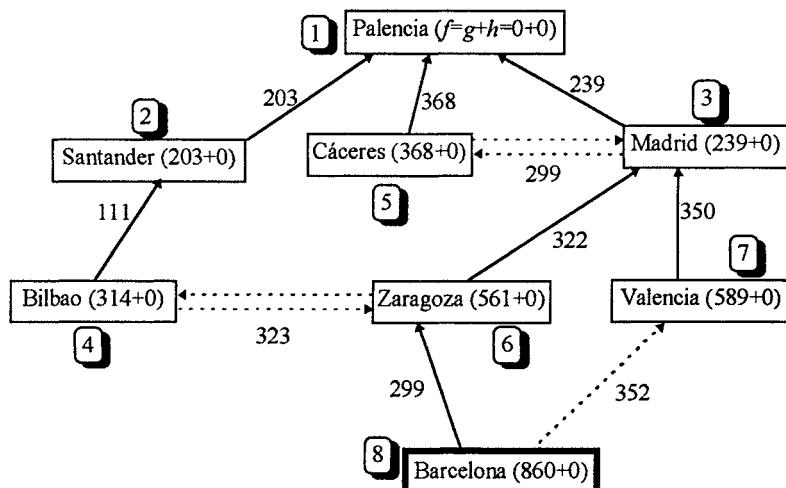


Figura 2-36

1b) El camino que recorra el menor número de ciudades se puede encontrar aplicando el algoritmo de búsqueda en amplitud.

2) Se puede recurrir al algoritmo A* de manera que, aprovechando la nueva información sobre la distancia aérea estimada a la meta, se obtendrá el siguiente grafo de búsqueda:

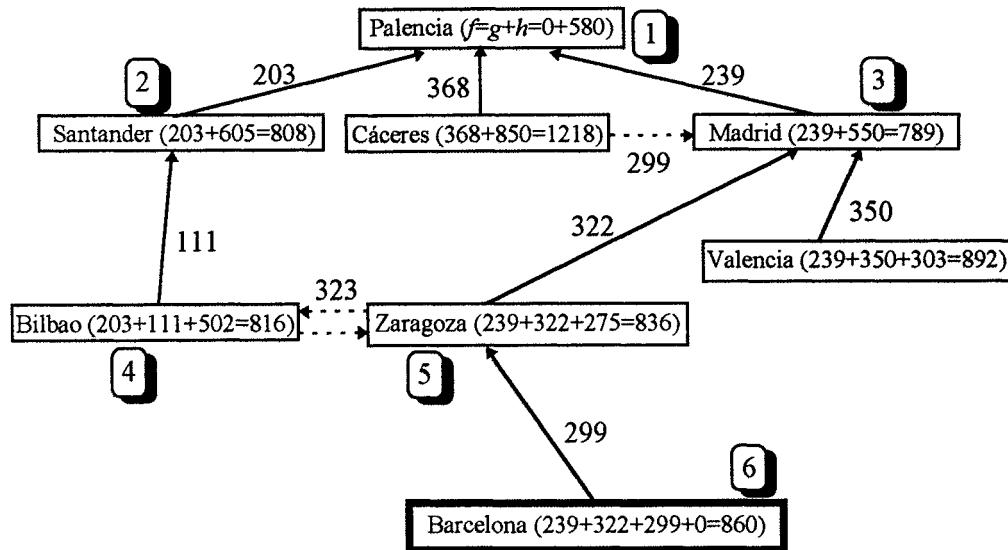


Figura 2-37

Se ha elegido A* ya que se cumple que $h(n) \leq h^*(n)$, pues la distancia aérea es siempre una estimación optimista de la distancia real.

• PROBLEMA 2.8: (adaptado de [Nilsson, 1980])

Considérese de nuevo el grafo Y/O del apartado teórico. Describir paso a paso el desarrollo de la exploración de dicho grafo mediante el algoritmo YO*. Para ello supóngase que el coste de cada arco es 1 y que se tienen los siguientes valores para la función heurística h de estimación del coste del grafo solución óptima desde cada nodo:

$$h(n_0) = 0$$

$$h(n_3) = 4$$

$$h(n_6) = 2$$

$$h(n_1) = 2$$

$$h(n_4) = 1$$

$$h(n_7) = 0$$

$$h(n_2) = 4$$

$$h(n_5) = 1$$

$$h(n_8) = 0$$

Recuérdese que n_0 es el nodo inicial y los nodos terminales son n_7 y n_8 .

SOLUCIÓN:

¿Qué pretende este problema? El presente problema y el 2.9 muestran detalladamente los pasos que se siguen en el algoritmo YO* para la obtención de un grafo solución.

El grafo que se pretende explorar con los valores iniciales de h al lado de cada nodo es:

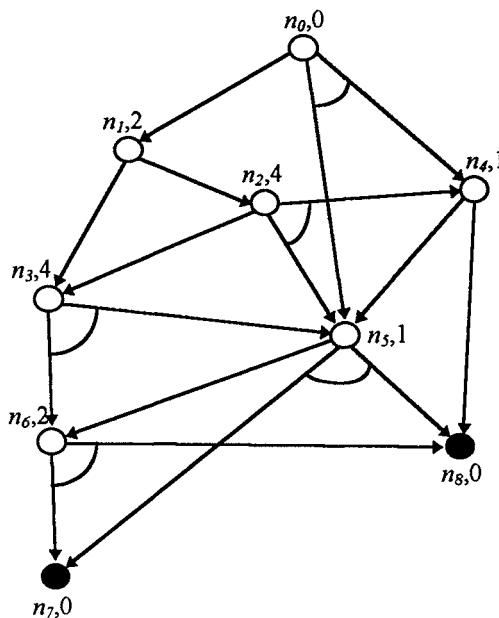


Figura 2-38 Nodo inicial: n_0 . Nodos terminales: n_7 y n_8 .

- Inicialmente:



Figura 2-39

- Primer ciclo:

La única posibilidad es expandir n_0 :

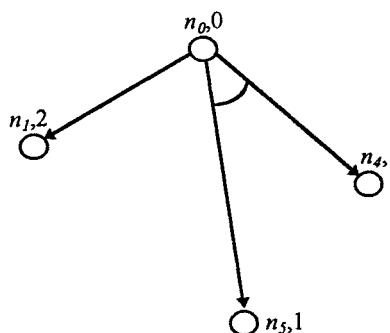


Figura 2-40

Ahora habrá que determinar cómo afecta la expansión realizada al nodo desde el que se realizó la misma y a todos sus antepasados. Para ello se hace uso del conjunto S , que inicialmente estará formado por:

$$S = \{n_0\}$$

Sacando n_0 de S :

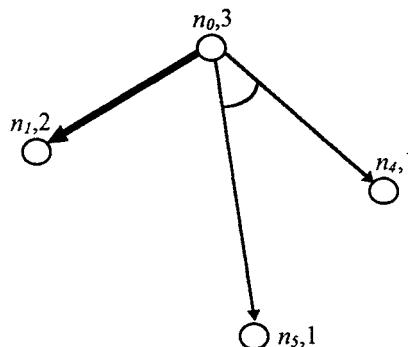


Figura 2-41

con lo que se determina el subárbol de menor coste que parte de n_0 , cuyo nuevo coste será 3. Como n_0 no tiene predecesores para ser introducidos en S , el ciclo acaba.

- Segundo ciclo:

Siguiendo los enlaces del subgrafo solución parcial creado hasta ahora, la única posibilidad es expandir n_1 :

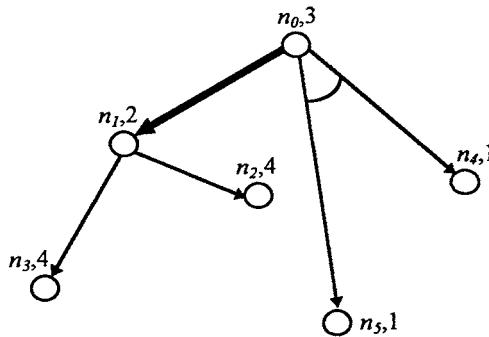


Figura 2-42

$$S = \{n_1\}$$

Al sacar n_1 de S se puede elegir el enlace que va a n_3 o el que va a n_2 , ya que por los dos el coste es el mismo. Supóngase que se elige el primero de ellos. Como el coste desde n_1 ha cambiado (de 2 a 5), habrá que introducir n_0 en S .

$$S = \{n_0\}$$

Al sacar n_0 de S y recalcular su coste, el subgrafo formado por el enlace Y se hace más prometedor (coste 4 frente a 6 por el otro camino de la izquierda):

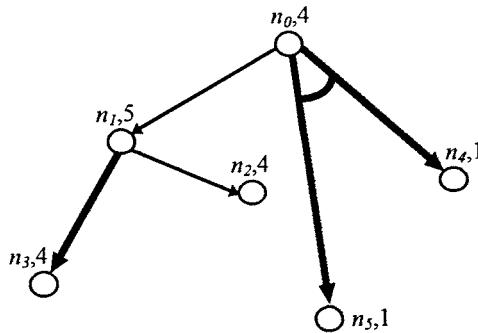


Figura 2-43

- Tercer ciclo:

Siguiendo nuevamente el subgrafo parcial solución encontrado desde n_0 , se llega a los nodos hoja $\{n_5, n_4\}$. Si se expande n_5 :

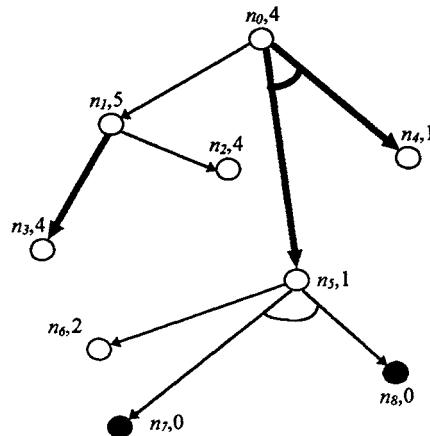


Figura 2-44

Se marcan con un círculo negro los nodos resueltos (en este caso $n_5,1$ y $n_5,2$). Se tendrá:

$$S = \{n_5\}$$

Al sacar n_5 de S quedará marcado como resuelto (se elige el camino formado por el enlace Y), por lo que n_0 es introducido en S . Finalmente, al sacar n_0 de S se recalcula su coste.

La situación queda del siguiente modo:

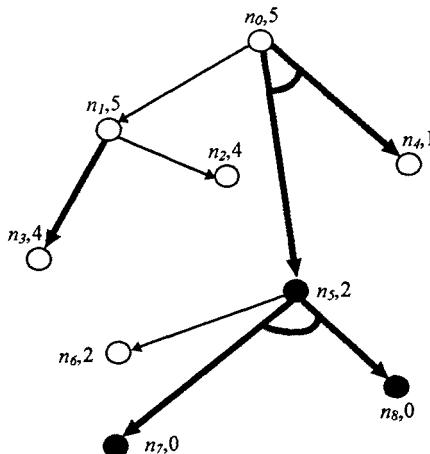


Figura 2-45

• Cuarto y último ciclo:

La única opción ahora es expandir n_4 :

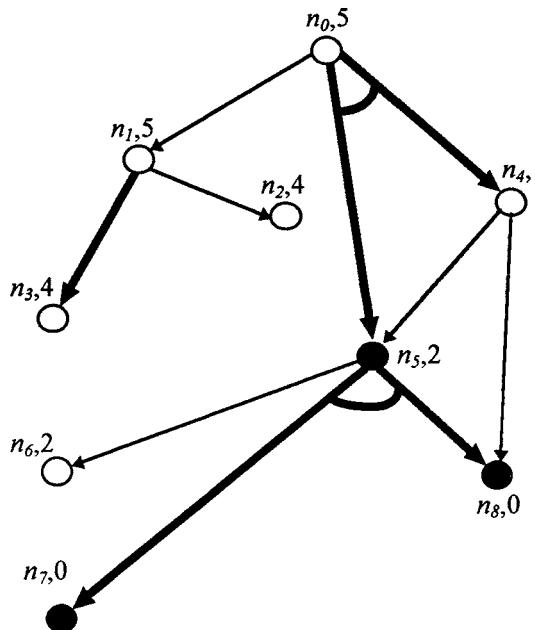


Figura 2-46

Se tiene:

$$S = \{n_4\}$$

Al sacar n_4 de S se elige el enlace a n_8 para la expansión del subgrafo parcial solución, pues $0(n_8) < 2(n_5)$. Como n_8 es un nodo terminal, n_4 también lo será y su predecesor n_0 tendrá que ser introducido en S .

$$S = \{n_0\}$$

Al sacar n_0 de S y recalcular los costes del subgrafo parcial solución que parte desde este nodo, la situación final queda con n_0 resuelto y un grafo solución hallado, tal como se indica en la figura 2-47 (recuérdese como en el apartado teórico aparecían otros dos subgrafos solución: figuras 2.2 y 2.3 cuyo coste era superior a 5, que es el coste del subgrafo solución hallado en este problema).

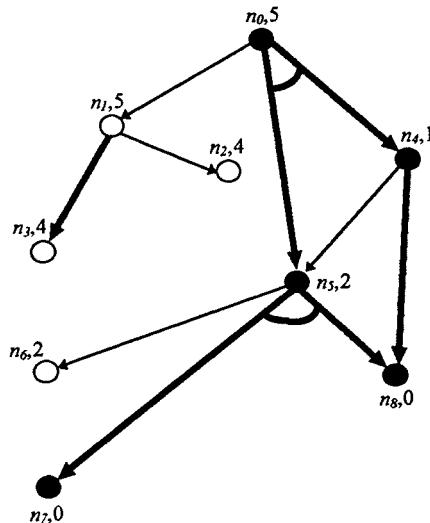


Figura 2-47

• PROBLEMA 2.9:

Dado el grafo Y/O de la figura 2-48, explorarlo mediante el algoritmo YO*.

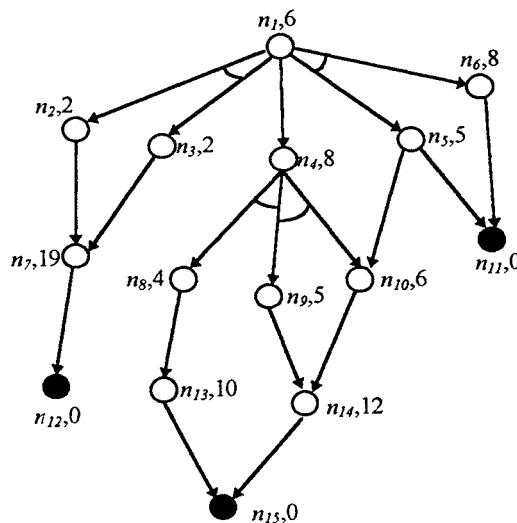


Figura 2-48

n_1 es el nodo inicial y n_{11} , n_{12} y n_{15} los nodos terminales. Junto a cada nodo figura el valor de la función heurística h en el mismo. Si hubiera habido en el grafo algún nodo sin sucesores que no fuera terminal, se le habría asignado un valor de h de magnitud *coste-máximo* (ver algoritmo YO* del apartado teórico).

SOLUCIÓN:

- Inicialmente:



Figura 2-49

A continuación se describirá el comportamiento del algoritmo a lo largo de cada ciclo. En cada uno de ellos, primero se expande un nodo hoja cualquiera del grafo solución parcial que cuelga de n_1 , y luego se realiza, si procede, una actualización hacia arriba de los costes de los grafos solución parciales que cuelgan de los nodos antepasados del nodo expandido.

- Ciclo 1:

- Se expande n_1 .
- La evolución del conjunto S es la siguiente:

- 1) $S = \{n_1\}$

- 2) n_1 es sacado de S . Su nuevo coste es $1+1+2+2=6$ (igual que el anterior).

- 3) $S = \{\}$

- La situación final es la siguiente:

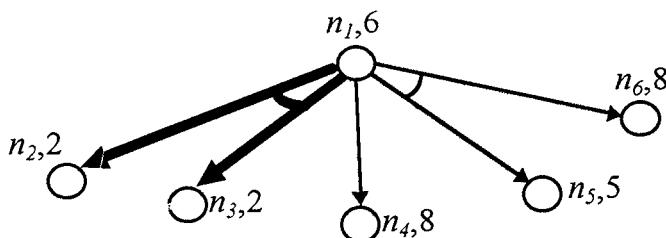


Figura 2-50

• Ciclo 2:

- Se expande n_2 , aunque también podría haberse expandido n_3 (ver figura 2-51).

- Evolución del conjunto S :

$$1) S = \{n_2\}$$

2) Se saca n_2 de S . Su nuevo coste es: $1+19=20$. Al haber cambiado este coste respecto al antiguo, se introducen los predecesores de n_2 en S .

$$3) S = \{n_1\}$$

4) n_1 es sacado de S . Su nuevo coste es 9 por el enlace central.

$$5) S = \{\}$$

- Finalmente,

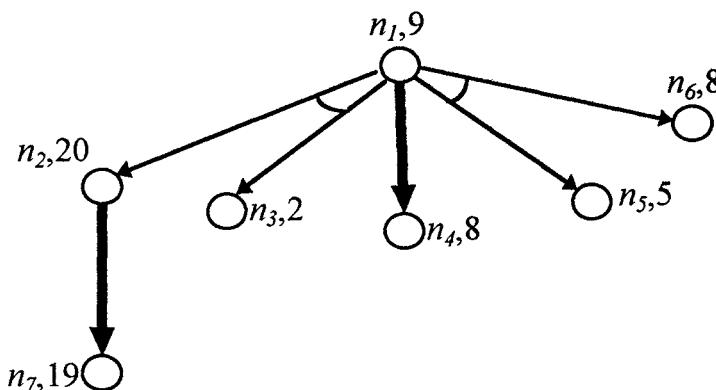


Figura 2-51

• Ciclo 3:

- Se expande n_4 .

- Evolución de S :

$$1) S = \{n_4\}$$

2) Nuevo coste de n_4 : $1+1+4+5=11$

$$3) S = \{n_1\}$$

4) Nuevo coste de n_1 : $1+11=12$

5) $S = \{\}$

- Al final de este ciclo:

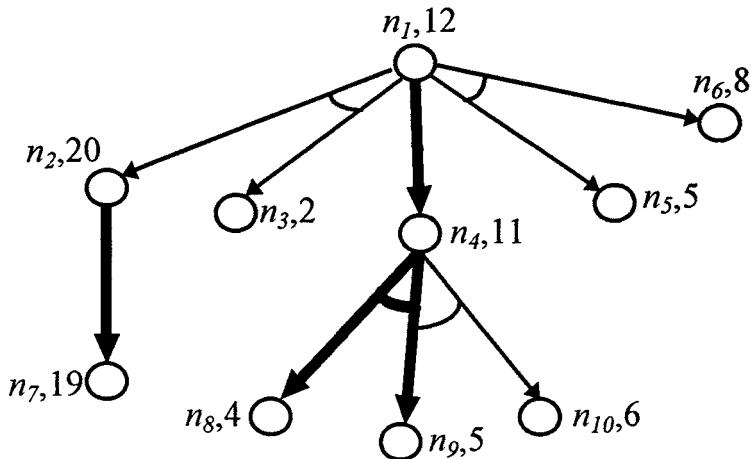


Figura 2-52

• Ciclo 4:

- Se expande n_8 , aunque también se podría haber expandido n_9 .

- Evolución de S :

1) $S = \{n_8\}$

2) Nuevo coste de n_8 : $1+10=11$

3) $S = \{n_4\}$

4) Nuevo coste de n_4 : $1+1+5+6=13$ por el enlace Y de la derecha.

5) $S = \{n_1\}$

6) Nuevo coste de n_1 : $1+13=14$

7) $S = \{\}$

- Finalmente,

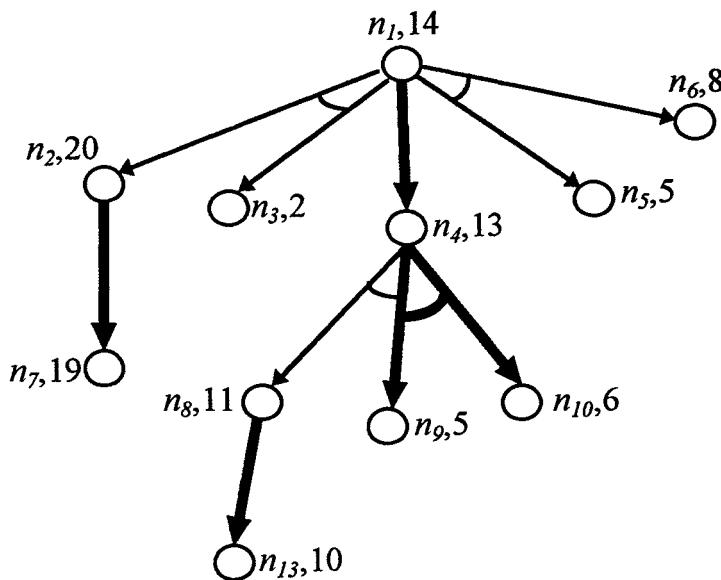


Figura 2-53

• Ciclo 5:

- Se expande n_{10} , aunque también se podría haber expandido n_9 .
 - Evolución de S :
 - 1) $S = \{n_{10}\}$
 - 2) Nuevo coste de n_{10} : $1+12=13$
 - 3) $S = \{n_4\}$
 - 4) Nuevo coste de n_4 : $1+1+11+5=18$ por el enlace Y de la izquierda.
 - 5) $S = \{n_1\}$
 - 6) Nuevo coste de n_1 : $1+1+5+8=15$ por el enlace Y de la derecha.
 - 7) $S = \{\}$
- Finalmente,

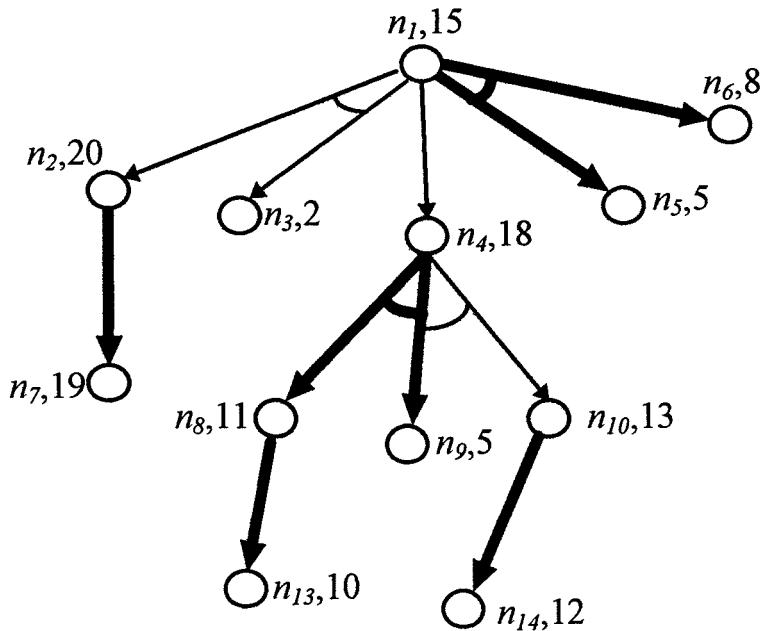


Figura 2-54

• Ciclo 6:

- Se expande n_5 , aunque n_6 podría haber sido el nodo elegido para su expansión.

- Evolución de S a lo largo del ciclo:

- 1) $S = \{n_5\}$

2) Nuevo coste de n_5 : $0+1=1$, quedando resuelto este nodo.

- 3) $S = \{n_1\}$

4) Nuevo coste de n_1 : $1+1+1+8=11$.

- 5) $S = \{\}$

- Finalmente,

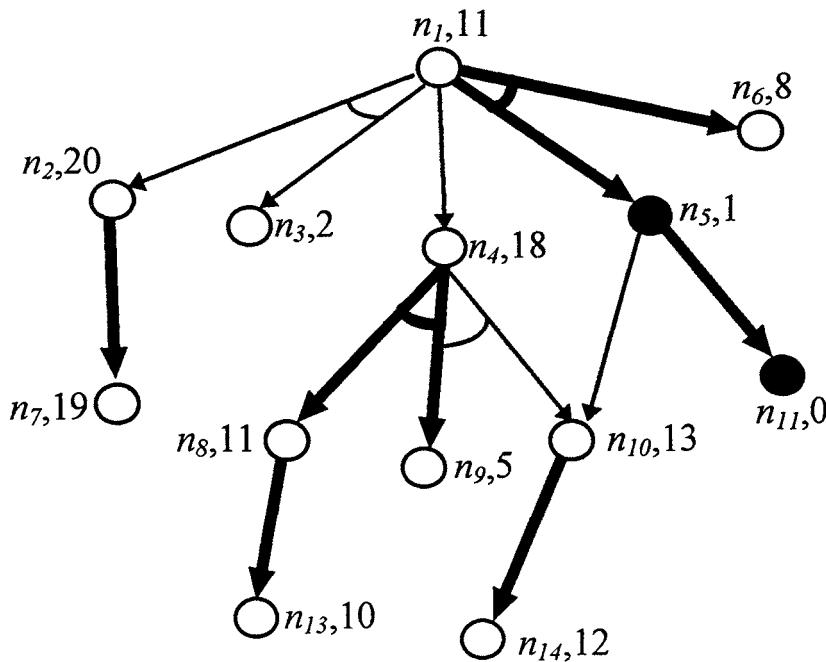


Figura 2-55

• Ciclo 7:

- En esta ocasión se expande n_6 .
- Evolución de S a lo largo del ciclo:

- 1) $S = \{n_6\}$

- 2) Nuevo coste de n_6 : $0+1=1$, quedando este nodo resuelto.

- 3) $S = \{n_1\}$

- 4) Nuevo coste de n_1 : $1+1+1+1=4$, quedando n_1 resuelto.

- Finalmente,

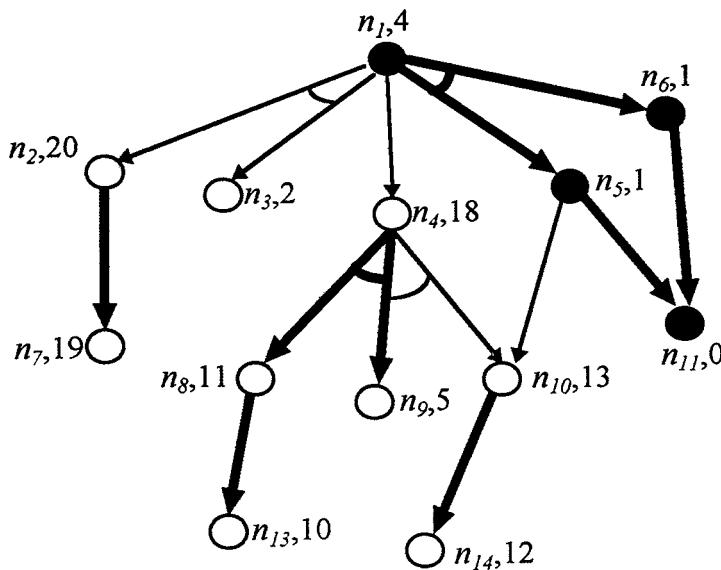


Figura 2-56

con lo que el nodo inicial queda resuelto y el algoritmo finaliza. El subárbol solución encontrado es el indicado por las líneas gruesas a partir de n_1 .

• **PROBLEMA 2.10: (*)**

Dado el árbol de la figura 2-57:

- a) Señale en dicho árbol las estrategias ganadoras —si es que las hubiera— para un jugador MAX y para un jugador MIN. Se considera que un estado extremo —o nodo terminal (hojas del árbol)— es ganador para MAX si su valor es superior o igual a 6. Contrariamente, un estado ganador para MIN es aquél cuyo valor es inferior a 6. Razone las respuestas.

- b) Recorra el árbol, de izquierda a derecha, siguiendo el método de poda α - β , indicando claramente (por ejemplo, tachándolos con una \times) los nodos en que se produce un corte en el proceso de búsqueda. Explique el razonamiento seguido en dichas situaciones, concretando el tipo de corte producido. Encuadre o marque los nodos terminales que no ha sido necesario considerar. Finalmente, señale el valor de la decisión inicial más acertada para MAX.

c) Realice el mismo estudio que en el apartado anterior b), suponiendo ahora que la raíz del árbol es un nodo MAX.

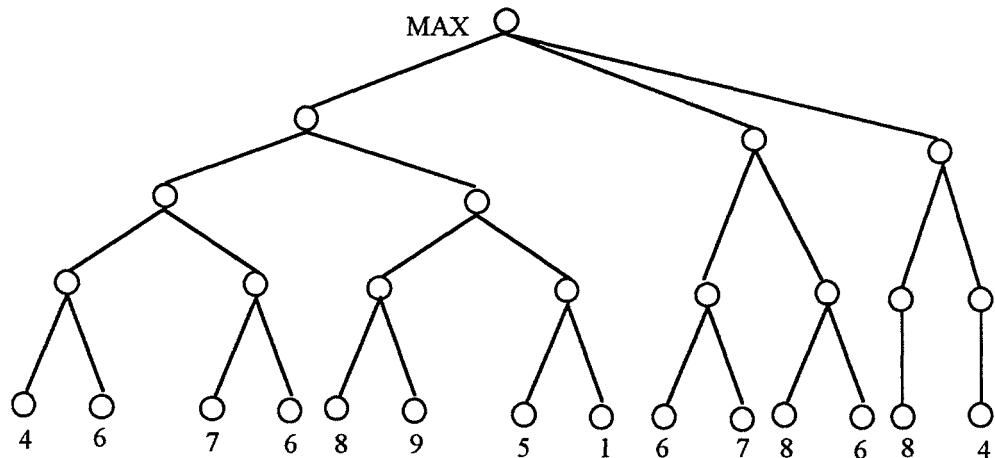


Figura 2-57

SOLUCIÓN:

¿Qué pretende este problema? Al ser éste el primer ejercicio sobre poda α - β , se introducirá a lo largo del mismo la notación gráfica que se va a emplear para este algoritmo en el resto del capítulo.

Para la resolución de este problema ténganse en cuenta que el árbol es no equilibrado, es decir, no todos los caminos desde el nodo raíz hasta un nodo hoja tienen la misma profundidad.

a) Estrategia ganadora para un jugador MAX:

Para MAX existen dos estrategias ganadoras que aparecen marcadas en la figura 2-58; una de ellas sería tomar el camino central y la otra el de la izquierda (se elegiría finalmente la central, ya que es la que conduciría a MAX a un valor más alto). En la figura aparecen marcados los enlaces que conducen a MAX a ganar el juego.

Para MIN no habrá ninguna estrategia ganadora, por haber al menos una estrategia ganadora para MAX.

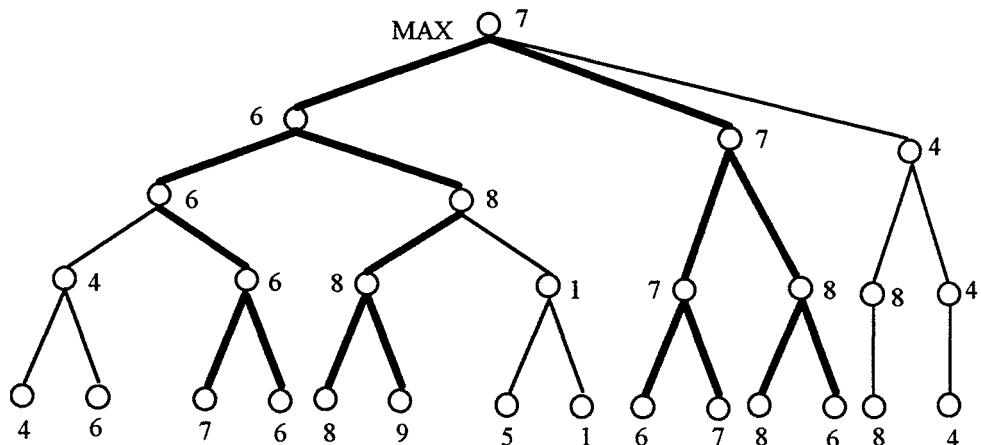


Figura 2-58

Aunque el enunciado del problema no lo pide, ¿qué pasaría si el nodo inicial fuese MIN? (ver figura 2-59)

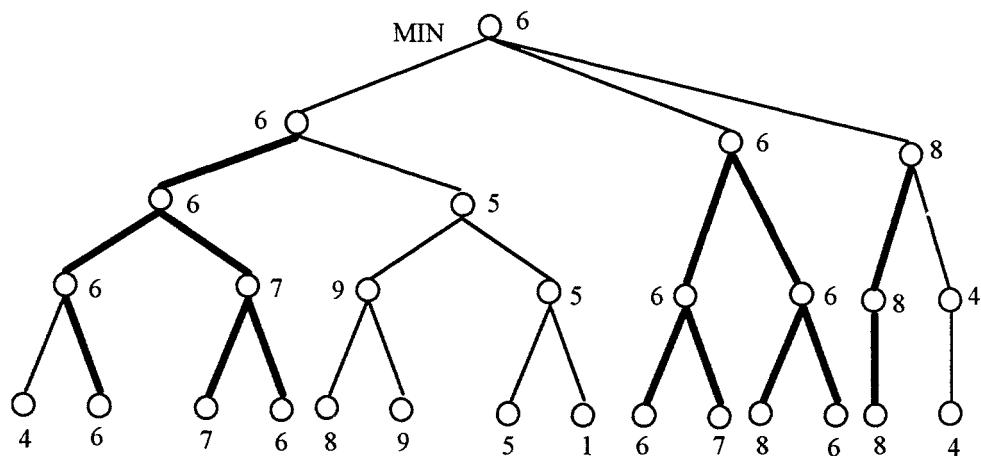


Figura 2-59

En este caso no existe estrategia ganadora para MIN, ya que desde el nodo raíz los nodos terminales con valor más bajo a los que se puede acceder son de valor 6, con lo

cual MAX ganaría la partida. Los enlaces marcados vuelven a representar la estrategia ganadora para MAX.

b) Con respecto a cada nodo, se especificarán en el árbol de búsqueda dos tipos de datos:

- El primero lo constituyen los valores de α y β con los que se hace cada llamada recursiva desde ese nodo. Se adoptará el siguiente convenio: si las llamadas se hacen desde un nodo MAX, α será el valor que podrá ir aumentando tras cada llamada recursiva, hasta igualar o superar a β , en cuyo caso se producirá una poda α . Este tipo de llamadas se representarán de la forma $(\alpha = \text{valor}_1, \text{valor}_2)$, ya que lo que se pretende resaltar es que es el cambio de α el que determina el que haya poda o no. Si, por el contrario, las llamadas recursivas se hacen desde un nodo MIN, será β la variable que podrá ir disminuyendo tras cada llamada. La poda β se producirá si β iguala o queda por debajo de α (que permanece fijo) en algún momento. En este caso, para resaltar el hecho de que es β la variable que determina la existencia o no de poda, este tipo de llamadas se representa de la forma $(\text{valor}_1, \beta = \text{valor}_2)$.
 - El segundo dato relacionado con cada nodo, que es importante especificar, es el resultado que se devuelve al nodo padre, como consecuencia de la ejecución del algoritmo.

Teniendo en cuenta las anteriores consideraciones, el árbol de búsqueda quedaría de la siguiente forma (mientras no se indique lo contrario, y para el resto de problemas que abordan este tipo de búsqueda, se supondrá que la búsqueda con retroceso realizada se lleva a cabo de izquierda a derecha):

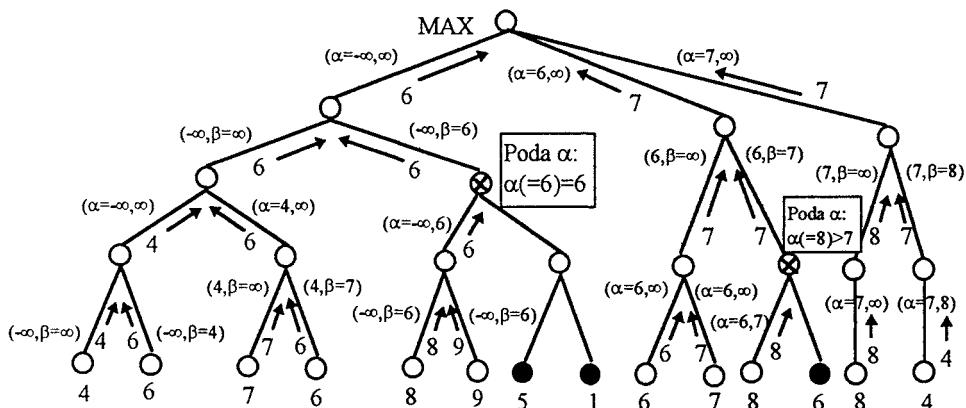


Figura 2-60

La decisión más acertada sería seguir el camino central, a través del cual se conseguiría llegar a un nodo terminal de valor 7. Éste es el camino a elegir debido a que a través de él se produce la última actualización (incremento) del valor de α mandado desde el nodo raíz en cada llamada recursiva.

c)

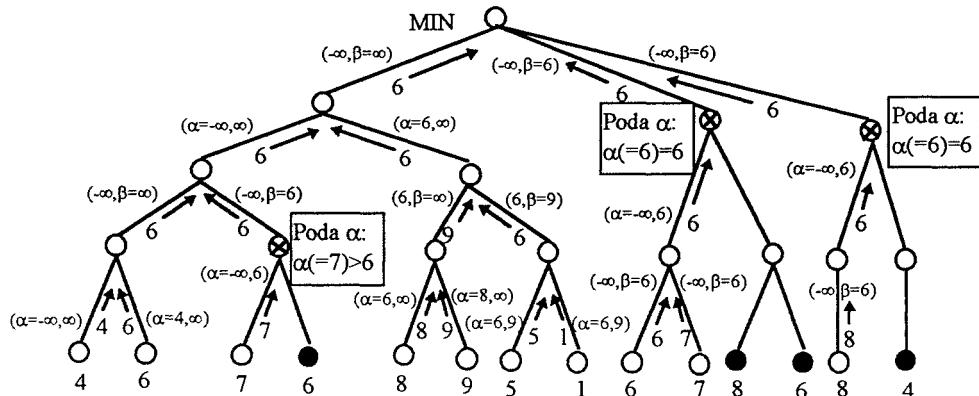


Figura 2-61

No existe estrategia ganadora para MIN, ya que el nodo terminal con el valor más bajo al que puede acceder tiene un valor de 6, que significa pérdida de la partida para MIN.

• PROBLEMA 2.11:

Considere el siguiente árbol:

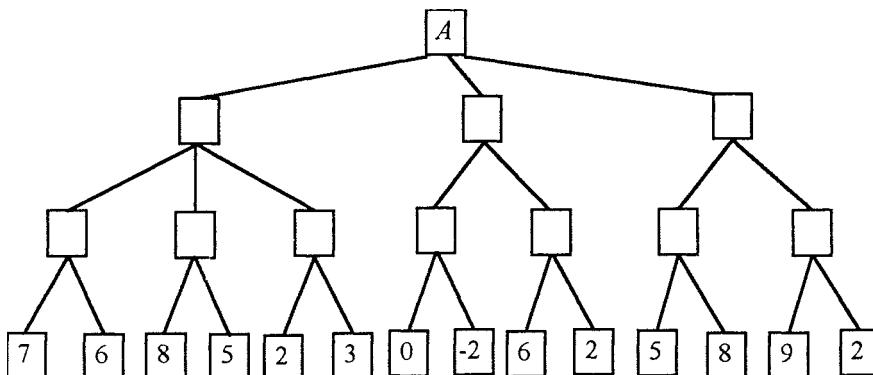


Figura 2-62

Suponiendo que A es el jugador MAX:

a) ¿Qué movimiento debería escoger?

b) En el árbol anterior, ¿qué nodos tendrían que ser examinados usando la estrategia de poda $\alpha\text{-}\beta$?

SOLUCIÓN:

a) El método MINIMAX realizaría un recorrido en profundidad del árbol, seleccionando en cada momento los siguientes valores:

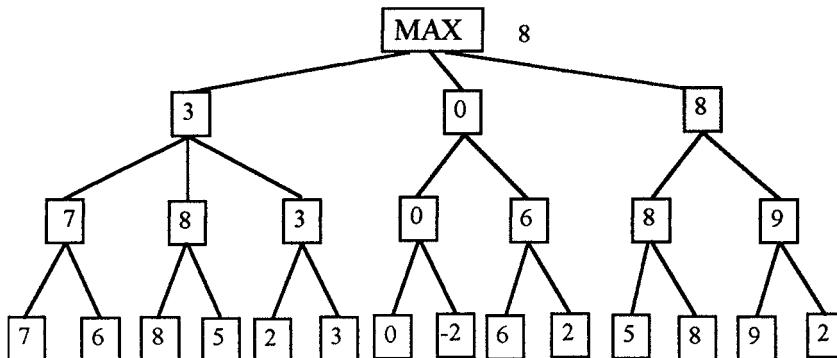


Figura 2-63

Se debería escoger, por tanto, el camino de la derecha para poder llegar al nodo terminal de valor 8.

b)

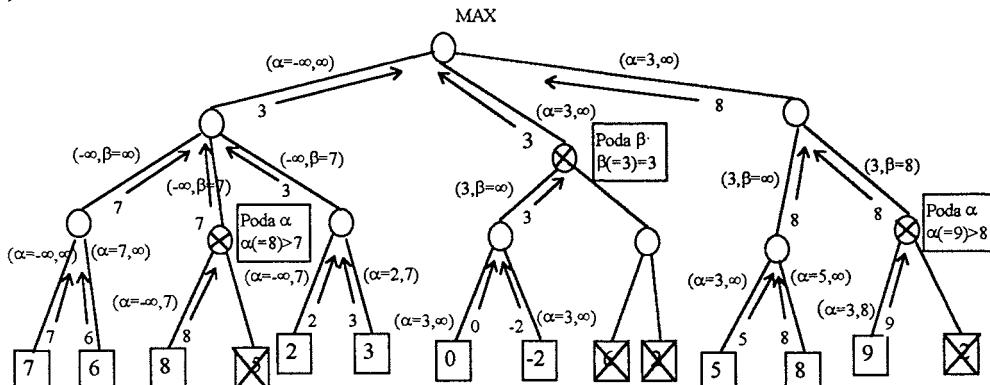


Figura 2-64

Tras haber recorrido el árbol de izquierda a derecha, de nuevo se comprueba que el mejor camino es el de la derecha. Por este camino se podrá acceder a un nodo terminal de valor 8.

¿Qué dificultades puede haber encontrado en este problema?

Aplicación 1 de la regla A:

El lector no entiende a qué se refieren los términos “jugador MAX” y “movimiento” del enunciado del problema.

Acciones:

Leer el texto comprendido entre el inicio del apartado 2.2.4 y la figura 2-4. En caso de persistir las dudas acudir a: [Mira *et al.*, 1995], páginas 164-170 o [Rich & Knight, 1991], páginas 339-342.

Aplicación 1 de la regla B:

No se entiende el significado de los números en los nodos hoja del árbol del enunciado (figura 2-62).

Acciones:

Examinar figura 2-4.

Aplicación 1 de la regla C:

El lector no sabe cómo aplicar el método MINIMAX para la resolución del problema.

Acciones:

Realizar una traza del algoritmo correspondiente al método MINIMAX que aparece en el apartado teórico 2.2.4.1 para el siguiente árbol:

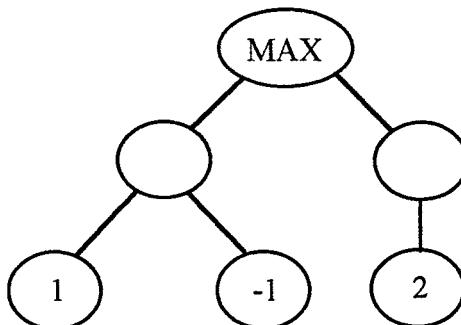


Figura 2-65

Aplicación 2 de la regla A:

Al leer el apartado b), el lector ignora lo que significa una “estrategia de poda alfabética”.

Acciones:

Acudir por este orden a: sección 2.2.4.2 del apartado teórico, [Mira *et al.*, 1995], páginas 172-176 y [Rich & Knight, 1991], páginas 343-349.

Aplicación 1 de la regla D:

Al leer la solución al apartado a) del problema, no se sabe cómo realizar un recorrido en profundidad del árbol del enunciado.

Acciones:

Acudir al problema 1.1 donde aparece un ejemplo de aplicación de búsqueda en profundidad.

Aplicación 2 de la regla B:

No se entiende el proceso de selección de valores llevado a cabo en la figura 2-63.

Acciones:

Examinar la figura 2-4

Aplicación 2 de la regla D:

El lector ignora por qué se debe escoger el camino de la derecha.

Acciones:

Examinar el siguiente ejemplo:

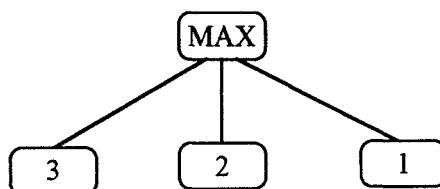


Figura 2-66

En este caso MAX tomaría el camino de la izquierda, ya que le conduciría al nodo con una situación de la partida más prometedora (3).

Aplicación 3 de la regla B:

El lector no consigue captar el significado de la figura 2-64.

Acciones:

Examinar la figura 2-60 y toda la explicación que aparece en el problema 2.10 referente a la misma.

Aplicación 3 de la regla D:

No se sabe qué significa un “recorrido de izquierda a derecha”.

Acciones:

Examinar el siguiente ejemplo donde se realiza un recorrido en profundidad de derecha a izquierda del siguiente árbol:

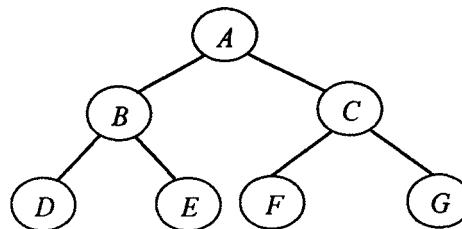


Figura 2-67

El resultado final sería: *A, C, G, F, B, E, D.*

Aplicación 4 de la regla D:

En la resolución del apartado b) no se sabe por qué se elige como mejor camino el de la derecha.

Acciones:

- 1) Leer la parte final del apartado b) del problema 2.10. En caso de persistir la duda:
- 2) Considérese el siguiente ejemplo donde el recorrido del árbol es de izquierda a derecha:

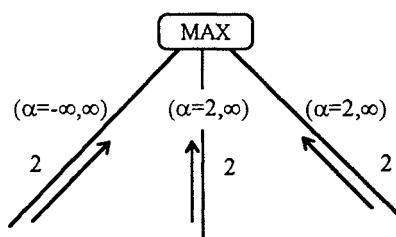


Figura 2-68

La última (y en este caso única) rama por la que el valor devuelto es mayor que el que toma α al hacer la llamada recursiva es la de la izquierda. Este camino será, por tanto, el elegido.

• PROBLEMA 2.12: (*)

Explore el árbol siguiente mediante el *procedimiento alfa-beta*, recorriendo el árbol de izquierda a derecha; marque con una \times los nodos en que se produce una poda, señalando su tipo. Encuadre o marque los nodos terminales considerados.

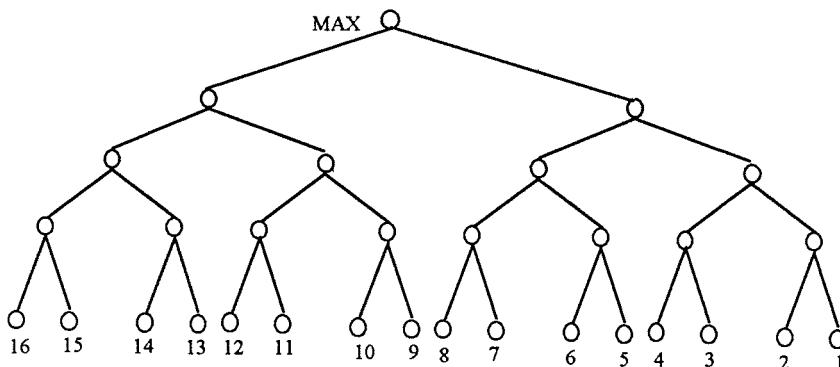


Figura 2-69

Realice el mismo proceso para el árbol siguiente:

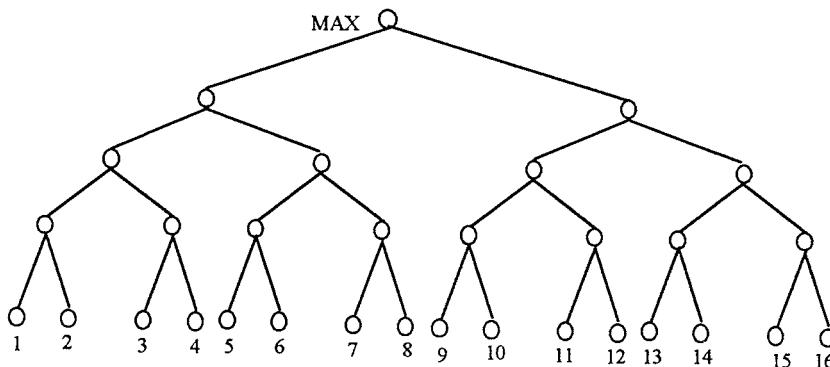


Figura 2-70

Compare las diferencias en el número de terminales considerados en ambos recorridos. Razoné las causas que justifican dichas diferencias.

SOLUCIÓN:

El árbol de búsqueda quedaría de la siguiente forma:

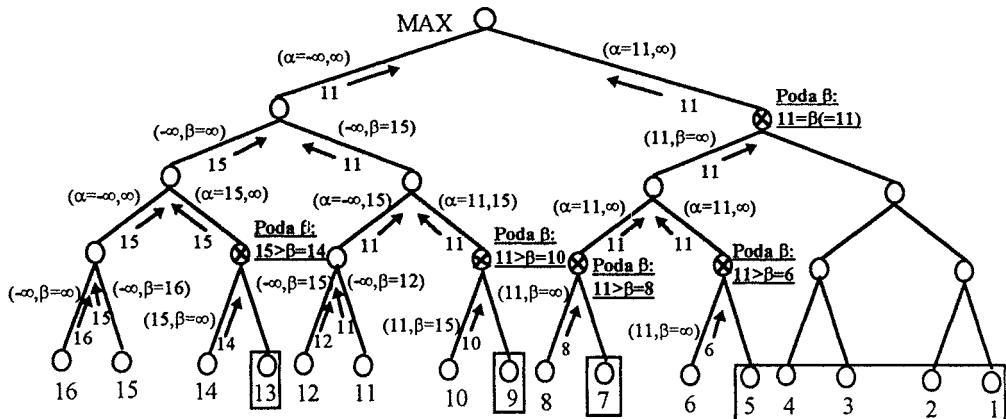


Figura 2-71

Una vez completado el algoritmo, $\alpha = 11$ y $\beta = \infty$, siendo la mejor jugada la del camino izquierdo. Para el segundo árbol del enunciado:

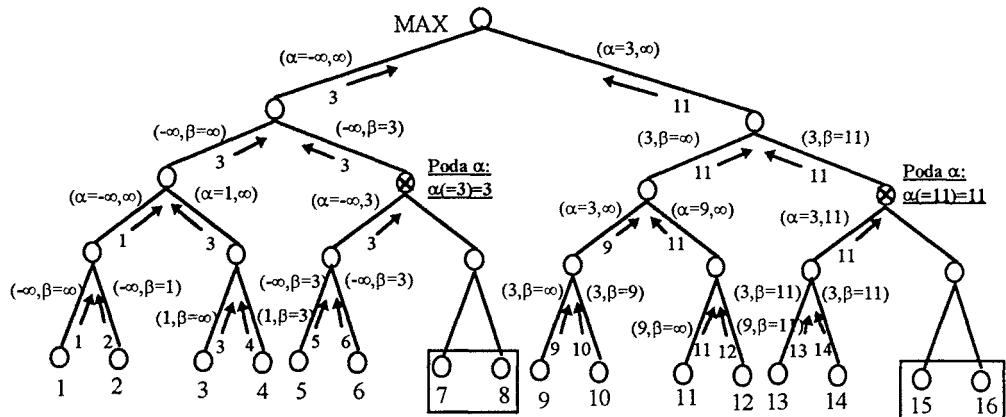


Figura 2-72

La mejor jugada en este caso corresponderá al camino derecho, que conducirá a un nodo terminal de valor 11. Se puede comprobar fácilmente que el nodo “2” no habría sido visitado si en vez de llamar inicialmente al procedimiento $\alpha\beta$ con $\alpha = -\infty$ y $\beta = +\infty$ se hubiera hecho con $\alpha = 1$ y $\beta = 16$. En el primer recorrido se consideran menos nodos terminales debido a que los nodos más prometedores se visitan al principio del proceso de búsqueda. La importancia del ordenamiento hace que haya otros procedimientos que realizan una ordenación previa de los nodos para optimizar el proceso de búsqueda.

• PROBLEMA 2.13:

Aplicar la estrategia de poda alfa-beta al siguiente árbol:

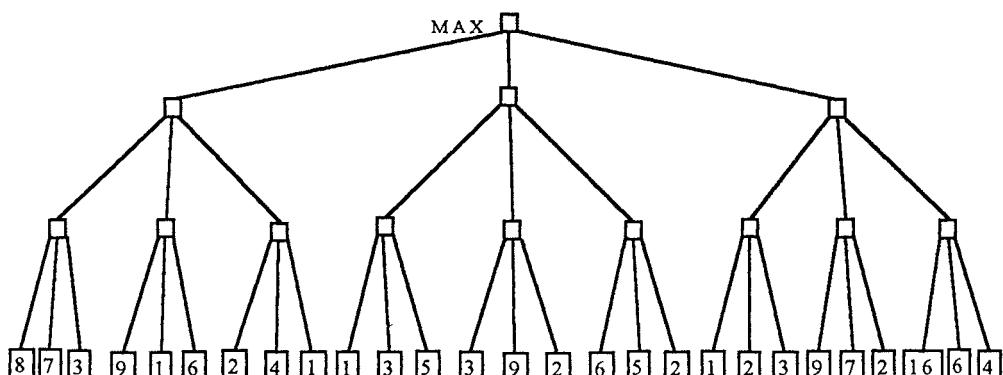


Figura 2-73

SOLUCIÓN:

¿Qué pretende este problema? En primer lugar se resolverá este ejercicio por el método descrito en el apartado teórico y posteriormente se introducirá un nuevo método más eficiente computacionalmente, aunque menos didáctico que el anterior.

El mejor camino es el central (ver figura 2-74), ya que la última actualización del valor de α que almacena el nodo raíz se produce a través de este camino, que permite llegar a un nodo terminal de valor 5.

En el algoritmo para la estrategia de poda alfa-beta aplicado en la figura anterior, debe tenerse en cuenta si las llamadas recursivas se hacen desde un nodo MAX o un nodo MIN, para determinar qué tipo de acciones llevar a cabo en cada caso. Existe, por otra parte, la posibilidad de modificar el algoritmo anterior con el fin de aplicar un

procedimiento único con independencia de si se está en un nodo MAX o MIN. Las modificaciones que habría que efectuar son las siguientes:

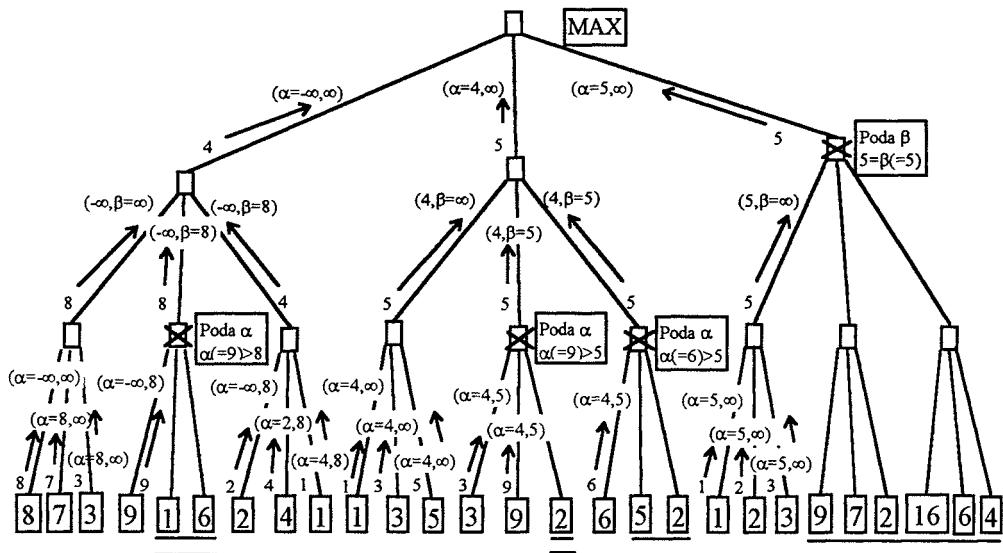


Figura 2-74

- En el caso del algoritmo anterior, el valor devuelto desde cada nodo hoja era siempre el de la función de evaluación heurística para ese nodo, según el criterio del jugador MAX, independientemente de si el nodo hoja era MAX o MIN. En el nuevo algoritmo, este valor devuelto dependerá de si el nodo hoja considerado es MAX o MIN (por ejemplo, en el presente problema, en vez de devolver un 8 desde el nodo hoja MIN situado más a la izquierda del árbol, habrá que devolver -8, que es el valor de la función de evaluación heurística para ese nodo según el criterio del jugador MIN).

- Así como con anterioridad se empleaban dos variables α y β , ahora el papel de estas variables lo desempeñarán l_s : “límite_siguientes” y l_a : “límite_actual”. Con independencia de si se está en un nodo MAX o MIN, se irá actualizando el valor de l_s hasta que supere o iguale a l_a , momento en el que habrá que efectuar una poda.

El nuevo cambio que hay que tener en cuenta respecto al algoritmo anterior se refiere al modo de hacer las llamadas recursivas cuando se cambia de nivel, que será de la siguiente forma:

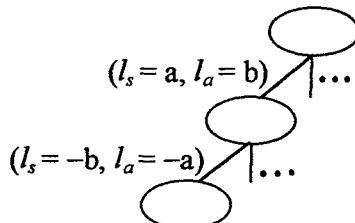


Figura 2-75

Obsérvese que en el algoritmo anterior no había que hacer ningún cambio:

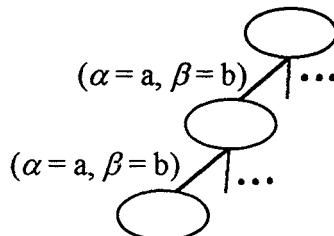


Figura 2-76

- La última modificación que hay que tener en cuenta es que los valores o mensajes numéricos mandados hacia arriba en cada paso del algoritmo son ahora multiplicados siempre por -1 antes de ser aceptados por el nodo padre del nodo que manda el valor. Por lo demás, el nuevo algoritmo opera de forma análoga al que había sido considerado hasta ahora:

alfabeta(m, profundidad, jugador, límite_siguientes, límite_actual)

1. Si m no tiene sucesores o si se considera que m ha alcanzado el límite de profundidad en *profundidad*, devolver:

$$\alpha\beta\nu(m) = \begin{cases} \bullet \text{fev}(m) & \text{si } m \text{ es un jugador MAX} \\ \bullet -\text{fev}(m) & \text{si } m \text{ es un jugador MIN} \end{cases}$$

2. Generar los sucesores de m :

2.1 Para cada sucesor n de m :

- (1) $\alpha\beta\nu(n) = \text{alfabeta}(n, \text{profundidad} + 1, C(\text{jugador}), -\text{límite_actual}, -\text{límite_siguientes})$, siendo $C(\text{jugador})$ una función que cambia de jugador.

$$(2) \text{ límite_siguientes} = \max[-\alpha\beta\nu(n), \text{ límite_siguientes}]$$

(3) Si $\text{límite_siguientes} \geq \text{límite_actual}$, abandonar este nivel y devolver límite_siguientes .

3. Una vez que se han analizado recursivamente todos los sucesores, se devuelve el que ha obtenido un mejor valor:

$$\alpha\beta\nu(m) = \text{límite_siguientes}$$

La primera llamada recursiva se hace con:

m: Inicial (situación inicial de la partida)

profundidad: 0

jugador: MAX

límite_siguientes: fev_{min} (valor mínimo que puede alcanzar fev)

límite_actual: fev_{max} (valor máximo que puede alcanzar fev)

La solución al presente problema, aplicando el nuevo algoritmo, sería (en la figura siguiente, el primer valor de cada llamada recursiva es l_s y el segundo l_a):

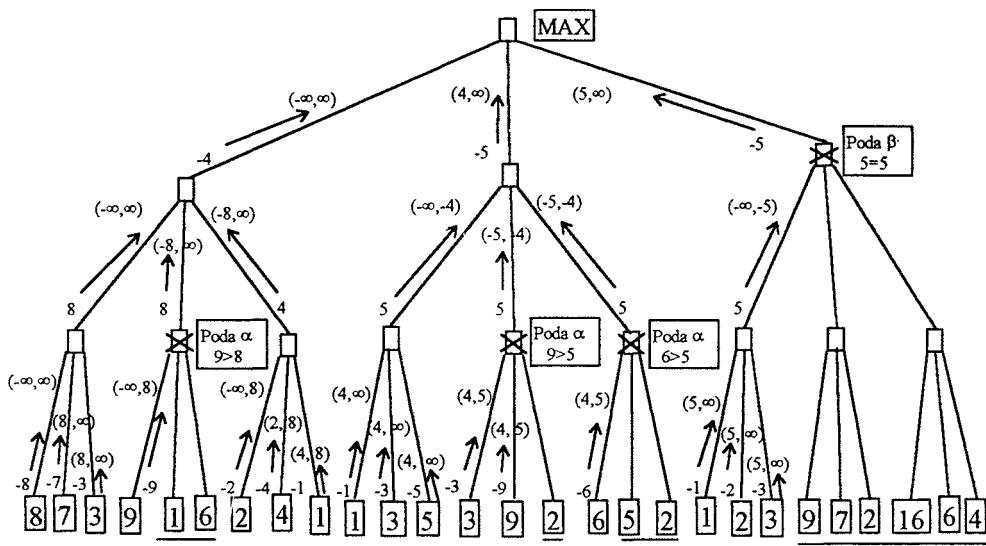


Figura 2-77

El mejor camino sería el central y conduciría a un nodo de valor 5.

• **PROBLEMA 2.14:**

Señalar los arcos y los nodos terminales visitados mediante la estrategia de poda $\alpha\text{-}\beta$ en el siguiente árbol si se recorre el mismo de izquierda a derecha.

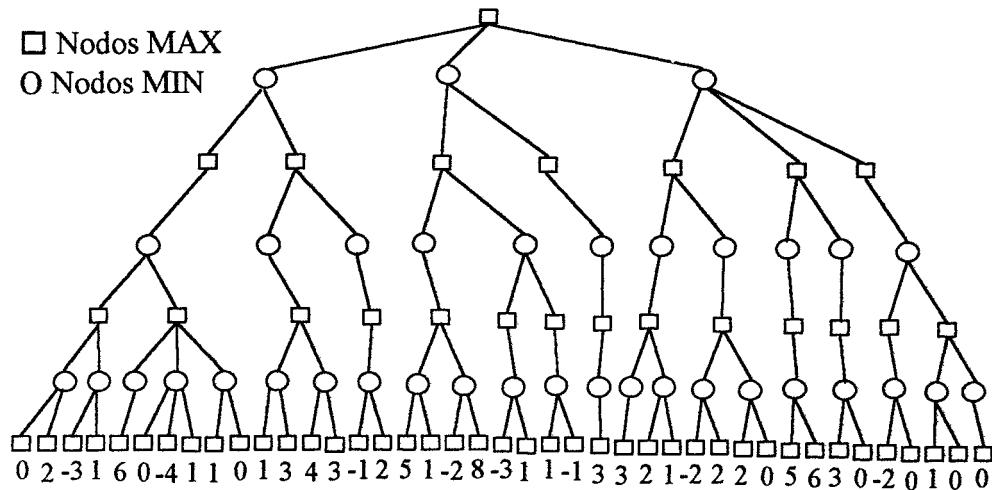


Figura 2-78

SOLUCIÓN:

En la figura 2-79 aparece la solución a este problema.

2.4 Contexto adicional

2.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

- Saber qué es una función de evaluación heurística y cuáles son las ventajas que produce su utilización en un proceso de búsqueda.
- Diferenciar los diversos modos en que se puede hacer uso de una función de evaluación heurística y que dan lugar a estrategias irrevocables, de exploración de alternativas o de búsqueda con adversarios.
- Identificar aquellos problemas o tareas para cuya solución es adecuado el empleo de grafos Y/O.

- Saber por qué y cuándo está justificado realizar una poda para el caso de la estrategia de poda alfa-beta.

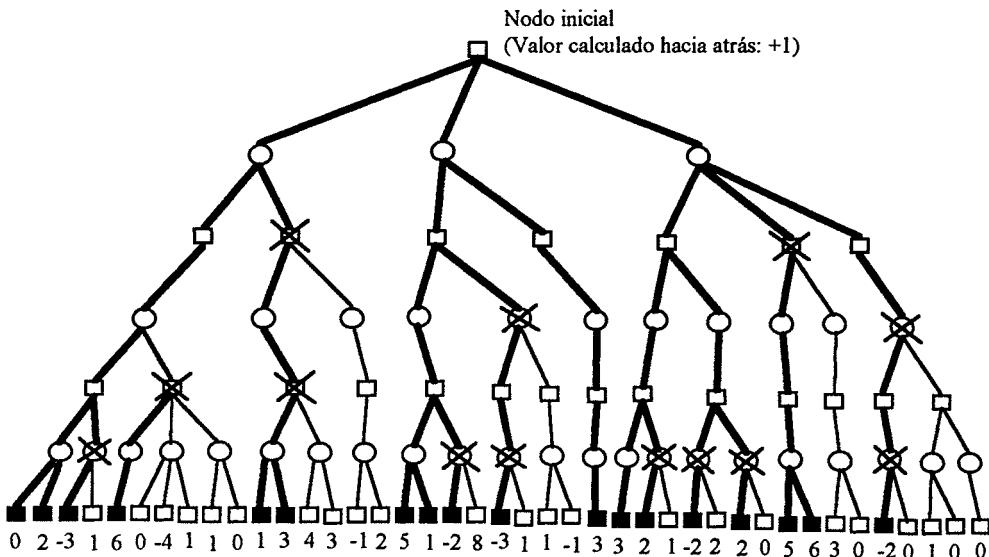


Figura 2-79

2.4.2 Pistas de autoevaluación

- ¿Cómo se modificaría la solución del problema 2.1 si el valor de la función heurística asociado al nodo F fuera 4 en lugar de 2? ¿Se alcanzaría en este caso un nodo objetivo o meta?
- Realizar el apartado b) del problema 2.3 suponiendo que se expanden en cada paso del algoritmo sólo los 3 nodos más prometedores.
- ¿Cómo quedaría modificada la solución del problema 2.4 si se suprime el enlace que une los nodos I y B ?
- Hallar la solución al problema 2.8 si el enlace que une n_4 y n_8 no existiera.
- En el problema 2.12 realice el recorrido del árbol de la figura 2-65 de derecha a izquierda. Los nodos visitados y podas realizadas deberían ser los mismos que en el caso del árbol de la figura 2-66.

3 LÓGICA

3.1 Contexto previo

3.1.1 ¿Por qué está aquí este capítulo?

En las décadas de los setenta y ochenta se empezaron a aplicar técnicas propias de la inteligencia artificial en la resolución de problemas del mundo real con complejidad mayor que la de los llamados “mundos formales” (juegos, mundo de los bloques...). Esta época vio surgir los llamados sistemas basados en conocimiento de los cuales los sistemas expertos constituyen un ejemplo. Una importante conclusión a la que se llegó fue la importancia de saber modelar la gran cantidad de conocimiento específico del dominio que aparecía en cada caso.

Entre los principales formalismos de representación utilizados en los sistemas basados en conocimiento figuran: la lógica, las reglas, las redes asociativas, los marcos y los guiones. Asociados con cada uno de estos formalismos existen métodos específicos de manipulación del conocimiento representado. El proceso de obtención de nueva información a partir de la ya disponible se denomina *razonamiento* o *inferencia*. En los próximos capítulos aparecen ejemplos concretos de representación de conocimiento e inferencia a partir de los métodos o formalismos enumerados previamente. Además, en el capítulo inicial se ha ofrecido una explicación del papel de la representación formal de las entidades de un modelo y de su posición relativa en el proceso global de desarrollar una versión computable de un sistema basado en conocimiento a partir de un modelo conceptual de la tarea básica y del método asociado, ambos descritos a nivel de conocimiento, en el sentido de Newell.

3.1.2 ¿Dónde hay conocimiento teórico del campo?

El lector podrá adquirir una visión más extensa del campo de la lógica clásica y sus ampliaciones en:

Referencias básicas:

- [Aranda *et al.*, 1993]
- [Brewka, 1991]

- [Gensler, 1990]
- [Klir & Yuan, 1995]
- [Mira *et al.*, 1995], capítulo 5.

Referencias complementarias:

- [Jackson *et al.*, 1989]
- [Lucas & Van der Gaag, 1991], capítulo 2.
- [Nilsson, 1987], capítulos 4 y 5.
- [Norvig, 1992], capítulo 14.
- [Rich & Knight, 1994], capítulos 5 y 7.
- [Winston, 1994], capítulo 13.

3.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

- Conocimientos sobre lógica de proposiciones y de predicados a nivel medio.
- Nociones básicas de teoría de conjuntos.

3.1.4 ¿Qué software de apoyo está disponible?

- **Prolog-2**

Versión compatible con el Prolog de Edimburgo, disponible en la Open University del Reino Unido. Puede obtenerse en:

<http://www.dia.uned.es/~jgb/util/prolog.html>

- **COMMON LISP**

El lenguaje LISP, junto con Prolog, es uno de los lenguajes más empleados en inteligencia artificial. Esta versión puede obtenerse en:

<http://www.dia.uned.es/~jgb/util/lisp.html>

- **RICE (*Routines for Implementing C Expert Systems*)**

Motor de inferencia para lógica difusa escrito en C/C++. Puede obtenerse en:

<http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/fuzzy/systems/rice/0.html>

- **IFR_fuzzy**

Módulo de control difuso para MATLAB que permite trabajar con variables difusas, reglas difusas... Puede obtenerse en:

<http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/fuzzy/systems/ifr/0.html>

- **INFER**

Librería general sobre razonamiento difuso desarrollada en la Universidad de Tokio.
Puede obtenerse en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/fuzzy/systems/infer3/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/fuzzy/systems/infer3/0.html)

3.2 Contenido teórico

3.2.1 Introducción

Este capítulo intenta dar una visión de cómo la *lógica clásica*, con la cual suponemos que el lector está ya familiarizado, constituye un mecanismo con capacidad insuficiente para abordar ciertos problemas que se presentan en inteligencia artificial. La deficiencia mencionada con anterioridad ha provocado en las últimas décadas la aparición de diferentes lógicas que pretenden servir de ampliación a la lógica clásica.

3.2.2 Lógica de predicados con identidad

Supone la ampliación de la lógica de predicados tradicional con el predicado diádico “=”, de manera que si se escribe $x=y$, lo que se quiere expresar es que “ x ” e “ y ” se refieren al mismo elemento. Algunos ejemplos de la utilización de este nuevo predicado podrían ser:

1. “Alguien que no es Roberto es rico.”

$$\exists x (x \neq r \wedge Rx)$$

donde “ r ” representa “Roberto”, “ R ” representa “ser rico” y \neq equivale a la negación lógica de la identidad.

2. “Sólo Roberto es rico.”

$$Rr \wedge \neg \exists x (x \neq r \wedge Rx)$$

3. “Por lo menos dos personas son ricas.”

$$\exists x \exists y (x \neq y \wedge Rx \wedge Ry)$$

4. “Exactamente dos personas son ricas.”

$$\exists x \exists y ((x \neq y \wedge Rx \wedge Ry) \wedge \neg \exists z (z \neq x \wedge z \neq y \wedge Rz))$$

Se necesitan dos reglas nuevas cuando se amplía la lógica de predicados con la identidad:

regla a): $a = a$ siempre es cierto, sea cual sea la constante “a”.

regla b): $F_a, a = b \rightarrow F_b$ para cualesquiera dos constantes “a” y “b”. Por ejemplo:

C_j “Juan es ciego.”

$j = a$ “Juan es tu mejor amigo.”

$\therefore C_a$ Por tanto, “Tu mejor amigo es ciego.”

donde \therefore representa “se infiere” o “se deduce”.

PROBLEMAS RELACIONADOS: 3.4 a 3.10

3.2.3 Lógica modal

La *lógica modal* constituye una extensión de la lógica de predicados con la capacidad de evaluar argumentos que implican *necesidad* o *posibilidad*. Para ello se introducen dos nuevos símbolos:

$\Box A$: “Es *necesario* que A.”

$\Diamond A$: “Es *possible* que A.”

Necesidad y posibilidad son dos “modos de verdad”; de ahí el nombre de esta lógica. “ $\Box A$ ” es un concepto más fuerte que “A es cierto” y significa “A tiene que ser cierto”. Por otro lado, “ $\Diamond A$ ” es más débil que “A es cierto” y significa: “A podría ser cierto”. Para el tratamiento de los conceptos de necesidad y posibilidad, la lógica modal se apoya en la idea de “mundo”. Un *mundo* es una descripción consistente y completa de cómo las cosas podrían haber sido o podrían, de hecho, ser. De entre todos los mundos de los que se puede hacer uso en un razonamiento con lógica modal, cabe destacar el llamado “mundo real”, que describe las cosas tal como realmente éstas son. Un enunciado *necesario* sería aquél que es cierto en todos los mundos posibles. Un enunciado *verdadero* es aquél que es cierto en el *mundo real*. Finalmente, un enunciado *possible* es aquél que es cierto en algún posible mundo. Un enunciado *possible* puede o no ser cierto en el mundo real. En lógica modal se consideran las nuevas reglas de inferencia siguientes para cualquier fórmula válida “A”:

$$1. \neg\Diamond A \leftrightarrow \Box\neg A$$

$$2. \neg\Box A \leftrightarrow \Diamond\neg A$$

3. $\Box A \rightarrow A$ (Si “A” es necesario, entonces es cierto en cualquier posible mundo M que escojamos, incluyendo el mundo real.)

4. $\Diamond A \rightarrow A$ (Si “A” es posible, entonces es cierto en un mundo al que se asignará un nombre que no haya sido especificado con anterioridad.)

Ejemplo de uso de las reglas de inferencia:

Se pretende probar lo siguiente:

“Necesariamente, si hay materia, hay mal.”

“Es necesario que haya materia.”

Por tanto, “Es necesario que haya mal.”

La demostración se hará por reducción al absurdo:

1. “Necesariamente, si hay materia, hay mal.”

2. “Es necesario que haya materia.”

[∴ “Es necesario que haya mal.”

3. | supóngase: “No es necesario que haya mal.”

4. | ∴ “Es posible que no haya mal.” (a partir de 3 y la *regla 2*)

5. | ∴ En algún mundo W : “No hay mal.” (a partir de 4 y la *regla 4*)

6. | ∴ En W : “Hay materia.” (a partir de 2 y la *regla 3*)

7. | ∴ En W : “Si hay materia, hay mal.” (a partir de 1 y la *regla 3*)

8. | ∴ En W : “Hay mal.” (a partir de 6 y 7, contradiciendo a 5)

9. ∴ “Es necesario que haya mal.” (a partir de 3, 5 y 8)

La línea 5 usa el principio de que aquello que es *possible* es verdad en *algún* mundo. Las líneas 6 y 7, por su parte, usan el principio de que aquello que es *necesario* es verdad en *cualquier posible* mundo. En el mundo W del ejemplo se ha llegado a una contradicción, ya que contiene “No hay mal” (línea 5) y “Hay mal” (línea 8). Por tanto, la conclusión inicial era válida. A partir de ahora se utilizará la siguiente representación:

$W: P$ (“P es verdad en el mundo W .”)

$W2$ supóngase: P (“Se supone que P es verdad en el mundo $W2$.”)

El ejemplo anterior, utilizando el simbolismo:

$$M \equiv \text{"hay materia "}$$

$$E \equiv \text{"hay mal"}$$

quedaría de la siguiente forma:

Dadas las premisas: $\Box(M \supset E)$ y $\Box M$, deducir $\Box E$.

Por reducción al absurdo:

1. $\Box(M \supset E)$
2. $\Box M$
3. $\Box E$
supóngase: $\neg\Box E$
4. $\therefore \Diamond\neg E$ (a partir de 3 y la *regla 2*)
5. $W \therefore \neg E$ (a partir de 4 y la *regla 4*)
Esta conclusión parcial se cumple sólo en el mundo W .
6. $W \therefore M$ (a partir de 2 y la *regla 3*)
7. $W \therefore (M \supset E)$ (a partir de 1 y la *regla 3*)
8. $W \therefore E$ (a partir de 6 y 7, contradiciendo a 5)
9. $\therefore \Box E$ (a partir de 3, 5 y 8)

Nótese que para que una contradicción sea válida debe aparecer sólo en un mundo. Los dos hechos siguientes, por ejemplo, no conducirían a una contradicción:

$$W1 \quad A$$

$$W2 \quad \neg A$$

Como se puede deducir a partir del ejemplo anterior, la mejor estrategia para probar un razonamiento en lógica modal consiste en intentar sacar los operadores modales fuera en cada enunciado y posteriormente hacerlos desaparecer especificando un mundo en el que el enunciado se cumple. Por otra parte, sólo se podrá usar una regla de inferencia dentro de un mundo dado. Si se tuviera " $(A \supset B)$ " y " A " en el mismo mundo, entonces se podría inferir " B ", pero sólo en ese mundo.

PROBLEMAS RELACIONADOS: 3.11 a 3.14

3.2.4 Lógica difusa

La lógica clásica no ofrece un marco adecuado para la representación de conocimiento de tipo impreciso y subjetivo. Además, no permite la realización de razonamientos donde la incertidumbre deba ser tenida en cuenta. Considérese, por ejemplo, la frase:

“Las personas altas generalmente son bastante pesadas.”

Si nos encontramos con una persona que mida 1 metro y 70 centímetros, deberíamos poder decir algo a partir de este dato y el conocimiento representado en la frase anterior. Sin embargo, surgen una serie de preguntas:

- ¿Es alta una persona de 170 cms. de altura?
- ¿Cuál es el rango de pesos donde entran las personas *bastante pesadas*?
- Dada una persona considerada como *alta*, ¿cuándo se podrá decir que es bastante pesada? Es decir, ¿cuál es el efecto real del adverbio *generalmente* sobre el resto de la frase?

La lógica difusa intenta resolver las deficiencias que aparecen en la lógica clásica al abordar problemas de características similares a las mencionadas con anterioridad.

• Conjuntos borrosos

En un *conjunto borroso* cada elemento tiene asociado un *grado de pertenencia* al mismo comprendido en el intervalo $[0, 1]$.

Ejemplo Considérese el universo de edades $U=\{0, 10, 20, 30, 40, 50, 60, 70, 80\}$. Se puede definir el conjunto borroso C , que representa el concepto “viejo”, del siguiente modo:

$$C=\{0|0, 10|0, 20|0, 30|0'1, 40|0'2, 50|0'5, 60|0'9, 70|1, 80|1\}$$

donde al lado de cada elemento (edad en años) aparece el grado de pertenencia del mismo al conjunto representado. A la función que asocia a cada elemento su grado de pertenencia se la llama *función de pertenencia*. Esta función da idea de la compatibilidad de cada elemento con el concepto representado. Por tanto, en general se puede definir un conjunto borroso C sobre un universo U de la siguiente forma:

$$C=\{x|\mu_C(x) \text{ } \forall x \in U \text{ si } \mu_C(x) \neq 0\}$$

donde μ_C es la función de pertenencia asociada al conjunto C .

• **Operaciones definidas sobre conjuntos borrosos**

Dados los conjuntos borrosos F y G cuyos elementos pertenecen a un universo U , se definen las siguientes operaciones:

a) INCLUSIÓN:

$$F \subset G \text{ si } \mu_F(x) \leq \mu_G(x) \quad \forall x \in U$$

b) IGUALDAD:

$$F = G \text{ si } \mu_F(x) = \mu_G(x) \quad \forall x \in U$$

c) DESIGUALDAD:

$$F \neq G \text{ si } \mu_F(x) \neq \mu_G(x) \quad \forall x \in U$$

d) COMPLEMENTO:

$$c(F) = \{x | \mu_{c(F)}(x) = 1 - \mu_F(x) \quad \forall x \in U\}$$

e) UNIÓN:

$$F \cup G = \{x | \mu_{F \cup G}(x) = \max(\mu_F(x), \mu_G(x)) \quad \forall x \in U\}$$

f) INTERSECCIÓN:

$$F \cap G = \{x | \mu_{F \cap G}(x) = \min(\mu_F(x), \mu_G(x)) \quad \forall x \in U\}$$

Dado un universo de discurso U y los conjuntos **no borrosos** X , Y y Z definidos en él, éstos cumplen las siguientes propiedades a partir de la teoría de conjuntos:

Involución

$$c(c(X)) = X$$

Comutatividad

$$X \cup Y = Y \cup X, X \cap Y = Y \cap X$$

Asociatividad

$$(X \cup Y) \cup Z = X \cup (Y \cup Z), (X \cap Y) \cap Z = X \cap (Y \cap Z)$$

Distributividad

$$X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z), X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$$

Idempotencia

$$X \cup X = X, X \cap X = X$$

Absorción

$$X \cup (X \cap Y) = X, X \cap (X \cup Y) = X$$

Absorción por U y \emptyset

$$X \cup U = U, X \cap \emptyset = \emptyset$$

Identidad

$$X \cup \emptyset = X, X \cap U = X$$

Ley de contradicción

$$X \cap c(X) = \emptyset$$

Ley del tercio excluso

$$X \cup c(X) = U$$

Leyes de De Morgan

$$c(X \cap Y) = c(X) \cup c(Y), c(X \cup Y) = c(X) \cap c(Y)$$

Para conjuntos difusos todas estas propiedades siguen cumpliéndose, excepto la de contradicción y la del tercio excluso. Por tanto, los conjuntos borrosos dotados de las operaciones de unión, intersección y complemento no pueden formar un *álgebra de Boole*.

- **Modificadores lingüísticos**

Los *modificadores lingüísticos* (“casi”, “muy”...) se modelan en lógica borrosa a partir de determinadas operaciones sobre la función de pertenencia asociada al predicado que se está modificando. Algunas de las operaciones sugeridas para este fin son:

Tipo de operación	Representación	Cálculos realizados
Negación	NEG($\mu(x)$)	$1 - \mu(x)$
Concentración	CON($\mu(x)$)	$\mu^2(x)$
Dilatación	DIL($\mu(x)$)	$2\mu(x) - \mu^2(x)$
Intensificación	INT($\mu(x)$)	$2\mu^2(x)$ si $0 \leq \mu(x) \leq 0.5$ $1 - 2(1 - \mu(x))^2$ si $\mu(x) > 0.5$

Tabla 3-1

de manera que se pueden definir modificadores difusos de la siguiente forma:

Modificador difuso	Operación asociada
“no”	NEG($\mu(x)$)
“muy”	CON($\mu(x)$)
“algo”	DIL($\mu(x)$)
“casi”	DIL(DIL($\mu(x)$)))
“bastante”	INT(CON($\mu(x)$)))
...	...

Tabla 3-2

- Relaciones borrosas

Dados n universos U_1, \dots, U_n se define una *relación difusa* R por una función de pertenencia μ_R que asocia a cada (x_1, \dots, x_n) , con $x_i \in U_i$, un valor en el intervalo $[0, 1]$.

Ejemplo Se define el *producto cartesiano* de dos conjuntos difusos como la relación binaria difusa siguiente:

$$F \times G = \{(x, y) | \mu_{F \times G}(x, y) = \min(\mu_F(x), \mu_G(y)) \quad \forall x \in U_1, \forall y \in U_2\}$$

Las relaciones binarias difusas pueden ser representadas mediante matrices.

Ejemplo Considérense los siguientes universos:

$$U_1 = \{\text{Pedro, María}\}, \quad U_2 = \{\text{Juan, Ana, Antonio}\}$$

La relación binaria borrosa “ $x \in U_1$ le cae simpático a $y \in U_2$ ” podría tener la siguiente representación matricial:

R	Juan	Ana	Antonio
Pedro	0	0'5	0'7
María	1	0'5	0'2

Se define la *composición de relaciones borrosas* de la siguiente forma:

$$R1(U_1, U_2) \circ R2(U_2, U_3) = R3(U_1, U_3)$$

donde

$$R3(U_1, U_3) = \{(x, z) | \mu_{R1 \circ R2}(x, z) \quad \forall x \in U_1, \forall y \in U_2, \forall z \in U_3\}$$

siendo

$$\mu_{R_1 \circ R_2}(x, z) = \max[\forall y \in U_2: \min(\mu_{R_1}(x, y), \mu_{R_2}(y, z))]$$

Existe una analogía entre la operación definida anteriormente y el producto matricial usual, sin más que asociar la suma con la operación *max* y el producto con la operación *min*. Debido a lo anterior, a la composición de relaciones borrosas se la conoce también por el nombre de *producto matricial max-min*.

Ejemplo Sean los siguientes universos:

$$U_1 = \{a, b, c, d\}$$

$$U_2 = \{e, f\}$$

$$U_3 = \{g, h, i\}$$

Se definen las siguientes relaciones:

$$R_1(U_1, U_2) = \begin{pmatrix} 0'3 & 0 \\ 0'2 & 0'1 \\ 0'1 & 0'2 \\ 0 & 0'3 \end{pmatrix}$$

$$R_2(U_2, U_3) = \begin{pmatrix} 0'1 & 0'7 & 0'5 \\ 0'2 & 0 & 0 \end{pmatrix}$$

Por tanto,

$$R_3(U_1, U_3) = R_1(U_1, U_2) \circ R_2(U_2, U_3) = \begin{pmatrix} 0'3 & 0 \\ 0'2 & 0'1 \\ 0'1 & 0'2 \\ 0 & 0'3 \end{pmatrix} \circ \begin{pmatrix} 0'1 & 0'7 & 0'5 \\ 0'2 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0'1 & 0'3 & 0'3 \\ 0'1 & 0'2 & 0'2 \\ 0'2 & 0'1 & 0'1 \\ 0'2 & 0 & 0 \end{pmatrix}$$

Como caso particular de la operación anterior se encuentra la composición entre un conjunto borroso y una relación borrosa, empleado en *inferencia borrosa*. En este caso, cada conjunto borroso se representaría por una matriz de una fila.

- Interpretación de sentencias compuestas

A cada predicado borroso, por ejemplo “ser viejo”, se le puede asociar un conjunto borroso, o una relación borrosa si el predicado es poliádico. También es posible asignar valores de verdad a sentencias borrosas compuestas:

a) Disyunción:

$$I(P(x) \vee Q(y)) = \max(\mu_P(x), \mu_Q(y)) \quad \forall x \in U_1, \forall y \in U_2$$

b) Conjunción:

$$I(P(x) \wedge Q(y)) = \min(\mu_P(x), \mu_Q(y)) \quad \forall x \in U_1, \forall y \in U_2$$

c) Condicional:

Existen diferentes interpretaciones posibles para el condicional en la literatura de lógica difusa:

$$I(P(x) \rightarrow Q(y)) = \max(\min(\mu_P(x), \mu_Q(y)), 1 - \mu_P(x)) \quad \forall x \in U_1, \forall y \in U_2$$

$$I(P(x) \rightarrow Q(y)) = \min(\mu_P(x), \mu_Q(y)) \quad \forall x \in U_1, \forall y \in U_2$$

(En lo que resta emplearemos esta interpretación, que coincide con el producto cartesiano.)

$$I(P(x) \rightarrow Q(y)) = \mu_P(x) \cdot \mu_Q(y) \quad \forall x \in U_1, \forall y \in U_2$$

$$I(P(x) \rightarrow Q(y)) = \min(1, 1 - \mu_P(x) + \mu_Q(y)) \quad \forall x \in U_1, \forall y \in U_2$$

- Inferencia en lógica difusa

Algunas de las reglas básicas de inferencia que se emplean en lógica difusa son las siguientes:

a) Principio de herencia: dado un conjunto borroso A asociado a un predicado, que es un subconjunto borroso de B , cualquier elemento con la propiedad A hereda la propiedad B .

$$\frac{\begin{array}{c} A(x) \\ A \subseteq B \end{array}}{B(x)} \quad \forall x \in U, A \subset U, B \subset U$$

Se dice que un conjunto borroso A es un *subconjunto* de otro conjunto borroso B , definidos sobre un mismo universo U , si el grado de pertenencia de cada elemento del universo al conjunto A es menor o igual que su grado de pertenencia al conjunto B :

$$A \subset B \text{ si } \mu_A(x) \leq \mu_B(x) \quad \forall x \in U$$

b) Regla de composición: dado un elemento que cumple la propiedad representada por el conjunto borroso A y que está relacionado a través de una relación binaria R con otro elemento de un universo diferente, se le puede asociar a este segundo elemento la propiedad resultante de componer A y R .

$$\begin{array}{l} A(x) \\ \underline{R(x, y)} \\ (A \circ R)(y) \quad \forall x \in U, \forall y \in V, A \subset U, R \subset U \times V \end{array}$$

c) Modus Ponens generalizado:

$$\begin{array}{l} A(x) \\ \underline{B(x) \rightarrow C(y)} \\ (A \circ (B \times C))(y) \quad \forall x \in U, \forall y \in V, A \subset U, B \subset U, C \subset V \end{array}$$

siendo

$$\mu_{A \circ (B \times C)}(y) = \max_{x \in U} [\min(\mu_A(x), \mu_{B \times C}(x, y))] = \max_{x \in U} [\min(\mu_A(x), \min(\mu_B(x), \mu_C(y)))]$$

(Si “a” es el número de elementos del universo donde está definido el conjunto A y “b” es lo mismo, pero referente al conjunto B , la matriz correspondiente a $A \times B$ tendrá “a” filas y “b” columnas).

d) Modus Tollens generalizado:

$$\begin{array}{l} D(y) \\ \underline{B(x) \rightarrow C(y)} \\ (D \circ (C \times B))(x) \quad \forall x \in U, \forall y \in V, D \subset V, B \subset U, C \subset V \end{array}$$

(Aunque aquí se ha incluido esta definición del *Modus Tollens* generalizado, hoy en día existe gran controversia sobre cómo debería ser definida esta regla. No existe un consenso general a tal respecto.)

e) Silogismo hipotético:

$$\begin{array}{l} A(x) \rightarrow B(y) \\ \underline{B(y) \rightarrow C(z)} \\ ((A \times B) \circ (B \times C))(x, z) \quad \forall x \in U, \forall y \in V, \forall z \in W, A \subset U, B \subset V, C \subset W \end{array}$$

PROBLEMAS RELACIONADOS: 3.15 a 3.22

3.2.5 Lógicas no monótonas

Una de las principales características de la lógica clásica es su carácter *monótono*. Es decir, dado un conjunto de sentencias S_1 del que se puede inferir la sentencia C , al

añadir a S_1 otro conjunto de sentencias S_2 , se tiene que poder seguir infiriendo C a partir de $S_1 \cup S_2$. El que la lógica clásica sea monótona supone un inconveniente a la hora de abordar con este formalismo gran cantidad de problemas que se presentan en inteligencia artificial y que tienen carácter no monótono. Un ejemplo es el *razonamiento de sentido común* empleado por el ser humano, caracterizado por establecer conclusiones a partir de información parcial, que muchas veces tienen que ser revisadas o desecharadas una vez que se obtiene nueva información o evidencia del dominio. Por ejemplo, si alguien nos dice que tiene un pájaro, nosotros enseguida pensaremos que ese pájaro vuela. Si posteriormente se nos dice que el pájaro es en realidad un pingüino, tendremos que dejar de pensar que puede volar y que seguramente su amo lo tenía encerrado en una jaula.

A lo largo de los últimos años han aparecido diversos formalismos cuyo objetivo es intentar servir de base al razonamiento no monótono. Entre otros podemos citar los siguientes:

- La *lógica por defecto* de Reiter.
- Los mecanismos de *circunscriptión* de McCarthy.
- La *lógica modal no monótona* de McDermott y Doyle.
- La *lógica autoepistémica* de Moore.
- Las *redes de herencia* de Touretzky.

PROBLEMAS RELACIONADOS: 3.23 a 3.26

3.3 Problemas resueltos

• PROBLEMA 3.1: (*)

Codificar en lógica de primer orden la base de conocimientos de un sumador binario de 2 bits como el que aparece en la figura 3-1.

Verificar que el sumador funciona correctamente indicando la aplicación de las reglas utilizadas en cada paso y los elementos contenidos en la base de hechos (base de afirmaciones) a lo largo del proceso, partiendo de los números siguientes:

$$\{Y_0 = 1, Y_1 = 0\}$$

$$\{X_0 = 0, X_1 = 1\}$$

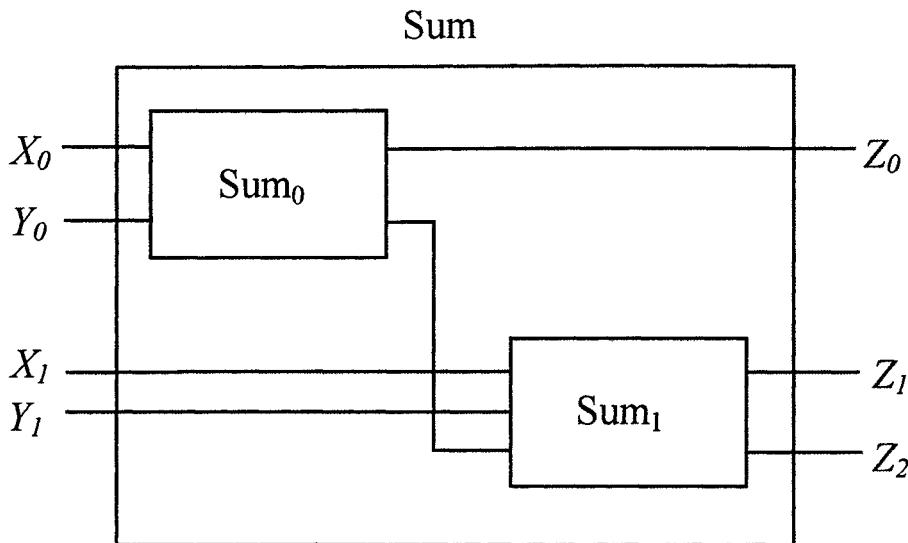


Figura 3-1

SOLUCIÓN:

¿Qué pretende este problema? La labor que hay que realizar consiste en representar parte del conocimiento correspondiente al dominio de circuitos digitales mediante lógica de primer orden.

a) Primero habrá que representar cuáles son los componentes que aparecen en el sumador de 2 bits y cómo están interconectados. En total hay 3 componentes: el propio sumador de 2 bits, un sumador de 1 bit y un semisumador.

1. sumador2bit(Sum)
2. semisumador(Sum₀)
3. sumador1bit(Sum₁)

Por otra parte, existen 8 conexiones entre los 3 componentes anteriores:

4. conexión(e(1, Sum), e(1, Sum₀))
5. conexión(e(2, Sum), e(2, Sum₀))
6. conexión(e(3, Sum), e(1, Sum₁))
7. conexión(e(4, Sum), e(2, Sum₁))
8. conexión(s(1, Sum), s(1, Sum₀))

9. conexión(s(2, Sum), s(1, Sum₁))
10. conexión(s(3, Sum), s(2, Sum₁))
11. conexión(s(2, Sum₀), e(3, Sum₁))

donde, por ejemplo, “conexión(s(2, Sum₀), e(3, Sum₁))” significa que la salida segunda del semisumador Sum₀ y la entrada tercera del sumador Sum₁ están conectadas.

Inicialmente la base de hechos consta de los siguientes datos (entradas al sumador):

12. e(2, Sum)
13. e(3, Sum)

Al no estar e(1, Sum) y e(4, Sum) en la base de hechos, se supone que su valor lógico es 0 (axioma del mundo cerrado).

La descripción del funcionamiento de los componentes del circuito sería, teniendo en cuenta la siguiente tabla para un sumador de 1 bit:

a	b	c _i	s	c _{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

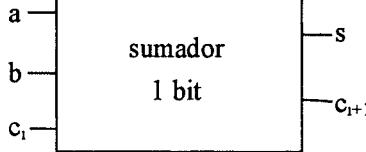


Figura 3-2

- Sumador:

$$s = a \oplus b \oplus c_i$$

$$c_{i+1} = (a + b) \cdot c_i + a \cdot b$$

- Semisumador (c_i=0):

$$s = a \oplus b$$

$$c = a \cdot b$$

que en simbolismo lógico quedaría del siguiente modo:

14. $\forall x, y (((x \wedge \neg y) \vee (\neg x \wedge y)) \Rightarrow \text{xor}(x, y))$
15. $\forall x ((\text{semisumador}(x) \wedge e(1, x) \wedge e(2, x)) \Rightarrow s(2, x))$
16. $\forall x ((\text{semisumador}(x) \wedge \text{xor}(e(1, x), e(2, x))) \Rightarrow s(1, x))$
17. $\forall x (\text{sumador1bit}(x) \wedge \text{xor}(e(1, x), \text{xor}(e(2, x), e(3, x))) \Rightarrow s(1, x))$
18. $\forall x (\text{sumador1bit}(x) \wedge (((e(1, x) \vee e(2, x)) \wedge e(3, x)) \vee (e(1, x) \wedge e(2, x))) \Rightarrow s(2, x))$

Finalmente, la propagación de los valores lógicos a través de las conexiones queda regida por la siguiente regla:

19. $\forall x, y (\text{conexión}(x, y) \wedge x) \Rightarrow y$

b) Como lo que realmente interesa es conocer los valores lógicos que van apareciendo en las conexiones a lo largo del proceso de inferencia, se va a hacer referencia únicamente a estos valores en la base de hechos. En cada paso se mencionará qué reglas pueden ser aplicadas y cuáles son los nuevos hechos que van a ser inferidos. El proceso completo de inferencia se resume en la figura 3-3:

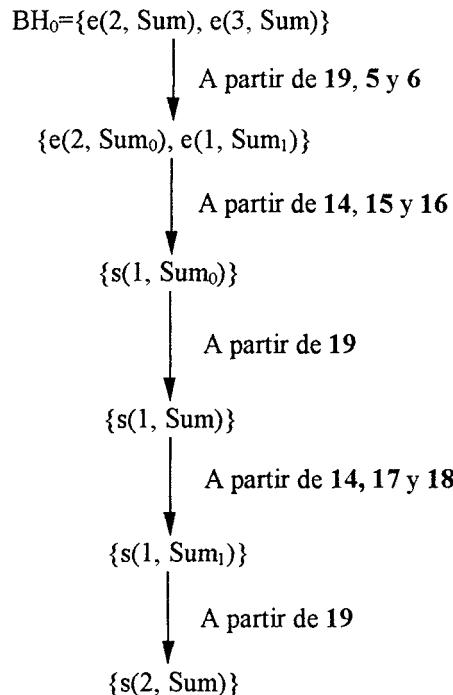


Figura 3-3

Por tanto, el sumador de 2 bits tendrá finalmente los siguientes valores en su salida:

$$Z_0 = 1, Z_1 = 1, Z_2 = 0$$

• **PROBLEMA 3.2:** (adaptado de [Norvig, 1992])

Describir brevemente los problemas que presenta el cálculo de predicados con respecto a las siguientes cuestiones: decidibilidad, tratabilidad, incertidumbre, monotonía, consistencia y expresividad.

SOLUCIÓN:

¿Qué pretende este problema? La lógica de predicados es un formalismo de representación que no resulta adecuado para el tratamiento de ciertas tareas de las que se ocupa la inteligencia artificial. Algunas características de esta deficiencia tienen que ver con los aspectos mencionados en el enunciado.

1) Decidibilidad

Dado un conjunto de axiomas y un objetivo, puede ocurrir en lógica de predicados que ni el objetivo ni su negación puedan ser obtenidos a partir de los axiomas. En estos casos se dice que el formalismo es semidecidible o indecidible.

2) Tratabilidad

Incluso cuando un objetivo puede ser probado, puede llevar demasiado tiempo encontrar la prueba usando los mecanismos de inferencia disponibles.

3) Incertidumbre

Puede ser conveniente tratar con relaciones que son probables hasta un cierto grado, pero que no se sabe si son con certeza ciertas o falsas.

Este concepto adquiere especial relevancia dentro de la inteligencia artificial, pues muchos de los problemas que aborda requieren la formalización de la incertidumbre intrínseca que aparece en los mismos. Como ejemplos pueden citarse: problemas de diagnóstico en determinados campos de la Medicina, problemas de comprensión de cierto tipo de imágenes, etc.

4) Monotonía

En el cálculo de predicados puro, una vez que se prueba un teorema, es cierto para siempre. Pero sería conveniente disponer de una manera de obtener teoremas provisionales que se basen en suposiciones y ser capaces de retirarlos cuando se comprueba que aquellas suposiciones son falsas.

5) Consistencia

El cálculo de predicados puro no admite contradicciones. Si por accidente se obtiene P y $\neg P$, entonces cualquier teorema puede ser probado. En efecto, una simple contradicción corrompe toda la base de afirmaciones.

6) Expresividad

El cálculo de predicados de primer orden hace poco elegante el tratamiento de ciertos dominios tales como las relaciones y proposiciones del mismo lenguaje natural.

• PROBLEMA 3.3: (adaptado de [Jackson *et al.*, 1989])

Supóngase que se desea construir un sistema basado en lógica con la capacidad de utilizar *razonamiento temporal*. Enumerar aquellos requisitos que debería reunir esta lógica temporal con la que se quiere dotar al sistema para ser lo más eficaz posible.

SOLUCIÓN:

1) Como toda lógica temporal se pretende que sea finalmente un lenguaje de representación del conocimiento, debería ser posible diseñar un motor de inferencia que realizará de una manera eficiente aquellas inferencias permitidas por dicha lógica.

2) Un segundo requisito es que el lenguaje de la lógica sea lo suficientemente expresivo a la hora de poder representar el conocimiento temporal.

Obviamente, estos dos requisitos no son independientes el uno del otro. Los lenguajes de representación más expresivos son más difíciles de implementar eficientemente y los lenguajes para los cuales existe una implementación eficiente no son a menudo lo suficientemente expresivos. Por ello debe buscarse una solución de compromiso entre ambos factores.

3) El conocimiento de carácter temporal puede ser preciso o impreciso. Como ejemplo de imprecisión puede citarse el hecho de que podría saberse que un determinado acontecimiento ocurrirá en el futuro, pero no en qué preciso instante. La representación final deberá permitir este tipo de imprecisión, así como el caso opuesto en que se conoce exactamente la coordenada temporal en que un hecho tiene lugar.

4) Otro requisito conveniente para una lógica temporal es la posibilidad de combinarla con cualquier otro tipo de lógica (por ejemplo, lógica modal). Al fin y al cabo, se podría querer usar una lógica temporal en una aplicación en la que se pretende razonar sobre la necesidad y posibilidad de ciertas proposiciones, como es el caso de la lógica modal.

5) Una lógica temporal adecuada debería tener en cuenta los cambios que se producen en el mundo en relación a: las propiedades de cualquier elemento del mismo, las relaciones entre los diferentes elementos, la existencia de dichos elementos... Algunos autores argumentan que un requisito adicional para una representación temporal adecuada sería que fuera capaz de abordar la cuestión de la *persistencia*, es decir, debería tener en cuenta el hecho de que la mayoría de las cosas siguen siendo verdaderas a medida que el tiempo pasa. El problema de la persistencia es idéntico al *problema del contexto* (*frame problem* en inglés). El problema del contexto consiste en decidir qué condiciones cambian y qué condiciones no cambian cuando un sistema experimenta alguna modificación.

• PROBLEMA 3.4: (adaptado de [Gensler, 1990])

Dar una prueba del siguiente razonamiento:

$$\begin{aligned} a &= b \\ \therefore b &= a \end{aligned}$$

SOLUCIÓN:

¿Qué pretende este problema? Desde este problema hasta el 3.10 se presentan varias demostraciones por reducción al absurdo de razonamientos donde se ha incluido el predicado identidad. Lo que se pretende es poner en práctica las leyes que permiten ampliar la lógica de predicados tradicional a la lógica de predicados con identidad.

1. $a = b$
- $\therefore b = a$
2. | supóngase: $b \neq a$
3. | $\therefore b \neq b$ (a partir de 1, 2 y la *regla b*)
4. | $\therefore b = b$ (a partir de la *regla a*), contradiciendo a 3)
5. $\therefore b = a$ (a partir de 2, 3 y 4)

Partiendo de la premisa “1” se pretende demostrar que “ $b=a$ ”, para lo cual se supone cierta la condición contraria en “2” y se comprueba que lleva a una contradicción.

• PROBLEMA 3.5: (adaptado de [Gensler, 1990])

Probar:

$$\begin{aligned} Fa \\ \therefore (\neg Fb \supset b \neq a) \end{aligned}$$

SOLUCIÓN:

$$1. \quad F_a$$

$$[\therefore (\neg F_b \supset b \neq a)]$$

$$2. \quad \left| \begin{array}{l} \text{supóngase: } \neg(\neg F_b \supset b \neq a) \\ \therefore \neg F_b \text{ (a partir de 2)} \end{array} \right.$$

$$4. \quad \left| \begin{array}{l} \therefore b = a \text{ (a partir de 2)} \\ \therefore F_b \text{ (a partir de 1 y 4, contradiciendo a 3)} \end{array} \right.$$

$$6. \quad \therefore (\neg F_b \supset b \neq a) \text{ (a partir de 2, 3 y 5)}$$

• **PROBLEMA 3.6:** (adaptado de [Gensler, 1990])

Dar una prueba para:

$$a = b$$

$$b = c$$

$$\therefore a = c$$

SOLUCIÓN:

$$1. \quad a = b$$

$$2. \quad b = c$$

$$[\therefore a = c]$$

$$3. \quad \left| \begin{array}{l} \text{supóngase: } a \neq c \\ \therefore a \neq b \text{ (a partir de 2 y 3, contradiciendo a 1)} \end{array} \right.$$

$$5. \quad \therefore a = c \text{ (a partir de 1, 3 y 4)}$$

• **PROBLEMA 3.7:** (adaptado de [Gensler, 1990])

Dar una prueba para:

$$a = b$$

$$\forall x \ (Fx \supset Gx)$$

$$\neg Ga$$

$$\therefore \neg Fb$$

SOLUCIÓN:

$$1. \quad a = b$$

2. $\forall x (Fx \supset Gx)$
3. $\neg Ga$
- [$\therefore \neg Fb$
4. supóngase: Fb
5. $\therefore (Fb \supset Gb)$ (a partir de 2)
6. $\therefore Gb$ (a partir de 4 y 5)
7. $\therefore Ga$ (a partir de 1 y 6, contradiciendo a 3)
8. $\therefore \neg Fb$ (a partir de 4, 3 y 7)

• PROBLEMA 3.8: (adaptado de [Gensler, 1990])

Demostrar el siguiente razonamiento lógico:

“Yo peso 70 kilogramos.”

“Mi cerebro no pesa 70 kilogramos.”

Por tanto, “Yo no soy idéntico a mi cerebro.”

SOLUCIÓN:

Estableciendo la siguiente representación:

$P \equiv$ “pesar 70 kilogramos”

$i \equiv$ “yo”

$c \equiv$ “mi cerebro”

la demostración se llevará a cabo por refutación (el establecimiento de la negación de la conclusión, junto con las premisas dadas, se comprobará que lleva a una contradicción):

1. Pi
2. $\neg Pc$
- [$\therefore i \neq c$
3. supóngase: $i = c$
4. $\therefore Pc$ (a partir de 1,3 y la regla b), contradiciendo a 2)

5. $\therefore i \neq c$ (a partir de 3, 2 y 4¹)

• PROBLEMA 3.9: (adaptado de [Gensler, 1990])

Dar una prueba para el siguiente razonamiento:

“Algunas personas son listas.”

“Algunas personas no son listas.”

\therefore “Hay más de una persona.”

SOLUCIÓN:

Representando

$L \equiv$ “ser listo”

se tiene:

1. $\exists x Lx$

2. $\exists x \neg Lx$

$[\therefore \exists x \exists y x \neq y$

3. supóngase: $\neg \exists x \exists y x \neq y$

4. $\therefore \forall x \neg \exists y x \neq y$ (a partir de 3)

5. $\therefore La$ (a partir de 1)

6. $\therefore \neg Lb$ (a partir de 2)

7. $\therefore \neg \exists y a \neq y$ (a partir de 4)

8. $\therefore \forall y a = y$ (a partir de 7)

9. $\therefore a = b$ (a partir de 8)

10. $\therefore Lb$ (a partir de 5, 9 y la *regla b*), contradiciendo a 6)

11. $\therefore \exists x \exists y x \neq y$ (a partir de 3, 6 y 10)

• PROBLEMA 3.10: (adaptado de [Gensler, 1990])

Probar:

“Exactamente una persona vive en Navarra.”

¹ Aquellas reglas de inferencia que pertenecen a la lógica de predicados sin identidad no se están mencionando explícitamente.

“Pablo vive en Navarra.”

“Pablo es fontanero.”

∴ “Todo el mundo que vive en Navarra es fontanero.”

SOLUCIÓN:

Se utilizará la siguiente representación:

$N \equiv$ “vivir en Navarra”

$p \equiv$ “Pablo”

$F \equiv$ “ser fontanero”

de modo que:

1. $\exists x (Nx \wedge \neg \exists y (y \neq x \wedge Ny))$
2. Np
3. Fp
4. supóngase: $\neg \forall x (Nx \supset Fx)$
5. $\therefore \exists x \neg(Nx \supset Fx)$ (a partir de 4)
6. $\therefore \neg(Na \supset Fa)$ (a partir de 5)
7. $\therefore Na$ (a partir de 6)
8. $\therefore \neg Fa$ (a partir de 6)
9. $\therefore (Nb \wedge \neg \exists y (y \neq b \wedge Ny))$ (a partir de 1)
10. $\therefore Nb$ (a partir de 9)
11. $\therefore \neg \exists y (y \neq b \wedge Ny)$ (a partir de 9)
12. $\therefore \forall y \neg(y \neq b \wedge Ny)$ (a partir de 11)
13. $\therefore \neg(a \neq b \wedge Na)$ (a partir de 12)
14. $\therefore \neg(p \neq b \wedge Np)$ (a partir de 12)
15. $\therefore a = b$ (a partir de 7 y 13)
16. $\therefore p = b$ (a partir de 2 y 14)

17. | ∴ Fb (a partir de 3,16 y la *regla b*)
 18. | ∴ Fa (a partir de 15,17 y la *regla b*), contradiciendo a 8)
 19. ∴ $\forall x (Nx \supset Fx)$ (a partir de 4, 8 y 18)

• **PROBLEMA 3.11:** (adaptado de [Gensler, 1990])

Dado el siguiente razonamiento:

“Necesariamente, si Juan es español, es europeo.”

“Es posible que Juan no sea europeo.”

Por tanto, “Es posible que Juan no sea español.”

dar una prueba del mismo.

SOLUCIÓN:

¿Qué pretende este problema? Desde el presente problema hasta el 3.14 aparecen varias demostraciones por reducción al absurdo de razonamientos donde figuran los operadores de necesidad y posibilidad. En tales demostraciones se hace uso constante de las leyes que permiten ampliar la lógica de predicados tradicional a la lógica modal.

Considérese:

$$A \equiv \text{“Juan es español.”}$$

$$B \equiv \text{“Juan es europeo.”}$$

El problema, entonces, quedaría reducido a probar:

$$\begin{aligned} & \Box(A \supset B) \\ & \Diamond\neg B \\ & \therefore \Diamond\neg A \end{aligned}$$

La prueba pedida sería:

1. $\Box(A \supset B)$
 2. $\Diamond\neg B$
 [∴ $\Diamond\neg A$
 3. | supóngase: $\neg\Diamond\neg A$
 4. | ∴ $\Box A$ (a partir de 3 y de la *regla 1*)
 5. | $W \therefore \neg B$ (a partir de 2 y de la *regla 4*)

6. $W \therefore (A \supset B)$ (a partir de 1 y de la *regla 3*)
7. $W \therefore A$ (a partir de 4 y de la *regla 3*)
8. $W \therefore B$ (a partir de 6 y 7, contradiciendo a 5)
9. $\therefore \Diamond \neg A$ (a partir de 3, 5 y 8)

• **PROBLEMA 3.12:** (adaptado de [Gensler, 1990])

Demostrar que el siguiente razonamiento es válido:

“Necesariamente, si no como, moriré.”

“Es posible que no coma.”

“Por tanto, es posible que muera.”

SOLUCIÓN:

Siendo:

$$A \equiv \text{“no como”}$$

$$B \equiv \text{“moriré”}$$

el problema consiste en demostrar

1. $\Box(A \supset B)$
2. $\Diamond A$
3. $\therefore \Diamond B$

Podría construirse la siguiente prueba:

1. $\Box(A \supset B)$
2. $\Diamond A$
3. $\therefore \Diamond B$
4. supóngase: $\neg \Diamond B$
5. $W \therefore A$ (a partir de 2 y de la *regla 4*)
6. $W \therefore (A \supset B)$ (a partir de 1 y de la *regla 3*)
7. $W \therefore \neg B$ (a partir de 4 y de la *regla 3*)
8. $W \therefore B$ (a partir de 5 y 6, contradiciendo a 7)

9. $\therefore \Diamond B$ (a partir de 3, 7 y 8)

• **PROBLEMA 3.13:** (adaptado de [Gensler, 1990])

Dar una prueba formal del siguiente razonamiento:

“No es posible que Ana vaya a la fiesta y que Pedro no vaya.”

“Por tanto, necesariamente, si Ana va a la fiesta, Pedro también.”

SOLUCIÓN:

Sea

$$A \equiv \text{“Ana va a la fiesta”}$$

$$B \equiv \text{“Pedro va a la fiesta”}$$

Entonces, el problema consistiría en probar

$$\neg\Diamond(A \wedge \neg B)$$

$$\therefore \Box(A \supset B)$$

Considérese la siguiente prueba:

$$1. \quad \neg\Diamond(A \wedge \neg B)$$

$$[\therefore \Box(A \supset B)]$$

$$2. \quad \text{supóngase: } \neg\Box(A \supset B)$$

$$3. \quad \therefore \Diamond\neg(A \supset B) \text{ (a partir de 2 y de la regla 2)}$$

$$4. \quad \therefore \Box\neg(A \wedge \neg B) \text{ (a partir de 1 y de la regla 1)}$$

$$5. \quad W \therefore \neg(A \supset B) \text{ (a partir de 3 y de la regla 4)}$$

$$6. \quad W \therefore \neg(A \wedge \neg B) \text{ (a partir de 4 y de la regla 3)}$$

$$7. \quad W \therefore A \text{ (a partir de 5)}$$

$$8. \quad W \therefore \neg B \text{ (a partir de 5)}$$

$$9. \quad W \therefore B \text{ (a partir de 6 y 7, contradiciendo a 8)}$$

$$10. \quad \therefore \Box(A \supset B) \text{ (a partir de 2, 8 y 9)}$$

• **PROBLEMA 3.14:** (adaptado de [Gensler, 1990])

Dada una lógica modal que formalice los conceptos de *necesidad* y *posibilidad*, ¿cómo podría ser utilizada para abordar el tratamiento del razonamiento temporal?

SOLUCIÓN:

En lógica modal se introducen nuevas modalidades de verdad como son la *necesidad* o la *posibilidad* de un determinado enunciado:

$$\Box A \equiv \text{"es necesario que } A\text{"}$$

$$\Diamond A \equiv \text{"es posible que } A\text{"}$$

Una semántica ampliamente aceptada para esta lógica supone la introducción del concepto de *mundos posibles* (Kripke, 1963). De esta manera:

$$\Box A \equiv \text{"A es verdad en todos los mundos posibles"}$$

$$A \equiv \text{"A es verdad en el } mundo \ real\text{"}$$

$$\Diamond A \equiv \text{"A es verdad en algunos mundos posibles"}$$

Un *mundo posible* es una descripción completa y consistente de cómo las cosas podrían ser realmente. En cada posible mundo se describirán una serie de situaciones que son posibles todas juntas, es decir, pueden o no ser verdad. El *mundo real* contiene una descripción completa de cómo las cosas son en realidad.

Un *enunciado necesario* es aquél que es verdadero en todos los posibles mundos. Un *enunciado verdadero* sería el que es cierto en el mundo real. Finalmente, un *enunciado posible* es verdadero en un determinado mundo y puede o no ser verdadero en el mundo real.

Cuando son aplicados a lógica temporal, los *posibles mundos* son equiparados con puntos o instantes temporales y los operadores modales son por lo general alguna variante de los siguientes:

$$Fp: p \text{ es verdadero en algún punto del futuro}$$

$$Gp: p \text{ es verdadero en todos los puntos del futuro}$$

$$Pp: p \text{ es verdadero en algún punto del pasado}$$

$$Hp: p \text{ es verdadero en todos los puntos del pasado}$$

La característica destacada de estos sistemas es que las fórmulas se interpretan con respecto a un punto temporal (generalmente llamado *ahora*) y pueden contener referencias a otros puntos temporales a través del uso de los operadores modales. Así, si p significara “Está lloviendo.”, la fórmula $\neg p \supset G\neg p$ significaría: “Si no está lloviendo ahora, entonces nunca lo hará.”. Se podría, por tanto, construir la siguiente definición: $\Box p \equiv Hp \wedge p \wedge Gp$.

• PROBLEMA 3.15:

Sean los conjuntos borrosos F y G con las funciones de pertenencia que aparecen en la figura, definidas sobre un universo U .

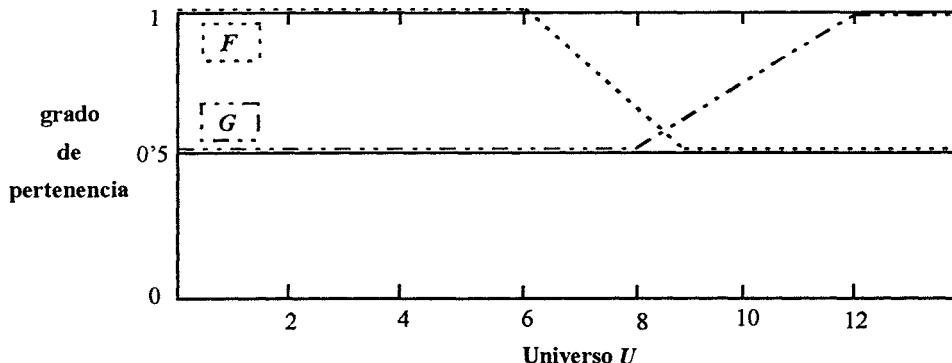


Figura 3-4

- Representar la función de pertenencia correspondiente a $F \cup G$.
- Comprobar gráficamente que el conjunto F no cumple la ley de contradicción.
- Comprobar gráficamente que $c(F \cap G) = c(F) \cup c(G)$.

SOLUCIÓN:

¿Qué pretende este problema? En este problema se revisan diversas operaciones sobre conjuntos difusos definidos a partir de sus funciones de pertenencia.

a)

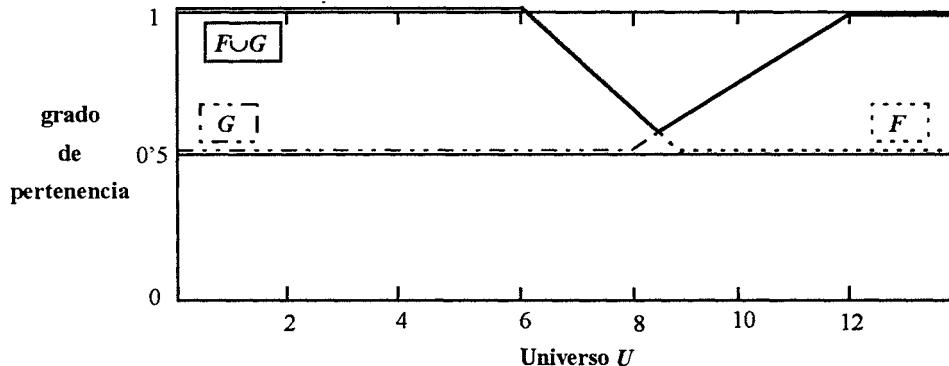


Figura 3-5

La función de pertenencia asociada a $F \cup G$ es:

$$\mu_{F \cup G}(x) = \max(\mu_F(x), \mu_G(x)) \quad \forall x \in U$$

con lo que se tendrá la gráfica que aparece en la figura 3-5.

b) Dados dos conjuntos borrosos A y B definidos sobre un universo U , la función de pertenencia asociada al conjunto intersección de A y B es:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \quad \forall x \in U$$

Por tanto, a $F \cap c(F)$ le corresponderá la siguiente gráfica:

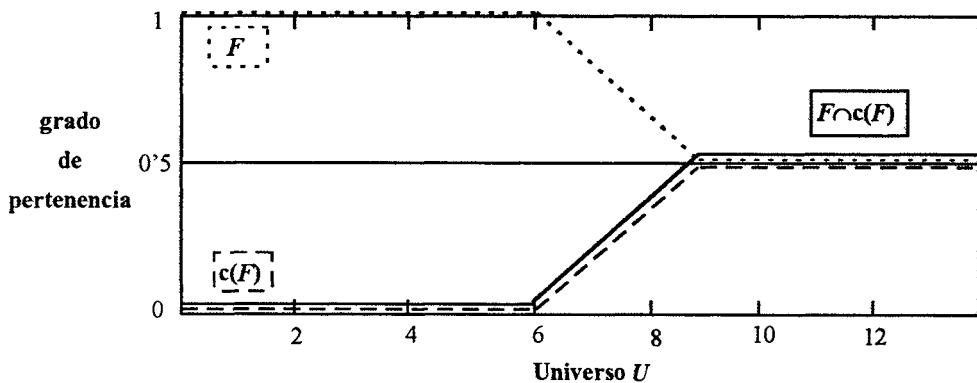


Figura 3-6

Como se puede comprobar a partir de la gráfica, $F \cap c(F) \neq \emptyset$, ya que hay elementos de este conjunto que tienen un grado de pertenencia al mismo mayor que cero. Por tanto, en el caso de conjuntos borrosos no se cumple en general la ley de contradicción.

c) Dados dos conjuntos borrosos A y B definidos sobre un universo U , la función de pertenencia asociada al conjunto unión de A y B es:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad \forall x \in U$$

En la figura 3-7 aparece la gráfica correspondiente a $c(F \cup G)$.

Por otra parte, a $c(F) \cap c(G)$ le correspondería una función de pertenencia que se representa en la figura 3-8.

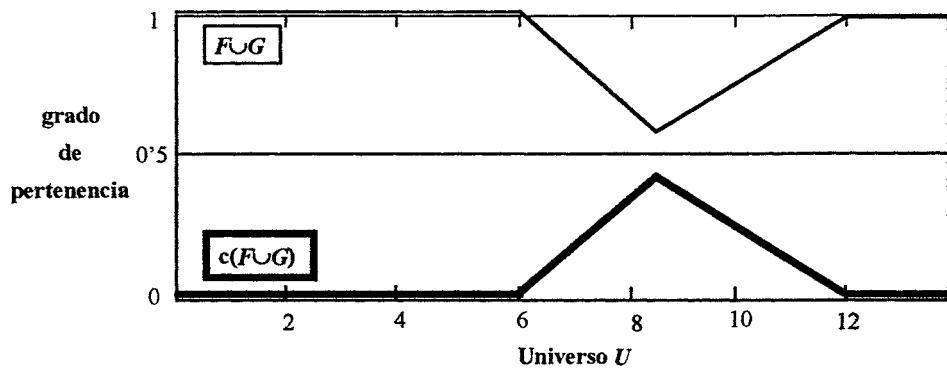


Figura 3-7

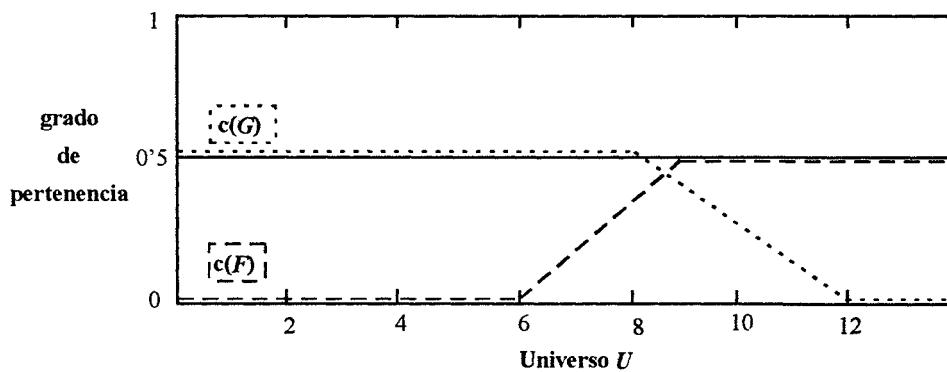


Figura 3-8

Por tanto, finalmente se obtiene:

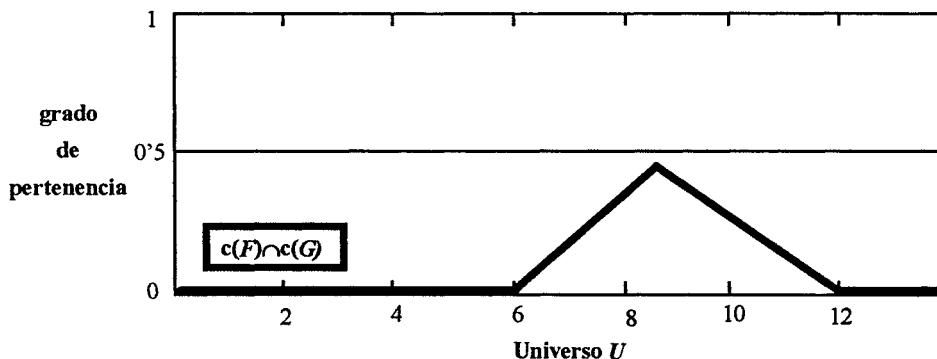


Figura 3-9

Como se puede comprobar a partir de las gráficas, se cumple la ley de De Morgan en este caso.

¿Qué dificultades puede haber encontrado en este problema?

Aplicación 1 de la regla A:

El lector no entiende los términos: “conjunto borroso”, “función de pertenencia”, “universo” y “unión, intersección o complemento de conjuntos borrosos”.

Acciones:

Leer el apartado teórico de lógica difusa que se extiende hasta la sección “modificadores lingüísticos”.

Aplicación 1 de la regla C:

El lector no es capaz de trabajar con las gráficas dibujadas, debido a que no está acostumbrado a manejar grados de pertenencia diferentes de 0 o 1.

Acciones:

Intentar hacer el mismo ejercicio cuando F y G son conjuntos no borrosos, es decir, cuando las gráficas correspondientes a los grados de pertenencia sólo pueden ser líneas horizontales en los valores 0 o 1.

• PROBLEMA 3.16:

Comprobar analíticamente que los conjuntos difusos violan la *ley de contradicción* y cumplen la de *absorción*.

SOLUCIÓN:

a) En el caso de la ley de contradicción basta con verificar que la ecuación

$$\min[A(x), 1-A(x)]=0$$

es violada por al menos un $x \in X$ (X : universo de discurso donde está definido el conjunto difuso A). Pero esto es obvio, ya que la ecuación no se cumple para cualquier valor $A(x)$ perteneciente al intervalo $[0, 1]$, satisfaciéndose sólo cuando $A(x)$ es 0 o 1. Por tanto, la ley de contradicción sólo se cumple para conjuntos no borrosos.

b) Verificar la ley de absorción:

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

requiere demostrar que la ecuación

$$\max[A(x), \min[A(x), B(x)]] = A(x)$$

se satisface para todo $x \in X$. Han de considerarse dos casos: que $A(x) \leq B(x)$ o que $A(x) > B(x)$. En el primer caso se obtiene:

$$\max[A(x), A(x)] = A(x)$$

lo cual es siempre cierto, mientras que en el segundo se tiene:

$$\max[A(x), B(x)] = A(x)$$

que también se cumple al haberse supuesto en este caso que $A(x) > B(x)$.

• PROBLEMA 3.17:

Una relación binaria difusa $R(X, X)$ se dice que es:

- **Reflexiva:**

$$\text{si } R(x, x) = 1 \quad \forall x \in X$$

- **Simétrica:**

$$\text{si } R(x, y) = R(y, x) \quad \forall x, y \in X$$

- **Transitiva** (o, más específicamente, max-min transitiva):

$$\text{si } R(x, z) \geq \max_{y \in X} \min[R(x, y), R(y, z)] \text{ se satisface para todo par } \langle x, z \rangle \in X^2$$

Entre las relaciones binarias difusas siguientes, indicar las que son:

a) simétricas

b) reflexivas

c) transitivas

$$R_1 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0'9 & 0'7 & 0'3 \\ 0 & 0 & 0 & 0'7 & 0'3 \\ 0 & 0 & 0 & 0 & 0'3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$R_2 = \begin{pmatrix} 0 & 0'3 & 1 & 0 & 0'5 \\ 0'3 & 0'3 & 0 & 0'8 & 0'1 \\ 1 & 0 & 0 & 0'2 & 1 \\ 0 & 0'8 & 0'2 & 1 & 0'4 \\ 0'5 & 0'1 & 1 & 0'4 & 0'4 \end{pmatrix}$$

$$R_3 = \begin{pmatrix} 1 & 0'5 & 0'5 & 0 & 0'7 \\ 0 & 1 & 0'7 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0'3 & 0'3 & 1 & 0 \\ 1 & 0'5 & 0'5 & 0 & 1 \end{pmatrix}$$

$$R_4 = \begin{pmatrix} 0 & 0 & 0'3 & 0'2 & 0 \\ 0'6 & 1 & 0'8 & 1 & 0'2 \\ 0'2 & 0 & 1 & 0'8 & 0'3 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0'2 & 0'6 & 0 \end{pmatrix}$$

$$R_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0'4 \\ 0 & 1 & 0'9 & 1 & 0 \\ 0 & 0'9 & 1 & 0'9 & 0 \\ 0 & 1 & 0'9 & 1 & 0 \\ 0'4 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_6 = \begin{pmatrix} 0 & 0'1 & 0 & 1 & 0'8 \\ 0'1 & 0 & 0 & 0 & 0'6 \\ 0 & 0 & 0 & 0'4 & 1 \\ 1 & 0 & 0'4 & 0 & 1 \\ 0'8 & 0'6 & 1 & 1 & 0 \end{pmatrix}$$

SOLUCIÓN:

¿Qué pretende este problema? A partir de aquí y hasta el problema 3.19 se revisan diferentes conceptos relacionados con las relaciones difusas. Una correcta comprensión de estos apartados permitirá una mayor facilidad a la hora de asimilar las nociones correspondientes a la inferencia difusa.

A partir de las definiciones dadas se obtienen los siguientes resultados:

	R_1	R_2	R_3	R_4	R_5	R_6
Simetría		X			X	X
Reflexividad			X		X	
Transitividad	X		X		X	

• PROBLEMA 3.18:

Realizar las siguientes operaciones correspondientes a composiciones max-min entre relaciones difusas:

a) $\begin{bmatrix} 0'3 & 0'5 & 0'8 \\ 0 & 0'7 & 1 \\ 0'4 & 0'6 & 0'5 \end{bmatrix} \circ \begin{bmatrix} 0'9 & 0'5 & 0'7 & 0'7 \\ 0'3 & 0'2 & 0 & 0'9 \\ 1 & 0 & 0'5 & 0'5 \end{bmatrix}$

b) $\begin{bmatrix} 0'1 & 0'2 & 0 & 1 & 0'7 \\ 0'3 & 0'5 & 0 & 0'2 & 1 \\ 0'8 & 0 & 1 & 0'4 & 0'3 \end{bmatrix} \circ \begin{bmatrix} 0'9 & 0 & 0'3 & 0'4 \\ 0'2 & 1 & 0'8 & 0 \\ 0'8 & 0 & 0'7 & 1 \\ 0'4 & 0'2 & 0'3 & 0 \\ 0 & 1 & 0 & 0'8 \end{bmatrix}$

SOLUCIÓN:

a) Llamando p_{ij} , q_{ij} y r_{ij} a los elementos correspondientes a la fila “i” y columna “j” de las matrices que intervienen en la operación, se quiere calcular:

$$[r_{ij}] = [p_{ik}] \circ [q_{kj}]$$

donde

$$r_{ij} = \max_k \min(p_{ik}, q_{kj})$$

Aplicando la expresión anterior, se tendría por ejemplo:

$$r_{11} = \max[\min(p_{11}, q_{11}), \min(p_{12}, q_{21}), \min(p_{13}, q_{31})] = \max[\min(0'3, 0'9), \min(0'5, 0'3), \min(0'8, 1)] = 0'8$$

$$r_{32} = \max[\min(p_{31}, q_{12}), \min(p_{32}, q_{22}), \min(p_{33}, q_{32})] = \max[\min(0'4, 0'5), \min(0'6, 0'2), \min(0'5, 0)] = 0'4$$

El resultado final sería:

$$\begin{bmatrix} 0'8 & 0'3 & 0'5 & 0'5 \\ 1 & 0'2 & 0'5 & 0'7 \\ 0'5 & 0'4 & 0'5 & 0'6 \end{bmatrix}$$

b) La matriz solución en este caso es:

$$\begin{bmatrix} 0'4 & 0'7 & 0'3 & 0'7 \\ 0'3 & 1 & 0'5 & 0'8 \\ 0'8 & 0'3 & 0'7 & 1 \end{bmatrix}$$

• PROBLEMA 3.19:

La composición max-min cumple la propiedad transitiva:

$$(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$$

Comprobar la propiedad anterior para las siguientes relaciones difusas:

$$R_1 = \begin{bmatrix} 0'1 & 0'2 & 0'1 \\ 0'3 & 0'4 & 1 \\ 0'7 & 0'8 & 0 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0'1 & 0'2 \\ 0'4 & 0'5 & 0'6 \end{bmatrix}$$

$$R_3 = \begin{bmatrix} 0'2 & 0'2 & 0'2 \\ 0'1 & 0 & 0'1 \\ 1 & 0 & 0'6 \end{bmatrix}$$

SOLUCIÓN:

Se tiene:

$$R_1 \circ R_2 = \begin{bmatrix} 0'1 & 0'1 & 0'2 \\ 0'4 & 0'5 & 0'6 \\ 0'7 & 0'7 & 0'2 \end{bmatrix}$$

$$(R_1 \circ R_2) \circ R_3 = \begin{bmatrix} 0'2 & 0'1 & 0'2 \\ 0'6 & 0'2 & 0'6 \\ 0'2 & 0'2 & 0'2 \end{bmatrix}$$

Por otra parte,

$$R_2 \circ R_3 = \begin{bmatrix} 0'2 & 0'2 & 0'2 \\ 0'2 & 0 & 0'2 \\ 0'6 & 0'2 & 0'6 \end{bmatrix}$$

$$R_1 \circ (R_2 \circ R_3) = \begin{bmatrix} 0'2 & 0'1 & 0'2 \\ 0'6 & 0'2 & 0'6 \\ 0'2 & 0'2 & 0'2 \end{bmatrix}$$

Por tanto, $(R_1 \circ R_2) \circ R_3 = R_1 \circ (R_2 \circ R_3)$ como se quería comprobar.

• PROBLEMA 3.20:

Sea U el universo de las posibles cantidades de dinero (en millones de pesetas) en que están valorados los bienes de una persona cualquiera. Sobre este universo se definen los siguientes conjuntos borrosos:

$$R \equiv \text{"ser rico"} = \{0|0, 1|0, 5|0'1, 10|0'2, 50|0'5, 100|0'9, 500|1\}$$

$$A \equiv \text{"ser adinerado"} = \{0|0, 1|0'1, 5|0'5, 10|0'8, 50|1, 100|1, 500|1\}$$

Por otra parte, sea V el universo de los posibles precios (en millones de pesetas) de un coche, en el que se considera el siguiente conjunto borroso:

$$C \equiv \text{"ser un coche caro"} = \{0'5|0, 1|0, 2|0'5, 5|0'9, 10|1\}$$

Comprobar si a partir de las premisas:

“Jorge es rico.”

“Si Jorge es adinerado, se comprará un coche caro.”

se puede concluir que

“Jorge se comprará un coche caro.”

SOLUCIÓN:

¿Qué pretende este problema? En este problema y hasta el 3.22 se aplican a casos concretos los conceptos vistos en el apartado teórico sobre inferencia difusa.

Suponiendo que x representa los “bienes en millones de pesetas de Jorge” e y el “coste en millones de pesetas del coche que Jorge se va a comprar”, habrá que comprobar si el siguiente razonamiento es correcto:

$$\begin{array}{c} R(x) \\ A(x) \rightarrow C(y) \\ C(y) \qquad \forall x \in U, \forall y \in V \end{array}$$

Considerando la regla de *Modus Ponens* generalizado, a partir de las dos premisas del enunciado se puede deducir $(R \circ (A \times C))(y)$. Por tanto, habrá que verificar que $R \circ (A \times C) \subset C$, ya que si esta condición se cumple, el *principio de herencia* garantiza la validez de $C(y)$. Operando sobre A , R y C se obtienen los siguientes resultados:

$A \times C$	0'5	1	2	5	10
0	0	0	0	0	0
1	0	0	0'1	0'1	0'1
5	0	0	0'5	0'5	0'5
10	0	0	0'5	0'8	0'8
50	0	0	0'5	0'9	1
100	0	0	0'5	0'9	1
500	0	0	0'5	0'9	1

De manera que para $R \circ (A \times C)$ se tiene:

$$(0 \ 0 \ 0'1 \ 0'2 \ 0'5 \ 0'9 \ 1) \circ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0'1 & 0'1 & 0'1 \\ 0 & 0 & 0'5 & 0'5 & 0'5 \\ 0 & 0 & 0'5 & 0'8 & 0'8 \\ 0 & 0 & 0'5 & 0'9 & 1 \\ 0 & 0 & 0'5 & 0'9 & 1 \\ 0 & 0 & 0'5 & 0'9 & 1 \end{pmatrix} = (0 \ 0 \ 0'5 \ 0'9 \ 1)$$

Por tanto, el predicado conclusión resultante es el conjunto borroso:

$$R \circ (A \times C) = \{0'5|0, 1|0, 2|0'5, 5|0'9, 10|1\}$$

que coincide con el conjunto C , de manera que la conclusión:

“Jorge se comprará un coche caro.”

es válida.

• **PROBLEMA 3.21:**

Sobre los siguientes universos:

$$H \equiv \text{“primeras horas con sol del día”} = \{6\text{hs.}, 8\text{hs.}, 10\text{hs.}, 12\text{hs.}\}$$

$$G \equiv \text{“temperaturas”} = \{-5^{\circ}\text{C}, 0^{\circ}\text{C}, 5^{\circ}\text{C}, 10^{\circ}\text{C}, 15^{\circ}\text{C}\}$$

se definen los siguientes conjuntos borrosos:

$$T \equiv \text{“hora temprana del día”} = \{6\text{hs.}|1, 8\text{hs.}|0'8, 10\text{hs.}|0'6, 12\text{hs.}|0'3\}$$

$$B \equiv \text{“temperatura baja”} = \{-5^{\circ}\text{C}|1, 0^{\circ}\text{C}|1, 5^{\circ}\text{C}|0'9, 10^{\circ}\text{C}|0'6, 15^{\circ}\text{C}|0'3\}$$

Determinar si a partir de las siguientes premisas:

- 1) Es muy temprano.
- 2) Si es temprano, la temperatura es baja.

se puede concluir:

- a) La temperatura es baja.
- b) La temperatura es bastante baja.

SOLUCIÓN:

Representando por:

t : variable “temperatura”

h : variable “hora del día”

MT : concepto “hora muy temprana del día”

T : concepto “hora temprana del día”

B : concepto “temperatura baja”

las premisas anteriores podrían reescribirse del siguiente modo:

- 1) $MT(h)$
- 2) $T(h) \rightarrow B(t)$

Aplicando la regla del *Modus Ponens* generalizado se obtiene la siguiente conclusión válida a partir de las premisas 1 y 2:

$$3) (MT \circ (T \times B))(t)$$

Los datos referentes a T y B aparecen en el enunciado del problema. El conjunto MT , por su parte, corresponde a aplicar el modificador difuso “muy” al conjunto T . Por tanto, para el conjunto MT que aparece en el enunciado del problema:

$$\begin{aligned} MT &\equiv \text{“hora muy temprana del día”} = \{h \mid \mu_T(h)^2, \forall h \in H\} = \\ &= \{6\text{hs.}|1, 8\text{hs.}|0'64, 10\text{hs.}|0'36, 12\text{hs.}|0'09\} \end{aligned}$$

De esta manera se tendrá:

$$MT \circ (T \times B) = \begin{pmatrix} 1 & 1 & 09 & 06 & 03 \\ 08 & 08 & 08 & 06 & 03 \\ 06 & 06 & 06 & 06 & 03 \\ 03 & 03 & 03 & 03 & 03 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 09 & 06 & 03 \\ 08 & 08 & 08 & 06 & 03 \\ 06 & 06 & 06 & 06 & 03 \\ 03 & 03 & 03 & 03 & 03 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 09 & 06 & 03 \\ 08 & 08 & 08 & 06 & 03 \\ 06 & 06 & 06 & 06 & 03 \\ 03 & 03 & 03 & 03 & 03 \end{pmatrix}$$

Por tanto, la conclusión es el conjunto borroso:

$$F = \{-5^\circ\text{C}|1, 0^\circ\text{C}|1, 5^\circ\text{C}|0'9, 10^\circ\text{C}|0'6, 15^\circ\text{C}|0'3\}$$

En cuanto a la conclusión a) del enunciado del problema, al coincidir F con el conjunto B , la conclusión “la temperatura es baja” es válida. Por otra parte, respecto a la conclusión b), “la temperatura es bastante baja”, no se puede decir que sea válida ya que:

$$\begin{aligned} \text{bastante } B &\equiv \text{“temperatura bastante baja”} = \{t \mid \text{INT}(\text{CON}(\mu_B(t))) \quad \forall t \in G\} = \\ &= \{-5^\circ\text{C}|1, 0^\circ\text{C}|1, 5^\circ\text{C}|0'9278, 10^\circ\text{C}|0'2592, 15^\circ\text{C}|0'0512\} \end{aligned}$$

que no coincide con la conclusión F ni tiene a esta conclusión como subconjunto suyo.

• PROBLEMA 3.22:

Sean las variables X e Y definidas sobre los universos $U_1 = \{x_1, x_2, x_3\}$ y $U_2 = \{y_1, y_2\}$, respectivamente. Dada la proposición:

“Si X es A , entonces Y es B .”

donde

$$A = \{x_1|0'5, x_2|1, x_3|0'6\}$$

$$B = \{y_1|1, y_2|0'4\}$$

Supóngase también la existencia del siguiente hecho:

“ Y es B' ”

donde

$$B' = \{y_1|0'9, y_2|0'7\}$$

Teniendo en cuenta todo lo anterior, ¿qué se puede inferir en relación a la variable X ?

SOLUCIÓN:

Considerando la regla del *Modus Tollens* generalizado se podrá establecer una conclusión del siguiente tipo a partir de las premisas que aparecen en el enunciado:

“ X es A' .”

Lo que habrá que determinar es el grado de pertenencia de x_1 , x_2 y x_3 al conjunto borroso asociado a la propiedad A' . De nuevo, según la regla del *Modus Ponens* generalizado:

$$A' = B' \circ (B \times A) = (0'9 \quad 0'7) \circ \begin{pmatrix} 0'5 & 1 & 0'6 \\ 0'4 & 0'4 & 0'4 \end{pmatrix} = (0'5 \quad 0'9 \quad 0'6)$$

Por tanto, se puede concluir que X es A' con

$$A' = \{x_1|0'5, x_2|0'9, x_3|0'6\}$$

- **PROBLEMA 3.23:** (adaptado de [Brewka, 1991])

El lenguaje Prolog se distingue de la lógica de primer orden en el tratamiento de la negación. $\text{NOT}(P)$ es cierto en Prolog siempre que P no pueda ser inferido (*axioma del mundo cerrado*). Comprobar, teniendo en cuenta la característica anterior, que Prolog es no monótono a partir del siguiente ejemplo:

Primera representación

- (1) VUELA(x) \leftarrow PÁJARO(x) \wedge not PÁJARO-EXCEPCIONAL(x)
- (2) PÁJARO-EXCEPCIONAL(x) \leftarrow PINGÜINO(x)
- (3) PÁJARO(TWEETY)

Representación ampliada

(1) VUELA(x) \leftarrow PÁJARO(x)² \wedge not PÁJARO-EXCEPCIONAL(x)

(2) PÁJARO-EXCEPCIONAL(x) \leftarrow PINGÜINO(x)

(3) PÁJARO(TWEETY)

(4) PINGÜINO(TWEETY)

SOLUCIÓN:

¿Qué pretende este problema? Desde el presente problema y hasta el 3.26 se estudiará el carácter no monótono de ciertos sistemas que aparecen bien dentro del campo de la lógica o fuera del mismo.

Si Prolog fuera un lenguaje monótono, cualquier hecho que se derivara de la primera representación del ejemplo del enunciado, debería seguir siendo cierta en la representación ampliada, ya que ésta última incluye a la anterior.

En la primera representación PÁJARO-EXCEPCIONAL(TWEETY) no puede ser probado, con lo que Prolog obtiene como cierto el hecho NOT PÁJARO-EXCEPCIONAL(TWEETY). Esto último no sería posible en la lógica de primer orden. Como a partir de (3) se tiene PÁJARO(TWEETY), se puede obtener VUELA(TWEETY) teniendo en cuenta (1). Al añadir el hecho (4) en la representación ampliada, no se puede obtener VUELA(TWEETY), ya que ahora PÁJARO-EXCEPCIONAL(TWEETY) pasa a ser cierto a partir de (2), lo que impide aplicar (1).

- **PROBLEMA 3.24:**

La hipótesis del mundo cerrado que se adopta en Prolog no puede ser aplicada a una base de conocimientos que no esté formada por *cláusulas de Horn*. Mostrar un ejemplo que ratifique la afirmación anterior.

SOLUCIÓN:

Una *cláusula* es una expresión de la forma:

$$B_1 \circ \dots \circ B_m \leftarrow A_1 \text{ y } \dots \text{ y } A_n$$

donde $B_1, \dots, B_m, A_1, \dots, A_n$ son *fórmulas atómicas*, con $n \geq 0$ y $m \geq 0$. Si $m=1$, es decir, sólo existe una conclusión en la expresión anterior, entonces se tiene *una cláusula de*

² Cualquier sistema actual relacionado con el lenguaje Prolog no admite tildes, diéresis, etc. en los nombres de los predicados. Ello es debido, sin duda, a que están diseñados en países cuya lengua no incluye estos signos.

Horn. Supóngase una base de conocimientos constituida únicamente por la siguiente disyunción:

$$P \vee Q$$

que no es una cláusula de Horn. Aplicando la hipótesis del mundo cerrado sobre P y la base de conocimientos anterior, se puede inferir $\neg P$, ya que no hay manera posible de demostrar P . De la misma forma, se podrá inferir también $\neg Q$. Por tanto, se tiene la siguiente base de conocimientos ampliada:

$$P \vee Q$$

$$\neg P \wedge \neg Q \equiv \neg(P \vee Q)$$

en la cual existe una clara contradicción.

- **PROBLEMA 3.25:** (adaptado de [Brewka, 1991])

Los sistemas de marcos han sido ampliamente utilizados en el diseño de sistemas expertos. Los marcos son representaciones de clases de objetos formadas por pares atributo-valor. Considérese el siguiente ejemplo:

```
(defmarco COCHE
  (atributos
    (RUEDAS 4)
    (ASIENTOS 5)
    (CILINDROS 4)))
```

Además, normalmente forman una jerarquía de clases. Podríamos asociar, por ejemplo, la siguiente subclase de coches a la clase anterior:

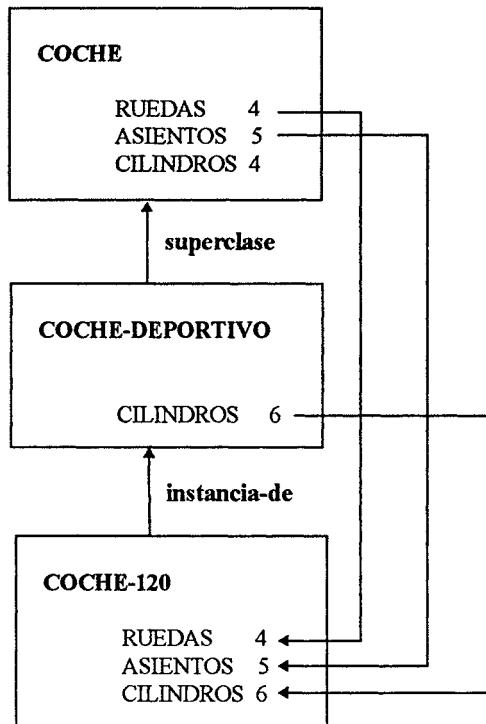
```
(defmarco COCHE-DEPORTIVO
  (super COCHE)
  (atributos
    (CILINDROS 6)))
```

donde la especificación “super” indica que COCHE-DEPORTIVO es una subclase de COCHE. Cada subclase hereda de la clase a la que pertenece aquellos pares atributo-valor que no figuren en su propia definición. En caso de que haya atributos repetidos, siempre prevalece la información más específica, es decir, la de la subclase. Se llama *instancia* a un ejemplo particular de un objeto perteneciente a una determinada clase.

Poner un ejemplo que dé idea del comportamiento no monótono de un sistema de marcos, basado en la idea de que la información más específica prevalece sobre la más general (utilizar la jerarquía de marcos dada en el ejemplo).

SOLUCIÓN:

Dada una instancia particular de COCHE-DEPORTIVO llamada COCHE-120, se tendría:

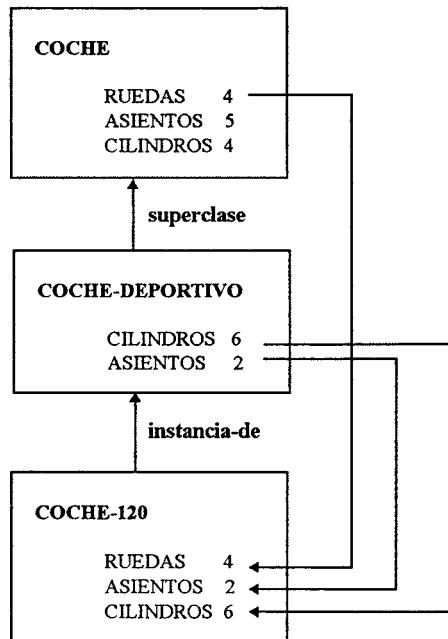
Base de conocimientos original:**Figura 3-10**

COCHE-120 tendría, por tanto, 6 cilindros, 5 asientos y 4 ruedas. Si se añadiera al sistema de marcos la información de que los coches deportivos suelen tener 2 asientos:

```

(defmarco COCHE-DEPORTIVO
  (super COCHE)
  (atributos
    (ASENTOS 2)
    (CILINDROS 6)))
  
```

se tendría la siguiente base de conocimientos:

Base de conocimientos ampliada:**Figura 3-11**

La conclusión es que COCHE-120 tiene 2 asientos y no 5 como había sido inferido con anterioridad al aplicar la herencia de propiedades al sistema de marcos. Este cambio ha sido producido por la adición de nueva información al sistema, lo cual demuestra el comportamiento no monótono de los sistemas de marcos.

- **PROBLEMA 3.26:** (adaptado de [Brewka, 1991])

El llamado “razonamiento autoepistémico” sigue el siguiente patrón:

- 1) Si la afirmación x fuera verdad, yo lo sabría.
- 2) No sé si x es verdad.
- 3) Por tanto, x no es cierto.

donde 2) no es una premisa, sino que tiene que ser determinado a partir del conocimiento que se tenga a mano.

Poner un ejemplo que muestre el carácter no monótono del razonamiento autoepistémico.

SOLUCIÓN:

Dada la siguiente base de hechos:

- (1) Si alguien es mi hermano, yo lo sé.
- (2) Juan es mi hermano.

No hay información en la base de datos que permita establecer que Pedro es mi hermano. Por tanto, siguiendo las leyes del razonamiento autoepistémico, se puede concluir: “Pedro no es mi hermano”, ya que “Yo no sé que Pedro sea mi hermano”. Sin embargo, añadiendo a la base de hechos la información:

- (3) Pedro es mi hermano.

la conclusión previa es imposible. Por tanto, este ejemplo muestra que el razonamiento autoepistémico es no monótono, debido a que al introducir en la base de hechos una afirmación que en principio no debería entrar en contradicción con las afirmaciones anteriores, se ha llegado a un inconsistencia como consecuencia del tipo de razonamiento empleado para obtener conclusiones.

3.4 Contexto adicional

3.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

- Captar los límites y deficiencias de la lógica clásica a la hora de servir de método de representación de conocimiento para ciertos dominios.
- Diferenciar las nuevas capacidades de representación e inferencia características de cada una de las lógicas introducidas en el capítulo.
- Concluir que en el futuro deberán seguir surgiendo nuevos formalismos lógicos que superen a los actuales en la labor de representar cualquier tipo de conocimiento.

3.4.2 Pistas de autoevaluación

- Realizar el problema 3.1 para un sumador binario de 4 bits.
- Dar una prueba del siguiente razonamiento:

“Hay más de un ser.”

“Por tanto, es falso que haya exactamente un ser.”

- Dar una prueba del siguiente razonamiento:

“Necesariamente, si no decides entonces decides.”
“Por tanto, necesariamente decides.”
- Comprobar en el problema 3.15 que F no cumple la ley del tercio excluso.
- Sugerir ejemplos que demuestren el carácter no monótono del razonamiento de sentido común humano.

4 REGLAS

4.1 Contexto previo

4.1.1 ¿Por qué está aquí este capítulo?

Las reglas constituyen un intento de resolver la ineficiencia propia del formalismo lógico, para lo cual se sacrifica expresividad a cambio de eficiencia. El formalismo de reglas de producción marcó la expansión de la inteligencia artificial desde un nivel teórico o de laboratorio hasta otro más práctico o comercial.

El sistema experto basado en reglas MYCIN (década de los setenta) ha sido, sin lugar a dudas, el sistema más influyente en la historia de los sistemas expertos y puede todavía considerarse paradigmático de una forma de representar conocimiento y de usarlo después en la inferencia.

En la actualidad, el diseño de sistemas expertos suele llevarse a cabo con la ayuda de herramientas comerciales que ofrecen facilidades para modelar el conocimiento del dominio mediante reglas y marcos e incorporan algunos de los conceptos propios de las redes semánticas. De hecho, todas las técnicas de representación que se estudian en este libro se integran en formalismos más completos y próximos al campo de la programación orientada a objetos. Las reglas pueden considerarse también como objetos.

4.1.2 ¿Dónde hay conocimiento teórico del campo?

Este capítulo de reglas puede ser completado con la lectura de los siguientes apartados teóricos:

Referencias básicas:

- [Borrajo *et al.*, 1993], capítulo 4.
- [Mira *et al.*, 1995], capítulo 6.
- [Rich & Knight, 1994], capítulos 6 y 8.

Referencias complementarias:

- [Jackson, 1990], capítulos 3 y 8.

- [Lucas & Van der Gaag, 1991], capítulos 3 y 5.
- [Nilsson, 1987], capítulo 6.
- [Norvig, 1992], capítulo 16.
- [Winston, 1994], capítulos 7 y 8.

4.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

- Es conveniente que el lector repase el capítulo 1, en concreto las técnicas de búsqueda un profundidad, búsqueda con retroceso y búsqueda en grafos Y/O.

4.1.4 ¿Qué software de apoyo está disponible?

- CLIPS (*C Language Integrated Production System*)

CLIPS 6.0 es un sistema escrito en ANSI C que permite encadenamiento hacia adelante, adición dinámica de reglas, definición de estrategias para la resolución de conflictos... Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/clips/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/clips/0.html)

- ES

Herramienta de fácil manejo para el desarrollo de sistemas expertos. Incorpora encadenamiento hacia adelante, hacia atrás, posibilidad de manejo de reglas borrosas y métodos de explicación del razonamiento. Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/es/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/es/0.html)

- EXPERT

Sistema escrito en ADA que utiliza encadenamiento hacia atrás. Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/expert/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/expert/0.html)

- OPS5

Entorno para la construcción de sistemas expertos con la capacidad de aplicar el algoritmo RETE. Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/ops5/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/ops5/0.html)

4.2 Contenido teórico

4.2.1 Introducción

Un sistema basado en reglas consta de los siguientes elementos:

- a) **Motor de inferencia:** coordina la información procedente del resto de módulos.
- b) **Base de conocimientos:** contiene las reglas utilizadas para representar el conocimiento disponible de un determinado dominio.
- c) **Base de afirmaciones:** contiene los hechos o afirmaciones conocidos inicialmente y aquéllos que se van creando en el proceso de inferencia.
- d) **Interfaz de usuario:** a través del cual se puede solicitar u ofrecer información al usuario.

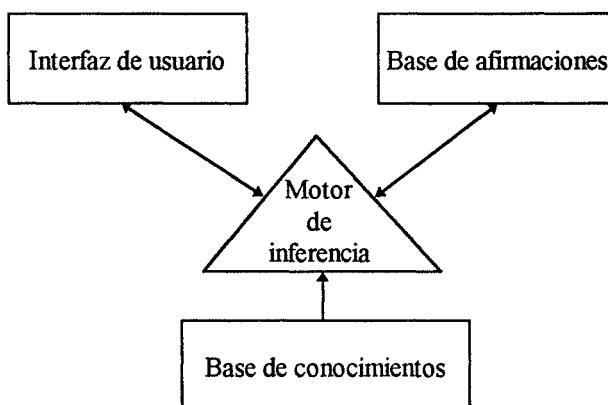


Figura 4-1

En la figura 4-1 aparece un esquema de la composición de un sistema basado en reglas, donde el sentido de las flechas da idea del flujo de información que se produce entre los diferentes módulos. Obsérvese que normalmente no existirá este flujo desde el motor de inferencia a la base de conocimientos, a no ser que el sistema tenga capacidad de aprendizaje.

Una regla, por lo general, constará de dos partes: *antecedente o parte izquierda*, y *consecuente o parte derecha*. En el antecedente figurarán cláusulas o condiciones que se deberán cumplir para que la regla pueda ser ejecutada. En el consecuente, por otra parte, aparecerán las conclusiones a establecer o acciones a desarrollar cuando la regla sea ejecutada.

4.2.2 Inferencia

El proceso de inferencia en un sistema basado en reglas consiste en establecer la verdad de determinadas conclusiones a partir de la información que se tiene en la base de afirmaciones y la base de conocimientos. Es el motor de inferencia el encargado de llevar a cabo dicho proceso, que por lo general puede realizarse de las dos formas siguientes:

a) Encadenamiento hacia adelante

Se basa en ejecutar aquellas reglas cuyo antecedente sea cierto a partir de la información que hay en el sistema. Por otra parte, la introducción de nueva información a partir del consecuente de cada regla ejecutada, permitirá la ejecución de otras reglas.

PROBLEMAS RELACIONADOS: 4.1, 4.2, 4.7 a 4.9

b) Encadenamiento hacia atrás

Se basa en seleccionar aquellas reglas cuyo consecuente permita demostrar cierta condición, si ésta no puede ser demostrada a partir de la base de afirmaciones. A su vez, las condiciones o cláusulas que figuren en los antecedentes de las reglas seleccionadas pasarán a convertirse en nuevos subobjetivos a demostrar, entrándose en un proceso recursivo que finaliza cuando se encuentra la información buscada en la base de afirmaciones o cuando se le solicita dicha información al usuario.

PROBLEMAS RELACIONADOS: 4.1, 4.3 a 4.6, 4.12 y 4.13

Una de las ventajas de las reglas frente a la lógica clásica, en la que sólo se realiza razonamiento monótono —nunca la información inferida contradice la ya existente—, es la posibilidad que ofrecen para “borrar” de la base de afirmaciones hechos como consecuencia de nuevas inferencias. Debido a lo anterior, habrá que estudiar qué hacer con aquellas conclusiones que se obtuvieron a partir de información que ahora se “borra”, anula o invalida; por tanto, va a surgir el concepto de *tipo de dependencia* de una regla, que podrá ser:

a) Dependencia reversible: si en cualquier instante las afirmaciones que hicieron cierto el antecedente de una regla son anuladas, habrá que anular también los hechos inferidos por el consecuente.

b) Dependencia irreversible: una vez inferido un hecho, no podrá ser invalidado o anulado.

4.2.3 Control del razonamiento o resolución de conflictos

Este apartado aborda el problema de qué regla ejecutar primero cuando hay varias disponibles para tal fin. Existen diferentes estrategias al respecto:

- **Ordenación de reglas:** se colocan en primer lugar las reglas que se quiere examinar antes.
- **Control de las agendas:** el conjunto de reglas listas para ser ejecutadas puede reunirse en una agenda en la que se ordenan según un valor de prioridad asignado a cada una de ellas.
- **Criterio de actualidad:** consiste en ejecutar primero aquellas reglas cuyo antecedente se apoya en información más reciente. Para que este criterio pueda aplicarse, el motor de inferencia deberá registrar en qué momento se ha generado cada afirmación.
- **Criterio de especificidad:** se prefieren para su ejecución las reglas más específicas. En el siguiente ejemplo:

R₁: Si *a* entonces *b*

R₂: Si *a* y *d* entonces *e*

Base de afirmaciones inicial: {*a*, *d*}

R₂ se ejecutaría antes que R₁ por ser más específica.

Normalmente, los sistemas basados en reglas disponen de un *mecanismo de refractariedad* que impide la ejecución repetida de una regla sin que se introduzca nueva información en el sistema. Otro mecanismo de control del razonamiento son los *conjuntos de reglas* (*rule sets* en inglés) que permiten activar o desactivar varias reglas en bloque. Finalmente, cabe mencionar el concepto de *metaregulas*, que son reglas que razonan acerca de otras reglas. También las metaregulas pueden ayudar al control del razonamiento, por ejemplo cambiando o asignando prioridades a otras reglas bajo determinadas circunstancias.

PROBLEMAS RELACIONADOS: 4.1 a 4.10

4.2.4 Apéndice 4.A: Algoritmo RETE

Los mecanismos de encadenamiento hacia atrás generalmente utilizan una búsqueda en profundidad con retroceso para seleccionar reglas, mientras que los de encadenamiento hacia adelante utilizan diversos mecanismos para agilizar la resolución de conflictos. Entre estos últimos, uno de los algoritmos más eficaces y extendidos es el llamado RETE, desarrollado por Charles Forgy.

Si en cada ciclo del proceso de inferencia llevado a cabo en encadenamiento hacia adelante hubiera que comprobar todas las reglas que pueden aplicarse en función de los elementos de la memoria de trabajo, el sistema de producción sería muy poco eficiente. Teniendo en cuenta que en cada ciclo los cambios producidos en la base de hechos son los únicos que influyen en la modificación del conjunto conflicto, se toman dichas actualizaciones como base del cálculo de las reglas que pasarán a formar parte del citado conjunto. De esta forma, los nuevos hechos que son introducidos en la memoria de trabajo son los que determinan las reglas aplicables. Es decir, la idea que constituye la base de este algoritmo es que, en lugar de buscar qué reglas satisfacen los hechos existentes en cada momento, son los nuevos hechos generados en cada ciclo los que buscan o determinan qué nuevas reglas se seleccionan para una posible ejecución (ver figura 4-2).

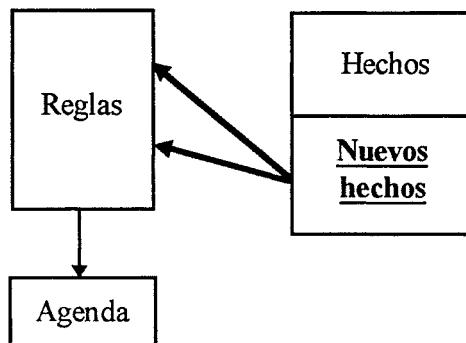


Figura 4-2 En el algoritmo RETE los nuevos hechos buscan reglas.

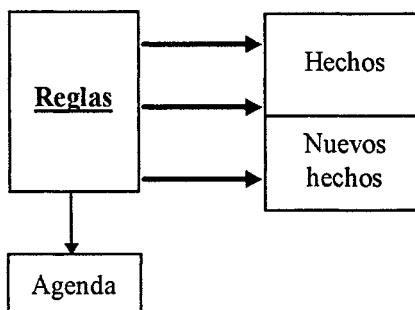


Figura 4-3 Por el método tradicional se verifica para cada regla si se cumple su antecedente a partir de la base de hechos total.

El sistema RETE mantiene una *red de condiciones* de las reglas, de forma que varias reglas pueden compartir las condiciones en dicha red, ahorrándose entonces el procesamiento repetido de los hechos coincidentes. La forma esquemática de funcionamiento de este algoritmo es la siguiente:

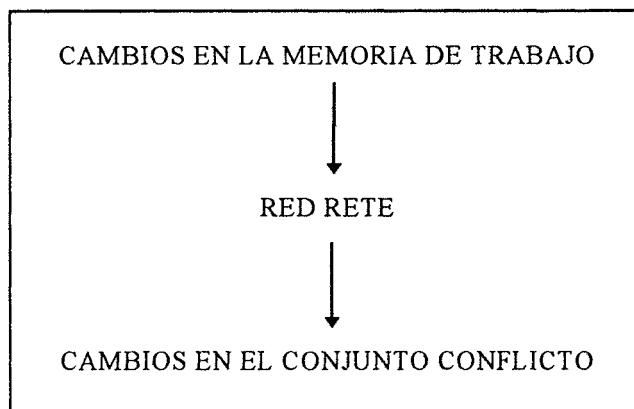


Figura 4-4

La comunicación a la red RETE de los cambios producidos en la memoria de trabajo se hace mediante una “señal” o “testigo” (en inglés *token*), que está formada por dos elementos:

- a) Un signo “+” o “-”, según se trate de una inclusión o una supresión de elementos.
- b) Los elementos de la memoria de trabajo que se han añadido o suprimido.

Por ejemplo, $(+(AB))$ significaría que se ha añadido a la memoria de trabajo el hecho (AB) . La red RETE se genera al principio de la ejecución del sistema de producción mediante un compilador. Considérense los siguientes ejemplos:

- (a) Si la base de conocimientos estuviera formada por la regla,

$$R_1: (AB) (CD) \longrightarrow \dots$$

(el antecedente de esta regla se cumplirá si en la base de afirmaciones figuran los hechos (AB) y (CD)), se generará la siguiente red RETE:

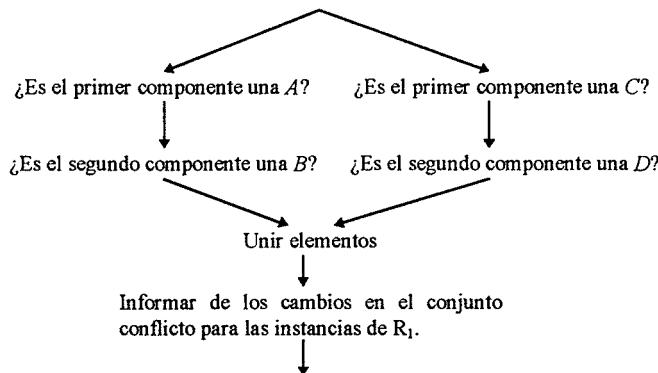


Figura 4-5

(b) $R_2: (A <x>) (B <y>) \rightarrow \dots$ ($<x>$ e $<y>$ representan variables)

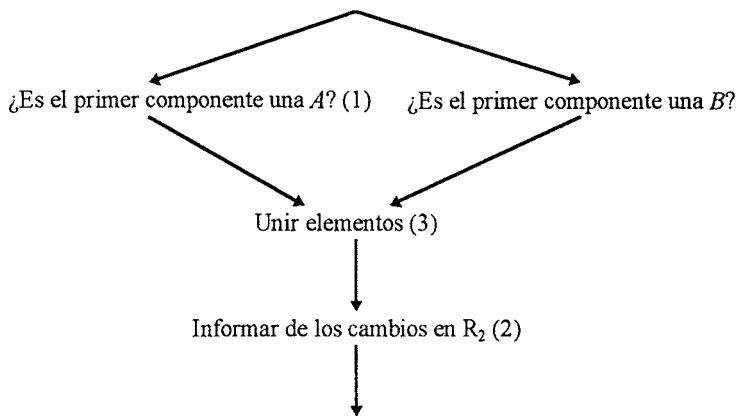


Figura 4-6

Existen varios tipos de nodos en la red RETE:

1) De una entrada, que pueden ser subdivididos en:

- Los que verifican el valor de una componente variable o constante perteneciente a un elemento de condición de una regla (es el caso del nodo (1) del ejemplo (b) anterior; ver figura 4-6).

- b) Los *terminales* que se sitúan al final de la red. Éstos se encargan de comunicar las modificaciones que se producen en el conjunto conflicto. Existe un nodo terminal para cada regla (ver el nodo (2) del ejemplo (b), figura 4-6).
- c) Los que comprueban la aparición de una misma variable dos o más veces en un elemento de condición. Estos nodos comparan si son iguales dos valores de un elemento de la memoria de trabajo. Por ejemplo:

$$R_3: (A <x> B <x> <x>) \longrightarrow \dots$$

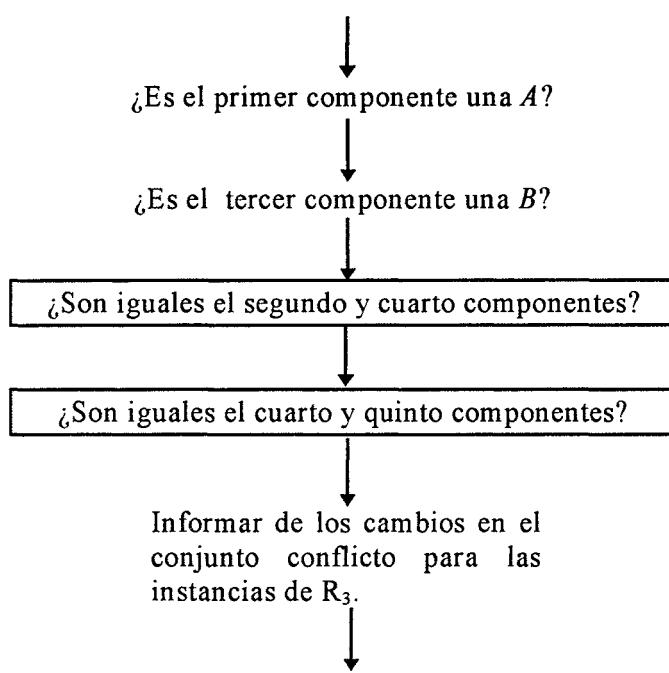


Figura 4-7

- d) El *nodo raíz*, que se sitúa al inicio de la red. Su misión es recibir los “testigos” para distribuirlos por la red.

2) De dos entradas

Se sitúan siempre en la red después de los nodos de una entrada. Estos nodos tienen asociada para cada entrada una memoria donde se almacenan los “testigos”. Existen, de nuevo, varios tipos de nodos dentro de esta categoría:

a) De unión de elementos de condición pertenecientes a una regla que contiene o no variables (ver el nodo (3) del ejemplo (b), figura 4-6).

b) De unión de elementos pertenecientes a una regla que contiene variables iguales. Estos nodos comprueban si los valores ligados a una variable que aparecen en dos elementos de condición de una regla son iguales. Sea el siguiente ejemplo:

$$R_7: (A < x > < x >) (< x > D < y >) (B < x > < y >) \longrightarrow \dots$$

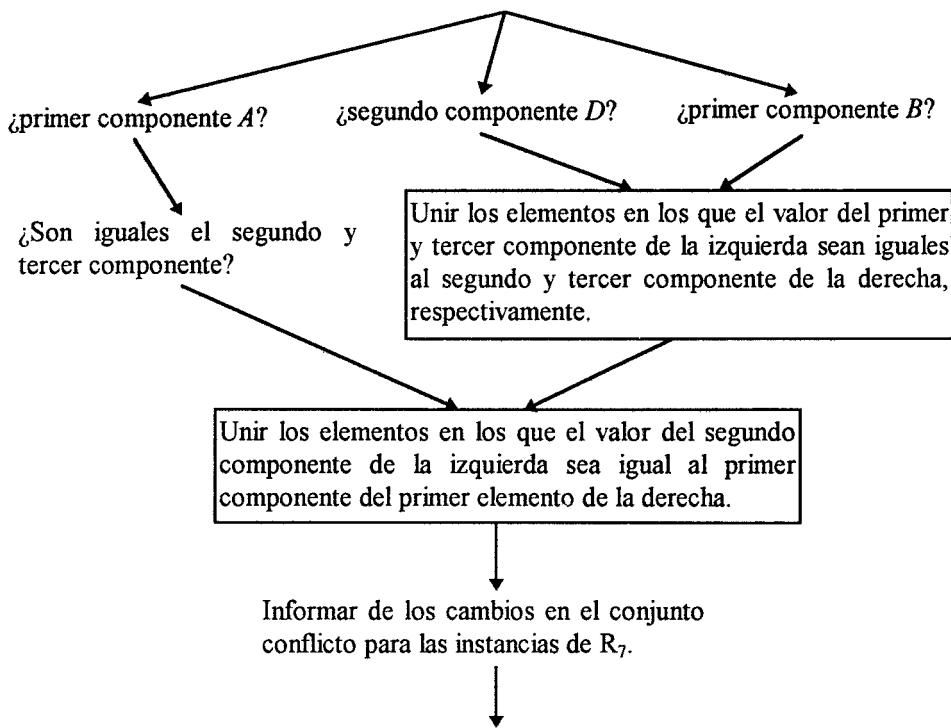


Figura 4-8

c) De unión de elementos de condición negados con los restantes elementos de condición en un regla. Disponen de memoria, al igual que los nodos de unión de elementos no negados, aunque el funcionamiento de la misma es diferente y se adapta ahora a las nuevas circunstancias. Considérese el siguiente ejemplo:

$$R_9: (A < x >) -(< x > < y >) \longrightarrow \dots$$

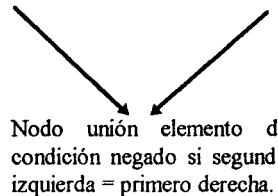


Figura 4-9

Por tanto, en la construcción de RETE se hace uso de la similitud en las estructuras de las reglas. En concreto, lo que se hace es compartir partes comunes, en vez de duplicar nodos. Considerese, finalmente, el siguiente ejemplo. Sean las reglas:

$$R_1: (AB <x>) (C <x> DE <y>) \longrightarrow \dots$$

$$R_2: (AB <x>) (C <x> DF <y>) \longrightarrow \dots$$

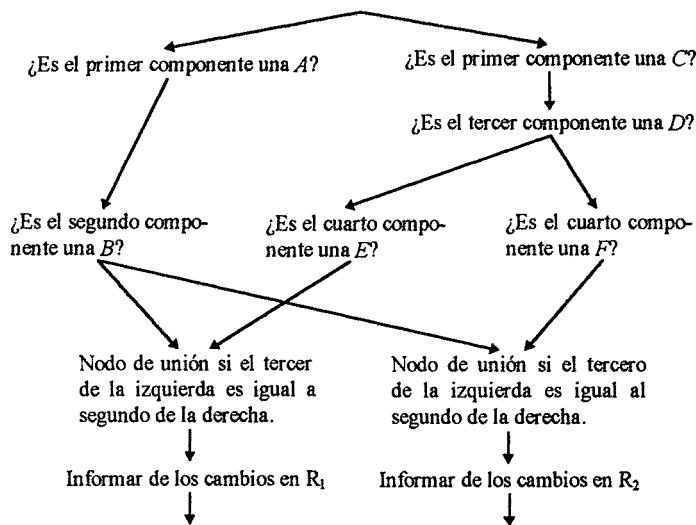


Figura 4-10

En [Borrajo *et al.*, 1993] aparece un completo apartado dedicado al algoritmo RETE cuya lectura puede resultar de interés para todo aquél que desee tener un conocimiento más profundo de este algoritmo.

PROBLEMAS RELACIONADOS: 4.11

4.2.5 Apéndice 4.B: Factores de certeza de MYCIN

MYCIN es un sistema experto diseñado para el tratamiento de infecciones en la sangre que fue desarrollado en la década de los 70 en la Universidad de Stanford. La base de conocimientos de este sistema experto se organiza alrededor de un conjunto de reglas, a cada una de las cuales se asocia un factor numérico que indica el grado de certeza que se tiene en el cumplimiento de la conclusión de la regla, una vez que se satisfacen las condiciones que aparecen en el antecedente de la misma. Estos “factores de certeza” toman en MYCIN valores pertenecientes al intervalo $[-1,1]$. Dada cualquier regla del tipo:

if evidencia then hipótesis

a la que se le asocia un factor de certeza $FC(hipótesis, evidencia)$, se tendrá que $FC(hipótesis) = \max\{0, FC(evidencia)\} \cdot FC(hipótesis, evidencia)$. Un factor de certeza asociado a una regla que sea igual a cero indicará que la evidencia no influye en el grado de creencia que se tiene en una determinada hipótesis. Dadas las reglas:

if e' then h_1

if e' then h_2

if e_1' then h

if e_2' then h

con factores de certeza $FC(h_1, e')$, $FC(h_2, e')$, $FC(h, e_1')$ y $FC(h, e_2')$ respectivamente, MYCIN hace uso de las siguientes fórmulas para el cálculo de factores de certeza compuestos:

$$FC(h_1 \text{ and } h_2, e') = \min\{FC(h_1, e'), FC(h_2, e')\}$$

$$FC(h_1 \text{ or } h_2, e') = \max\{FC(h_1, e'), FC(h_2, e')\}$$

$$FC(h, e_1' \text{ comb } e_2') = \begin{cases} FC(h, e_1') + FC(h, e_2') \cdot (1 - FC(h, e_1')) & \text{si } FC(h, e_i') > 0 \text{ con } i=1, 2 \\ \frac{FC(h, e_1') + FC(h, e_2')}{1 - \min\{|FC(h, e_1')|, |FC(h, e_2')|\}} & \text{si } -1 < FC(h, e_1') \cdot FC(h, e_2') \leq 0 \\ FC(h, e_1') + FC(h, e_2') \cdot (1 + FC(h, e_1')) & \text{si } FC(h, e_i') < 0 \text{ con } i=1, 2 \end{cases}$$

Este modelo de certeza introducido en MYCIN ha sido criticado por el hecho de que se basa más en la intuición que en una teoría matemática sólida. Incluso ha llegado a comprobarse que puede producir resultados contrarios al sentido común y a la teoría de la probabilidad. Aunque en un primer momento parecía que MYCIN producía resultados satisfactorios, nunca fue usado en hospitales por varios motivos: su base de conocimientos era incompleta (el espectro de enfermedades infecciosas es enorme), a mediados de los años 70 la mayoría de los hospitales no podían permitirse la compra de un equipo lo suficientemente potente en el que MYCIN pudiera funcionar y, finalmente, el interfaz existente no era lo suficientemente amigable como para que los médicos aceptaran el sistema de buen grado.

PROBLEMAS RELACIONADOS: 4.12, 4.13

4.3 Problemas resueltos

• PROBLEMA 4.1:

Dado un sistema basado en reglas con la siguiente base de conocimientos:

R₁: Si h_2 y h_5 entonces h_1

R₂: Si h_4 y h_3 entonces h_2

R₃: Si h_6 entonces h_3

donde cada h_i representa una situación o concepto y los números al lado de las reglas marcan la prioridad de ejecución de las mismas en caso de conflicto. La base de hechos inicial contiene los siguientes datos: h_6, h_7, h_9, h_8, h_4 y h_5 .

- Aplicar encadenamiento hacia adelante, mostrando cómo evoluciona el sistema en cada ciclo del proceso.
- Aplicando encadenamiento hacia atrás, determinar si es posible establecer la existencia de la situación h_1 a partir de la base de hechos inicial.

SOLUCIÓN:

¿Qué pretende este problema? A partir de este problema y hasta el 4.10 se estudiarán ejemplos de inferencia (encadenamiento hacia adelante y hacia atrás) en sistemas basados en reglas. Al mismo tiempo, en cada caso se aplicarán diferentes métodos de control del razonamiento o resolución de conflictos.

- El procedimiento aplicable en el caso de encadenamiento hacia adelante sería del siguiente tipo:

1. Hasta que el problema esté resuelto o no haya más reglas que aplicar:

1.1 Seleccionar aquellas reglas cuyo antecedente se cumple a partir de la base de hechos actual. Si hay más de una regla seleccionada, emplear una estrategia de resolución de conflictos que elimine todas las reglas anteriores menos una.

1.2 Ejecutar la regla resultante del paso anterior.

1) Inicialmente se tendría:

BH ₀
<i>h</i> ₆
<i>h</i> ₇
<i>h</i> ₉
<i>h</i> ₈
<i>h</i> ₄
<i>h</i> ₅

Figura 4-11

De las tres reglas existentes en la base de conocimientos sólo la 3 se selecciona. Tras su ejecución:

BH ₁
<i>h</i> ₆
<i>h</i> ₇
<i>h</i> ₉
<i>h</i> ₈
<i>h</i> ₄
<i>h</i> ₅
<i>h</i> ₃

Figura 4-12

Ahora existirá un conflicto entre las reglas 2 y 3. Al ser la primera de ellas más prioritaria, se ejecuta:

BH ₂
<i>h</i> ₆
<i>h</i> ₇
<i>h</i> ₉
<i>h</i> ₈
<i>h</i> ₄
<i>h</i> ₅
<i>h</i> ₃
<i>h</i> ₂

Figura 4-13

En esta situación, todas las reglas son seleccionadas. Finalmente, es la regla 1 la que se ejecuta:

BH ₃
<i>h</i> ₆
<i>h</i> ₇
<i>h</i> ₉
<i>h</i> ₈
<i>h</i> ₄
<i>h</i> ₅
<i>h</i> ₃
<i>h</i> ₂
<i>h</i> ₁

Figura 4-14

Esta base de hechos final no cambiaría si el procedimiento de selección de reglas continuara. Nótese que no se ha aplicado ningún mecanismo de refractariedad que impida que una misma regla se ejecute dos veces, debido a que no se pide explícitamente en el enunciado del problema, aunque el procedimiento normal hubiera sido aplicarlo.

- b) En el caso de encadenamiento hacia atrás, se parte de un determinado concepto objetivo cuya validez hay que verificar a partir de la base de conocimientos y la base

inicial de hechos. Para ello, en primer lugar se comprueba si dicho concepto pertenecía ya o no a la base de hechos. En caso negativo es necesario echar mano de las reglas existentes en la base de conocimientos, concretamente de aquéllas en cuyo consecuente figure el concepto objetivo. De esta manera, los conceptos del antecedente de dichas reglas pasan a ser considerados como subobjetivos. Si se demuestra la validez de estos subobjetivos para una determinada regla (obsérvese el carácter recursivo del encadenamiento hacia atrás), podrá inferirse el concepto objetivo global, que era el que se pretendía demostrar.

En el ejemplo del enunciado se quiere saber si se puede establecer la existencia de h_1 , que no figura en la base de hechos inicial; por tanto, será necesario acudir a la base de conocimientos. La regla 1 tiene h_1 en su consecuente. Ello nos permite fijar h_2 y h_5 como subobjetivos. Al encontrarse h_5 en la base de hechos inicial, sólo quedaría por demostrar h_2 .

La regla 2, que tiene h_2 en su consecuente, establece h_4 y h_3 como nuevos subobjetivos. El hecho h_4 está contenido en la base de hechos inicial. Habrá que verificar, por tanto, la existencia de h_3 .

La regla 3 fija ahora como subobjetivo h_6 . Al estar h_6 en la base de hechos inicial, se ha llegado al final del proceso, con lo que queda demostrada la validez de h_1 sin más que aplicar, en este orden, las reglas 3, 2 y 1 a la base de hechos inicial.

- **PROBLEMA 4.2:** (adaptado de [Borrajo *et al.*, 1993])

Sea el siguiente conjunto de reglas:

R₁: Si h_8 y h_6 y h_5 entonces h_4

R₂: Si h_6 y h_3 entonces h_9

R₃: Si h_7 y h_4 entonces h_9

R₄: Si h_8 entonces h_1

R₅: Si h_6 entonces h_5

R₆: Si h_9 y h_1 entonces h_2

R₇: Si h_7 entonces h_6

R₈: Si h_1 y h_7 entonces h_9

R₉: Si h_1 y h_8 entonces h_6

La base de hechos inicial es la que aparece en la figura 4-15 y el concepto meta es h_2 .

BH ₀
<i>h</i> ₇
<i>h</i> ₈

Figura 4-15

Aplicando encadenamiento hacia adelante y suponiendo que el sistema está dotado de un mecanismo de refractariedad (no se debe ejecutar dos veces la misma regla de la misma forma), describir el proceso de inferencia resultante cuando las estrategias para resolución de conflictos son las siguientes:

- a) Es más prioritaria aquella regla con subíndice menor.
- b) Es más prioritaria aquella regla con más condiciones en su antecedente (en caso de igualdad se aplicaría el criterio del apartado “a”).

SOLUCIÓN:

Cada ciclo del proceso de encadenamiento hacia adelante llevará a cabo los mismos pasos, hasta que el problema se resuelva o no queden más reglas por aplicar:

- Comparación de la base de hechos actual con los antecedentes de todas las reglas de la base de conocimientos para la determinación de aquéllas que pasan a ser candidatas para la ejecución (a este conjunto de reglas se le llama “conjunto conflicto”).
 - Aplicación al conjunto conflicto de la estrategia para resolución de conflictos correspondiente, de manera que sólo una de las reglas se ejecute.
 - Actualización de la base de hechos a partir del consecuente de la regla ejecutada.
- a) A continuación se describirá el desarrollo iterativo de los pasos descritos anteriormente para el caso particular del presente problema y la primera estrategia de resolución de conflictos:

1)

BH ₀
<i>h</i> ₇
<i>h</i> ₈

Figura 4-16

- Conjunto conflicto: R_4, R_7 .
- La regla seleccionada para ejecución será R_4 , por ser $4 < 7$. Debido al principio de refractariedad, R_4 pasará a estar inactiva para el resto del proceso de inferencia, por lo que no formará parte de ningún otro conjunto conflicto, aunque su antecedente se cumpla.

- Ejecución de R_4 .

La nueva base de hechos será:

BH ₁
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁

Figura 4-17

2) A partir del contenido de BH₁ se desarrollarán los siguientes pasos:

- Conjunto conflicto: R_7, R_8, R_9 .
- Regla seleccionada para ejecución: R_7 .
- Ejecución de R_7 .

La base de hechos resultante de la ejecución de este segundo paso del proceso de inferencia es:

BH ₂
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁
<i>h</i> ₆

Figura 4-18

3) A partir de BH₂:

- Conjunto conflicto: R_5, R_8, R_9 .
- Regla seleccionada para ejecución: R_5 .

- Ejecución de R_5 .

Base de hechos resultante:

BH ₃
<i>h₇</i>
<i>h₈</i>
<i>h₁</i>
<i>h₆</i>
<i>h₅</i>

Figura 4-19

4) A partir de BH₃:

- Conjunto conflicto formado: R₁, R₈, R₉.
- Regla seleccionada para ejecución: R₁.
- Ejecución de R₁.

Nueva base de hechos:

BH ₄
<i>h₇</i>
<i>h₈</i>
<i>h₁</i>
<i>h₆</i>
<i>h₅</i>
<i>h₄</i>

Figura 4-20

5) A partir de BH₄:

- Conjunto conflicto: R₃, R₈, R₉. Por el principio de refractariedad, R₁ no pasa a formar parte de este conjunto.
- Regla seleccionada, según la estrategia correspondiente: R₃.
- Ejecución de R₃.

Base de hechos actualizada:

BH ₅
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁
<i>h</i> ₆
<i>h</i> ₅
<i>h</i> ₄
<i>h</i> ₉

Figura 4-21

6) A partir de BH₅:

- Conjunto conflicto: R₆, R₈, R₉.
- Regla seleccionada: R₆.
- Ejecución de R₆.

Base de hechos actualizada:

BH ₆
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁
<i>h</i> ₆
<i>h</i> ₅
<i>h</i> ₄
<i>h</i> ₉
<i>h</i> ₂ ¡ÉXITO!

Figura 4-22

Debido a que se ha generado el hecho o concepto meta, el proceso de inferencia finaliza.

b) Teniendo en cuenta la segunda estrategia para resolución de conflictos mencionada en el enunciado, se tiene:

1) Inicialmente la base de hechos es:

BH ₀
<i>h</i> ₇
<i>h</i> ₈

Figura 4-23

De acuerdo con esta base de hechos:

- Conjunto conflicto: R₄, R₇.
 - En esta ocasión, tanto R₄ como R₇ tienen una única condición en su antecedente. Por tanto, al ser 4<7, la regla seleccionada para ejecución será R₄ y no R₇.
 - Ejecución de R₄.

Base de hechos actualizada:

BH ₁
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁

Figura 4-24

2) A partir de BH₁:

- Conjunto conflicto: R₇, R₈, R₉.
 - La regla seleccionada para ejecución es R₈, debido a que tiene dos condiciones en su antecedente (R₇ tiene tan solo una) y 8<9 (R₉ tiene también dos condiciones en su antecedente.).
 - Ejecución de R₈.

Base de hechos actualizada:

BH ₂
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁
<i>h</i> ₉

Figura 4-25

3) A partir de BH₂:

- Conjunto conflicto: R₆, R₇, R₉.
- Regla seleccionada R₆.
- Ejecución de R₆.

Nueva base de hechos:

BH ₃
<i>h</i> ₇
<i>h</i> ₈
<i>h</i> ₁
<i>h</i> ₉
<i>h</i> ₂ ¡EXITO!

Figura 4-26

En este momento el proceso de inferencia finalizaría y *h*₂ habría quedado demostrado.

• PROBLEMA 4.3: (adaptado de [Borrajo *et al.*, 1993])

Considérese el conjunto de reglas del problema anterior, así como la misma base de hechos inicial y concepto meta:

R₁: Si *h*₈ y *h*₆ y *h*₅ entonces *h*₄

R₂: Si *h*₆ y *h*₃ entonces *h*₉

R₃: Si *h*₇ y *h*₄ entonces *h*₉

R₄: Si *h*₈ entonces *h*₁

R₅: Si *h*₆ entonces *h*₅

R₆: Si h_9 y h_1 entonces h_2

R₇: Si h_7 entonces h_6

R₈: Si h_1 y h_7 entonces h_9

R₉: Si h_1 y h_8 entonces h_6

Concepto meta u objetivo: h_2

Base de hechos inicial:

BH ₀
h_7
h_8

Figura 4-27

Aplicar encadenamiento hacia atrás y describir el proceso de inferencia resultante. Para la resolución de conflictos utilizar la estrategia consistente en seleccionar primero aquella regla con un subíndice menor.

SOLUCIÓN:

Cuando en un proceso de inferencia con reglas se emplea encadenamiento hacia atrás para intentar demostrar un hecho o concepto objetivo, se llevan a cabo los siguientes pasos:

- Dado el hecho que se pretende demostrar, se averigua en primer lugar si dicho hecho figura en la base de hechos actual, con lo que quedaría demostrado.

- Si lo anterior no se cumple, se crea un conjunto conflicto con aquellas reglas en cuyo consecuente figure el hecho que se pretende demostrar y se llevarían a cabo los siguientes pasos hasta que no queden más reglas por aplicar o el objetivo haya sido demostrado:

- a) Aplicando la estrategia de resolución de conflictos que se considere conveniente, se selecciona una regla.

- b) En un proceso recursivo, se intentaría demostrar cada una de las condiciones que figuran en el antecedente de la regla seleccionada.

El proceso de encadenamiento hacia atrás con reglas descrito anteriormente conduce de forma natural a una estructura de árbol en la que los nodos representan conceptos particulares, habiendo dos posibles tipos de enlaces:

- **Enlaces O:** se dan en el caso de que exista una determinada submeta y la base de hechos contenga varias reglas con dicha submeta en sus consecuentes. Basta con que una de esas reglas produzca una demostración de la submeta mencionada para considerar que la misma ha sido verificada. Por ejemplo, dado:

$$R_1: A \text{ y } B \longrightarrow C$$

$$R_2: D \longrightarrow C$$

Submeta: C ; $BH=\{D\}$

se representaría:

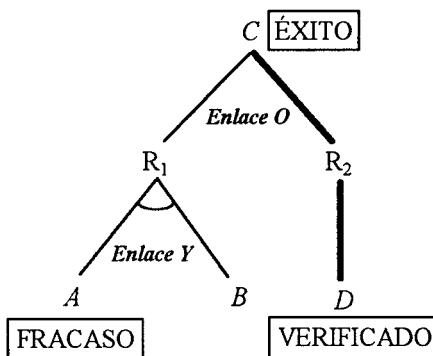


Figura 4-28

- **Enlaces Y:** sirven para representar el hecho de que, dada una submeta y una regla con dicha submeta en su consecuente, habrá que pasar a demostrar el antecedente de dicha regla. En dicho antecedente pueden figurar conjunciones de condiciones (o también disyunciones que originarían enlaces O), dando lugar a la aparición de enlaces Y en el árbol. Si el antecedente no puede ser demostrado, tampoco lo podrá ser la submeta que estaba siendo considerada, a no ser que haya otras reglas que lo permitan (en la figura 4-28 puede observarse un enlace Y).

El proceso de encadenamiento hacia atrás de reglas conduce a una exploración en profundidad del árbol que se forma a partir de la base de conocimientos. En el caso de este problema se tendría:

- 1) El hecho objetivo es h_2 , con lo que habrá que averiguar si se encuentra en BH_0 . Como no es así, será necesario crear un conjunto conflicto con aquellas reglas que puedan llevarnos a demostrar h_2 :

- Conjunto conflicto: R_6 . Se tiene:



Figura 4-29

- Nuevo subobjetivo marcado: h_9 .

2)

- h_9 no está en BH_0 .
- Conjunto conflicto formado para demostrar h_9 : R_2, R_3, R_8 .
- Regla seleccionada R_2 .

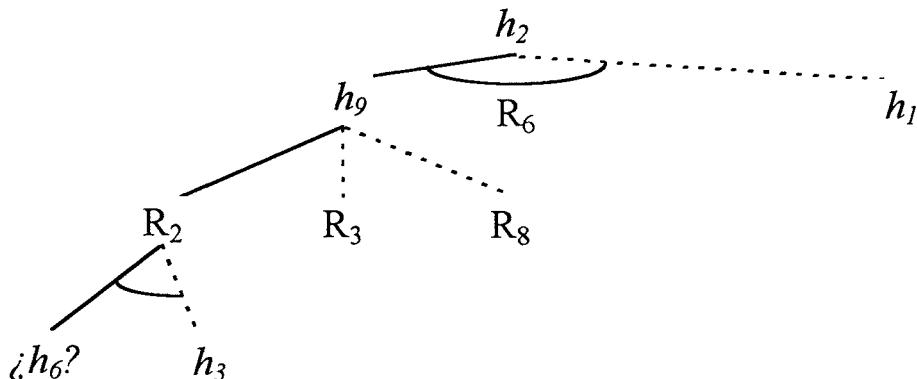


Figura 4-30

- Aparecen dos nuevos subobjetivos en el antecedente de R_2 : h_6 y h_3 . Se estudiará en primer lugar h_6 (esta elección es arbitraria).

3)

- h_6 no está en BH_0 .
- Conjunto conflicto formado para demostrar h_6 : R_7, R_9 .
- Regla seleccionada: R_7 .

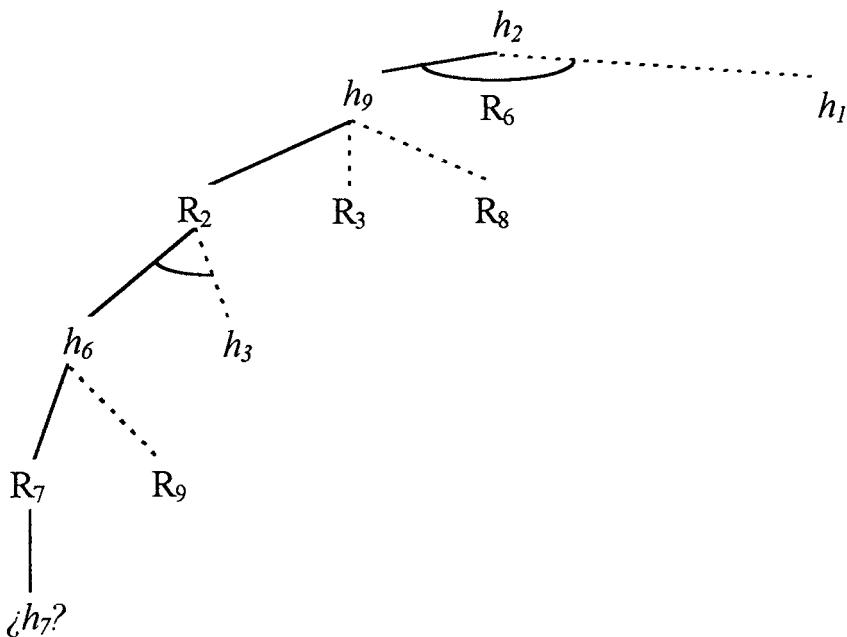


Figura 4-31

- Nuevo subobjetivo: h_7 .

4) h_7 está en BH_0 ; por tanto, h_6 quedará demostrado. Ahora el nuevo subobjetivo será h_3 :

- h_3 no está en BH_0 .
- No existe ninguna regla en cuyo consecuente aparezca h_3 .

Como consecuencia de lo anterior, h_3 no puede ser demostrado y el antecedente de R_2 no se cumple.

Para intentar demostrar h_9 habrá que recurrir a R_3 . h_7 , una de las condiciones del antecedente de R_3 , está en BH_0 . Queda por demostrar h_4 , que pasa a ser el nuevo hecho objetivo.

En la figura 4-32 aparece reflejado el razonamiento anterior. En dicha figura las líneas gruesas reflejan los hechos que se van verificando. Las líneas con trazo normal se refieren a zonas investigadas que se pondrán con trazo grueso sólo si hay

verificación. Finalmente, las líneas con trazo entrecortado reflejan zonas que no ha sido necesario investigar.

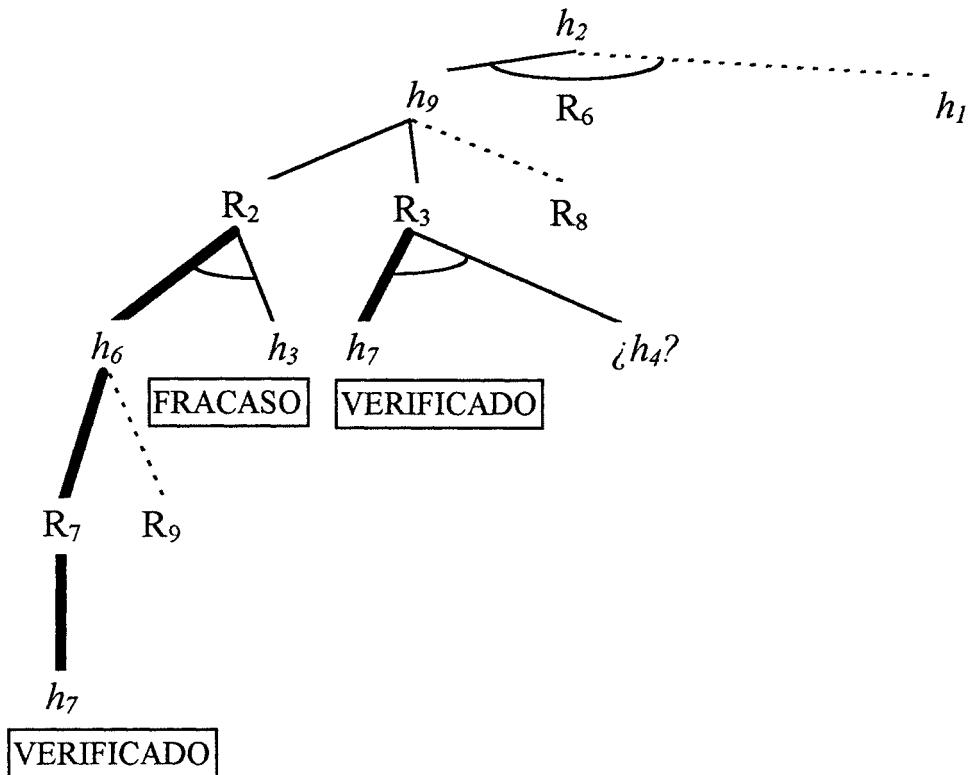


Figura 4-32

5)

- h_4 no está en BH_0 .
- Conjunto conflicto formado para demostrar h_4 : R_1 .
- Regla seleccionada: R_1 (ver figura 4-33).
- Nuevo subobjetivo: h_5 (h_8 está en BH_0 y h_6 había sido demostrado en un paso previo.).

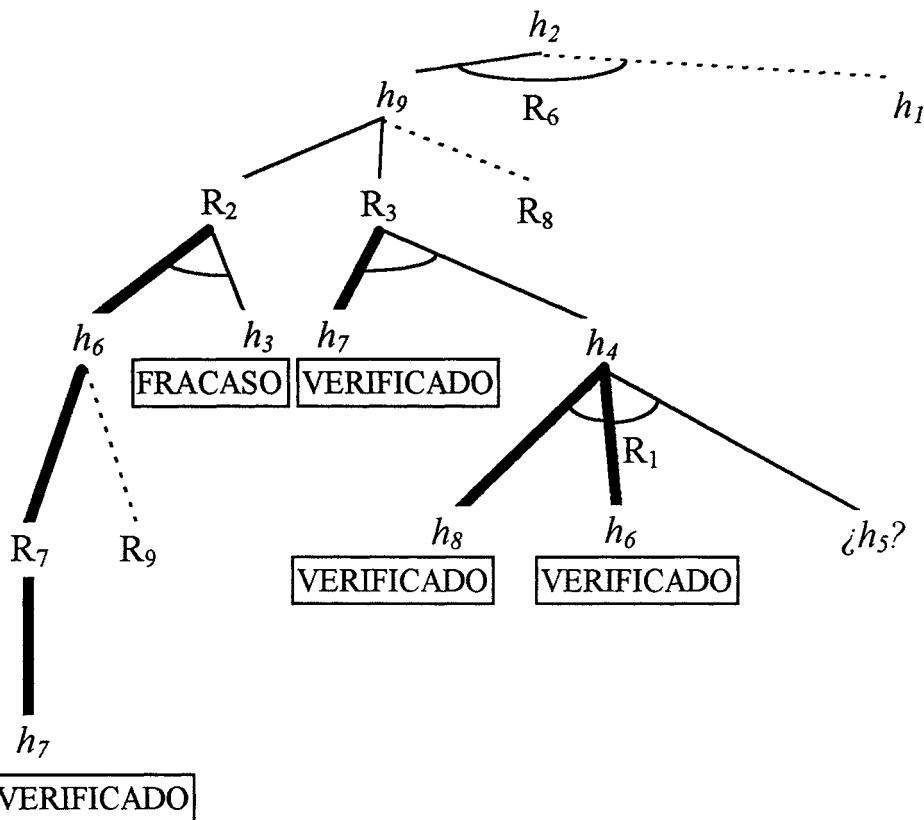


Figura 4-33

6)

- h_5 no está en BH_0 .
- Nuevo conjunto conflicto: R_5 .
- Regla seleccionada: R_5 .
- Nuevo subobjetivo: h_6 , que ya se había verificado en un paso anterior. Por tanto, h_5 queda demostrado y, como consecuencia de ello, h_4 también. A su vez, h_9 quedará también demostrado. La nueva situación queda reflejada en la figura 4-34.

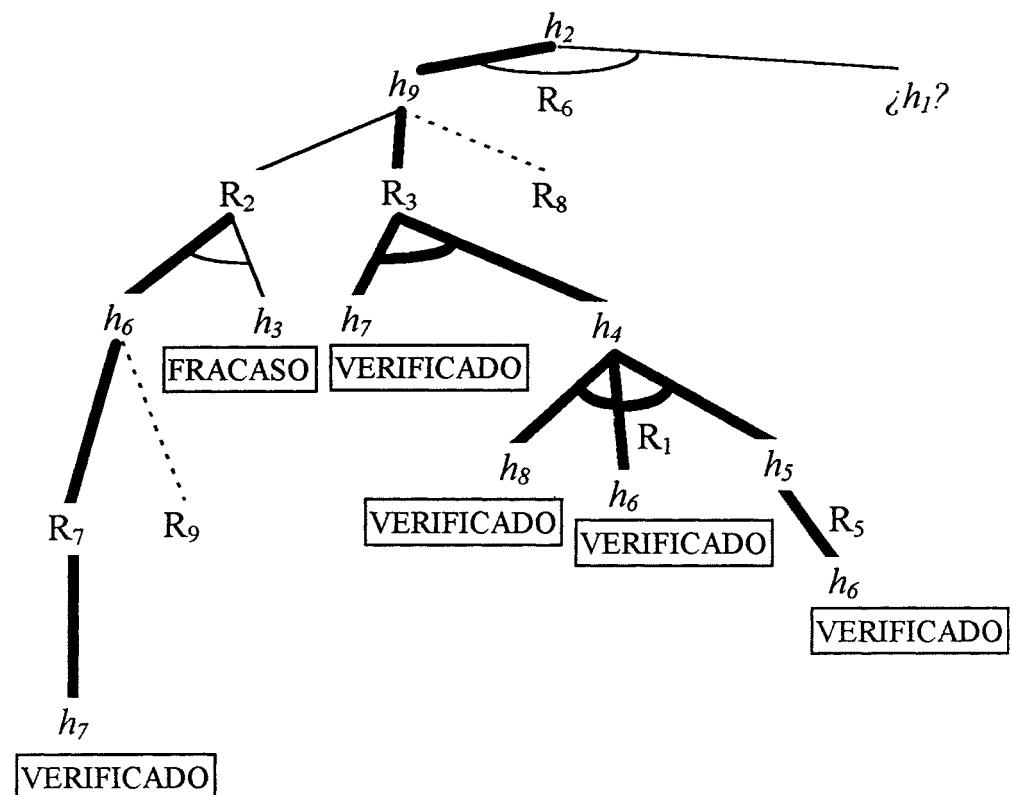


Figura 4-34

- Nuevo subobjetivo: h_1 .

7)

- h_1 no está en BH_0 .
- Conjunto conflicto: R_4 .
- Regla seleccionada: R_4 .
- Nuevo subobjetivo: h_8 , que está en BH_0 . Por tanto, h_2 queda finalmente verificado (ver figura 4-35).

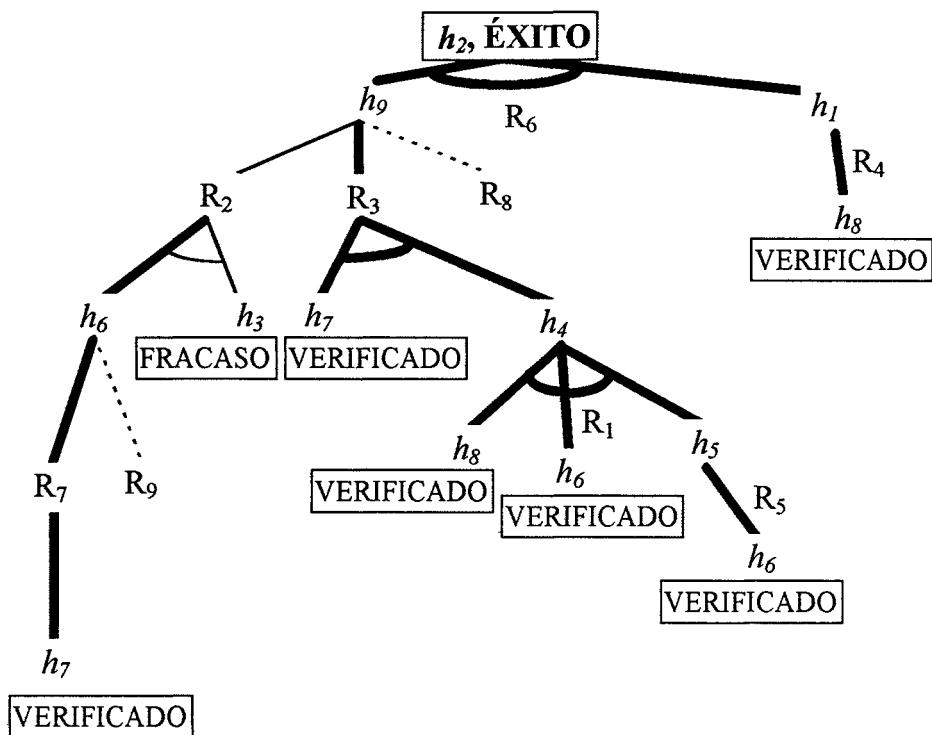


Figura 4-35

¿Qué dificultades puede haber encontrado en este problema?

Aplicación 1 de la regla A:

El lector no entiende los términos: “base de hechos”, “concepto meta”, “encadenamiento hacia atrás” o “resolución de conflictos”.

Acciones:

Leer las secciones 4.2.1, 4.2.2 y 4.2.3 del apartado teórico. Ampliar conocimientos con las lecturas recomendadas al final de este apartado teórico.

Aplicación 1 de la regla C:

Una vez consultado el enunciado, el lector es incapaz de resolver el problema.

Acciones:

Intentar resolver el problema para una base de conocimientos más reducida: $\{R_4, R_6, R_8\}$.

Aplicación 1 de la regla D:

El lector no consigue asimilar el concepto de “búsqueda en profundidad en un árbol Y/O” que aparece en la solución del problema.

Acciones:

Realizar la traza del algoritmo de búsqueda en profundidad del capítulo 1 para un árbol de estructura simple y repasar el concepto de árbol Y/O que aparece en el mismo capítulo.

• **PROBLEMA 4.4:** (adaptado de [Lucas & Van der Gaag, 1991])

Sea el conjunto de variables x_1, x_2, x_3 y x_4 , multivaluadas y la variable x_5 de un único valor. Dado un sistema basado en reglas cuya base de conocimientos contiene:

R₁: Si $x_1=a$ y $x_1=b$ entonces $x_3=f$

R₂: Si $x_1=b$ entonces $x_3=g$

R₃: Si $x_1=d$ y $x_5>0$ entonces $x_3=e$

R₄: Si $x_1=c$ y $x_5<30$ entonces $x_4=h$

R₅: Si $x_2=d$ y $x_5<10$ entonces $x_4=i$

R₆: Si $\text{conocido}(x_1)$ y $x_3\neq e$ entonces $x_2=d$

La base de hechos inicial (base de afirmaciones) contiene:

BH _i
$x_5=5$

Figura 4-36

Supóngase que el motor de inferencia puede solicitar los valores para todas las variables, excepto para x_4 y para x_2 . Considérese, también, que las respuestas del usuario benefician el proceso de inferencia y que la resolución del conjunto conflicto (reglas seleccionadas) se realiza aplicando primero las reglas de menor subíndice (así, R₁ se aplicaría antes que cualquier otra).

1) Teniendo como único objetivo el valor de la variable x_4 , aplique la estrategia de inferencia adecuada y justifique su utilización frente a otras. Señale el valor que finalmente tendrán la base de hechos y el valor del objetivo indicado.

2) ¿Qué hubiera ocurrido si se añadiera la siguiente regla?:

R₈: Si $\text{conocido}(x_5)$ entonces $x_1=c$ y retractar(x_2)

¿Qué supone utilizar reglas de este tipo?, ¿qué relación tiene con algún tipo de razonamiento lógico? Describa dicho tipo de razonamiento.

3) ¿Qué hubiera ocurrido si se añadieran las siguientes reglas?:

$R_9:$ Si conocido(x_1) entonces $x_2=e$

$R_{10}:$ Si conocido(x_2) entonces $x_2=f$ y retractar(x_1)

SOLUCIÓN:

1) En este caso se debe aplicar encadenamiento hacia atrás, ya que se conoce el objetivo a alcanzar. Si se aplicara encadenamiento hacia adelante, habría pasos en el proceso de inferencia que realmente no serían necesarios para llegar al objetivo.

a) La primera regla aplicable para determinar el valor de x_4 es R_4 , en cuyo consecuente figura $x_4=h$. En primer lugar habrá que examinar si las condiciones que aparecen en el antecedente de R_4 se cumplen o no para que esta regla pueda aplicarse.

$x_5 < 30$: Efectivamente esto se cumple ya que $BH_i = \{x_5=5\}$.

$x_1=c$: No se puede saber si esta condición es cierta, primero a partir de la base de afirmaciones y después a partir de la base de conocimientos, con lo que el motor de inferencia solicitará al usuario que le proporcione un valor para x_1 . El usuario, al intentar favorecer el proceso de inferencia, dará el valor c para x_1 .

Al final, por tanto, R_4 podrá ser aplicada, al cumplirse las condiciones de su antecedente, con lo que $x_4=(h)$ y:

BH
$x_5=5$
$x_1=(c)$
$x_4=(h)$

Figura 4-37

b) Dado que la variable x_4 es multivaluada, se sigue intentando aplicar el resto de las reglas de las que pueden derivarse nuevos valores para esta variable. R_5 sería la siguiente regla a considerar. Examinando las condiciones de su antecedente:

$x_5 < 10$: Se cumple ya que $x_5=5$ a partir de BH.

$x_2=d$: Como siempre, primero se comprueba si $x_2=d$ pertenece a la base de afirmaciones y, si no, se intenta inferir. R_6 es la única regla a través de la cual se puede obtener un valor para x_2 . Hay que estudiar las condiciones de su antecedente:

conocido(x₁): Se cumple al ser $x_1=(c)$.

$x_3 \neq e$: Esta condición es un poco especial en el sentido de que se trata de la negación de un hecho, en este caso $x_3=e$. Se supondrá que el sistema utiliza el *axioma del mundo cerrado* para el tratamiento de las negaciones, de modo que en realidad intentará deducir la condición $x_3=e$ y, si no lo consigue, supondrá que $x_3 \neq e$. Como en la base de afirmaciones no hay ningún valor para x_3 , hay que acudir a la base de conocimientos, en concreto a R_3 , para comprobar si $x_3=e$. Al no cumplirse la condición $x_1=d$ de R_3 , no se puede deducir $x_3=e$. Por tanto, a partir del axioma del mundo cerrado el sistema inferiría $x_3 \neq e$. Si se preguntara al usuario por x_3 , éste, al favorecer el proceso de inferencia, daría como respuesta que $x_3 \neq e$.

Como consecuencia de todo lo anterior, R_6 puede ejecutarse y, por tanto, R_5 también.

Finalmente: $x_4=(h, i)$ y:

BH
$x_5=5$
$x_1=(c)$
$x_3 \neq e$
$x_2=(d)$
$x_4=(h, i)$

Figura 4-38

- 2) Reglas como R_8 suponen realizar un razonamiento de tipo no monótono, pudiendo ocurrir que en un determinado momento se decida sacar cierta información de la base de afirmaciones que se considere que ya no pasa a ser cierta, con el consiguiente problema que ocasionaría comprobar todos los hechos deducidos anteriormente a partir del que ahora se ha descartado.

La inclusión de R_8 no introduciría ninguna diferencia en el resultado final, ya que ahora $x_1=c$ pasa a ser demostrado por R_8 y no es un dato introducido por el usuario. La acción $\text{retractar}(x_2)$ del consecuente de R_8 no tiene ningún efecto, ya que no hay ningún valor de x_2 en la base de afirmaciones cuando se considera esta regla.

3) La introducción de R_9 y R_{10} no produce ningún cambio respecto a los razonamientos hechos en el apartado anterior, ya que estas dos nuevas reglas son de menor prioridad que todas las que había con anterioridad en la base de conocimientos, las cuales eran suficientes para producir la activación de R_4 y R_5 y, además, en el proceso de deducción de x_4 no es necesario inferir $x_2=e$ y $x_2=f$.

• **PROBLEMA 4.5:** (adaptado de [Lucas & Van der Gaag, 1991])

Sea el conjunto de variables x_1, x_2, x_3, x_4 y x_5 multivaluadas y la variable x_6 univaluada. El motor de inferencia puede solicitar valores al usuario únicamente para x_1, x_3 y x_5 . La variable objetivo es x_6 . Considérese el siguiente conjunto de reglas:

R_1 : Si $x_5=a$ y $x_1=b$ entonces $x_4=c$

R_2 : Si $x_5=d$ y $x_4=c$ entonces $x_2=e$

R_3 : Si $x_4=c$ entonces $x_6=k$

R_4 : Si $x_1=j$ y $x_2=e$ entonces $x_6=h$

R_5 : Si $x_3=f$ y $x_1=g$ entonces $x_6=i$

(Los subíndices de cada regla marcan prioridades para casos de conflicto.)

Suponer que la base de hechos está inicialmente vacía y que las respuestas que da el usuario a las preguntas del sistema no favorecen el proceso de inferencia.

Describir la actuación de un motor de inferencia que aplicara encadenamiento hacia atrás en su intento de hallar un valor para x_6 .

SOLUCIÓN:

Inicialmente el motor de inferencia hallaría el siguiente conjunto conflicto de reglas: $\{R_3, R_4, R_5\}$. La primera regla en ser aplicada sería R_3 . Para determinar si la condición de esta regla, $x_4=c$, es cierta, se puede acudir al consecuente de R_1 . El nuevo conjunto conflicto estaría formado únicamente por esta regla. En su antecedente figuran las variables x_1 y x_5 para las que se solicitaría un valor al usuario, dado que no hay ninguna regla que permita inferirlas. Éste introduciría un valor distinto de a para x_5 y otro distinto de b para x_1 , con lo cual R_1 no podría ejecutarse.

La siguiente regla seleccionada del conjunto conflicto creado a partir de x_6 sería R_4 , que no pasaría a ser ejecutada, debido a que el usuario introduciría un valor para x_1 distinto de j . Si el valor devuelto hubiera sido por equivocación j (por ejemplo $x_7=j$) podría haberse contestado por el usuario al intentar aplicar previamente R_1), se debería haber examinado la segunda condición, $x_2=e$. R_2 es la única regla que tiene en su consecuente $x_2=e$. Su primera condición, $x_5=d$, no sería cierta, ya que el usuario daría un valor para x_5 distinto de d . Si por equivocación el valor devuelto hubiera sido d , habría que examinar la segunda condición, $x_4=c$. Esto último ya se describió cuando R_3 había sido la regla seleccionada del conjunto conflicto. Es conveniente observar que un buen motor de inferencia habría anotado que la condición $x_4=c$ no se satisfacía, de manera que si posteriormente esta condición volvía a ser examinada, ya se sabría que en realidad era falsa. Por tanto, R_4 finalmente no se ejecutaría.

La última regla del conjunto conflicto es R_5 . Su condición $x_3=f$ nunca sería cierta, debido a los valores que para la misma introduciría el usuario. Por tanto, no se puede asignar ningún valor a x_6 al final del proceso.

Lo que se ha pretendido en realidad con este problema es ir examinando todas las posibilidades que el motor de inferencia tendría para asignarle un valor a la variable objetivo. Si, por ejemplo, después de examinar el antecedente de R_3 , esta regla hubiera podido ejecutarse, se habría asignado a x_6 el valor k y el proceso se hubiera dado por finalizado. Esto sería así ya que x_6 es una variable univaluada y bastaría con que le asignara un único valor aquella regla más prioritaria del conjunto conflicto que realmente pudiera ejecutarse. Si x_6 hubiera sido una variable multivaluada, se habría intentado aplicar todas las reglas del conjunto conflicto.

- **PROBLEMA 4.6:** (adaptado de [Lucas & Van der Gaag, 1991])

Dado un sistema basado en reglas con la siguiente base de conocimientos:

R₁: Si $x_3 < 20$ y no-conocido(x_2) entonces $x_4 = b$

R₂: Si $x_1 = c$ y conocido(x_2) entonces $x_4 = d$

R₃: Si $x_1 \neq b$ y $x_3 > 100$ entonces $x_2 = f$

Las variables x_1 , x_2 y x_3 son univaluadas y la variable x_4 es multivaluada. Las variables x_1 y x_3 son las únicas para las que el motor de inferencia puede solicitar un valor al usuario. La variable objetivo es x_4 .

Suponiendo que la base de hechos inicial es:

BH _i
$x_1=c$
$x_3=5$

Figura 4-39

a) Determinar la base de hechos que resulta de aplicar encadenamiento hacia atrás a este conjunto de reglas. ¿Qué reglas se ejecutan y cuáles no? En caso de conflicto entre reglas, aplicar la estrategia de menor subíndice para la resolución del mismo.

b) Supóngase que la siguiente regla se añade a las anteriores:

R₄: Si no-conocido(x_2) entonces $x_2=a$

Empleando de nuevo encadenamiento hacia atrás y teniendo la misma base de hechos inicial, ¿qué base de hechos se obtiene?

SOLUCIÓN:

a) Siendo x_4 la variable objetivo, el primer conjunto conflicto que se forma está constituido por las reglas en cuyo consecuente figura dicha variable: {R₁, R₂}. Al ser R₁ más prioritaria, son las condiciones de su antecedente los primeros subobjetivos que se fijan:

$x_3 < 20$: Es cierto al ser $x_3=5$, según la base de hechos inicial.

no-conocido(x_2): En primer lugar, se sabe que x_2 no está en la base de hechos inicial. Habrá que averiguar, no obstante, si se puede inferir algún valor para x_2 a partir de la base de conocimientos. R₃ tiene esta variable en su consecuente, con lo que pasa a formar un nuevo conjunto conflicto. Los nuevos subobjetivos que se forman son: $x_1 \neq b$ y $x_3 > 100$. A partir de la base de hechos se sabe que, aunque $x_1 \neq b$, x_3 no es mayor que 100. Por tanto, R₃ no puede ejecutarse y no ha podido inferirse ningún valor para x_2 que es una variable de la que no se puede solicitar un valor al usuario. Como consecuencia de todo lo anterior, no-conocido(x_2) es verdadero.

Al cumplirse las condiciones que figuran en el antecedente de R₁, esta regla pasa a ser ejecutada, quedando la nueva base de hechos como sigue:

BH_2
$x_1=c$
$x_3=5$
$x_4=(b)$

Figura 4-40

Queda por examinar R_2 de las reglas que formaban el conjunto conflicto. Los nuevos subobjetivos que hay que evaluar son:

$x_1=c$: Es cierto a partir de BH_2 .

conocido(x_2): Ya se había comprobado con anterioridad que esta condición es falsa.

Por tanto, R_2 no puede ejecutarse y $BH_{final}=BH_2$.

b) La diferencia que introduce R_4 con respecto al proceso seguido en el apartado anterior es que hace que la condición **no-conocido(x_2)** de R_1 sea ahora falsa. En efecto, el conjunto conflicto que surge para la evaluación de la mencionada condición es $\{R_3, R_4\}$. Ya se comprobó con anterioridad que R_3 no podía ejecutarse. No ocurre así para R_4 , ya que su condición **no-conocido(x_2)** es verdadera (R_3 , como ya se había visto, no puede proporcionar ningún valor para x_2), con lo que pasa a ser $x_2=a$. Como ahora el valor de x_2 sí es conocido, R_1 no podrá ejecutarse y sí lo podrá ser R_2 . Al final:

BH_{final}
$x_1=c$
$x_3=5$
$x_2=a$
$x_4=(d)$

Figura 4-41

- **PROBLEMA 4.7:** (adaptado de [Lucas & Van der Gaag, 1991])

Dadas las siguientes reglas:

R_1 : Si $x_1=a$ y **conocido(x_2)** entonces $x_2=b$

R_2 : Si $x_1=c$ y $x_3<15$ entonces $x_4=d$

R_3 : Si $x_2=b$ y $x_3<5$ entonces $x_4=f$

donde todas las variables son univaluadas y la siguiente base de hechos inicial:

BH _i
1: $x_1=a$
2: $x_2=b$
3: $x_3=10$
4: $x_1=c$

Figura 4-42

(al lado de cada afirmación figura el ciclo en que se dedujo la misma), empleando encadenamiento hacia adelante:

- a) Dar todas las instancias de reglas creadas inicialmente a partir de BH_i y del conjunto de reglas dado.
- b) Ordenar el conjunto de instancias hallado en a) según el criterio de actualidad. ¿Qué instancia de regla se seleccionará para su ejecución?
- c) Dar la base de hechos resultante después de la evaluación de la instancia hallada en el apartado b). ¿Terminará finalmente la inferencia? Explicar la respuesta dando los cambios sucesivos que tienen lugar en la base de hechos.

SOLUCIÓN:

- a) Se tienen inicialmente las siguientes instancias de reglas:

$$(R_1, \{1: x_1=a, 2: x_2=b\})$$

$$(R_2, \{4: x_1=c, 3: x_3=10\})$$

- b) A través del criterio de actualidad se pretende seleccionar aquella instancia de regla tal que el conjunto de sus condiciones haya sido el más reciente en formarse. El mecanismo que se empleará será el de ordenar, para cada instancia, los tiempos en que cada condición se ha cumplido (de mayor a menor), completando con ceros a la derecha si fuera necesario. La ordenación final entre instancias se realiza teniendo en cuenta el valor de los números formados en el paso anterior.

Para el presente caso se tendría:

$$R_1: 1 \ 2$$

$$R_1: 2 \ 1$$

$$R_2: 4 \ 3 \longrightarrow (\text{Ordenación de las condiciones en cada regla}) \longrightarrow R_2: 4 \ 3$$

El orden final sería R_2 (43), R_1 (21). R_2 será, por tanto, la primera regla seleccionada para ejecución.

c) Tras la ejecución de la regla seleccionada en b):

BH
1: $x_1=a$
2: $x_2=b$
3: $x_3=10$
4: $x_1=c$
5: $x_4=d$

Figura 4-43

A continuación se ejecutará de nuevo la misma instancia del apartado anterior, ya que nada ha cambiado respecto a las variables que aparecen en los antecedentes de cada una de las 3 reglas. Si no se aplica ningún mecanismo de refractariedad que impida que dicha instancia vuelva a ser aplicada, ésta se ejecutará indefinidamente. Está claro que esto último no sería deseable, por lo que convendría dotar al sistema de un mecanismo que evitara la repetición continuada de la ejecución de la misma instancia de regla.

• **PROBLEMA 4.8:** (adaptado de [Lucas & Van der Gaag, 1991])

Considérese la siguiente base de hechos:

BH _i
1: $x_1=a$
2: $x_1=b$
3: $x_2=4$

Figura 4-44

y el siguiente conjunto de reglas:

R_1 : Si $x_1=a$ y $x_1=b$ entonces $x_3=e$

R_2 : Si $x_3=e$ y $x_4=g$ entonces $x_3=f$

R_3 : Si $x_2<10$ y ($x_1=a$ o $x_1=b$) entonces $x_4=g$

Si el motor de inferencia aplica encadenamiento hacia adelante y todas las variables son univalueadas:

- a) ¿Qué instancias de reglas se crean inicialmente?
- b) Aplicando el criterio de actualidad, ¿qué regla pasa a ejecutarse?
- c) Dar la base de hechos resultante al final del apartado b).

SOLUCIÓN:

- a) El conjunto conflicto inicial está formado por las siguientes instancias de reglas:

$$(R_1, \{1: x_1=a, 2: x_1=b\})$$

$$(R_3, \{3: x_2=4, 1: x_1=a\})$$

$$(R_3, \{3: x_2=4, 2: x_1=b\})$$

Obsérvese como han sido creadas dos instancias de la regla R_3 .

- b) Aplicando el criterio de actualidad para resolución de conflictos, la segunda instancia de R_3 sería la seleccionada para evaluación. Esto es así debido a que la etiqueta temporal del segundo hecho que figura en la misma es más reciente que la del mismo hecho que aparece en la primera instancia de R_3 .

- c) La evaluación de la segunda instancia de R_3 provoca la inclusión de $4: x_4=g$ en la base de hechos inicial:

BH _{final}
1: $x_1=a$
2: $x_1=b$
3: $x_2=4$
4: $x_4=g$

Figura 4-45

• PROBLEMA 4.9: (adaptado de [Lucas & Van der Gaag, 1991])

Dadas

R_1 : Si $x_1=a$ y $x_2=10$ entonces $x_3=b$

R_2 : Si $x_3=c$ y $x_2<20$ entonces $x_1=a$

y la siguiente base de datos:

BH _i
1: $x_1=a$
2: $x_3=c$
3: $x_2=10$
4: $x_3=c$

Figura 4-46

se utiliza encadenamiento hacia adelante para inferir nuevos hechos.

a) ¿Qué instancias de reglas se crearán en el primer paso de la inferencia y cuál de ellas se seleccionará para ejecución si se aplica el criterio de actualidad? Dar el nuevo conjunto de hechos obtenido después de la evaluación de la instancia obtenida. ¿Terminará finalmente la inferencia?

b) Suponer que se añade la siguiente regla a las anteriores, antes de consultar la base de hechos inicial:

$$R_3: \text{Si } x_3=b \text{ entonces } x_3=c$$

Dar la base de hechos resultante. ¿Terminará la inferencia? Comparar los resultados con los del apartado a).

SOLUCIÓN:

a) En el primer paso de la inferencia aparecen las siguientes instancias:

$$(R_1, \{1: x_1=a, 3: x_2=10\})$$

$$(R_2, \{2: x_3=c, 3: x_2=10\})$$

$$(R_2, \{4: x_3=c, 3: x_2=10\})$$

Aplicando el criterio de actualidad se seleccionará en primer lugar la última instancia de las que aparecen arriba:

$$R_1: 1 \ 3 \qquad R_1: 3 \ 1$$

$$R_2: 2 \ 3 \longrightarrow R_2: 3 \ 2 \longrightarrow 43 > 32 > 31 \Rightarrow (R_2, \{4: x_3=c, 3: x_2=10\})$$

$$R_2: 4 \ 3 \qquad R_2: 4 \ 3$$

La nueva base de hechos después de aplicar R_2 sería:

BH
1: $x_1=a$
2: $x_3=c$
3: $x_2=10$
4: $x_3=c$
5: $x_1=a$

Figura 4-47

Para esta nueva base de hechos se tendría el siguiente conjunto de instancias de reglas:

$$(R_1, \{1: x_1=a, 3: x_2=10\})$$

$$(R_1, \{5: x_1=a, 3: x_2=10\})$$

$$(R_2, \{2: x_3=c, 3: x_2=10\})$$

$$(R_2, \{4: x_3=c, 3: x_2=10\})$$

En este caso la segunda instancia sería la elegida para ejecutarse, añadiéndose a la base de hechos: 6: $x_3=b$

En los siguientes pasos siempre la instancia elegida será ésta misma, por lo que, si el sistema de inferencia dispusiera de un mecanismo de refractariedad que impida la ejecución de la misma instancia dos veces seguidas, el proceso de inferencia finalizaría.

b) Ahora se tiene:

$$R_1: \text{Si } x_1=a \text{ y } x_2=10 \text{ entonces } x_3=b$$

$$R_2: \text{Si } x_3=c \text{ y } x_2<20 \text{ entonces } x_1=a$$

$$R_3: \text{Si } x_3=b \text{ entonces } x_3=c$$

BH _i
1: $x_1=a$
2: $x_3=c$
3: $x_2=10$
4: $x_3=c$

Figura 4-48

Las instancias de reglas iniciales son las siguientes:

$(R_1, \{1:x=a, 3:y=10\})$

$(R_2, \{2:z=c, 3:y=10\})$

$(R_2, \{4:z=c, 3:y=10\}) \Leftarrow$ Instancia seleccionada

La nueva base de hechos después de ejecutar R_2 estaría constituida por:

BH ₁
1: $x_1=a$
2: $x_3=c$
3: $x_2=10$
4: $x_3=c$
5: $x_1=a$

Figura 4-49

En el siguiente ciclo:

$(R_1, \{1: x_1=a, 3: x_2=10\})$

$(R_1, \{5: x_1=a, 3: x_2=10\}) \Leftarrow$ Instancia seleccionada

$(R_2, \{2: x_3=c, 3: x_2=10\})$

$(R_2, \{4: x_3=c, 3: x_2=10\})$

Por tanto,

BH ₂
1: $x_1=a$
2: $x_3=c$
3: $x_2=10$
4: $x_3=c$
5: $x_1=a$
6: $x_3=b$

Figura 4-50

En el resto de pasos del proceso de inferencia se irán seleccionando cíclicamente las reglas R_3 , R_2 y R_1 debido al mecanismo de actualidad, con lo que el sistema nunca llegará a parar.

• PROBLEMA 4.10:

Se ha diseñado un sistema basado en reglas para el tratamiento de un determinado problema, habiéndose considerado apropiado dividir el tratamiento de este problema en varias etapas. Como consecuencia de ello, cualquier regla del sistema tiene la siguiente estructura:

```

IF etapai-está-activa AND
  condición1 AND
  condición2 AND
  ...
  THEN acción1 AND
        acción2 AND
        ...
  
```

Para el cambio de una etapa a otra se ha decidido diseñar reglas del siguiente tipo:

```

IF etapai-está-activa
THEN desactivar-etapai, AND
      activar-etapai+1
  
```

a) ¿Qué problema pueden plantear este tipo de reglas en relación al proceso de inferencia?

b) Sugerir una posible solución al problema del apartado anterior.

SOLUCIÓN:

a) El problema que se plantea es que si estamos en una determinada etapa j , podrá ejecutarse cualquier regla que tenga *etapa_i-está-activa* en su antecedente. Entre ellas se encuentra aquella regla que permite el paso de la etapa j a la $j+1$. Como en principio no hay nada que impida que esta regla se ejecute antes que todas las demás pertenecientes a la misma etapa, ocurriría que se estaría impidiendo que éstas últimas llevaran a cabo el trabajo para el que fueron diseñadas. Por tanto, existiría la posibilidad de pasar de una etapa a otra, aun habiendo reglas en la etapa anterior cuyo antecedente se cumpla. Esto, evidentemente, no es recomendable si se quiere que el sistema funcione correctamente.

b) Una solución al problema planteado en el apartado anterior sería emplear el criterio de especificidad para resolución de conflictos entre reglas. De esta manera, al ser la

regla de cambio de etapa menos específica que cualquiera otra de su misma etapa, se ejecutaría cuando ya no quedaran ninguna de éstas últimas cuyo antecedente se cumpliera.

• **PROBLEMA 4.11:** (adaptado de [Borrajo *et al.*, 1993])

Construir la red de redundancia temporal correspondiente a las siguientes reglas:

$$R_1: (A < x >) - (HB) \longrightarrow \dots$$

$$R_2: (C < y > D) (E < y >) \longrightarrow \dots$$

$$R_3: (F < y >) (AB) \longrightarrow \dots$$

$$R_4: (C < x > E) - (GK) \longrightarrow \dots$$

$$R_5: (HB) (C < y > D) (E < y >) \longrightarrow \dots$$

$$R_6: (AB) (F < y >) (HB) \longrightarrow \dots$$

Representar el estado de la red, si en la memoria de trabajo se encuentran los siguientes elementos:

(AB)

(CDD)

(FD)

(DE)

(HB)

Explicar qué ocurre, si secuencialmente:

- a) Se suprime (HB).
- b) Se añade (CDE).
- c) Se añade (AD).
- d) Se añaden (GK) (HB).

SOLUCIÓN:

¿Qué pretende este problema? En este problema se pretende resaltar cómo el diseño de una red de redundancia temporal, a partir de la base de conocimientos de un sistema basado en reglas, facilita el proceso de construcción del conjunto conflicto que se forma en cada ciclo del algoritmo encargado de aplicar encadenamiento hacia adelante en el sistema.

La red de redundancia temporal correspondiente a las seis reglas descritas se muestra en la figura 4-51.

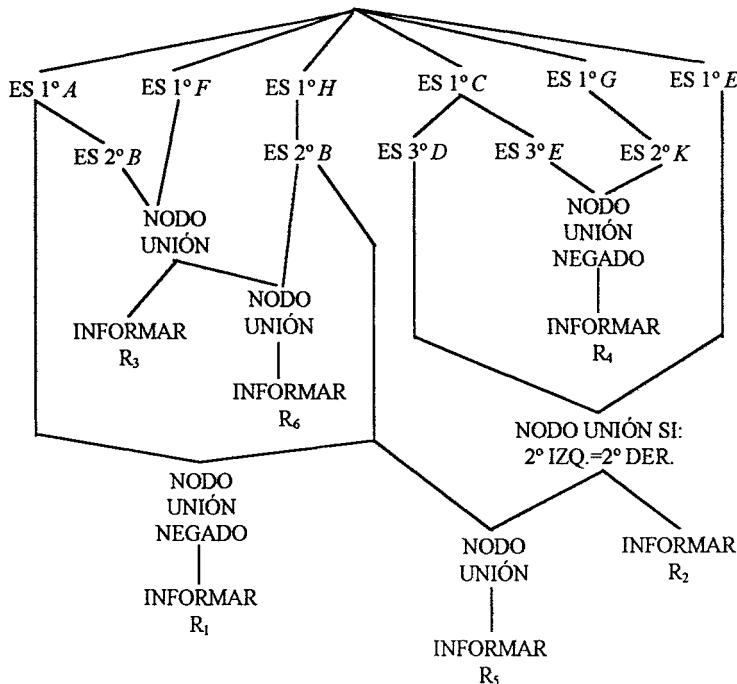


Figura 4-51

La segunda parte del problema solicita el estado de la red si se tuviera en este momento un número determinado de elementos. Así, el conjunto conflicto que se presentaría con esos elementos sería:

- $(R_2, \{(CDD) (ED)\})$
- $(R_3, \{(FD) (AB)\})$
- $(R_5, \{(HB) (CDD) (ED)\})$
- $(R_6, \{(AB) (FD) (HB)\})$

La tercera parte solicita, por orden secuencial, la ejecución de una serie de acciones; véase el conjunto conflicto que resultaría de la ejecución de cada una de las acciones, partiendo del conjunto conflicto de la segunda parte. Realizando la acción: suprimir (HB) , el conjunto conflicto sería:

(R₁, {(AB)})

(R₂, {(CDD) (ED)})

(R₃, {(FD) (AB)}))

Si ahora se añade (CDE), el conjunto conflicto quedaría:

(R₁, {(AB)})

(R₂, {(CDD) (ED)})

(R₃, {(FD) (AB)}))

(R₄, {(CDE)}))

Si a continuación se añade (AD), el conjunto conflicto resultante vendría a ser el siguiente:

(R₁, {(AB)})

(R₁, {(AD)}))

(R₂, {(CDD) (ED)})

(R₃, {(FD) (AB)}))

(R₄, {(CDE)}))

Si ahora se añaden (GK) y (HB), el conjunto conflicto final sería:

(R₂, {(CDD) (ED)})

(R₃, {(FD) (AB)}))

(R₅, {(HB) (CDD) (ED)})

(R₆, {(AB) (FD) (HB)}))

- **PROBLEMA 4.12:** (adaptado de [Lucas & Van der Gaag, 1991])

Dadas las reglas:

R₁: Si h_7 y (h_6 o h_5) entonces h_1 (0'7)

R₂: Si h_4 y h_3 entonces h_6 (0'5)

R₃: Si h_3 o h_2 entonces h_1 (0'3)

R₄: Si h_7 entonces h_4 (0'6)

R₅: Si h_8 entonces h_2 (0'2)

los valores que aparecen entre paréntesis al lado de las mismas son los factores de certeza asociados a cada regla y las prioridades en caso de conflicto se establecen favoreciendo los subíndices menores.

Se supone que h_1 es la hipótesis objetivo. Al aplicar encadenamiento hacia atrás, el usuario será consultado para que ofrezca información adicional sobre h_3 , h_5 , h_7 y h_8 . Se asume que, usando el conocimiento a priori e' , el usuario proporciona los siguientes factores de certeza:

$$FC(h_3, e') = 0'6$$

$$FC(h_5, e') = 0'4$$

$$FC(h_7, e') = 0'9$$

$$FC(h_8, e') = -0'3$$

Determinar el factor de certeza que resulta para h_1 según el método desarrollado en MYCIN.

SOLUCIÓN:

¿Qué pretende este problema? Junto con el problema 4.13, en este ejercicio se muestra cómo se aborda en el sistema experto MYCIN el tratamiento de la incertidumbre asociada al conocimiento del dominio.

La primera regla seleccionada para aplicación, usando encadenamiento hacia atrás, es R_1 . Habrá que determinar cuál es el grado de certeza para la condición h_7 y (h_6 o h_5), pero sólo se conocen los factores para las variables h_7 y h_5 ($0'9$ y $0'4$, respectivamente). Para calcular un factor de certeza para h_6 se acude a R_2 . Es necesario ahora hallar los factores asociados a las variables h_4 y h_3 por separado, para así poder hacerlo para la condición h_4 y h_3 y, por tanto, también posteriormente para h_6 . Se sabe que el factor de certeza asociado a h_3 es $0'6$ y, aplicando R_4 , se obtiene para h_4 : $0'6 \cdot 0'9 = 0'54$.

Llamando e_1' a toda la evidencia disponible hasta este punto, se tiene:

$$FC(h_4 \text{ y } h_3, e_1') = \min\{FC(h_4, e_1'), FC(h_3, e_1')\} = \min\{0'54, 0'6\} = 0'54$$

Luego,

$$FC(h_6, e_1') = FC(h_6, h_4 \text{ y } h_3) \cdot \max\{0, FC(h_4 \text{ y } h_3, e_1')\} = 0'5 \cdot 0'54 = 0'27$$

Ahora que se conocen los valores de los factores para h_7 , h_6 y h_5 por separado, se puede calcular:

$$FC(h_6 \text{ o } h_5, e_1') = \max\{FC(h_6, e_1'), FC(h_5, e_1')\} = 0'4$$

$$FC(h_7 \text{ y } (h_6 \text{ o } h_5), e_1') = \min\{FC(h_7, e_1'), FC(h_6 \text{ o } h_5, e_1')\} = 0'4$$

A partir de R₁ se obtiene el siguiente factor de certeza parcial para h₁:

$$FC(h_1, e_1') = FC(h_1, h_7 \text{ y } (h_6 \text{ o } h_5)) \cdot \max\{0, FC(h_7 \text{ y } (h_6 \text{ o } h_5), e_1')\} = 0'7 \cdot 0'4 = 0'28$$

Del mismo modo, a partir de la otra regla con h₁ en su consecuente, es decir, R₃, se obtiene el siguiente factor de certeza:

$$FC(h_1, e_2') = FC(h_1, h_3 \text{ o } h_2) \cdot \max\{0, FC(h_3 \text{ o } h_2, e_2')\} = 0'3 \cdot 0'6 = 0'18$$

En el proceso de desarrollo del último cálculo, se ha hallado un factor de certeza para h₂ igual a 0; esto es debido a que FC(h₈, e') = -0'3.

Combinando los resultados obtenidos para h₁:

$$FC(h_1, e_1' \text{ comb } e_2') = FC(h_1, e_1') + FC(h_1, e_2') \cdot (1 - FC(h_1, e_1')) = 0'28 + 0'18 \cdot 0'72 = 0'41$$

Cabe resaltar que se ha obtenido para h₁ un factor de certeza mayor que los que se habían obtenido por separado a partir de R₁ y R₃ (0'28 y 0'18, respectivamente).

- **PROBLEMA 4.13:** (adaptado de [Lucas & Van der Gaag, 1991])

Un sistema basado en reglas emplea el modelo de factores de certeza de MYCIN para tratar la incertidumbre que aparece en un determinado dominio. Sean las reglas:

R₁: Si h₂ o h₃ entonces h₆ (0'4)

R₂: Si h₆ y h₇ entonces h₁ (0'7)

R₃: Si h₄ o h₅ entonces h₁ (0'1)

Supóngase, además, que los atributos h₂, h₃, h₄, h₅ y h₇ se han establecido con factores de certeza 0'3, 0'6, 0'2, 0'8 y 0'9, respectivamente. El atributo h₁ es el atributo objetivo, empleándose encadenamiento hacia atrás. Calcular el factor de certeza que resulta para el atributo h₁.

SOLUCIÓN:

El conjunto conflicto inicial estaría formado por {R₂, R₃}. Examinando en primer lugar R₂, que es la regla más prioritaria, sería necesario conocer el factor de certeza para h₆ y h₇. Se conoce este factor para h₇, a partir de la evidencia inicial, pero no para h₆. Este último valor puede calcularse mediante la regla R₁. En efecto,

$$FC(h_2, e') = 0'3, FC(h_3, e') = 0'6 \Rightarrow FC(h_2 \text{ o } h_3, e') = \max\{0'3, 0'6\} = 0'6$$

Por tanto, a partir de R_1 :

$$FC(h_6, e_1') = FC(h_6, h_2 \text{ o } h_3) \cdot \max\{0, FC(h_2 \text{ o } h_3, e_1')\} = 0'4 \cdot 0'6 = 0'24$$

Luego,

$$FC(h_6 \text{ y } h_7, e_1') = \min\{FC(h_6, e_1'), FC(h_7, e_1')\} = \min\{0'24, 0'9\} = 0'24$$

$$FC(h_1, e_1') = FC(h_1, h_6 \text{ y } h_7) \cdot \max\{0, FC(h_6 \text{ y } h_7, e_1')\} = 0'7 \cdot 0'24 = 0'168$$

Ahora habrá que considerar la otra regla que formaba parte del conjunto conflicto inicial: R_3 . Se tiene:

$$FC(h_4, e_2') = 0'2, FC(h_5, e_2') = 0'8 \Rightarrow FC(h_4 \text{ o } h_5, e_2') = \max\{0'2, 0'8\} = 0'8$$

$$FC(h_1, e_2') = FC(h_1, h_4 \text{ o } h_5) \cdot \max\{0, FC(h_4 \text{ o } h_5, e_2')\} = 0'1 \cdot 0'8 = 0'08$$

Combinando las dos reglas que producen evidencia sobre h_1 , se obtiene finalmente:

$$\begin{aligned} FC(h_1, e_1' \text{ comb } e_2') &= FC(h_1, e_1') + FC(h_1, e_2') \cdot (1 - FC(h_1, e_1')) = \\ &= 0'168 + 0'08 \cdot 0'832 = 0'234 \end{aligned}$$

4.4 Contexto adicional

4.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

- Cuál es la estructura de una regla y de un sistema basado en reglas.
- Cómo se realiza la inferencia en un sistema basado en reglas y cómo operan los diferentes mecanismos de resolución de conflictos o control del razonamiento con los que puede ser dotado el sistema.
- En qué se basa y en qué consiste el algoritmo RETE.
- Cómo se realiza el tratamiento de la incertidumbre en el sistema experto basado en reglas MYCIN.

4.4.2 Pistas de autoevaluación

- Aunque queda fuera de los objetivos del presente texto, podría ser interesante que el lector intentara establecer las características con las que habría que dotar a un sistema basado en reglas para que tuviera capacidad de aprendizaje. Esto permitiría trazar un flecha en la figura 4-1 desde el motor de inferencia hasta la base de

conocimientos. El lector interesado en el campo del aprendizaje automático puede consultar las referencias bibliográficas dadas a lo largo del libro, donde en algunos capítulos se trata de manera más o menos profunda este tema.

- Se sugiere al lector que intente poner en práctica las ideas explicadas en este capítulo construyendo un pequeño sistema experto basado en reglas (ver apartado de software de apoyo).

- Repetir los problemas 4.2 y 4.3 si se suprimen las reglas R_2 y R_8 .

5 REDES ASOCIATIVAS

5.1 Contexto previo

5.1.1 ¿Por qué está aquí este capítulo?

Las redes asociativas encontraron su primera aplicación en inteligencia artificial en el campo del tratamiento del lenguaje natural. A finales de la década de los sesenta Ross Quillian denominó “red semántica” a un sistema que intentaba representar el significado de las palabras de la lengua inglesa; posteriormente, dicho término fue aplicándose a otros tipos diferentes de redes donde se representaban conceptos tan variados como proposiciones, estructura de objetos físicos... En realidad las redes han estado omnipresentes en toda la historia de la inteligencia artificial. Las redes taxonómicas y las redes bayesianas constituyen sendos ejemplos de cómo, variando el significado dado a nodos y enlaces de una red, se pueden obtener métodos de representación del conocimiento adecuados para tareas muy diversas.

La razón profunda por la que las redes siguen estando presentes en la mayoría de las técnicas de representación está en que incluidos en las mismas están los grafos y éstos son la estructura abstracta más común en toda la computación. Cualquier sistema que pueda describirse en términos de un conjunto de estados, lugares, entidades conceptuales, vértices, etc. (a los que llamamos **nodos**) y de las **conexiones** entre estos nodos, asociadas a relaciones de distinto tipo, puede ser representado por un **grafo** y una tabla de semántica donde se describe el significado en el dominio del observador y a nivel de conocimiento de cada nodo y cada arco.

5.1.2 ¿Dónde hay conocimiento teórico del campo?

En las siguientes referencias bibliográficas el lector puede ampliar conocimientos en relación a los diferentes tipos de redes empleados en inteligencia artificial:

Referencias básicas:

- [Mira *et al.*, 1995], capítulo 7.
- [Rich & Knight, 1994], capítulos 8, 9 y 10.

Referencias complementarias:

- [Díez, 1994]
- [Jackson, 1990], capítulo 9.
- [Lucas & Van der Gaag, 1991], capítulo 5.
- [Nilsson, 1987], capítulo 9.
- [Pearl, 1988]
- [Winston, 1994], capítulo 2.

5.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

- Conceptos básicos sobre árboles y grafos.
- Nociones elementales sobre probabilidad (conceptos básicos y teorema de Bayes).

5.1.4 ¿Qué software de apoyo está disponible?**• BAYES**

Permite la construcción de redes bayesianas únicamente con forma de árbol. Puede obtenerse en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/reasonng/probabl/bayes/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasonng/probabl/bayes/0.html)

• S-ElimBel

Se puede obtener información sobre este sistema y las redes bayesianas en general:
<http://www.spaces.uci.edu/thierry/elimbel>

• SNePS (*Semantic Network Processing System*)

En este sistema se implementa una teoría intensional sobre representación y razonamiento en conocimiento proposicional. SNePS incluye un módulo para la creación y acceso a redes proposicionales. Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/kr/systems/sneps/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/kr/systems/sneps/0.html)

5.2 Contenido teórico

5.2.1 Introducción

Una red está formada por un conjunto de nodos unidos entre sí por enlaces de diferente tipo. En las *redes asociativas* cada nodo representa un concepto y los enlaces

corresponden generalmente a diferentes tipos de relaciones entre los conceptos mencionados.

5.2.2 Redes relacionales

5.2.2.1 Modelo de memoria semántica de Quillian

El modelo de memoria semántica de Quillian intenta representar el significado de cada palabra tal como figura en un diccionario. Cada definición de un término se representa en un plano donde aparece un *nodo tipo* (encerrado en un óvalo) que representa el concepto definido y varios *nodos réplica* que desarrollan la definición. Existen seis tipos de enlaces en este formalismo:

1) **Subclase**: une un nodo tipo con la clase a la que pertenece.

2) **Modificación**: indica que un nodo réplica modifica a otro nodo réplica del mismo plano.

3) **Disyunción**.

4) **Conjunción**.

5) **Propiedad**: es de la forma que aparece en la figura 5-1.

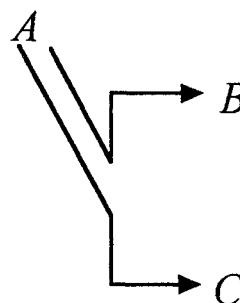


Figura 5-1

donde *A* es una relación, *B* el sujeto y *C* el objeto.

6) **Referencia al tipo**: une un nodo réplica con su correspondiente nodo tipo en un plano diferente.

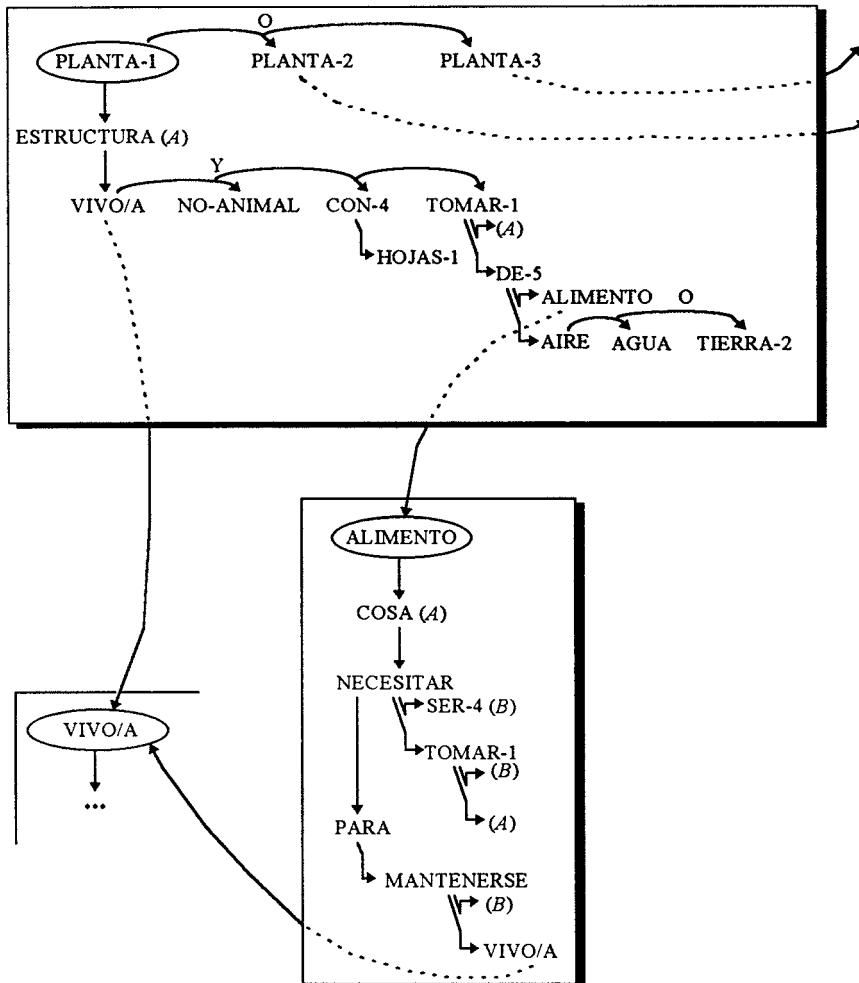


Figura 5-2 Red semántica de Quillian correspondiente al significado de dos palabras:
Planta-1: estructura viva que no es un animal, tiene hojas y toma su alimento del aire, del agua o de la tierra.

Alimento: una cosa que un ser tiene que tomar dentro de sí para mantenerse vivo.

La estructura de una red semántica de Quillian dependerá del idioma particular que se esté considerando. Lo ideal sería que esto no fuera así, pero para ello habría que pasar de la representación de palabras a la representación de conceptos. En las redes semánticas de Quillian no se puede extraer nueva información mediante inferencia, sino simplemente realizar comparaciones del significado de las palabras.

PROBLEMAS RELACIONADOS: 5.1, 5.2

5.2.2.2 Grafos de dependencia conceptual de Schank

Los grafos de dependencia conceptual están destinados a la comprensión del lenguaje natural, aunque más que intentar representar palabras, como en el caso del modelo de memoria semántica de Quillian, intentan representar conceptos. En los grafos de dependencia conceptual, dadas dos sentencias diferentes con el mismo significado, tendrán la misma representación y, por otra parte, cualquier información implícita deberá hacerse explícita en la representación del significado de una sentencia. Se distinguen 6 categorías conceptuales:

- PP (objetos del mundo real)
- ACT (acciones del mundo real)
- PA (atributos de objetos)
- AA (atributos de acciones)
- T (tiempos)
- LOC (posiciones)

Algunas de las leyes que rigen la formación de **relaciones** a partir de las categorías anteriores son las siguientes:

PP \longleftrightarrow ACT : un actor actúa.

PP \longleftrightarrow PA : un objeto posee un cierto atributo.

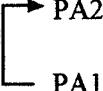
ACT \xleftarrow{O} PP : objeto de una acción.

ACT \xleftarrow{R} $\begin{array}{l} \text{PP} \\ \text{PP} \end{array}$: indica quiénes son el receptor y el dador en el curso de una acción.

ACT \xleftarrow{D} $\begin{array}{l} \text{PP} \\ \text{PP} \end{array}$: indica la dirección de un objeto en una acción.

 ACT : indica la conceptualización instrumental de una acción.

 : indica que la conceptualización X ha causado la Y.

 PP : indica un cambio de estado de un objeto.

PP1 ← PP2 : indica que o bien PP2 es parte de PP1 o bien PP2 es el poseedor de PP1.

Los principales modificadores de enlace son:

- p (pasado)
- f (futuro)
- ningún modificador (presente)
- ts = x (una relación que comienza en el tiempo x)
- tf = x (la relación termina en el instante x)
- c (condicional)
- / (negación)
- ? (pregunta)

Finalmente, las acciones primitivas que Schank considera suficientes para representar gran parte de las sentencias del lenguaje natural son:

ATRANS: Transferencia de una relación abstracta tal como posesión, propiedad, control.

PTRANS: Cambio de la posición física de un objeto.

MTRANS: Transferencia de información mental (decir, contar, comunicar).

PROPEL: Aplicación de una fuerza física a un objeto.

MOVEL: Movimiento de una parte del cuerpo.

GRASP: Acto mediante el cual un actor coge un objeto.

INGEST: Acto de ingerir.

CONC: Conceptualización o pensamiento acerca de una idea por un actor.

EXPTEL: Expulsión desde el cuerpo de un animal al exterior.

MBUILD: Construcción por un sujeto de una nueva información a partir de otra ya existente.

ATTEND: Acción de dirigir un órgano de los sentidos hacia un objeto.

SPEAK: Acción de producir sonidos con la boca.

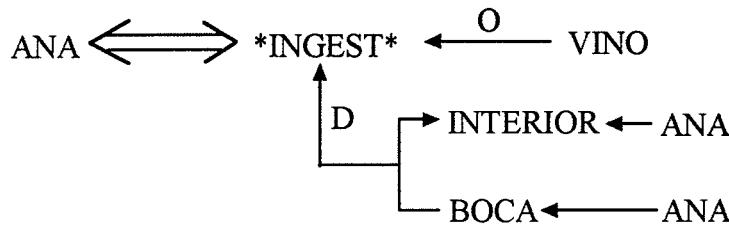


Figura 5-3 Grafo de dependencia conceptual correspondiente a la frase: “Ana bebe vino”.

PROBLEMAS RELACIONADOS: 5.3 a 5.5, 5.8

5.2.3 Redes proposicionales

5.2.3.1 Redes de Shapiro y grafos de Sowa

Tanto las *redes de Shapiro* como los *grafos de Sowa* constituyen un caso particular de red proposicional. En este tipo de redes se pueden expresar relaciones globales entre frases; no ocurre así en las llamadas *redes relacionales*, donde sólo aparecen aquellas relaciones que se pueden establecer dentro de una frase como son: sujeto, objeto directo, etc. Por tanto, en las redes proposicionales van a existir nodos que representan frases enteras, proposiciones, párrafos o incluso historias completas. Por otra parte, los enlaces que se establecen entre estos nodos van a servir para expresar características del lenguaje natural, tales como conjunciones, complementos de verbo, tiempos y modos verbales, etc. La frase “Juan piensa que Andrés está comiendo un helado” se representaría en estos dos formalismos de la siguiente forma:

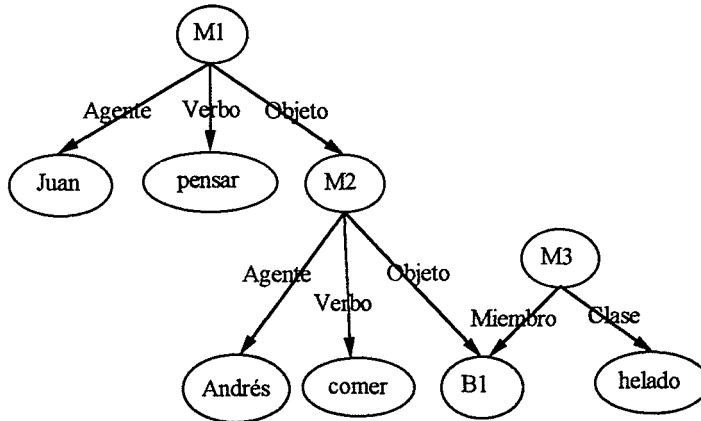


Figura 5-4 Ejemplo de red proposicional de Shapiro.

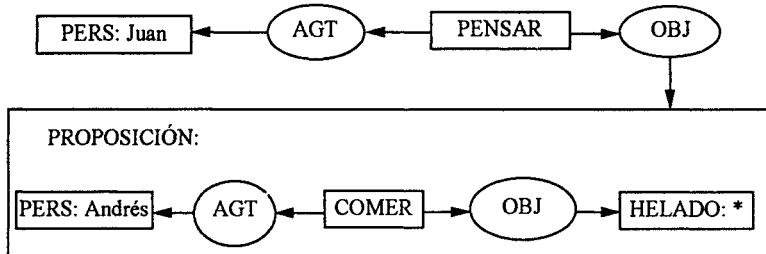


Figura 5-5 Ejemplo de grafo de Sowa.

Las representaciones gráficas de Shapiro y Sowa pueden expresarse en *notación lineal*, de manera que los nodos aparecerán entre corchetes y las etiquetas de los arcos entre paréntesis. Esta notación tiene como ventaja que ocupa menos espacio que los grafos correspondientes y es más fácil de manipular por un programa de ordenador. La frase-ejemplo anterior en esta representación quedaría de la siguiente forma:

[PENSAR] -
 → (AGT) → [PERS: Juan]
 → (OBJ) → [PROPOSICIÓN: [COMER] -
 (AGT) → [PERS: Andrés]
 (OBJ) → [HELADO]]

PROBLEMAS RELACIONADOS: 5.6 a 5.8, 5.10

5.2.4 Redes de clasificación

En este tipo de redes los arcos establecen una relación de orden parcial sobre los nodos, formando lo que se llama una *jerarquía de conceptos*. Un arco trazado desde un nodo *A* hasta un nodo *B* especifica que el concepto *A* es más general que el *B*. Una de las principales utilidades de agrupar un conjunto de conceptos en una *red jerárquica* es que un concepto pueda heredar las propiedades de sus antepasados, lo cual constituye un tipo particular de inferencia.

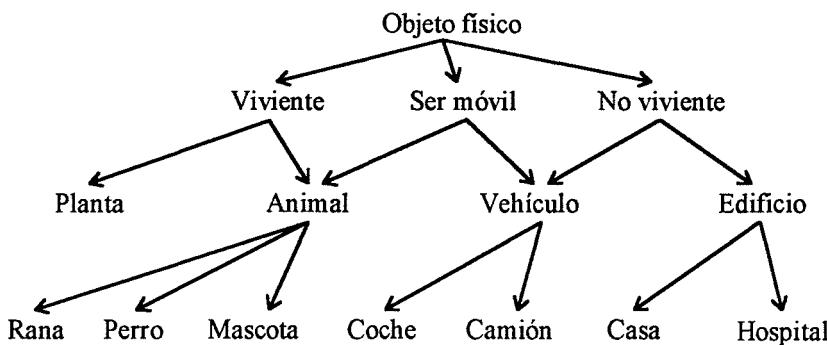


Figura 5-6 Ejemplo de red de clasificación.

5.2.5 Redes causales

En este tipo de redes los nodos representan variables y los enlaces relaciones de influencia o causalidad entre las mismas. Debido a lo anterior, son adecuados para abordar problemas de diagnóstico.

CASNET (Causal ASsociational NETwork) fue el primer sistema experto basado en una red causal. Desarrollado en los años 70, servía de apoyo a los médicos en el diagnóstico y tratamiento del glaucoma, una enfermedad ocular.

Las redes bayesianas constituyen el tipo de red causal que más importancia ha cobrado en los últimos años. A ellas está dedicado el siguiente apartado.

5.2.5.1 Redes bayesianas

Los primeros sistemas de diagnóstico basados en métodos probabilísticos —surgidos antes de la aparición de las redes bayesianas— se basaban en el *método clásico de diagnóstico probabilista* para la realización de la inferencia. En este

método, a la hora de determinar cuál es la probabilidad de un determinado diagnóstico dados ciertos síntomas, se parte del *teorema de Bayes* y se añaden las hipótesis de *exclusividad* (dos diagnósticos no pueden ser ciertos a la vez) y *exhaustividad* (no hay otros diagnósticos posibles aparte de los considerados), además de otra hipótesis adicional de *independencia condicional* (la probabilidad de que dado un diagnóstico se encuentre un determinado hallazgo es independiente de que se hayan encontrado o no otros hallazgos). Por tanto, para un diagnóstico d_i y m hallazgos E_1, \dots, E_m se tiene la siguiente fórmula:

$$P(d_i|e_1, \dots, e_m) = \frac{P(e_1|d_i) \cdot \dots \cdot P(e_m|d_i) \cdot P(d_i)}{\sum_j P(e_1|d_j) \cdot \dots \cdot P(e_m|d_j) \cdot P(d_j)}$$

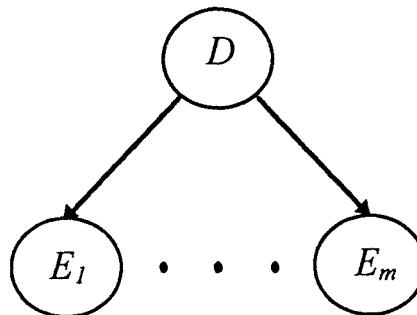


Figura 5-7 Método clásico de diagnóstico probabilista. La variable D puede tomar n valores: los n diagnósticos d_i posibles.

Las hipótesis introducidas en el método clásico de diagnóstico probabilista no se ajustan, en general, a la realidad. Este inconveniente fue superado a partir de la aparición de las redes bayesianas, que pueden diagnosticar varias alteraciones presentes simultáneamente y aplican la hipótesis de independencia condicional, no globalmente sobre el conjunto de diagnósticos y hallazgos, sino sólo localmente sobre los efectos inmediatos de una alteración.

- **Definición de red bayesiana**

Una red bayesiana es un grafo dirigido acíclico conexo más una distribución de probabilidad sobre sus variables, la cual cumple la propiedad de *separación direccional*.

nal. La propiedad de separación direccional expresa que la probabilidad de una variable X , una vez determinados los valores de los padres de X , es independiente de los demás nodos o variables que no son descendientes de X .

- **Inferencia en redes bayesianas**

Consiste en fijar en la red el valor de las variables conocidas y calcular la probabilidad de las variables cuyo valor se desconoce. Para redes pequeñas se puede conseguir este objetivo realizando cálculos sencillos (ver problemas más adelante), aunque para redes mayores se utilizan algoritmos más eficientes basados en el paso de mensajes numéricos entre nodos a través de sus enlaces y donde el cálculo de la probabilidad se puede hacer de forma distribuida, asociando cada nodo a un procesador físico.

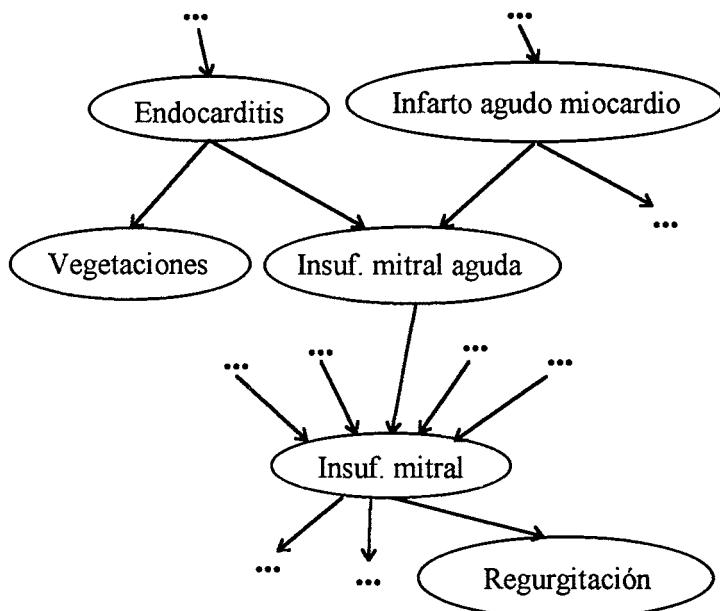


Figura 5-8 Porción de la red bayesiana del sistema experto DIAVAL, desarrollado por F. J. Díez Vegas, que sirve de ayuda al diagnóstico de enfermedades del corazón, especialmente valvulopatías.

5.3 Problemas resueltos

- **PROBLEMA 5.1: (*)**

Represente el esquema de una red semántica de Quillian correspondiente a las definiciones de diccionario de los términos siguientes:

Lima-1: alimento de sabor algo dulce y jugoso en forma esferoidal.

Lima-2: árbol que da la lima.

Lima-3: instrumento hecho de acero para pulir materiales.

Esferoidal: superficie que tiene forma de esfera.

Analice las ventajas de los grafos de dependencia conceptual de Schank frente a este tipo de redes.

SOLUCIÓN:

¿Qué pretende este problema? En este problema y en el 5.2 aparecen representaciones gráficas de redes de Quillian que ayudan a entender las ideas en que se basa este formalismo de representación.

Para el caso concreto presentado en el enunciado del problema se tiene la figura 5-9.

Como principales ventajas de los grafos de dependencia conceptual de Schank frente a las redes semánticas de Quillian se pueden citar:

- La estructura de una red de Quillian depende del idioma particular que se esté considerando. Esto no es así en los grafos de dependencia conceptual.
- La utilización de primitivas en los grafos de dependencia conceptual de Schank permite la creación de un intérprete capaz de realizar inferencias. Sin embargo, en las redes semánticas de Quillian no se pueden realizar inferencias que aporten nueva información al sistema, sino sólo comparaciones del significado de las palabras representadas.

- **PROBLEMA 5.2:**

Considérese la siguiente oración:

SAQUÉ EL LIBRO DE LA BIBLIOTECA.

Esta sentencia puede tener dos significados distintos:

a) *Saqué prestado un libro de la biblioteca.*

b) *Saqué de la biblioteca un libro que me pertenecía y que había llevado allí para poder estudiar con él.*

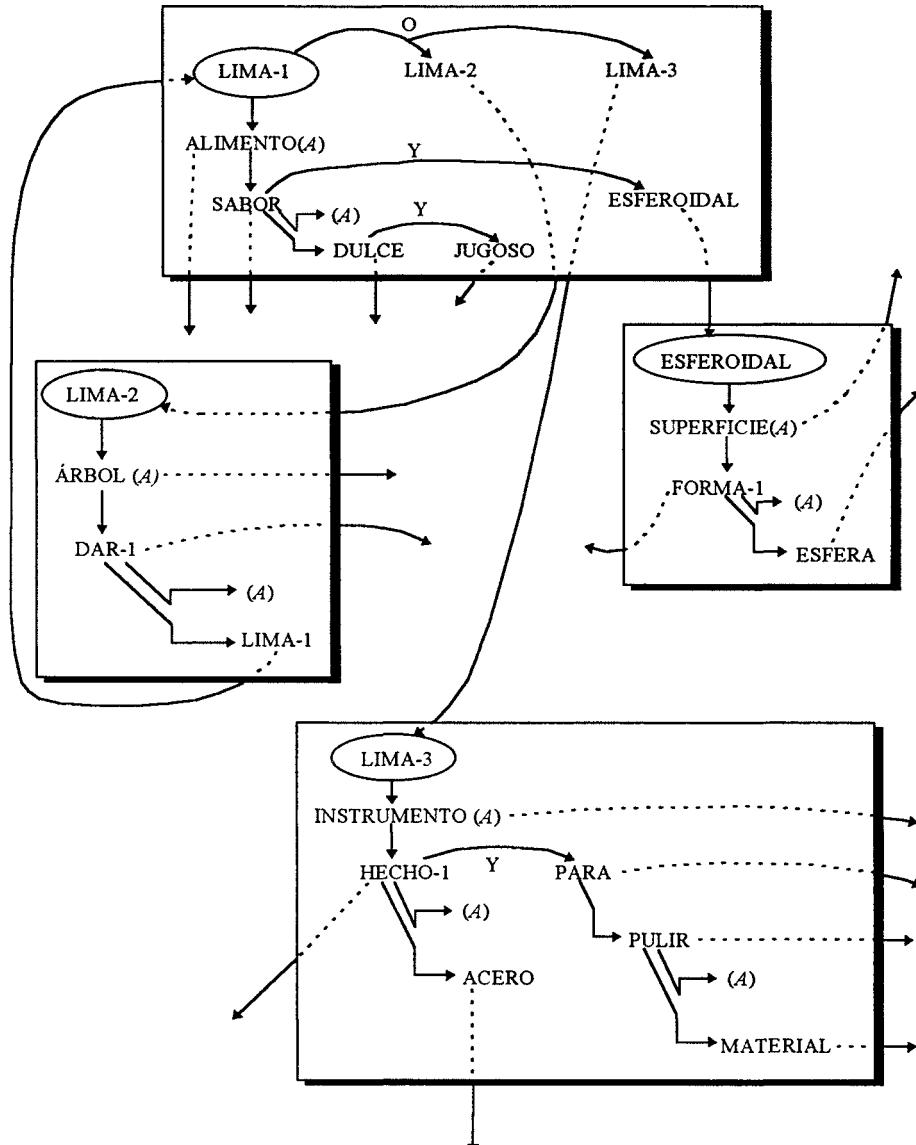


Figura 5-9

Utilizando el formalismo gráfico del modelo semántico de Quillian, representar los dos significados de la oración mencionada.

SOLUCIÓN:

- a) En este caso “DE LA BIBLIOTECA” modifica al sustantivo “LIBRO”, ya que dicho libro pertenece a los depósitos de la biblioteca. Por tanto, se tendría la siguiente representación:

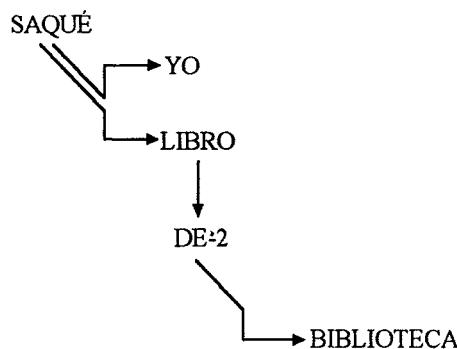


Figura 5-10

- b) Ahora “DE LA BIBLIOTECA” modifica al verbo “SAQUÉ”. Por otra parte, la preposición “DE”, que antes tenía un significado de pertenencia (DE-2), pasa a hacer referencia a una determinada localización (DE-3). Se tiene:

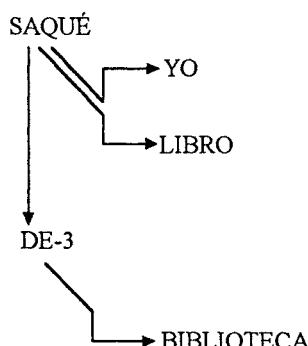


Figura 5-11

Los números en “DE-2” y “DE-3” se han escogido arbitrariamente.

• PROBLEMA 5.3:

Representar mediante grafos de dependencia conceptual las situaciones siguientes:

- Juan toma un calmante.
- Juan paga la factura.
- Luis sacó el perro a la calle.

SOLUCIÓN:

¿Qué pretende este problema? Desde este problema y hasta el 5.5 se consideran varios ejemplos de representación de oraciones simples mediante grafos de dependencia conceptual. En el problema planteado podrían construirse los siguientes grafos:

- Juan toma un calmante.

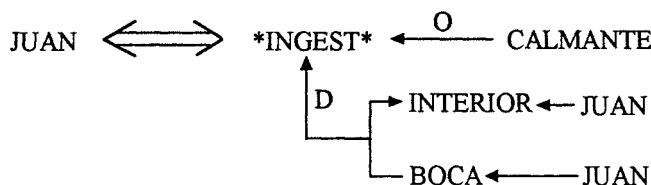


Figura 5-12

En el grafo aparece explícitamente la forma en que el calmante es ingerido, es decir, desde la boca se introduce en el interior de Juan.

- Juan paga la factura.

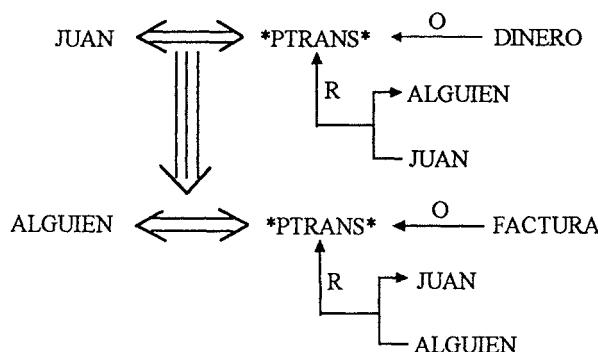


Figura 5-13

Obsérvese cómo el hecho de que Juan reciba la factura provoca el que pague el dinero correspondiente. Se está suponiendo, además, que el pago es en efectivo.

- c) Luis sacó el perro a la calle.

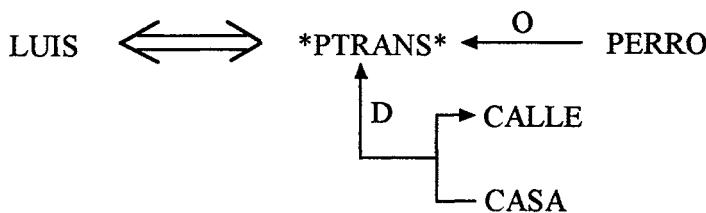


Figura 5-14

¿Qué dificultades puede haber encontrado en este problema?

Aplicación 1 de la regla A:

El lector no entiende el término “grafo de dependencia conceptual”.

Acciones:

Leer sección 5.2.2.2 del apartado teórico. Es posible ampliar conocimientos sobre este tema en: [Mira *et al.*, 1995], sección 7.1.3 o en [Rich & Knight, 1991], sección 10.1.

Aplicación 1 de la regla C:

El lector no es capaz de dibujar los grafos pedidos.

Acciones:

Conviene que repase detenidamente de qué acciones primitivas, tipos de enlace y categorías conceptuales se dispone en este formalismo.

• PROBLEMA 5.4:

Representar mediante grafos de dependencia conceptual el significado de las siguientes frases:

- ¿Juan bebe vino?
- José impidió que Ana le diera un libro a Luis.
- Antonio dará clases de francés a su hermano.

SOLUCIÓN:

a) ¿Juan bebe vino?

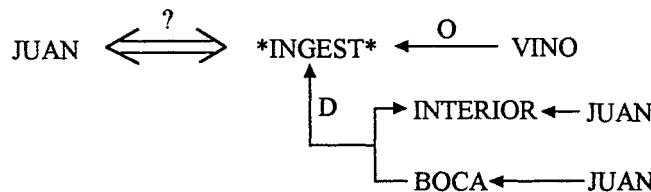


Figura 5-15

b) José impidió que Ana le diera un libro a Luis.

Como no se sabe qué es lo que hizo exactamente José para impedir que Ana le diera un libro a Luis, se utiliza la acción *DO* para expresar este hecho.

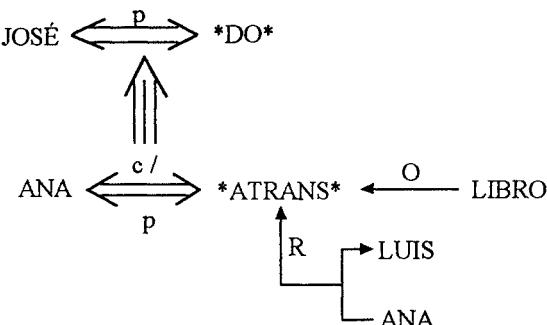


Figura 5-16

c) Antonio dará clases de francés a su hermano.

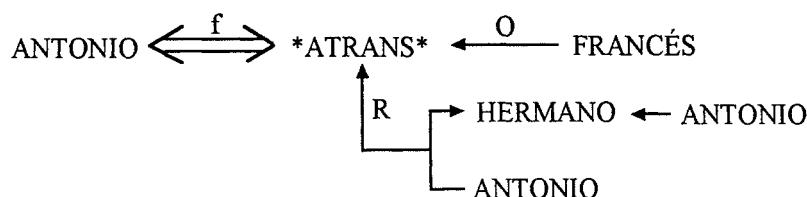


Figura 5-17

• PROBLEMA 5.5: (adaptado de [Rich & Knight, 1991])

Dada la oración:

“Debido a que fumar puede matar, yo lo dejé.”

representar su significado mediante un grafo de dependencia conceptual.

SOLUCIÓN:

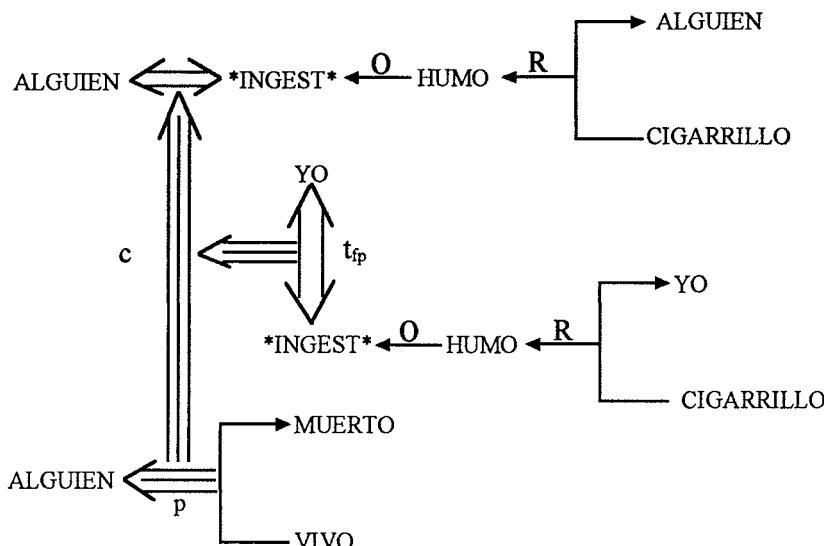


Figura 5-18

El enlace de causalidad vertical indica que fumar mata a las personas. La “c” que acompaña a este enlace indica que sabemos que fumar no necesariamente va a matar a una persona, aunque existe la posibilidad de que lo haga. El enlace de causalidad horizontal indica que la causalidad vertical es el motivo de que “yo dejara de fumar”. La etiqueta t_{fp} asociada a la dependencia entre INGEST y YO muestra que el acto fumar (una instancia de INGEST) ha acabado y que realmente acabó en el pasado.

• PROBLEMA 5.6:

Represente la siguiente frase:

“Andrés observa que un niño juega con una pelota.”

- Mediante una red de Shapiro.
- Mediante un grafo de Sowa.

SOLUCIÓN:

¿Qué pretende este problema? El presente problema, junto con el 5.7, muestra ejemplos de representación mediante redes de Shapiro y grafos de Sowa.

a) Dicha frase se representaría del siguiente modo utilizando el formalismo de Shapiro:

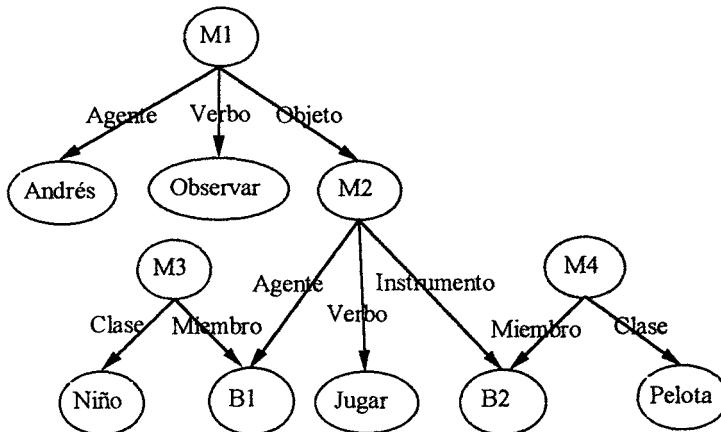


Figura 5-19

b) Mediante grafos de Sowa se obtendría la siguiente representación:

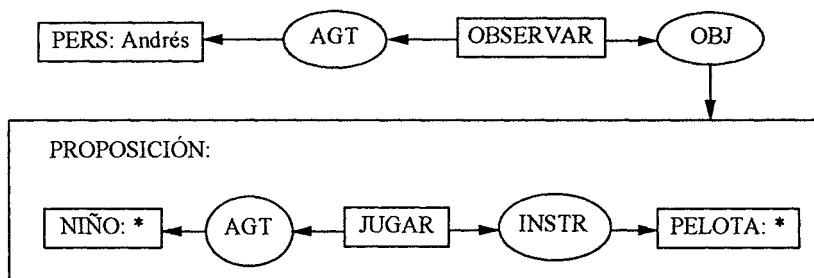


Figura 5-20

En los grafos de Sowa, si se conoce la identidad del objeto representado por un nodo, se indica el nombre explícitamente en el mismo junto con la clase a la que pertenece. Cuando aparece un asterisco al lado de la clase, lo que se está expresando es

la existencia de un determinado elemento de dicha clase cuya identidad no se conoce. En la práctica el asterisco suele omitirse.

• PROBLEMA 5.7:

Representar mediante:

- una red de Shapiro
- un grafo de Sowa

la siguiente frase:

“Ana piensa que Juan cree que un perro está comiendo un hueso”

SOLUCIÓN:

- La frase del enunciado del problema se representaría según el formalismo de redes de Shapiro del siguiente modo:

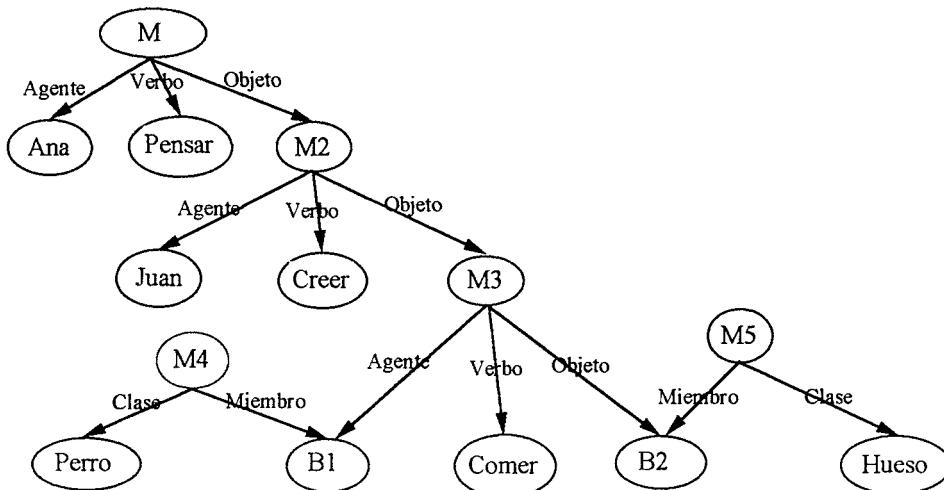


Figura 5-21

Los nodos M1, M2, M3, M4 y M5 son nodos que representan proposiciones. M1 corresponde a la afirmación de que Ana está pensando algo. Lo que Ana piensa se representa en el nodo M2. Este nodo indica que Juan cree M3, siendo M3 la proposición correspondiente a un perro que come un hueso. Algunas cuestiones a tener en cuenta son las siguientes:

- Los nodos “Ana” y “Juan” no muestran que los conceptos que representan correspondan a personas. Si fuera necesario establecer dicha información, habría que hacerlo a través de proposiciones separadas.
 - Los nodos B1 y B2 representan cierto perro y cierto hueso indeterminados.
 - Los nodos M4 y M5 establecen que B1 es un miembro de la clase “Perro” y que B2 es un miembro de la clase “Hueso”.
 - Nótese que los conceptos que representan verbos sólo están unidos a nodos proposición y que las relaciones del tipo Agente u Objeto se enlazan, no a los nodos verbo, sino a los nodos proposición. Los componentes de una proposición se unen a un nodo externo que representa dicha proposición.
- b) En el formalismo de Sowa, todo grafo que representa una proposición queda anidado dentro de un nodo concepto etiquetado con la palabra “proposición”, del siguiente modo:

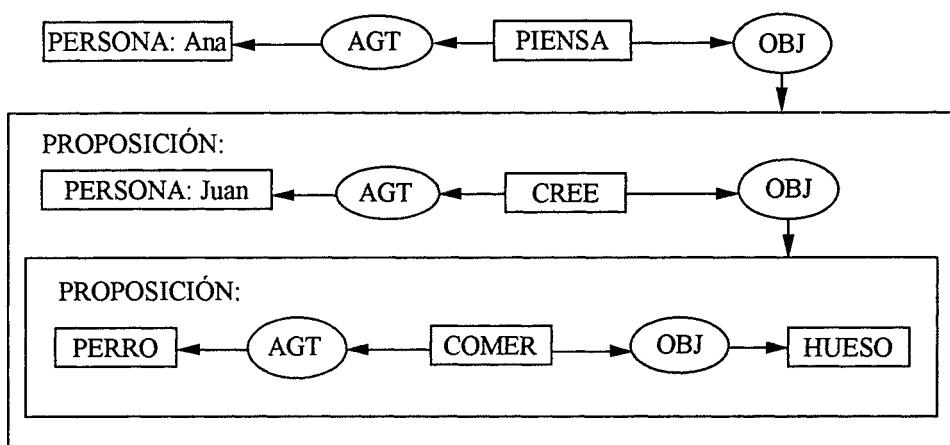


Figura 5-22

En este formalismo PERSONA: Ana (encerrado en un rectángulo) se refiere a algo llamado “Ana” de tipo PERSONA. Por otra parte, PERRO (encerrado en un rectángulo) se refiere a algo que no tiene nombre y que es de tipo PERRO. Obsérvese el tratamiento diferente respecto al apartado a) que se da ahora a los verbos en lo que se refiere a los enlaces asociados a los mismos.

• PROBLEMA 5.8:

Analizar las principales diferencias entre los grafos de dependencia conceptual y las redes de Shapiro.

SOLUCIÓN:

• La idea principal en los grafos de dependencia conceptual es representar cualquier frase mediante un número determinado de primitivas (6 categorías conceptuales, 16 reglas sintácticas y 12 acciones primitivas). Esta idea no aparece en las redes de Shapiro, en las que no existe un número de primitivas previamente establecido.

• En los grafos de dependencia conceptual el verbo es la pieza central de la representación, mientras que en las redes de Shapiro lo es la proposición.

• En los grafos de dependencia conceptual se utilizan tipos diferentes de arcos, cada uno para un tipo de relación. En las redes de Shapiro se emplean arcos simples etiquetados, con una etiqueta diferente para cada tipo de relación.

• En los grafos de dependencia conceptual cada nodo representa un determinado concepto, mientras que en las redes de Shapiro pueden representar además una variable, un sintagma, una frase, un párrafo o hasta historias completas.

• En los grafos de dependencia conceptual se representan únicamente relaciones de carácter local, es decir, relaciones que se pueden dar entre los diferentes elementos que forman una frase. En las redes de Shapiro, por su parte, se pueden representar además relaciones de tipo global que se dan entre frases u oraciones, de manera que una proposición puede estar contenida dentro de otra proposición, existiendo varios niveles anidados (se pueden formar conjunciones, disyunciones, implicaciones... en que intervengan varias proposiciones; el contexto de cada una de ellas viene dado por su posición en la red). Por tanto, una limitación de los grafos de dependencia conceptual frente a las redes de Shapiro es la dificultad o imposibilidad de tratar la interacción entre más de dos proposiciones. Este rasgo caracteriza igualmente al resto de los modelos considerados como grafos relationales y es una ventaja de las redes proposicionales frente a aquéllos.

• En las redes de Shapiro existe la posibilidad de establecer el *alcance* de un determinado operador, es decir, los nodos a los que afecta (considérese por ejemplo el caso de los cuantificadores universal y existencial). Esto permite implementar todas las capacidades de la lógica de primer orden mediante este formalismo. En los grafos de dependencia conceptual no existe el concepto de *alcance* mencionado anteriormente y, por tanto, no pueden reproducir las capacidades de la lógica de primer orden.

• PROBLEMA 5.9:

Representar en *notación lineal* las siguientes expresiones:

- a) Si un hombre mata a otro, será encarcelado.
- b) ¿Quién me ha robado el dinero que estaba en el cajón?
- c) Me iré cuando me apetezca.

SOLUCIÓN:

¿Qué pretende este problema? Se introduce a continuación una nueva notación llamada “lineal”, que permite una más fácil manipulación simbólica de las redes de Shapiro y de los grafos de Sowa por un computador.

- a) Si un hombre mata a otro, será encarcelado.

Se tendría la siguiente representación:

[CONDICIONAL:-

(SI) —→ [PROPOSICIÓN: [MATAR]-
 (AGT) —→ [PERS: *X]
 (OBJ) —→ [PERS: *]]
 (ENT) —→ [PROPOSICIÓN: [ENCARCELAR]-
 (OBJ) —→ [T: *X]
 (FUTURO)]]

donde se han empleado variables (X en este caso) para facilitar el proceso de representación. Además, la “T” se ha utilizado para representar un nodo sin especificar la clase a la que pertenece.

- b) ¿Quién me ha robado el dinero que estaba en el cajón?

En este caso:

[ROBAR]-

(AGT) → [PERS: ?]
 (RCP) → [PERS: #yo]
 (OBJ) → [DINERO: #] → (LOC) → [LUGAR: #cajón]
 (PASADO)

Aparece la etiqueta RCP (receptor de una acción, en este caso ROBAR) y no POS (posesión, en este caso del DINERO) debido a que se ha considerado que el dinero no tenía por qué ser posesión de la persona que pronuncia la frase.

c) Me iré cuando me apetezca.

[IRSE]-

(AGT) —→ [PERS: #yo]

(TIEMPO) —→ [PROPOSICIÓN: [APETECER]-

(AGT) —→ [PERS: #yo]

(FUTURO)]

(FUTURO)

• PROBLEMA 5.10: (*)

Dados los enunciados siguientes:

- a) “Juan y Antonio juegan con una pelota.”
- b) “Alguien juega con una pelota.”
- c) “Un niño juega con una pelota de tenis.”
- d) “Juan y Antonio juegan.”
- e) “Un niño juega con una raqueta.”
- f) “Un niño juega con una raqueta y una pelota de tenis.”

explique las operaciones básicas de inferencia utilizando los grafos de Sowa que permiten obtener unos enunciados a partir de otros. Señale el tipo de inferencia aplicado en cada caso y represente en forma lineal los grafos correspondientes a cada enunciado.

SOLUCIÓN:

Las operaciones básicas de inferencia en grafos de Sowa son la de *generalización*, *restricción* (proceso recíproco al de generalización) y la de *unión* y *simplificación*. Algunas de estas operaciones no siempre son válidas semánticamente. Por ejemplo, aunque la proposición “Un niño juega” sea verdadera, es posible que su restricción “Antonio juega” sea falsa.

Es claro que aplicando generalización sobre el enunciado a) se pueden obtener tanto el enunciado b) como el d). También aplicando generalización sobre f) se puede obtener c). Por otra parte, el enunciado c) puede ser obtenido a partir del b) aplicando una restricción. Del mismo modo, f) se obtiene a partir de b) mediante una restricción. Finalmente, si se parte de los enunciados c) y e), aplicando unión y simplificación, se obtiene el enunciado f). En cuanto a la representación mediante grafos de Sowa de los seis enunciados presentados, sería la siguiente:

a) "Juan y Antonio juegan con una pelota."

[JUGAR]-

$$\begin{aligned} \longrightarrow (\text{AGT}) &\longrightarrow [\text{PERS}: \{\text{Juan, Antonio}\}] \\ \longrightarrow (\text{INSTR}) &\longrightarrow [\text{PELOTA}: *] \end{aligned}$$

b) "Alguien juega con una pelota."

[JUGAR]-

$$\begin{aligned} \longrightarrow (\text{AGT}) &\longrightarrow [\text{PERS}: *] \\ \longrightarrow (\text{INSTR}) &\longrightarrow [\text{PELOTA}: *] \end{aligned}$$

c) "Un niño juega con una pelota de tenis."

[JUGAR]-

$$\begin{aligned} \longrightarrow (\text{AGT}) &\longrightarrow [\text{NIÑO}: *] \\ \longrightarrow (\text{INSTR}) &\longrightarrow [\text{PELOTA}: *] \longrightarrow (\text{ATR}) \longrightarrow [\text{DE-TENIS}] \end{aligned}$$

d) "Juan y Antonio juegan."

[JUGAR]-

$$\longrightarrow (\text{AGT}) \longrightarrow [\text{PERS}: \{\text{Juan, Antonio}\}]$$

e) "Un niño juega con una raqueta."

[JUGAR]-

$$\begin{aligned} \longrightarrow (\text{AGT}) &\longrightarrow [\text{NIÑO}: *] \\ \longrightarrow (\text{INSTR}) &\longrightarrow [\text{RAQUETA}: *] \end{aligned}$$

f) "Un niño juega con una raqueta y una pelota de tenis."

[JUGAR]-

$$\begin{aligned} \longrightarrow (\text{AGT}) &\longrightarrow [\text{NIÑO}: *] \\ \longrightarrow (\text{INSTR}) &\longrightarrow [\text{RAQUETA}: *] \\ \longrightarrow (\text{INSTR}) &\longrightarrow [\text{PELOTA}: *] \longrightarrow (\text{ATR}) \longrightarrow [\text{DE-TENIS}] \end{aligned}$$

• PROBLEMA 5.11:

Considérese el problema de diagnosticar una enfermedad D , que puede dar origen a la aparición de un síntoma S o un signo radiológico R . Cada una de estas tres variables puede tomar dos valores: "presente" o "ausente". Se conocen los siguientes datos:

$$P(+d) = 0'01$$

$$P(+s | +d) = 0'92 \quad P(+s | \neg d) = 0'16$$

$$P(+r | +d) = 0'98 \quad P(+r | \neg d) = 0'05$$

(obsérvese que, para esta enfermedad, el signo radiológico R es más sensible y específico que el síntoma S), donde:

$P(+d)$: prevalencia de D , es decir, la probabilidad a priori de que, dado un paciente cualquiera, éste padezca la enfermedad.

$P(+s | +d)$: sensibilidad de S respecto de D , es decir, probabilidad de que, dado un paciente que padezca la enfermedad D , presente el síntoma S .

$P(+s | -d)$: probabilidad de que, dado un paciente que no padezca la enfermedad D , presente el síntoma S .

$P(-s | -d)$: especificidad de S respecto de D .

Aplicando el método clásico de diagnóstico probabilista, calcular:

$$P(+d | +s)$$

$$P(+d | -r)$$

$$P(+d | +s, +r)$$

$$P(+d | +s, -r)$$

SOLUCIÓN:

En el presente problema se tiene:

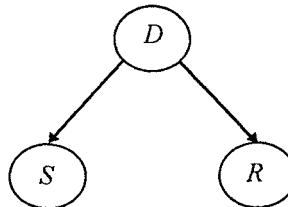


Figura 5-23

de manera que

$$\begin{aligned}
 P(+d|+s) &= \frac{P(+s|+d) \cdot P(+d)}{P(+s|+d) \cdot P(+d) + P(+s|-d) \cdot P(-d)} = \frac{0'92 \cdot 0'01}{0'92 \cdot 0'01 + 0'16 \cdot (1 - 0'01)} = \\
 &= 0'054
 \end{aligned}$$

$$P(+d|\neg r) = \frac{P(\neg r|+d) \cdot P(+d)}{P(\neg r|+d) \cdot P(+d) + P(\neg r|\neg d) \cdot P(\neg d)} = \\ = \frac{(1 - 0'98) \cdot 0'01}{(1 - 0'98) \cdot 0'01 + (1 - 0'05) \cdot (1 - 0'01)} = 0'00021$$

$$P(+d|+s, +r) = \frac{P(+s|+d) \cdot P(+r|+d) \cdot P(+d)}{P(+s|+d) \cdot P(+r|+d) \cdot P(+d) + P(+s|\neg d) \cdot P(+r|\neg d) \cdot P(\neg d)} = \\ = \frac{0'92 \cdot 0'98 \cdot 0'01}{0'92 \cdot 0'98 \cdot 0'01 + 0'16 \cdot 0'05 \cdot (1 - 0'01)} = 0'53$$

$$P(+d|+s, \neg r) = \frac{P(+s|+d) \cdot P(\neg r|+d) \cdot P(+d)}{P(+s|+d) \cdot P(\neg r|+d) \cdot P(+d) + P(+s|\neg d) \cdot P(\neg r|\neg d) \cdot P(\neg d)} = \\ = \frac{0'92 \cdot (1 - 0'98) \cdot 0'01}{0'92 \cdot (1 - 0'98) \cdot 0'01 + 0'16 \cdot (1 - 0'05) \cdot (1 - 0'01)} = 0'012$$

Por tanto, se podría construir la siguiente tabla (compruébese el resto de resultados):

evidencia	$P(+d \text{evidencia})$
\emptyset	0'01
$+s$	0'054
$\neg s$	0'00096
$+r$	0'165
$\neg r$	0'00021
$+s, +r$	0'53
$+s, \neg r$	0'0012
$\neg s, +r$	0'018
$\neg s, \neg r$	0'00002

Obsérvese que la existencia de evidencia a favor de S y R conjuntamente produce una probabilidad a favor del diagnóstico D ($0'53$) bastante mayor que cuando sólo se posee evidencia a favor de S o R por separado ($0'054$ o $0'165$ respectivamente). Del mismo modo, la existencia de evidencia en contra de S y R conjuntamente produce una probabilidad a favor de D ($0'00002$) bastante menor que cuando sólo hay evidencia en contra de S o R por separado ($0'00096$ y $0'00021$ respectivamente).

• **PROBLEMA 5.12:**

Dibujar el grafo de una red bayesiana para un dominio que considere las siguientes variables:

R : “renta per cápita de un país”

E : “extensión en km^2 del mismo”

I : “número de casos por 1000 habitantes (o prevalencia) de enfermedades infantiles en ese país”

K : “kilómetros de autopista en el país”

Comentar las dependencias e independencias probabilísticas representadas (implícita o explícitamente) en la red y enumerar, además, qué tablas de probabilidad intervienen en la misma.

SOLUCIÓN:

¿Qué pretende este problema? En este problema y en el 5.13 se estudian ejemplos de representación de conocimiento causal de un dominio mediante redes bayesianas.

Parece lógico pensar que la extensión de un país influirá en el número de kilómetros de autopista y que la renta per cápita lo hará sobre las enfermedades infantiles y sobre los kilómetros de autopista del mismo, sin que exista ninguna otra relación de dependencia entre las cuatro variables consideradas (la renta per cápita de un país, así como la proporción de casos de enfermedades infantiles en el mismo, son independientes de su extensión y los kilómetros de autopista tampoco influyen en la prevalencia o proporción de enfermedades infantiles que se presentan).

De acuerdo con las relaciones que se han establecido, el grafo pedido sería el siguiente:

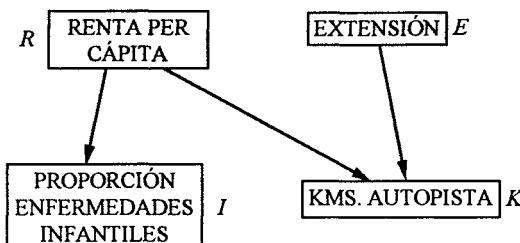


Figura 5-24

En cuanto a las probabilidades que intervienen en la red son:

$$P(r)$$

$$P(e)$$

$$P(i | r)$$

$$P(k | r, e)$$

donde las dos primeras son probabilidades a priori y las restantes son probabilidades condicionadas (las letras mayúsculas representan variables y las minúsculas los valores que éstas toman).

• PROBLEMA 5.13:

Dibujar el grafo de una red bayesiana que considere el sexo de una persona, su edad, sus ingresos mensuales, su estatura, el número de calzado que gasta y el tipo de coche que posee. Comentar las dependencias e independencias probabilísticas representadas (implícita o explícitamente) en la red. Enumerar, además, qué tablas de probabilidad forman parte de dicha red.

SOLUCIÓN:

Las variables que van a ser consideradas son:

S: Sexo

E: Edad

I: Ingresos

T: Estatura

C: Coche

N: Número de calzado

El grafo pedido tendría la siguiente forma:

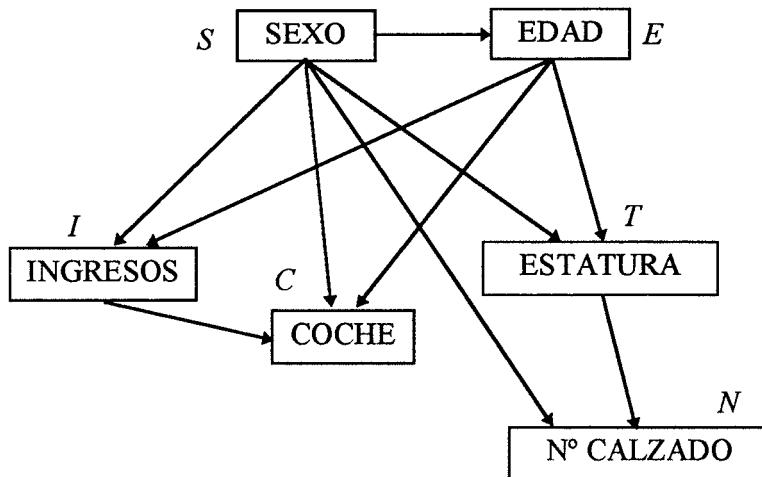


Figura 5-25

La estructura del grafo anterior refleja unas hipótesis más o menos acertadas, que vamos a enumerar a continuación, explicando por qué se traza o no un determinado enlace.

Evidentemente, el sexo de una persona influirá en sus ingresos (todavía en nuestra sociedad una importante parte de las profesiones siguen considerándose típicas de un determinado sexo e incluso dentro de la misma profesión las mujeres suelen ganar menos que los hombres), tipo de coche (en general los hombres prefieren coches potentes, mientras que las mujeres optan por aquéllos que les facilitan la conducción), la estatura (la estatura media del hombre es ligeramente superior a la de la mujer) y el número de calzado (las mujeres suelen usar números menores). También el sexo influye en la edad que una persona va a alcanzar; de hecho, las mujeres tienen una esperanza de vida varios años mayor que la de los hombres. Esto dará lugar a que, dada cierta persona de un determinado sexo, la probabilidad de que sea de edad avanzada será mayor si se trata de una mujer y no de un hombre.

Como consecuencia de todo lo comentado con anterioridad, se podrían ir trazando los siguientes enlaces:

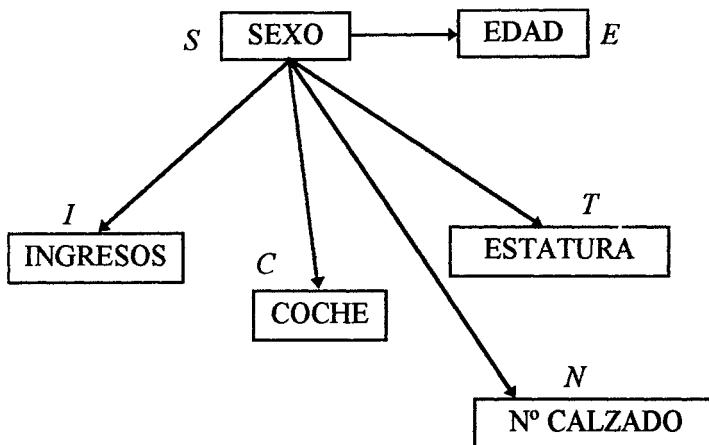


Figura 5-26

En cuanto a la edad, ésta va a influir en el nivel de ingresos de una persona (en general, con la edad cualquier trabajador alcanza una mayor experiencia y cualificación en su trabajo, que le permite elevar su nivel de ingresos), en el tipo de coche que posea (mientras que la gente de más edad prefiere coches más seguros, los jóvenes ven en la velocidad una cualidad importante) y en su estatura. Por tanto, se podría ampliar el grafo anterior de la siguiente forma:

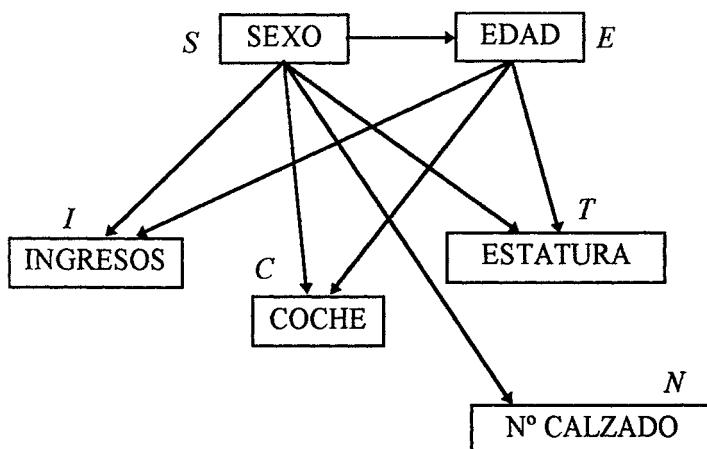


Figura 5-27

Obsérvese que no se ha trazado ningún enlace desde E hasta N . Esto es así debido a que, dada una determinada estatura para una persona, la edad de la misma no influirá en su número de calzado (por tanto, tal como se hace más adelante, habrá que trazar un enlace entre T y N). Finalmente, fijado el sexo, la estatura determina el número de calzado de una persona y, por otra parte, sus ingresos influyen sobre el tipo de coche que posee. La red final, por tanto, sería:

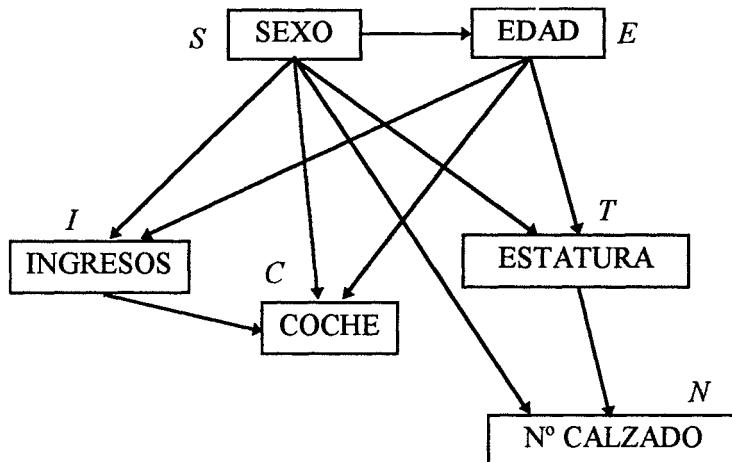


Figura 5-28

Obsérvese, por ejemplo, que no se ha trazado ningún enlace desde *Ingresos* hasta *Nº calzado*, ya que estas dos variables son condicionalmente independientes entre sí. Por otra parte, se podría haber supuesto que la estatura influye en el nivel de ingresos de una persona, pues una estatura adecuada contribuye a una mejor presencia física, requisito necesario para determinados puestos de trabajo (aquí, sin embargo, se ha considerado que un puesto donde se requiera buena presencia física no es sinónimo de puesto bien remunerado y por eso no se ha trazado un enlace de T a I). Tampoco se ha tenido en cuenta que la estatura ni el número de calzado influyan en el tipo de coche de una persona y, por tanto, no se han trazado los enlaces de T a C y de N a C . Las probabilidades que habría que añadir al grafo anterior para completar la red bayesiana son:

$$P(e)$$

$$P(s | e)$$

$$P(i | s, e)$$

$$P(t | s, e)$$

$$P(c | i, s, e)$$

$$P(n | t)$$

• PROBLEMA 5.14:

Considérese la siguiente red bayesiana:

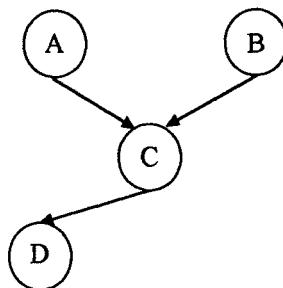


Figura 5-29

donde cada variable puede tomar dos valores: *presente* o *ausente*. Las distribuciones de probabilidades son:

$$P(+a) = 0'01$$

$$P(+b) = 0'006$$

$P(+c a, b)$	$+a$	$\neg a$
$+b$	0'99	0'9
$\neg b$	0'8	0'001

$P(+d c)$	$+c$	$\neg c$
$+d$	0'94	0'01

Calcular, a partir de la red bayesiana dada, las siguientes probabilidades:

- a) $P(+a | +c, \neg b)$
- b) $P(+c | +d)$
- c) $P(+c | +d, +a, \neg b)$

SOLUCIÓN:

¿Qué pretende este problema? Este problema muestra cómo se lleva a cabo el proceso de inferencia en una red bayesiana. Los métodos que aparecen aquí, aunque son didácticos, no son los utilizados normalmente para redes grandes debido a que son poco eficientes.

Antes de proceder con la resolución del problema es conveniente reseñar que, por la propiedad de separación direccional de las redes bayesianas, la probabilidad condicional de una variable X conocido el valor de sus padres, $\text{pa}(X)$, es independiente de los nodos que no son descendientes de X . Por ejemplo, para D tenemos:

$$P(d|c,a) = P(d|c,b) = P(d|c,a,b) = P(d|c)$$

porque el único parente de D es C , y ni A ni B son descendientes de D . Para A tenemos:

$$P(a|b) = P(a)$$

pues los padres de A son el conjunto vacío y B no es descendiente de A . Esta propiedad es equivalente a:

$$P(a,b) = P(a|b) \cdot P(b) = P(a) \cdot P(b)$$

es decir, A y B son independientes a priori.

a) Primer método:

Por la definición de probabilidad condicionada:

$$P(+a|\neg b,+c) = \frac{P(+a,\neg b,+c)}{P(\neg b,+c)}$$

Por la definición de probabilidad conjunta:

$$P(+a,\neg b,+c) = \sum_d P(+a,\neg b,+c, d)$$

$$P(\neg b,+c) = \sum_a \sum_d P(a,\neg b,+c, d)$$

Aplicando repetidamente la propiedad de separación direccional, podemos factorizar cada una de las probabilidades anteriores, con lo cual llegamos a:

$$P(+a,\neg b,+c) = \sum_d P(d|+c) \cdot P(+c|+a,\neg b) \cdot P(+a) \cdot P(\neg b)$$

$$P(\neg b, +c) = \sum_a \sum_d P(d|+c) \cdot P(+c|a, \neg b) \cdot P(a) \cdot P(\neg b)$$

Finalmente,

$$\begin{aligned} P(+a|\neg b, +c) &= \frac{P(+c|+a, \neg b) \cdot P(+a) \cdot P(\neg b) \cdot [P(+d|+c) + P(\neg d|+c)]}{P(\neg b) \cdot [P(+d|+c) \cdot P(+c|+a, \neg b) \cdot P(+a) + \dots]} = \\ &= \frac{0'8 \cdot 0'01 \cdot (1 - 0.006) \cdot 1}{(1 - 0'006) \cdot [0'94 \cdot 0'8 \cdot 0'01 + (1 - 0'94) \cdot 0'8 \cdot 0'01 + 0'94 \cdot 0'001 \cdot (1 - 0'01) + \dots]} = \\ &= 0'889 \end{aligned}$$

Obsérvese cómo ante la evidencia de que C está presente y una de sus causas, en este caso B , está ausente, la probabilidad de que la otra causa de C esté presente es bastante alta.

b) Primer método:

Nuevamente, por la definición de probabilidad condicionada:

$$P(+c|+d) = \frac{P(+c, +d)}{P(+d)}$$

A su vez, por la definición de probabilidad conjunta:

$$P(+c, +d) = \sum_a \sum_b P(a, b, +c, +d)$$

$$P(+d) = \sum_a \sum_b \sum_c P(a, b, c, +d)$$

Aplicando repetidamente la propiedad de separación direccional para factorizar cada una de las expresiones anteriores se llega a:

$$P(+c, +d) = \sum_a \sum_b P(a) \cdot P(b) \cdot P(+c|a, b) \cdot P(+d|+c)$$

$$P(+d) = \sum_a \sum_b \sum_c P(a) \cdot P(b) \cdot P(c|a, b) \cdot P(+d|c)$$

con lo cual, sustituyendo por los valores numéricos conocidos y operando se obtiene:

$$P(+c|+d) = \frac{0'01348}{0'02333} = 0'578$$

c) Primer método:

Operando del mismo modo que en los dos casos anteriores:

$$\begin{aligned} P(+c|+d, +a, \neg b) &= \frac{P(+c, +d, +a, \neg b)}{P(+d, +a, \neg b)} = \frac{P(+d|+c) \cdot P(+c|+a, \neg b) \cdot P(+a) \cdot P(\neg b)}{\sum_c P(+d, +a, \neg b, c)} = \\ &= \frac{0'94 \cdot 0'8 \cdot 0'01 \cdot (1 - 0'006)}{0'94 \cdot 0'8 \cdot 0'01 \cdot (1 - 0'006) + 0'01 \cdot (1 - 0'8) \cdot 0'01 \cdot (1 - 0'006)} = \frac{0'0074}{0'0074 + 0'00002} = \\ &= 0'9973 \end{aligned}$$

a) Segundo método:

Por la definición de probabilidad condicional:

$$P(+a|\neg b, +c) = \frac{P(+a, \neg b, +c)}{P(\neg b, +c)}$$

Igualmente por la definición de probabilidad condicional:

$$\begin{aligned} P(+a, \neg b, +c) &= P(+c|+a, \neg b) \cdot P(+a, \neg b) = P(+c|+a, \neg b) \cdot P(+a) \cdot P(\neg b) = \\ &= 0'8 \cdot 0'01 \cdot (1 - 0'006) = 0'008 \end{aligned}$$

Por la definición de probabilidad conjunta:

$$\begin{aligned} P(\neg b, +c) &= \sum_a P(a, \neg b, +c) = \sum_a P(+c|a, \neg b) \cdot P(a, \neg b) = \\ &= P(+c|+a, \neg b) \cdot P(+a) \cdot P(\neg b) + P(+c|\neg a, \neg b) \cdot P(\neg a) \cdot P(\neg b) = \\ &= 0'8 \cdot 0'01 \cdot (1 - 0'006) + 0'001 \cdot (1 - 0'01) \cdot (1 - 0'006) = 0'009 \end{aligned}$$

Luego,

$$P(+a|\neg b, +c) = \frac{0'008}{0'009} = 0'889$$

b) Segundo método:

Por el teorema de Bayes:

$$P(+c|+d) = \frac{P(+c) \cdot P(+d|+c)}{P(+d)}$$

Por otra parte,

$$\begin{aligned}
 P(+c) &= \sum_a \sum_b P(+c|a,b) = \sum_a \sum_b P(+c|a,b) \cdot P(a) \cdot P(b) = \\
 &= 0'99 \cdot 0'01 \cdot 0'006 + 0'8 \cdot 0'01 \cdot (1 - 0'006) + 0'9 \cdot (1 - 0'001) \cdot 0'006 + \dots = \\
 &= 0'0144
 \end{aligned}$$

$$\begin{aligned}
 P(+d) &= \sum_c P(+d|c) = \sum_c P(+d|c) \cdot P(c) = \\
 &= 0'94 \cdot 0'0144 + 0'01 \cdot (1 - 0'0144) = 0'0234
 \end{aligned}$$

$$P(+c|+d) = \frac{0'0144 \cdot 0'94}{0'0234} = 0'578$$

c) Segundo método:

$$\begin{aligned}
 P(+c|+d, +a, \neg b) &= \frac{P(+c, +d, +a, \neg b)}{P(+d, +a, \neg b)} = \\
 &= \frac{P(+d|c) \cdot P(+c|a, \neg b) \cdot P(+a) \cdot P(\neg b)}{\left(\sum_c P(+d|c) \cdot P(c|a, \neg b) \right) \cdot P(+a) \cdot P(\neg b)} = \\
 &= \frac{0'94 \cdot 0'8 \cdot 0'01 \cdot (1 - 0'006)}{(0'94 \cdot 0'8 + 0'01 \cdot (1 - 0'8)) \cdot 0'01 \cdot (1 - 0'006)} = 0'9973
 \end{aligned}$$

Aunque en la resolución del presente problema se han podido emplear cálculos sencillos, para redes de mayor tamaño en la práctica se utilizan otros métodos más eficientes; algunos de ellos permiten realizar el cálculo de la probabilidad de forma distribuida, de manera que cada nodo puede ser asociado a un procesador físico y a través de los enlaces se transmiten los mensajes numéricos necesarios para realizar los cálculos.

5.4 Contexto adicional

5.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

- Diferenciar los tipos de redes asociativas empleados a lo largo de la historia de la inteligencia artificial.
- Conocer los mecanismos de representación e inferencia de cada clase particular de red asociativa explicada en el capítulo.
- Identificar el tipo de problemas para el que cada red es adecuada.

5.4.2 Pistas de autoevaluación

- Representar mediante un grafo de dependencia conceptual la siguiente frase:
“Ernesto fertilizó el campo.”
- Pasar a notación lineal las oraciones que aparecen en los enunciados de los problemas 5.6, 5.7 y 5.10.
 - Se propone al lector la construcción de una red bayesiana simple (ver apartado de software de apoyo) y la propagación en la misma de evidencia disponible sobre el dominio.

6 MARCOS Y GUIONES

6.1 Contexto previo

6.1.1 ¿Por qué está aquí este capítulo?

Aunque las principales ventajas de la lógica eran su expresividad, su semántica y su eficacia en procesos deductivos, tenía como inconveniente la falta de estructuración. Por otra parte, el problema de la estructuración de la información también apareció en las redes, en aquellos casos en que nodos y enlaces empezaron a representar entidades cada vez más complejas. En realidad no existe una distinción clara entre una red semántica y un sistema de marcos; en general, se tiende a decir que se tiene un sistema de marcos cuando la información se encuentra estructurada dentro de la red.

Jerarquía y herencia son los dos conceptos básicos asociados a la representación por marcos. Desde que se acepta la distinción metodológica entre dos niveles de conocimiento (el específico del dominio y el genérico, propio de la tarea y el método usado para su descomposición hasta el nivel de primitivas), es comúnmente aceptado que la forma más completa y eficiente de representar el conocimiento específico de un dominio de aplicación es usar objetos estructurados y, siempre que sea posible, establecer jerarquías en estos objetos y en sus relaciones. Por otro lado, la herencia es el principal mecanismo de inferencia en la representación por marcos (en algunos casos es el único). Heredar propiedades de otras entidades más generales es el más primitivo de los mecanismos de inferencia. Hay que añadir el uso de conocimiento “por defecto” y la posibilidad de asociar procedimientos para completar las capacidades de los marcos.

Finalmente, en los marcos encontramos, en mayor o menor grado, todos los otros mecanismos de representación (lógica y reglas). Por todas estas razones está aquí este capítulo.

6.1.2 ¿Dónde hay conocimiento teórico del campo?

Información adicional referente a marcos y guiones aparece en:

Referencias básicas:

- [Mira *et al.*, 1995], capítulo 8.

- [Rich & Knight, 1994], capítulos 9 y 10.

Referencias complementarias:

- [Jackson, 1990], capítulo 9.
- [Lucas & Van der Gaag, 1991], capítulo 4.
- [Nilsson, 1987], capítulo 9.
- [Winston, 1994], capítulo 9 y 10.

6.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

- Conviene haber leído todos los capítulos que preceden al presente para un mejor aprovechamiento de los contenidos del mismo.

6.1.4 ¿Qué software de apoyo está disponible?

• BABYLON

Entorno para el desarrollo de sistemas expertos implementado en Common Lisp que incluye: objetos, reglas con encadenamiento hacia adelante y hacia atrás... Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/babylon/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/babylon/0.html)

• FRulekit

Sistema basado en marcos escrito en Common Lisp. Permite la aplicación del algoritmo RETE y la definición por parte del usuario de estrategias propias para resolución de conflictos. Incorpora un sistema de agendas para el control del orden de ejecución de las reglas. Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/frulekit/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/frulekit/0.html)

• MIKE (*Micro Interpreter for Knowledge Engineering*)

Sistema diseñado para fines académicos por la Open University del Reino Unido. Incluye, entre otras características, encadenamiento hacia adelante y hacia atrás de reglas con estrategias definibles de resolución de conflictos, un lenguaje para representación de marcos con herencia, demonios y diversas estrategias de herencia. Se puede obtener en:

[http://www.cs.cmu.edu/afs/cs/project/
ai-repository/ai/areas/expert/systems/mike/0.html](http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/mike/0.html)

6.2 Contenido teórico

6.2.1 Marcos

Los *marcos* constituyen un mecanismo de representación del conocimiento y razonamiento propuesto por Marvin Minsky en 1975. Desde su aparición, el concepto de marco ha ido evolucionando de tal manera que hoy en día algunas de las características sugeridas en su momento por Minsky han desaparecido y han surgido otras nuevas que se alejan bastante de la propuesta inicial.

En general, puede decirse que un marco es una colección de atributos, que reciben el nombre de *campos*, y valores asociados a los mismos. Dicha colección pretende describir o representar una determinada entidad o concepto del dominio que se esté considerando. No es usual el empleo de marcos aislados, sino que se suelen construir sistemas de marcos conectados entre sí. En la propuesta original de Minsky las conexiones se establecían exclusivamente a partir del hecho de que un marco podía ocupar un campo de otro (por ejemplo, el marco *DORMITORIO* podía ser un campo del marco *CASA*). En las herramientas comerciales actuales aparece un nuevo tipo de conexión que se establece a partir de la creación de una organización jerárquica entre los marcos (por ejemplo, se podría tener un marco raíz *ANIMAL* y dos marcos hijos *VERTEBRADOS* e *INVERTEBRADOS* unidos a él a través de enlaces “superclase”).

Los marcos propuestos por Minsky eran útiles en tareas de reconocimiento como las que pueden presentarse, por ejemplo, en visión artificial. La idea central era que la información recibida desde el exterior sirviera para activar determinados marcos; esto provocaría la activación de otros marcos conectados con los primeros. Al final se tendría una *red de activación* cuyo objetivo sería predecir la información que se podría encontrar más adelante. En los marcos que aparecen en las herramientas comerciales actuales para el diseño de sistemas expertos, no se diferencia entre *marcos activos* o *inactivos*; sin embargo, si surgen dos nuevas características:

- Existen dos tipos de marcos:
 - a) Las *clases*, que representan conceptos o entidades generales.
 - b) Las *instancias*, que vienen a ser ejemplos particulares de marcos clase (ver figura 6-1).
- Se dota a la red jerárquica de marcos de un *mecanismo de herencia* gracias al cual cada marco hereda los campos de sus antepasados en la red (por ejemplo, en la figura 6-1 la instancia *PERRO* hereda el campo *PESO* de la clase *ANIMAL* a la que

pertecece). Algunas herramientas comerciales permiten también la existencia de herencia ascendente en la red.

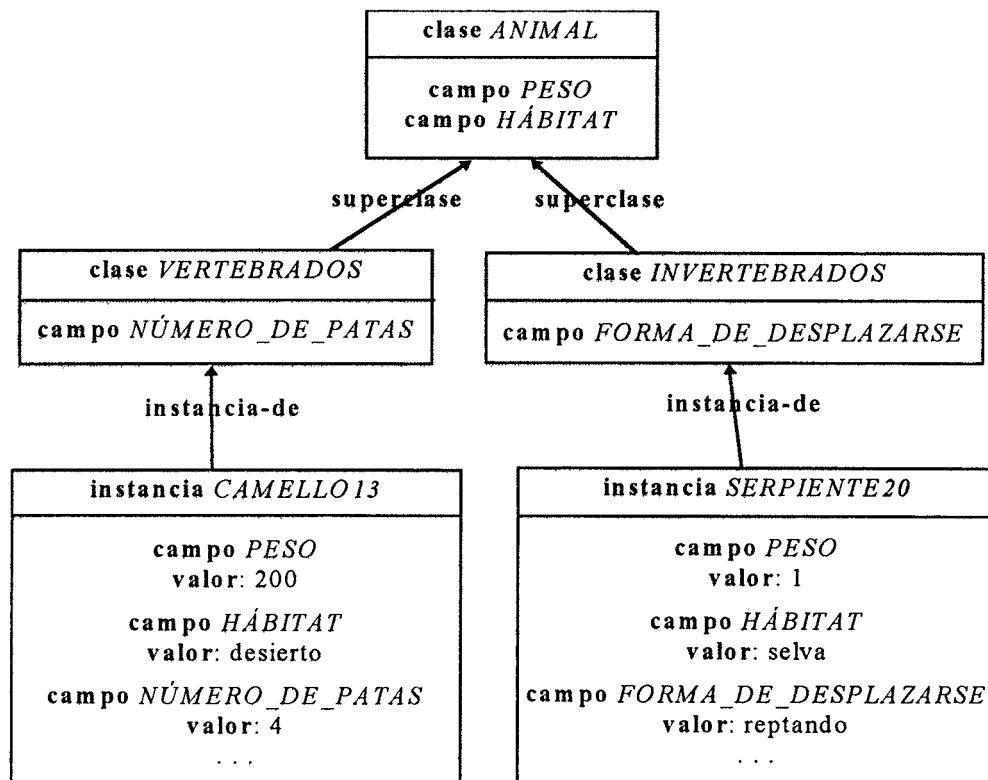


Figura 6-1 Jerarquía de marcos dotada de un mecanismo de herencia.

Se denomina *facetas* a las diferentes propiedades que tiene un campo. Las más usadas son las siguientes:

a) Valor por defecto

Es el valor que toma el campo si no ha sido asignado al mismo ningún otro valor con anterioridad.

b) Multivaluado

Especifica si el campo admite o no varios valores simultáneamente.

c) Restricciones

Se refiere a las limitaciones que se imponen al rango de valores que puede tomar un campo.

d) Certeza

Hace referencia a la credibilidad del valor asignado al campo.

e) Facetas de interfaz

Contiene información destinada a la interacción con el usuario.

Otra de las características importantes de los sistemas de marcos actuales es la posibilidad de asociar procedimientos llamados *demonios* a cualquier elemento de la red jerárquica. La utilidad de estos procedimientos es muy diversa, aunque en general sirven para mantener la consistencia de todo el sistema. Algunos ejemplos de este tipo de procedimientos son:

- **Demonios de necesidad**

Se ejecutan cada vez que se quiere saber cuál es el valor de un campo y éste no tiene ningún valor asignado.

- **Demonios de modificación**

Se ejecutan siempre que varíe el valor de un campo.

- **Demonios de borrado**

Se ejecutan al ser eliminado el valor de un campo.

- **Demonios de asignación**

Se ejecutan al añadir un valor a un campo.

- **Demonios de acceso**

Se ejecutan cuando se solicita el valor de un campo, aunque ya tuviera un valor asignado.

Gran parte de las características que se han descrito para sistemas basados en marcos son ofrecidas también por los lenguajes de *programación orientada a objetos*, de tal modo que estos dos formalismos son hoy en día bastante similares. En un entorno que disponga de un lenguaje para programación orientada a objetos, los marcos pasan a ser considerados como *objetos* autónomos que tienen la capacidad de comunicarse entre ellos mediante el *paso de mensajes*. Un mensaje consistirá en un mandato a un objeto

para que ejecute un procedimiento local al mismo. Estos procedimientos se llaman *métodos*, son dependientes del dominio o problema concreto que se esté tratando y se diferencian de los demonios en que no se activan automáticamente al realizar ciertas operaciones sobre el valor de un campo, sino que se activan ante la recepción por el objeto al que pertenecen de un determinado mensaje. Una solución a un problema que emplee programación orientada a objetos consistirá en identificar los objetos del dominio, determinar los mensajes a los que cada objeto puede dar respuesta, diseñar los procedimientos o métodos que cada objeto ejecutará en respuesta a la recepción de cada mensaje y, finalmente, establecer la secuencia de mensajes necesaria para dar una solución al problema inicial planteado. Reseñar, por último, que existen mensajes que, enviados a una clase, permiten la creación dinámica de una instancia de dicha clase, que los métodos también son heredados en la jerarquía de objetos correspondiente y que al definir los objetos se suele establecer una distinción entre *datos privados* a los que sólo puede acceder el propio objeto y *datos compartidos*. Considérese el ejemplo de la figura 6-2 donde aparece una jerarquía de clases e instancias de objetos correspondientes al dominio de las figuras geométricas y donde se representan para cada clase aquellos procedimientos o métodos que pueden ser invocados mediante mensajes.

El método “mostrar” de la clase “Arco” ejecuta acciones diferentes al método “mostrar” de la clase “Círculo”. No es necesario dar nombres diferentes a estos procedimientos, ya que el sistema es capaz de determinar el tipo de objeto que se referencia en cualquier mensaje y, por tanto, qué procedimiento debe ser aplicado. Esta cuestión se conoce con el nombre de *polimorfismo* en el contexto de la programación orientada a objetos.

Obsérvese aquí la diferencia esencial entre la perspectiva orientada a objetos y los desarrollos metodológicos en el campo de los sistemas basados en conocimiento, donde también se usan objetos estructurados y métodos, pero con un significado distinto. En la perspectiva orientada a objetos, los métodos se asocian a cada objeto y la arquitectura es plana y de carácter distribuido. La solución del problema queda repartida sobre los distintos objetos y sus métodos. Por el contrario, en los desarrollos metodológicos en torno a KADS y CommonKADS y en todo el movimiento de búsqueda de bibliotecas de componentes reutilizables, se parte de una arquitectura multicapa donde la capa del dominio contiene los objetos estructurados, sin micro-métodos asociados. Sobre esta capa se superpone otra responsable del esquema inferencial y del control, resultado de la selección de un método (“propón-actúa-modifica” o “establece-refina”, por ejemplo) para descomponer la tarea de análisis, modificación o síntesis de la que se trate.

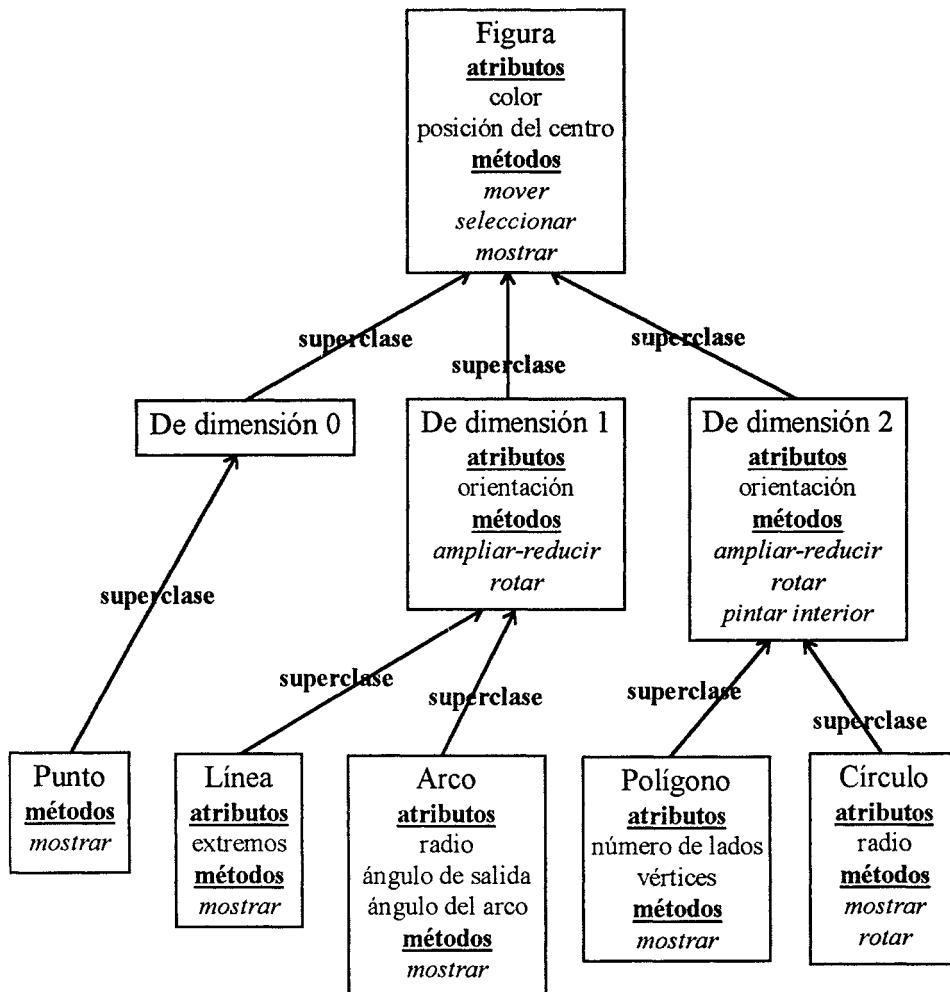


Figura 6-2

6.2.2 Guiones

Un guion es una estructura de conocimiento que organiza información referente a situaciones dinámicas estereotipadas como pueden ser: ir al cine, comer en un restaurante, ir de compras...; es decir, situaciones donde la secuencia de hechos que se desarrollan en las mismas es bastante fija, debido a lo cual no se necesita establecer un

plan previo para llevarlas a cabo, y los objetivos marcados en las mismas no son nada complejos. Los elementos que componen un guión son los siguientes:

- **Escenas:**

Son los sucesos descritos en el guión organizados en forma secuencial de manera que la realización de una escena permite que tenga lugar la siguiente.

- **Roles, objetos y lugares:**

Se corresponden con los personajes típicos que intervienen en el guión, los objetos que aparecen en los hechos descritos y los lugares donde acontecen las actividades propias del guión. Normalmente una persona, objeto o lugar concreto debe cumplir una serie de condiciones para poder entrar a formar parte del guión, las cuales se definen a través de un conjunto de restricciones que se asocian a cada rol, objeto o lugar del mismo.

- **Cabeceras:**

Permiten determinar si un guión es adecuado para explicar cierta situación, la cual no ha sido aún identificada.

Existen varios tipos de cabeceras:

- La que da **nombre** al guión.

Un ejemplo de su uso se puede dar en el proceso de comprensión de un texto en lenguaje natural, de modo que si apareciera en el mismo, por ejemplo, la palabra “restaurante”, el guión \$RESTAURANTE sería inmediatamente activado. La activación de este guón permite una mejor comprensión del resto de la información presente en el texto.

- La que representa **condiciones**.

Teniendo de nuevo en cuenta el guión \$RESTAURANTE, una de las posibles condiciones para ir a un restaurante sería “tener hambre”.

- La que representa **instrumentos**.

Cualquier guón correspondiente al viaje en un determinado medio de transporte (\$TREN, \$AVIÓN...) podría activarse al tener constancia de que alguien está realizando cualquier tipo de trayecto. En este caso el tren, avión... son los posibles “instrumentos” utilizados en la acción descrita en el guón.

- La que representa **lugares**.

El guión \$CINE podría ser activado ante la aparición en un texto de palabras como: “película”, “film”...

- **Resultados:**

Son un conjunto de hechos que serán ciertos una vez que se haya completado la secuencia de sucesos descritos en el guión.

Además de constituir un método de representación de conocimiento, los guiones han sido empleados en sistemas diseñados para la comprensión del lenguaje natural. En estos sistemas el proceso de inferencia se lleva a cabo del siguiente modo:

1) Primero se selecciona aquel guión que mejor explica la historia que se está analizando. Es a través de las cabeceras de los guiones como se decide el que pasa a estar activado.

2) A continuación se asignan valores a las variables del guión; es decir, se distribuyen los roles, objetos y lugares que intervienen en la historia.

3) Finalmente, a partir del guión seleccionado se extrae toda la información que no se ha podido obtener del texto escrito.

6.2.3 Apéndice 6.A: DIAVAL, ejemplo de un sistema completo

DIAVAL es un sistema experto destinado a ayudar a los médicos en el diagnóstico de enfermedades cardíacas, especialmente valvulopatías. Para ello, el programa considera los datos personales del enfermo, sus antecedentes, los síntomas y signos y los hallazgos de distintas pruebas clínicas, sobre todo de la ecocardiografía. Los síntomas corresponden a sensaciones que el paciente experimenta, tales como el dolor, las náuseas o la disnea (sensación de ahogo al respirar). Los signos son hechos objetivos que el médico observa, como la cianosis (coloración violácea, que indica falta de oxígeno en la sangre), la fiebre, un soplo cardíaco detectado en la auscultación, etc.

La base de conocimiento de DIAVAL consta de dos redes quasi-ortogonales¹: una red de clasificación y una red bayesiana. Ésta última contiene unos 300 nodos; la figura 6-3 muestra algunos de los que están más directamente relacionados con la insuficiencia mitral; los puntos suspensivos indican que la red se extiende hacia otras causas y otros efectos. En ella podemos distinguir dos tipos de nodos: unos corresponden a las alteraciones (enfermedad reumática, endocarditis, infarto agudo de

¹ Decimos “cuasi-ortogonales” en el sentido de que el lugar que un nodo ocupa en una red es casi independiente del que ocupa en la otra. Los marcos utilizados en este sistema sirven para representar el conocimiento sobre el funcionamiento de la red y no el conocimiento del dominio.

miocardio, etc.), mientras que otros, encerrados en óvalos de trazo grueso, representan los **datos** disponibles; en esta porción de la red aparecen un dato personal (la edad), un antecedente (la enfermedad reumática, registrada en la historia clínica), cuatro hallazgos del ecocardiograma bidimensional (vegetaciones, disminución de la movilidad, elongación de las cuerdas tendíneas y disminución de la movilidad) y un hallazgo del eco Doppler (regurgitación mitral).

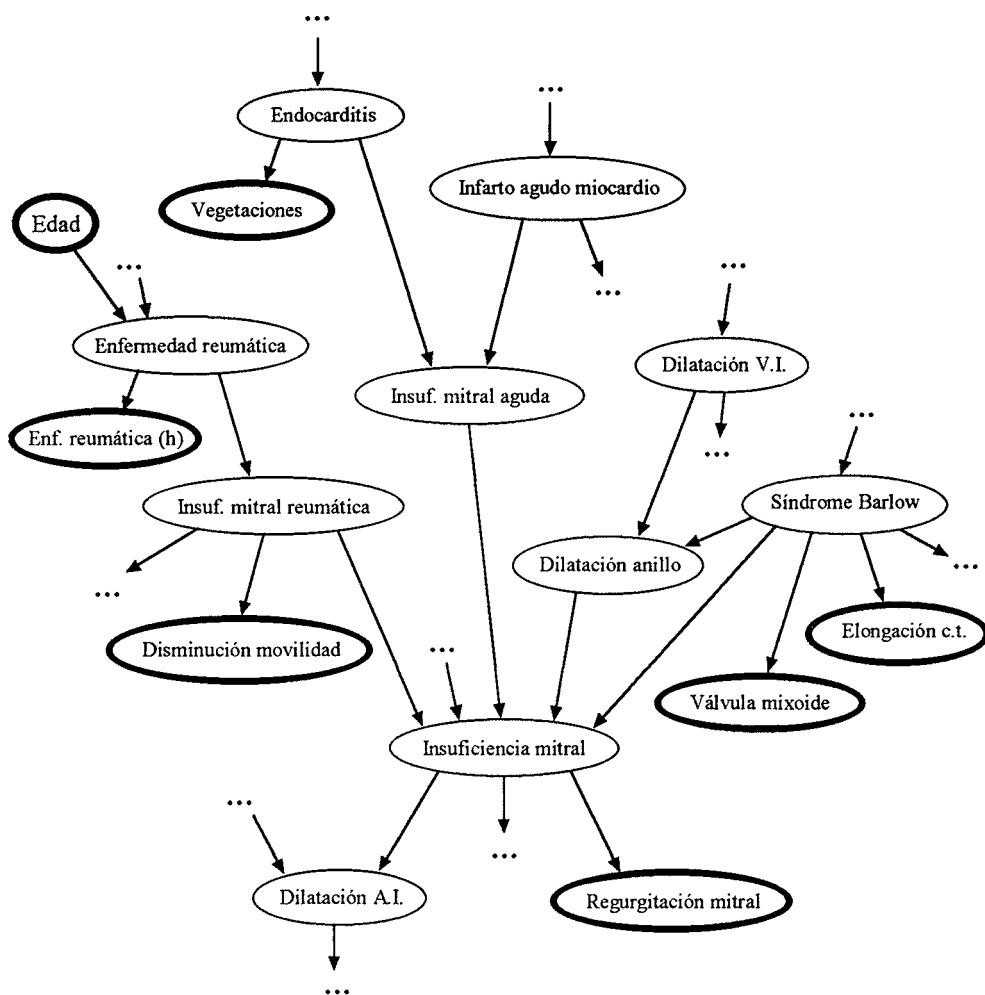


Figura 6-3 Red bayesiana de DIAVAL.

Los enlaces indican relaciones de *influencia* o de *causalidad* entre los nodos. Por ejemplo, la edad es uno de los *factores* que influyen en la probabilidad de padecer o haber padecido enfermedades reumáticas; la endocarditis, que en el eco se manifiesta en forma de vegetaciones, es *causa* de insuficiencia mitral aguda; etc. La distinción entre la alteración “enfermedad reumática [en el presente o en algún momento del pasado]” y el hallazgo “enfermedad reumática registrada en la historia clínica” es importante porque puede ocurrir que el paciente haya padecido la enfermedad sin que ningún médico la registrara y, a la inversa, puede figurar en la historia clínica un antecedente de enfermedad reumática anotado por error (aunque esto es más improbable). Cuando esta situación se da en pruebas de laboratorio suele hablarse de *falsos positivos* y *falsos negativos*; la frecuencia con que aparecen los falsos positivos y falsos negativos depende de la prevalencia de la enfermedad (la proporción de personas afectadas dentro de una determinada población) y de la sensibilidad y especificidad de la prueba en relación a la enfermedad.

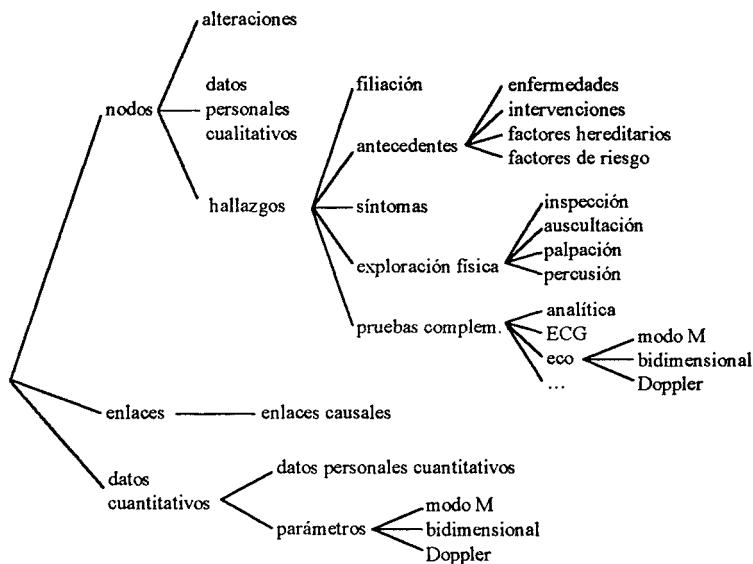


Figura 6-4 Red de clasificación de DIAVAL.

Como hemos mencionado al principio, los elementos que forman la base de conocimientos de DIAVAL se organizan en una **red de clasificación**, que se muestra en la figura 6-4. Cada una de las clases de esta red corresponde en la implementación del sistema experto a un marco, con campos heredados y campos propios; igualmente, cada elemento de una clase corresponde a una instancia. Por ejemplo, la clase-marco *nodos*

tiene tres subclases o submarcos: *alteraciones* (cuyas instancias son *endocarditis*, *infarto-agudo-miocardio*, *enf-reumática*, etc.), *datos-personales-cualitativos* (*sexo*, *edad*...) y *hallazgos*. El campo *nombre* del marco *nodo* contiene la cadena que representa el nodo en el interfaz de usuario; por ejemplo, el valor del campo *nombre* de la instancia *vm-insuf* es “insuficiencia mitral”. El campo *valores* indica los valores que puede tomar la variable. El valor por defecto de este campo es “ausente, presente”; por ejemplo, el campo *valores* de la instancia *endocarditis* no tiene valor asignado explícitamente, y por eso hereda el valor por defecto “ausente, presente”; en cambio, en la base de conocimiento se afirma que el campo *valores* de la instancia *vm-insuf* contiene la lista “ausente, leve, moderada, severa”. Otro de los campos del marco *nodo* se denomina *prob-posteriori*, y en él se registra la probabilidad a posteriori calculada para esa variable al propagar la evidencia; éste es, por tanto, un campo dinámico (se le asigna un valor durante el proceso de inferencia), a diferencia de los dos anteriores, *nombre* y *valores*, que eran estáticos (sus valores forman parte de la base de conocimientos).

La estructura de la red (el grafo) se implementa indicando los padres y los hijos de cada nodo. Concretamente, el campo *padres* se define dentro del marco *nodos*, para que lo hereden todos sus submarcos y todas sus instancias. En cambio, el campo *hijos* se define sólo dentro de los marcos *alteraciones* y *datos-personales-cualitativos*, pues si lo definiéramos dentro del marco *nodos*, lo heredaría también el submárcos *hallazgos*, lo cual no nos interesa porque —como dijimos anteriormente— en DIAVAL los hallazgos corresponden siempre a nodos terminales (sin hijos). Naturalmente, ambos campos son multivaluados.

Durante la creación o modificación de la red, a los campos *padres* e *hijos* se les puede asignar un demonio de modificación que compruebe que los valores insertados sean siempre instancias del marco *nodos*. Igualmente, otro demonio asignado al campo *padres* del marco *hallazgos* puede comprobar que este campo no esté vacío (porque cada nodo terminal debe tener al menos un parente, para no estar aislado). Obsérvese que cada campo se define y cada demonio se asigna en el lugar adecuado dentro de la jerarquía de marcos, con el fin de que lo hereden todas las instancias que deben llevarlo, y solamente ellas.

En la figura 6-4 aparece también el marco *enlaces*; por ejemplo, el enlace entre “edad” y “enfermedad reumática” es una instancia de este marco. Algunos de los campos que lleva asociados son *padre* (en el ejemplo anterior toma el valor “edad”), *hijo* (“enfermedad reumática”) y *explicación* (“Las personas de mayor edad tienen mayor probabilidad de haber padecido enfermedad reumática, por haber vivido más tiempo y porque hace años la incidencia de la enfermedades reumáticas era mayor.”).

Existe también un submarco *enlaces-causales* que, además de heredar los campos anteriores, tiene otros propios, como *sensibilidad*, *especificidad*, *valor-predictivo-positivo* y *valor-predictivo-negativo*. Si los enlaces no causales tuvieran características específicas (es decir, no aplicables a los enlaces causales), habría sido necesario crear un submarco *enlaces-no-causales*, con campos propios; sin embargo, como no es éste el caso, los enlaces no causales se definen directamente como instancias de *enlaces*.

Además de los nodos y enlaces de la red bayesiana, la base de conocimientos de DIAVAL contiene otro marco, *datos-cuantitativos*, con dos submarcos: *datos-personales-cuantitativos* (cuyas instancias son *edad*, *peso*, *estatura*, *superficie-corporal*, *DNI*...) y *parámetros* (cuyas instancias corresponden a las medidas realizadas en el ecocardiograma). Algunos de los campos del marco *datos-cuantitativos* son *valor*, *unidad-de-medida*, *mínimo* (cuyo valor por defecto es 0) y *máximo*. Por ejemplo, para la instancia *edad* el campo *unidad* tiene asignado el valor “años”, el *mínimo* es “0” (heredado por defecto) y el *máximo* “130”. El campo *valor* no tiene un valor asignado en la base de conocimiento, porque es distinto para cada paciente; en cambio, lleva asociado un demonio de modificación que se encarga de comprobar que el valor introducido en el campo *valor* de cada instancia es un número y que está comprendido entre el *máximo* y el *mínimo* de esa instancia.

Dentro del marco *datos-cuantitativos*, el campo *intervalos* está destinado a contener una lista que sirve para discretizar la variable y asignar así un valor al nodo de la red indicado en el campo *dato-cualitativo-asociado*. De este modo, cada dato personal numérico y cada dato medido en el ecocardiograma, una vez discretizado, se convierte en evidencia asociada a un nodo de la red bayesiana.

Una vez que se han introducido en DIAVAL todos los datos cuantitativos y cualitativos, asignando valores a los nodos correspondientes, comienza el proceso de inferencia. El algoritmo utilizado es el condicionamiento local [Diez, 1996]², que consiste en que cierto nodo, llamado “nodo-pivote”, solicita a cada uno de sus padres un mensaje π y a cada uno de sus hijos un mensaje λ . En general, cada nodo, para poder calcular el mensaje solicitado, debe solicitar a su vez mensajes a sus otros vecinos, y así sucesivamente. Cuando toda la evidencia disponible en la red ha llegado al nodo-pivote, éste envía un mensaje λ a cada uno de sus padres y un mensaje π a cada uno de sus hijos, y así sucesivamente. De este modo, por cada enlace de la red circulan dos mensajes, uno de ida y otro de vuelta. Con los mensajes recibidos, cada nodo

² [Diez, 1996] Díez Vegas, F. J., *Local conditioning in Bayesian Networks*. Artificial Intelligence, 87 (1996) 1-20.

calcula su probabilidad a posteriori. Existe un algoritmo, propio de DIAVAL, que se encarga de examinar todas las anomalías de la red con el fin de seleccionar entre ellas (en función de su probabilidad a posteriori y de su relevancia) las que van a formar parte del diagnóstico del enfermo.

Hay otros muchos detalles relativos a la implementación de DIAVAL cuya descripción superaría ampliamente el espacio del que disponemos. El lector interesado puede encontrar las referencias en Internet:

<http://www.dia.uned.es/~fjdiez/publicaciones.html>

Nos hemos limitado a explicar los aspectos del sistema experto que muestran cómo se puede implementar una red bayesiana mediante una estructura de marcos que constituye, a su vez, una red de clasificación; también hemos explicado cómo se realiza la inferencia mediante paso de mensajes. Ésta es una característica de muchos de los sistemas expertos de la actualidad, que combinan distintos métodos de representación del conocimiento y de inferencia, aprovechando las ventajas que ofrece cada uno de ellos.

6.3 Problemas resueltos

- **PROBLEMA 6.1:** (adaptado de [Winston, 1992])

En un periódico de información general se reciben noticias de sucesos de contenido muy variado. Se desearía disponer de algún método que captara el conocimiento general sobre dichos sucesos. Utilizar un sistema de marcos para llevar a cabo la labor mencionada.

SOLUCIÓN:

¿Qué pretende este problema? Se va a hacer hincapié en la estructuración del conocimiento de un dominio mediante una red jerárquica de conceptos dotada de herencia de propiedades.

El conocimiento general sobre las características de los sucesos que pueden dar lugar a una noticia se puede representar mediante una jerarquía de marcos donde cada nodo represente un tipo de suceso más o menos general y cada enlace una relación de superclase, de manera que se pueda construir una red de sucesos. En la parte alta de esta red figurarían los sucesos más generales, de los que otros sucesos más bajos en la misma podrían heredar información. A modo de ejemplo, en la red de sucesos debería quedar reflejada información del siguiente tipo:

El concepto más general que se manejará será el de suceso. Todo suceso se caracterizará por el lugar, día y hora en que ocurrió.

Existen diferentes tipos de sucesos: desastres naturales, acontecimientos deportivos, acontecimientos sociales...

Si se está considerando un desastre natural, habrá que tener en cuenta el número de muertos producidos, heridos, personas que se han quedado sin vivienda, daños materiales...

En todo acontecimiento deportivo interesaría reseñar el deporte del que se trata, quién es el ganador, el resultado final...

En un acontecimiento social podría ser interesante determinar el anfitrión, número de invitados...

Dentro de los desastres naturales cabría destacar los terremotos, inundaciones, huracanes...

En cuanto a cualquier terremoto es interesante conocer la falla que lo produjo, la magnitud del mismo...

Con respecto a las inundaciones es importante conocer, por ejemplo, el río que las ha producido...

En lo que se refiere a los huracanes, dos de sus características principales son su nombre y la máxima velocidad que ha alcanzado el viento.

Algunos tipos de acontecimientos sociales son las bodas, cumpleaños...

En una boda podría interesar saber quién es la novia, el novio, sus respectivos padres, cómo es el vestido de la novia...

Finalmente, en una fiesta de cumpleaños lo más reseñable es quién es la persona que cumple años y cuál es su edad.

Toda la información mencionada con anterioridad aparece de forma explícita en la jerarquía de marcos de la figura 6-5, donde los nombres de las clases se refieren a distintos tipos de sucesos y los campos de los que consta cada clase vienen a representar las características más importantes de cada suceso. En dicha figura cada enlace une un suceso con otro superior en la jerarquía, que es más general que el anterior.

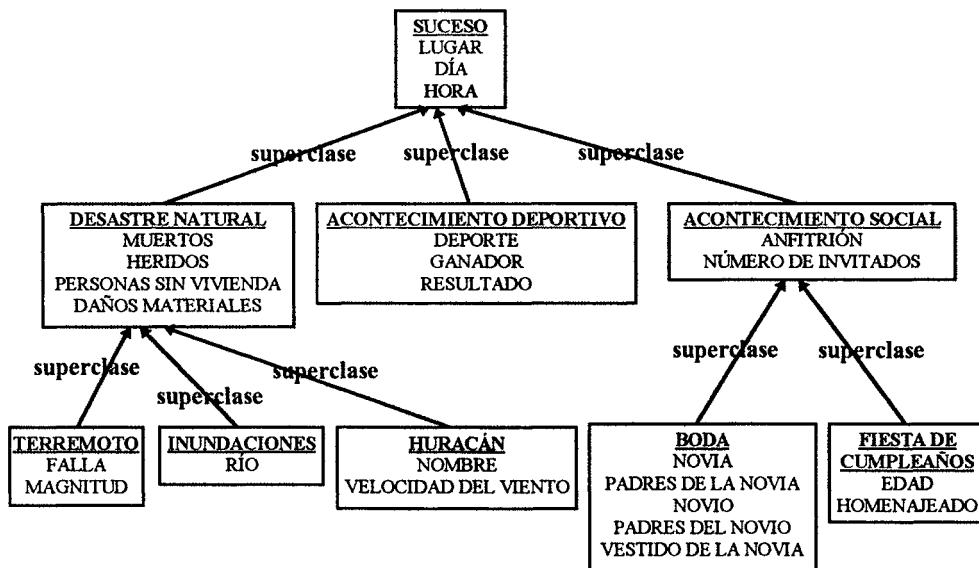


Figura 6-5

Como se puede observar en el árbol creado, en un nodo como pudiera ser “TERREMOTO” no es necesario señalar el lugar donde éste tuvo lugar al ser ésta una característica que hereda nodos superiores en la jerarquía.

- **PROBLEMA 6.2:** (adaptado de [Winston, 1992])

Dado el sistema de marcos construido en el problema anterior, ¿de qué capacidades habría que dotar al mismo para que constituyera un sistema automático para resumir noticias?

SOLUCIÓN:

¿Qué pretende este problema? Se presenta en este problema un ejemplo de cómo puede ser útil añadir a una red jerárquica de conceptos, conocimiento procedimental del dominio a través de los llamados “demonios”.

Para conseguir automatizar el proceso mencionado habría que ir completando las siguientes etapas:

- 1) Recepción de la noticia.
- 2) Identificación del tipo de noticia.

3) Llenado de los campos correspondientes de la jerarquía de marcos a partir de la información que aparece en la noticia.

4) Llenado de los huecos de un patrón resumen del suceso con los valores hallados para los campos de la jerarquía de marcos.

El paso de la etapa 1) a la 2) consiste en saber si la noticia hace referencia a una boda, terremoto... Para ello podría bastar con analizar el título de la noticia y las primeras frases de la misma. Una vez identificado el tipo de noticia, habría que llenar aquellos campos de la jerarquía de marcos construida en el problema anterior que tuvieran relación con el tipo de suceso en cuestión (paso de la etapa 2) a la 3)). Esto se puede conseguir mediante demonios de necesidad (procedimientos que actúan cuando se pretende conocer el valor de un campo y no puede obtenerse directamente o por herencia en la jerarquía). Algunos de los demonios mencionados serían:

Para llenar el campo HORA cuando se construye la noticia SUCESO, encuentre un número que incluya dos puntos y escribalo en el campo.

Para llenar el campo DÍA cuando se construye la noticia SUCESO, encuentre una palabra como “hoy”, “ayer”, “mañana” o el nombre de uno de los días de la semana y escribalo en el campo.

Para llenar el campo LUGAR cuando se construye la noticia SUCESO, encuentre un nombre que aparezca en un diccionario de lugares geográficos y escribalo en el campo.

Para llenar el campo DAÑOS MATERIALES cuando se construye la noticia DESASTRE NATURAL, encuentre el número que está junto a la palabra “pesetas” y escribalo en el campo.

Para llenar el campo MUERTOS cuando se construye la noticia DESASTRE, encuentre un entero cercano a una palabra que tenga que ver con muerte y escribalo en el campo.

Para llenar el campo FALLA cuando se construye la noticia TERREMOTO, encuentre un nombre propio cercano a la palabra “falla” y escribalo en el campo.

Para llenar el campo MAGNITUD cuando se construye la noticia TERREMOTO, encuentre un número decimal entre 1.0 y 10.0 y escribalo en el campo.

Finalmente, la noticia resumida se obtendría llenando los huecos del patrón correspondiente al suceso identificado en la etapa 2) con los valores de los campos obtenidos en la etapa anterior, la 3). Un ejemplo de patrón resumen correspondiente a un terremoto podría ser el siguiente:

Ocurrió un terremoto en *<valor del campo LUGAR>*, el día de *<valor del campo DÍA>*. Hubo *<valor del campo MUERTOS>* muertos y *<valor del campo DAÑOS-MATERIALES>* de pesetas de pérdidas materiales. La magnitud fue de *<valor del campo MAGNITUD>* en la escala Richter; la falla que provocó el terremoto fue la de *<valor del campo FALLA>*.

Gráficamente el proceso global que se ha descrito sería de la siguiente forma:

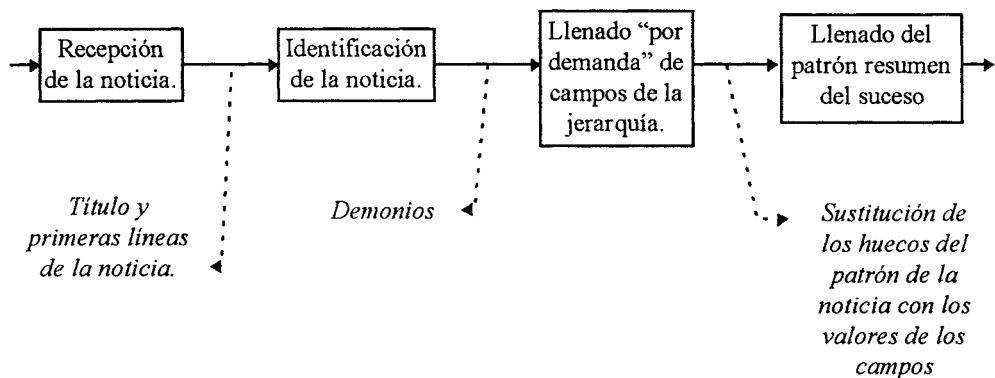


Figura 6-6

Considérese el siguiente ejemplo de aplicación del proceso por etapas visto para la siguiente noticia:

TERREMOTO EN PERÚ

Hoy, un serio terremoto de magnitud 8.5 sacudió Perú; el percance mató a 25 personas y ocasionó daños materiales por valor de 500 millones de pesetas. El Presidente del Perú dijo que el área más afectada, cercana a la falla de San Juan, ha sido durante años una zona de peligro.

En primer lugar, el sistema deduciría que el suceso noticiable se corresponde con un terremoto examinando el título de la noticia. A continuación se crearía una instancia de la clase TERREMOTO de manera que al intentarse hallar los valores de sus campos se haría uso de los demonios existentes en la jerarquía de marcos. El resultado final sería la creación de la siguiente instancia de la clase TERREMOTO:

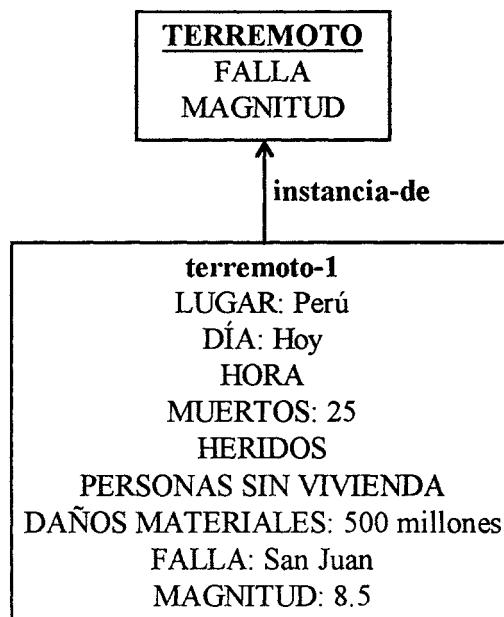


Figura 6-7

Una vez creada la instancia, no habría más que llevar a cabo un proceso de sustitución de los valores que aparecen en los campos de la misma en los huecos de la noticia resumen patrón, que quedaría de la siguiente forma:

Patrón resumen de terremoto sustituido

Ocurrió un terremoto en *Perú* el día de *hoy*. Hubo 25 muertos y *500 millones* de pesetas en pérdidas materiales. La magnitud fue de *8.5* en la escala Richter; la falla que provocó el terremoto fue la de *San Juan*.

- **PROBLEMA 6.3: (*)**

Definir un marco genérico que contenga la siguiente información referente a una persona: nombre, apellido, cónyuge, padres, hijos y si está vivo o no. Definir instancias del marco para representar la siguiente familia (sólo hace falta representar los elementos con asterisco *):

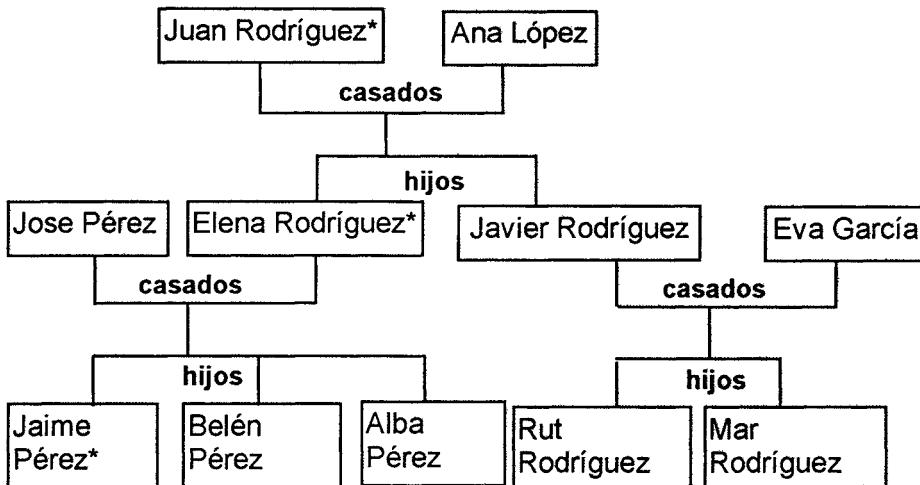


Figura 6-8

1) Definir demonios en pseudocódigo que actualicen en caso de fallecimiento las relaciones: padres, abuelos y tíos de una determinada persona.

2) Definir el conjunto de reglas que determinen cuáles son los herederos de una persona. Para ello se utilizará la jerarquía de objetos previamente definida y habrá que tener en cuenta las siguientes circunstancias:

- Si una persona tiene un cónyuge que está vivo, entre sus herederos estará el cónyuge.
- Si una persona tiene hijos que están vivos, éstos estarán entre sus herederos.
- Si una persona tiene un descendiente (hijo) que tiene hijos, los hijos del descendiente serán sus herederos, siempre que el cónyuge y sus hijos hayan fallecido.
- Si una persona no tiene hijos ni cónyuge ni padres vivos, sus herederos serán los herederos de sus padres.

3) Definir el conjunto de reglas que determinen cuáles son los hermanos políticos de una determinada persona.

La definición del conjunto de reglas debe realizarse en lógica de predicados y la notación que debe seguirse para la representación de marcos, instancias y demonios (con una notación de cláusulas de Prolog, tal como se indica en el ejemplo adjunto) es la siguiente:

Marcos:

```
(nombre-marco *sin-valor*
  (generalización
    (marco-1 marco-2 ... marco-n)
  (especialización
    (marco-1 marco-2 ... marco-n)
  (instancias (inst-1 inst-2 ... inst-n)
  (slot-1 *sin-valor*
    (slot-11 *sin-valor*)
    (slot-12 *sin-valor*
      (slot-121 *sin-valor*)
      (slot-122 *sin-valor*
        ....))))))
```

sin-valor es el valor por defecto que se asigna a los valores de los campos (o slots). Considérese el siguiente ejemplo de un marco que representa el concepto del animal doméstico “perro”:

```
(perro *sin-valor*
  (generalización
    (animal mamífero ... amigo-hombre)
  (especialización
    (cazadores guía ... policía)
  (instancias (toby ...))
  (longitud *sin-valor*)
  (altura *sin-valor*)
  (tamaño *sin-valor*
    (valor.asignado *sin-valor*
      (((tamaño ?x ?valor) :-
        (longitud ?x ?longitud)
        (≥ ?longitud 1m)
        (altura ?x ?altura)
        (≥ ?altura 1m)
        (asignar grande ?valor)))
      ((tamaño ?x ?valor) :-
        (longitud ?x ?longitud)
        (< ?longitud 1m)
        (altura ?x ?altura)
        (< ?altura 1m)
        (asignar pequeño ?valor)))))
  (amo *sin-valor*
    (valor.asignado *sin-valor*
```

```
((amо ?x ?amо):-  
  (dueño ?амо ?animales)  
  (elementо ?animales ?animal)  
  (igual ?animal ?x))  
  ....)))
```

Una instancia de la clase “perro” podría ser:

```
(perro toby  
  (generalización  
    (animal mamífero ... amigo-hombre)  
  (especialización  
    (cazadores guía ... policía)  
  (longitud 1.50m)  
  (altura 1.20m)  
  (tamaño grande)  
  (amo Fernando))))
```

Como puede apreciarse, el marco perro tiene asociado un demonio para el atributo tamaño que tiene dos reglas (todas las reglas introducidas se ejecutan secuencialmente hasta que una consiga satisfacer todas sus cláusulas) que sirven para determinar los dos valores alternativos de dicho atributo, que en este caso están en función de otros campos como longitud y altura. También existe un demonio para el atributo amo, para el que se han introducido los predicados igual y elemento, que junto con los predicados quitar (por ejemplo (quitar ?elemento ?lista) sirve para quitar un elemento de una lista) y asignar pueden ser útiles para resolver el problema planteado.

SOLUCIÓN:

En primer lugar definimos el marco genérico PERSONA con todos los campos señalados. Incluimos en la definición de este marco todos los demonios que permiten calcular los valores de los campos requeridos. Para ello seguimos la notación de cláusulas Prolog del ejemplo anterior. Hemos añadido una serie de comentarios clarificadores que están precedidos por el carácter de escape “;”.

Es importante darse cuenta de que tal y como se ha definido la propiedad AMOS del ejemplo ilustrativo, para obtener todos los *amos* de un determinado *perro* no es necesario definir la lista de amos posibles, la mera inclusión de la definición de la regla “(amo ?x ?amo)” hace que esta se ejecute tantas veces como sea necesario (es decir, una vez para cada instancia que cumpla las cláusulas de la regla correspondiente) y que ésta se genere automáticamente.

Para resolver el problema de los *demonios* hemos supuesto que todos son *demonios de acceso*; es decir, se ejecutan cada vez que se solicita el valor de un campo. Por ello, cada vez que una PERSONA pasa a estar muerta, no se actualiza automáticamente el

campo PADRES de sus hijos hasta que se haya preguntado de forma expresa por dicho campo. Entonces sí, el valor del campo PADRES lleva asociada la condición de que el *padre* debe ser un sujeto vivo. Por tanto, esta propiedad podría haberse llamado PADRES-VIVOS. Las mismas consideraciones son aplicables a los campos ABUELOS y TÍOS.

```
(persona *sin-valor*
  (generalización
    (animal-racional mamífero ser-vivo) ;;Podría tener más generalizaciones.
  (especialización
    (hombre mujer) ;;Son las dos especializaciones más elementales.

;; Los nombres de las instancias podrían haber sido simplemente el nombre de pila, ya que
;; no hay repeticiones en el ejemplo.

(instancias (juan_rodríguez ana_lópez josé_pérez elena_rodríguez javier_rodríguez
eva_garcía jaime_pérez belén_pérez alba_pérez rut_rodríguez mar_rodríguez)
(nombre *sin-valor*)
(apellido *sin-valor*)
(cónyuge *sin-valor*)

;; Una posibilidad para resolver el problema de los demonios que actualizan los valores de
;; los campos: PADRES, ABUELOS y TÍOS es preguntar por dichos valores cada vez que se
;; actualice el valor del campo VIVO (que indica si una persona ha fallecido o no). Para esto
;; podemos incluir cláusulas que pregunten por dichos valores dentro del subcampo
;; VALOR.ASIGNADO, que se aplica al asignar un valor al campo en cuestión. En su lugar
;; hemos optado por definir cada uno de los campos: PADRES, ABUELOS y TÍOS como
;; propiedades que tienen presente la propiedad estar VIVO.

(vivo *sin-valor*
  (valores-possibles (si no)))

;; La precisión de los valores posibles ayuda a evitar errores. Ahora cada uno de los
;; campos PADRES, HIJOS, ABUELOS y TÍOS deberían tener presente la propiedad estar
;; VIVO.

(padres *sin-valor*
  (valor.asignado *sin-valor*
    ((padres ?x ?padre) :-
      (hijos ?padre ?x)
      (vivo ?padre sí)))
  (hijos *sin-valor*
    (valor.asignado *sin-valor*
      ((hijos ?x ?hijo) :-
        (padres ?hijo ?x)
        (vivo ?hijo sí))))
```

:: Aunque no es una de las propiedades básicas del enunciado, la definición de ABUELOS :: podría simplificar la definición de otras propiedades.

```
(abuelos *sin-valor*
  (valor.asignado *sin-valor*
    ((abuelos ?x ?abuelo) :-
      (hijos ?padre ?x)
      (hijos ?abuelo ?padre)
      (vivo ?abuelo sí))))
```

:: Para definir la propiedad TÍOS podríamos habernos basado exclusivamente en las ::propiedades básicas del marco PERSONA, que según el enunciado son: NOMBRE, ::APELIDO, CÓNYUGE, PADRES, HIJOS y VIVO. En su lugar podría haberse optado por ::definir la propiedad adicional HERMANOS. De esta forma simplificaríamos la definición de ::las propiedades TÍOS y HERMANOS-POLÍTICOS.

```
(hermanos *sin-valor*
  (valor.asignado *sin-valor*
    ((hermanos ?x ?hermano) :-
      (hijos ?padre ?x)
      (hijos ?padre ?hermano))))
```

:: No obstante, utilizamos exclusivamente las propiedades básicas.

```
(tíos *sin-valor*
  (valor.asignado *sin-valor*
    ((tíos ?x ?tío) :-
      (hijos ?padre ?x)
      (hijos ?abuelo ?padre)
      (hijos ?abuelo ?tío)
      (vivo ?tío sí))))
```

:: A continuación se define la propiedad HEREDEROS teniendo presente todas las ::circunstancias mencionadas en el enunciado del problema. Cada uno de dichos puntos ::determina un tipo distinto de HEREDEROS.

```
(herederos *sin-valor*
  (valor.asignado *sin-valor*
    (((herederos ?x ?heredero) :-
      (cónyuge ?x ?heredero)
      (vivo ?heredero sí))
     ((herederos ?x ?heredero) :-
      (hijos ?x ?heredero)
      (vivo ?heredero sí))
     ((herederos ?x ?heredero) :-
      (cónyuge ?x ?cónyuge)
      (vivo ?cónyuge no)
      (hijos ?x ?hijo)))
```

:: Dado que se emplea lógica de predicados, podemos utilizar el predicado predefinido NO; así, al preguntar por si hay algún hijo vivo, se busca cualquier hijo que tuviera la propiedad VIVO con el valor SÍ. De esta forma se garantiza que no hay hijos vivos mediante:

```
(no (vivo ?hijo sí))
(hijos ?hijo ?heredero) ;Este ?heredero es en realidad un ?nieto.
(vivo ?heredero sí))
(herederos ?x ?heredero) :- 
(cónyuge ?x ?cónyuge)
(vivo ?cónyuge no)
(hijos ?x ?hijo)
```

:: Utilizando de nuevo la negación para garantizar la inexistencia de una propiedad:

```
(no (vivo ?hijo sí))
(padres ?x ?padre)
(no (vivo ?padre sí))
(herederos ?padre ?heredero))))
```

:: Finalmente sólo queda por determinar los hermanos políticos de la persona. Para simplificar los casos vamos a utilizar la propiedad HERMANOS definida previamente.

```
(hermanos-políticos *sin-valor* ;Este concepto equivale al de cuñado o cuñada.
(valor.asignado *sin-valor*)
((hermanos-políticos ?x ?hermano-p) :-
(cónyuge ?x ?cónyuge)
(hermanos ?cónyuge ?hermano-p)))
((hermanos-políticos ?x ?hermano-p) :-
(hermanos ?x ?hermano)
(cónyuge ?hermano ?hermano-p)))
((hermanos-políticos ?x ?hermano-p) :-
(cónyuge ?x ?cónyuge)
(hermanos ?cónyuge ?hermano-p1)
(cónyuge ?hermano-p1 ?hermano-p))))))
```

Una vez definida la entidad PERSONA, se definen las instancias según la gráfica adjunta. Todos aquellos atributos que no sean especificados en la definición de las instancias toman por defecto los valores señalados para la entidad genérica. Por ejemplo, partimos del hecho de que todas las personas de la jerarquía están vivas. Del conjunto de instancias posibles seleccionamos un conjunto reducido y representativo de las distintas posibilidades.

```
(persona juan_rodríguez
(nombre Juan)
(apellido Rodríguez)
(cónyuge ana_lopez)
(vivo sí)
(hijos (Elena Javier)))
```

```
(herederos (ana_lópez elena_rodríguez javier_rodríguez))

(persona elena_rodríguez
  (nombre Elena)
  (apellido Rodríguez)
  (cónyuge josé_pérez)
  (vivo sí)
  (padres (juan_rodríguez ana_lópez))
  (hijos (Jaime Belén Alba))
  (herederos (josé_pérez jaime_pérez belén_pérez alba_pérez))
  (hermanos-políticos (eva_garcía)))

(persona jaime_pérez
  (nombre Jaime)
  (apellido Pérez)
  (vivo sí)
  (padres (josé_pérez elena_rodríguez))
  ( tíos (javier_rodríguez eva_garcía))
  (abuelos (juan_rodríguez ana_lópez))
  (herederos (jose_pérez elena_rodríguez)))
```

• PROBLEMA 6.4: (*)

Considerar la siguiente red semántica que refleja parte del conocimiento manipulado por un sistema de diagnóstico médico. Represente el contenido de dicha red en forma de marcos y en lógica de predicados.

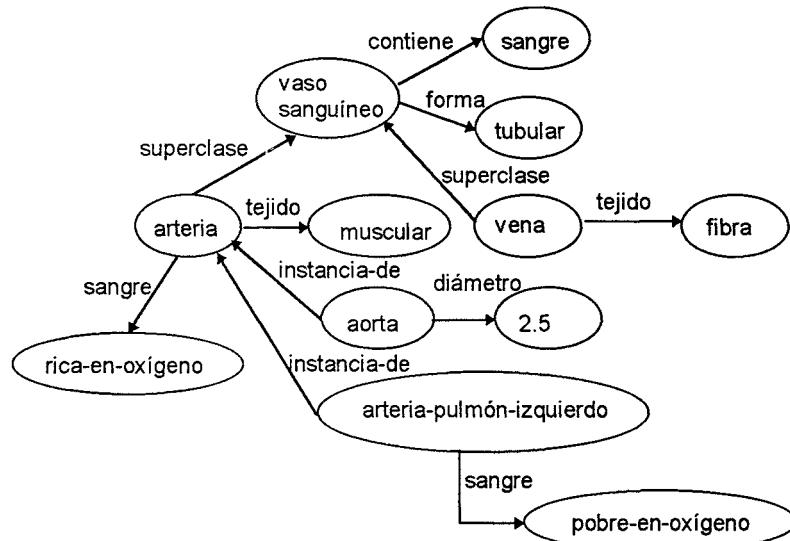


Figura 6-9

SOLUCIÓN:

¿Qué pretende este problema? En este problema se pretende mostrar que a la hora de representar conocimiento existen diferentes formalismos que pueden ayudar a completar dicha tarea.

El contenido de la red anterior en forma de marcos sería el siguiente:

clase vaso-sanguíneo es

subclase-de nil;

contiene=sangre;

forma=tubular

fin

clase arteria es

subclase-de vaso-sanguíneo;

tejido=muscular;

sangre=rica-en-oxígeno

fin

clase vena es

subclase-de vaso-sanguíneo;

tejido=fibra

fin

instancia aorta es

instancia-de arteria;

diámetro=2.5

fin

instancia arteria-pulmón-izquierdo es

instancia-de arteria;

sangre=pobre-en-oxígeno

fin

En forma gráfica tendríamos una jerarquía del siguiente tipo:

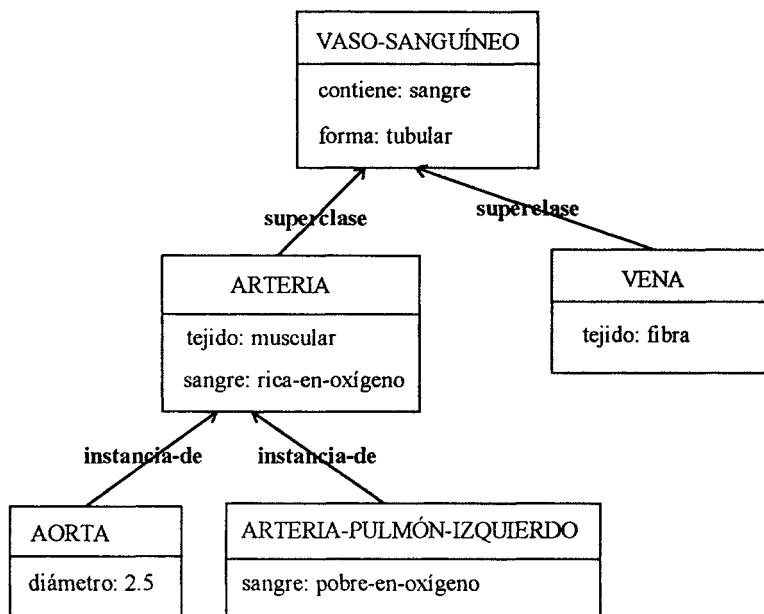


Figura 6-10

En lógica de predicados el contenido de la red sería el siguiente:

arteria(aorta)
diámetro(aorta, 2.5)

arteria (arteria-pulmón-izquierdo)
sangre(arteria-pulmón-izquierdo, pobre-en-oxígeno)

$\forall x \text{ arteria}(x) \rightarrow \text{tejido}(x, \text{muscular})$
 $\forall x \text{ arteria}(x) \rightarrow \text{sangre}(x, \text{rica-en-oxígeno})$
 $\forall x \text{ arteria}(x) \rightarrow \text{vaso-sanguíneo}(x)$

$\forall x \text{ vena}(x) \rightarrow \text{tejido}(x, \text{fibra})$
 $\forall x \text{ vena}(x) \rightarrow \text{vaso-sanguíneo}(x)$

$\forall x \text{ vaso-sanguíneo}(x) \rightarrow \text{contiene}(x, \text{sangre})$
 $\forall x \text{ vaso-sanguíneo}(x) \rightarrow \text{forma}(x, \text{tubular})$

Podría haberse pensado en representar las reglas de inferencia en sentido contrario, como por ejemplo:

$\forall x \text{ vaso-sanguíneo}(x) \wedge \text{sangre}(x, \text{rica-en-oxígeno}) \wedge \text{tejido}(x, \text{muscular}) \rightarrow \text{arteria}(x)$

Sin embargo, no hay nada en el enunciado que haga pensar que los antecedentes de esta regla sean condiciones suficientes de arteria.

¿Qué dificultades puede haber encontrado en este problema?

Aplicación 1 de la regla D:

El lector no entiende la nomenclatura empleada para la descripción de marcos.

Acciones:

Dicha nomenclatura se explica más adelante en los problemas 6.5 y 6.6.

Aplicación 2 de la regla D:

El lector no entiende la correspondencia entre el sistema de marcos del problema y la representación dada en lógica de predicados.

Acciones:

Probablemente el lector no ha captado el significado de los enlaces “superclase” e “instancia-de”. Convendría que repasara la sección 6.2.1 del apartado teórico.

• **PROBLEMA 6.5:** (adaptado de [Lucas & Van der Gaag, 1991])

Considérese el siguiente extracto proveniente del campo médico:

Arterias y venas son dos tipos de vasos sanguíneos del cuerpo humano. Arterias principales y arterias secundarias son, a su vez, dos tipos de arterias. Como ejemplos de arterias principales se pueden citar la aorta y la arteria braquial izquierda. La arteria braquial izquierda tiene un diámetro de unos 4 cms., está localizada en el brazo y contiene sangre rica en oxígeno.

Una característica importante de una arteria es su localización; como posibles localizaciones de las mismas se van a considerar: brazo, cabeza, pierna y tronco.

a) Estructurar el conocimiento encerrado en el extracto anterior mediante un sistema de marcos.

b) ¿Cómo habría que cambiar la representación anterior si se añadiera la siguiente información?:

Se sabe que la mayoría de las arterias contienen sangre rica en oxígeno.

c) Finalmente, cómo habría que ampliar nuestro sistema de marcos si:

Es sabido, también, que las arterias pulmonares derecha e izquierda son una excepción a la propiedad descrita en el apartado anterior; es decir, estas arterias transportan sangre pobre en oxígeno.

SOLUCIÓN:

- a) Una jerarquía de marcos que represente fielmente la información que aparece en el texto del enunciado sería:

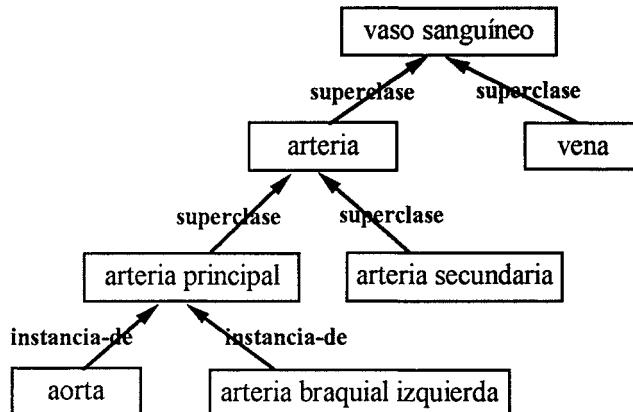


Figura 6-11

Se utilizará la siguiente nomenclatura o definición de marco:

<marco>	::=	<clase> <instancia>
<clase>	::=	clase <nombre-de-clase> es superclase <espec-super>; <atributos-de-clase> fin
<instancia>	::=	instancia <nombre-de-instancia> es instancia-de <espec-super>; <atributos-de-instancia> fin
<espec-super>	::=	<nombre-de-clase> nil
<atributos-de-clase>	::=	<declaración>{;<declaración>}* <vacío>

```
<atributos-de-instanci> ::= <par-atributo-valor>{;<par-atributo-valor>}* | <vacío>
<declaración>      ::= <par-atributo-tipo> | <par-atributo-valor>
<par-atributo-tipo> ::= <nombre-de-atributo> : <tipo>
<par-atributo-valor> ::= <nombre-de-atributo>=<valor>
<tipo>              ::= entero | real | cadena |<conjunto>|<nombre-de-clase>
<valor>             ::= <constante-elemental> | <nombre-de-instanci>
<vacío>            ::=
```

(Se usa **nil** para indicar que el marco correspondiente es la raíz de la jerarquía; por tanto, estamos suponiendo que la estructura de la red de marcos correspondiente a este problema es un árbol. **<conjunto>** estará formado por constantes elementales y nombres de instancia separados por comas y encerrados en llaves. Por otra parte, **<constante-elemental>** puede ser una constante de tipo real o entero, o una cadena de caracteres).

Se tiene:

```
clase vaso-sanguíneo es
    superclase nil;
fin
```

```
clase arteria es
    superclase vaso-sanguíneo;
    localización:{brazo, cabeza, pierna, tronco}
fin
```

```
clase vena es
    superclase vaso-sanguíneo;
fin
```

```
clase arteria-principal es
    superclase arteria;
fin
```

```
clase arteria-secundaria es
    superclase arteria;
fin
```

```

instancia aorta es
  instancia-de arteria-principal;
fin

```

```

instancia arteria-braquial-izquierda es
  instancia-de arteria-principal;
  diámetro=0.04;
  localización=brazo;
  sangre=rica-en-oxígeno
fin

```

Se podría considerar el valor *brazo* del atributo *localización* perteneciente a la clase *arteria* como una instancia de una clase llamada *miembro*:

```

clase miembro es
  superclase nil;
  posición:{inferior, superior}
fin

```

```

instancia brazo es
  instancia-de miembro;
  posición=superior
fin

```

b) La nueva representación sería ahora:

```

clase arteria es
  superclase vaso-sanguíneo;
  localización:{brazo, cabeza, pierna, tronco};
  sangre=rica-en-oxígeno
fin

```

```

instancia arteria-braquial-izquierda es
  instancia-de arteria-principal;
  diámetro=0.04;
  localización=brazo
fin

```

Lo que se ha conseguido con el nuevo atributo que aparece en la clase *arteria* es generalizar a todas ellas el hecho de que transporten sangre rica en oxígeno, tal como se menciona en el texto del enunciado.

c) Para tratar este tipo de excepciones basta con crear instancias donde quede reflejada la información relacionada con dichas excepciones:

```
instancia arteria-pulmonar-izquierda es
  instancia-de arteria;
  sangre=pobre-en-oxígeno
fin
```

```
instancia arteria-pulmonar-derecha es
  instancia-de arteria;
  sangre=pobre-en-oxígeno
fin
```

• **PROBLEMA 6.6:** (adaptado de [Lucas & Van der Gaag, 1991])

Dada la siguiente información:

Los vasos sanguíneos del cuerpo humano tienen forma tubular y contienen sangre. Arterias y venas son dos tipos de vasos sanguíneos.

Las arterias tienen pared muscular, sangre por lo general rica en oxígeno, presión sanguínea por lo general de valor 20, flujo sanguíneo normalmente de valor 4, un porcentaje del volumen sanguíneo de valor 20 y, finalmente, su valor de resistencia se calcula dividiendo su presión sanguínea entre su flujo sanguíneo.

Las venas tienen pared fibrosa y sangre por lo general pobre en oxígeno.

Las arterias de gran calibre, arterias de pequeño calibre y anastomosis arterio-venosas son tipos de arterias. Las anastomosis arterio-venosas también son un tipo de venas.

Las arterias de gran calibre tienen un porcentaje del volumen sanguíneo de valor 11 y una presión media de valor 100.

Las arterias de pequeño calibre poseen un porcentaje del volumen sanguíneo de valor 7.

Las anastomosis arterio-venosas tienen sangre mezclada (pobre y rica en oxígeno).

Como ejemplos de arterias de gran calibre se pueden citar la aorta, la arteria braquial izquierda y la arteria pulmonar izquierda.

La aorta posee un diámetro de valor 2.5.

La arteria braquial izquierda posee un diámetro de valor 0.04 y está localizada en el brazo.

La arteria pulmonar izquierda tiene sangre pobre en oxígeno.

Como ejemplo de arteria de pequeño calibre se citará la arteria cubital izquierda.

Crear un sistema de marcos que capte el conocimiento encerrado en el extracto anterior.

SOLUCIÓN:

Aparecen ahora varias novedades con respecto al anterior problema basado en este mismo tipo de dominio médico. Estas novedades son: introducción de *valores por defecto* (añadidos a un campo, sólo son válidos si no se ha podido asignar ningún valor al mismo), existencia de herencia múltiple (un marco puede tener más de un marco padre en la red) y existencia de procedimientos asociados a un campo (también conocidos como demonios). Las novedades antes reseñadas podrán ser reflejadas utilizando la siguiente definición de marco:

```

<marco>          ::=  <clase> | <instancia>
<clase>           ::=  clase <nombre-de-clase> es
                           superclase <espec-super>;
                           <atributos>
                           fin
<instancia>        ::=  instancia <nombre-de-INSTANCE> es
                           instancia-de <espec-super>;
                           <atributos>
                           fin
<espec-super>      ::=  <nombre-de-clase>{,<nombre-de-clase>}* | nil
<atributos>         ::=  <par-atributo-faceta>{;<par-atributo-faceta>}* | 
                           <vacío>
<par-atributo-faceta> ::=  <nombre-de-atributo>=(<faceta>{,<faceta>}*)
<faceta>            ::=  <nombre-de-faceta> <valor> |
                           demonio <tipo-de-demonio>
                           <llamada-a-demonio>
<nombre-de-faceta>  ::=  valor | valor-por-defecto
<tipo-de-demonio>   ::=  si-se-necesita | si-se-añade | si-se-borra

```

```
<valor> ::= <constante-elemental> | <nombre-de-instancia>  
<vacío> ::=
```

Con la anterior nomenclatura podría construirse el siguiente sistema de marcos:

clase vaso-sanguíneo es

superclase nil;
forma=(valor tubular);
contiene=(valor sangre)

fin

clase arteria es

superclase vaso-sanguíneo;
pared=(valor muscular);
sangre=(valor-por-defecto rica-en-oxígeno);
presión-sanguínea=(valor-por-defecto 20);
flujo-sanguíneo=(valor-por-defecto 4);
porcentaje-del-volumen-sanguíneo=(valor 20);
resistencia=(demonio si-se-necesita
R(presión-sanguínea, flujo-sanguíneo))

fin

clase vena es

superclase vaso-sanguíneo;
pared=(valor fibrosa);
sangre=(valor-por-defecto pobre-en-oxígeno)

fin

clase arteria-de-gran-calibre es

superclase arteria;
porcentaje-del-volumen-sanguíneo=(valor 11);
presión-media=(valor 100)

fin

clase arteria-de-pequeño-calibre es

superclase arteria;
porcentaje-del-volumen-sanguíneo=(valor 7)

fin

clase anastomosis-arterio-venosa es

superclase arteria, vena;

sangre=(valor mezclada)

fin

instancia aorta es

instancia-de arteria-de-gran-calibre;

diámetro=(valor 2.5)

fin

instancia arteria-braquial-izquierda es

instancia-de arteria-de-gran-calibre;

diámetro=(valor 0.04);

localización=(valor brazo)

fin

instancia arteria-pulmonar-izquierda es

instancia-de arteria-de-gran-calibre;

sangre=(valor pobre-en-oxígeno)

fin

instancia arteria-cubital-izquierda es

instancia-de arteria-de-pequeño-calibre

fin

Gráficamente se tendría la red de la figura 6-12.

- **PROBLEMA 6.7:** (adaptado de [Jackson, 1990])

Supóngase que el trabajo de un agente comercial consiste en valorar económicamente parcelas de terreno que aparecen en el mercado, incluso aunque no se posea toda la información necesaria para obtener el precio exacto. Para llevar a cabo esta labor el agente maneja la siguiente información:

La mayoría de las parcelas tienen forma de cuadriláteros convexos; además, en caso de conocerse la geometría exacta de la parcela, se dispone de un algoritmo que calcula de forma precisa su área:

Procedimiento-1:

El área de cualquier polígono de n lados se halla sumando las $n-2$ áreas de los triángulos que se pueden formar con el mismo.

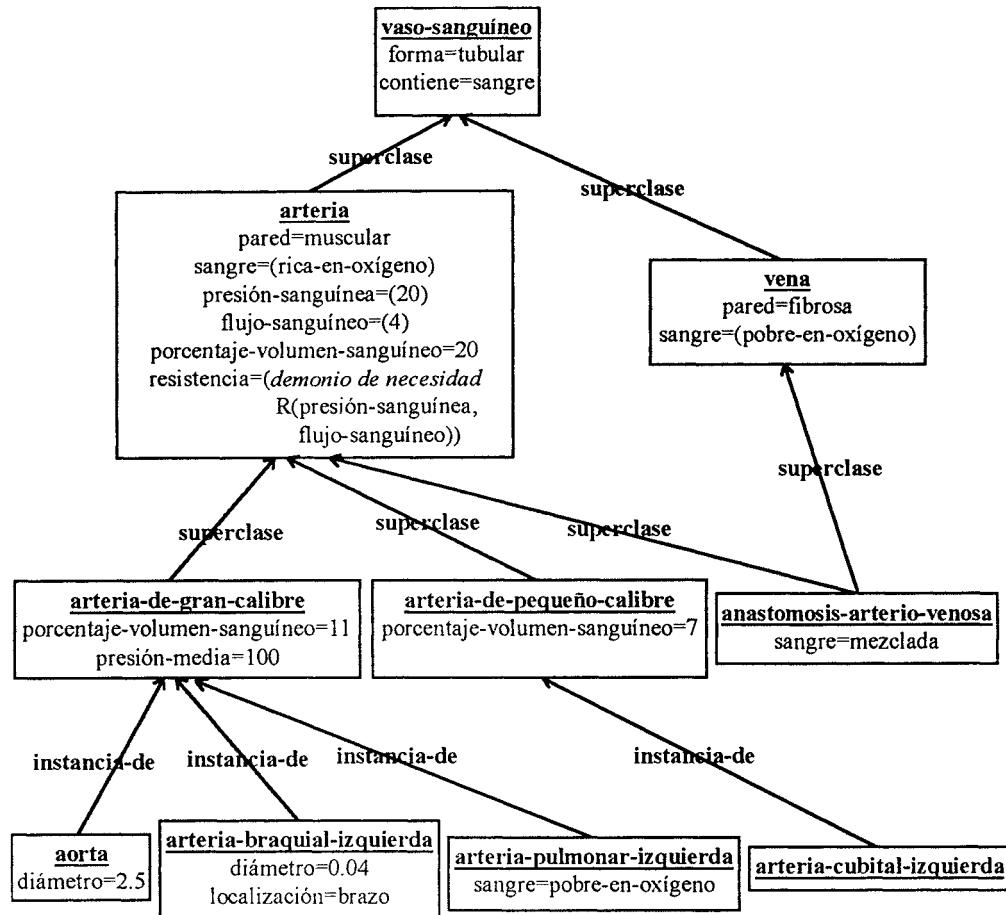


Figura 6-12

Si no fuera posible aplicar el algoritmo anterior, porque no se conociera la geometría exacta de la parcela, para el caso de cuadriláteros se dispone de una función heurística que realiza un cálculo estimado del área, una vez conocidas las cuatro longitudes de los lados:

Heurística-1:

Una estimación del área de un cuadrilátero convexo sería la multiplicación de la media de las longitudes de un par de lados opuestos por la media del otro par.

Finalmente, el precio de las parcelas se considera proporcional a su área.

Organice toda la información reseñada con anterioridad de forma estructurada mediante un sistema de marcos.

SOLUCIÓN:

En cuanto a la jerarquía de clases que se va a manejar, una parcela cualquiera va a ser asociada a un polígono. Como se sabe que la mayoría de las parcelas son cuadriláteros convexos, se construye el siguiente enlace:

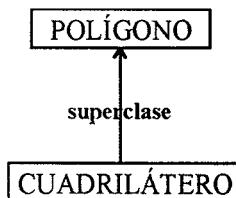


Figura 6-13

Ya que se sabe que en muchas ocasiones va a poderse conocer la geometría exacta de una parcela, sería conveniente ampliar la jerarquía anterior de la siguiente forma:

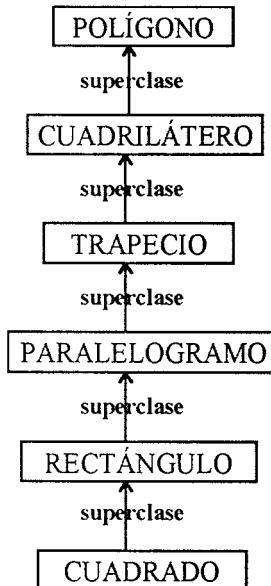


Figura 6-14

En cuanto a los campos que debería tener cada clase, podría resultar suficiente con diseñar la clase polígono de la siguiente forma:

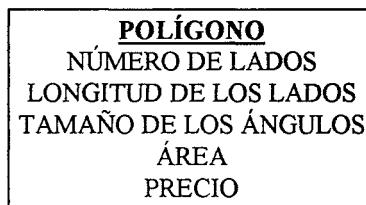


Figura 6-15

de manera que el resto de clases heredara estos campos.

¿Cómo habría que ampliar la jerarquía de marcos hasta ahora construida para que en ella quedara reflejado el resto de la información que maneja el agente comercial? En primer lugar, sería conveniente añadir un *valor por defecto* al campo NÚMERO DE LADOS de la clase POLÍGONO. Este valor sería 4, ya que se sabe que en la mayoría de los casos las parcelas van a ser de cuatro lados (ver figura 6-16). Evidentemente, para el resto de las clases, desde cuadrilátero hasta cuadrado, este valor pasaría a ser un valor real.

En cuanto al procedimiento de cálculo del área de un polígono de n lados como suma de las áreas de $n-2$ triángulos, podría pasar a ser un *demonio de necesidad* asociado al campo ÁREA de la clase POLÍGONO. Un requisito necesario para poder utilizar este demonio sería que se conocieran los valores de los campos LONGITUD DE LOS LADOS y TAMAÑO DE LOS ÁNGULOS de la clase POLÍGONO. Este procedimiento sería heredado por el campo ÁREA del resto de clases.

Con respecto al algoritmo que establece que el precio de una parcela es proporcional a su área, se implementará como un *demonio de asignación* asociado al campo ÁREA de la clase POLÍGONO; es decir, cada vez que se introduzca el valor del área de una parcela, se calculará un nuevo precio para la misma. De nuevo este demonio es heredado por el resto de campos del mismo nombre del resto de clases de la jerarquía:

Procedimiento-2:

$$\text{PRECIO} = k \cdot \text{ÁREA} \quad (k=\text{constante})$$

Ocurre ciertas veces que sólo se conocen los valores de la longitud de los lados de una parcela que se sabe que es un cuadrilátero. En estos casos es posible aplicar la heurística mencionada en el enunciado del problema para obtener una estimación del área de la parcela. De cualquier modo, será necesario comprobar, antes de utilizar esta

heurística, si el método exacto de cálculo de áreas puede ser aplicado. Por tanto, se podría asociar un demonio de necesidad al campo ÁREA de la clase CUADRILÁTERO que tendría la siguiente forma:

Procedimiento-3:

Si hay información sobre los lados y ángulos:

 Llamar al **procedimiento-1**, asociado a la clase POLÍGONO

 Acabar

Si hay información sólo sobre lados:

 Aplicar **heurística-1**

 Acabar

Si no existe información sobre lados ni ángulos:

 Permanecer en “situación de espera” hasta que llegue información

Finalmente, se podrían añadir demonios de necesidad más específicos para el cálculo del valor del campo ÁREA de clases como:

TRAPECIO:

Procedimiento-4:

ÁREA=base·altura·0'5

PARALELOGRAMO:

Procedimiento-5:(Es heredado por las clases RECTÁNGULO y CUADRADO).

ÁREA=base·altura

Se tendría, por tanto, la jerarquía de la figura 6-16. Obsérvese que en el caso de un trapecio, paralelogramo, rectángulo o cuadrado, para calcular el área de cualquiera de los mismos, se prefiere un algoritmo más específico antes que usar el algoritmo heredado que sumaba áreas de triángulos. Sin embargo, para el caso de un cuadrilátero, en general, al disponerse únicamente de una heurística y no de un procedimiento exacto, hay que dar prioridad al algoritmo heredado de la clase POLÍGONO.

• **PROBLEMA 6.8: (*)**

Supongamos que Juan Pérez y Antonio Fernández son dos jugadores de baloncesto. El primero es un pivot y juega con el Estudiantes y el segundo es un base y juega con el Cáceres. El entrenador de Juan es Pedro Pérez y el de Antonio es Daniel Gómez. Todos los equipos tienen un entrenador, un color del uniforme y un número de jugadores. Ambos tipos de jugadores —pivot y base— tienen asociado un porcentaje de canastas por partido. Los pivots se caracterizan por poner más de 5 tapones por partido y los bases por superar las 6 asistencias por partido. Por otro lado, cualquier jugador de baloncesto posee entre sus atributos más significativos: altura, número de asistencias

por partido y porcentaje de canastas. Cuando el número de tapones por partido más el de asistencias supera el valor de 10 y el porcentaje de canastas excede el 70%, entonces el jugador se considera excelente. Los jugadores de baloncesto son hombres adultos y como tales tienen entre sus rasgos la altura. Los hombres adultos son personas y éstas tienen como cualidad su edad. Además, las personas son mamíferos y como tales tienen la propiedad de estar vivos.

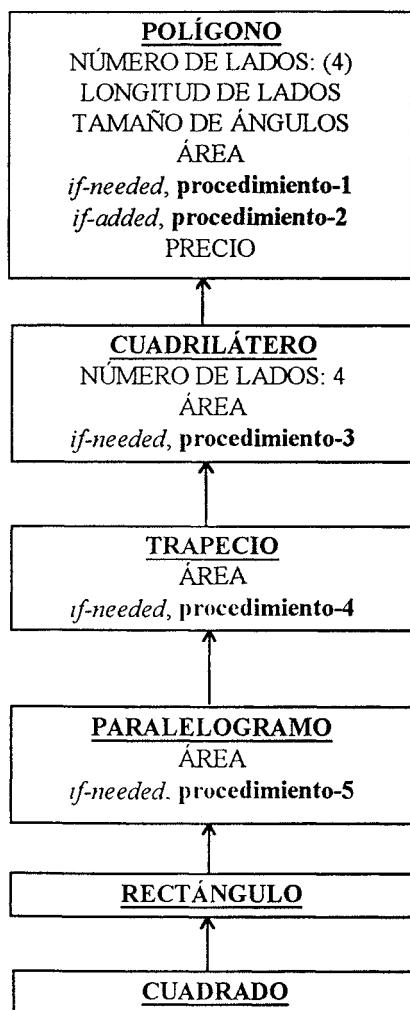


Figura 6-16

a) Partimos de que la clase más básica es *Clase* y de que ésta representa el conjunto de todas las clases. Represente todos los marcos que forman la red de herencia descrita en la narración previa, distinguiendo los que son clases y los que son instancias. Asigne de la forma más conveniente los atributos a cada una de las clases e instancias del conjunto, marcando con un asterisco * todas las propiedades que sean heredadas. Defina e identifique los demonios que necesite y señale su tipo.

b) Si se diera la circunstancia de que Juan Pérez fuera, además, jugador de balonmano y de que los jugadores de balonmano se caracterizaran por efectuar más de 8 asistencias por partido, ¿cómo solucionaría este conflicto?, ¿qué mecanismo genérico permite resolver este problema?

SOLUCIÓN:

a) La jerarquía de objetos se muestra en la figura 6-17 donde aparecen con asterisco "*" las propiedades heredadas a lo largo de la jerarquía. En dicho gráfico se observa que para averiguar el nombre del entrenador de un jugador de baloncesto concreto se puede preguntar por su *equipo* y desde esta entidad se accede al nombre buscado. Se han dibujado dos flechas con trazo discontinuo para expresar esta circunstancia. Si se hubiera realizado un planteamiento distinto, basado en introducir directamente un campo *entrenador* para cada jugador, cada vez que se asignara o cambiara el valor de dicho campo, habría que activar un *demonio* que comprobara que aquél es consistente con la información del equipo al que pertenece dicho jugador.

En segundo lugar, se definen los objetos del dominio. Hasta ahora, se han ilustrado las soluciones de otros ejercicios similares con dos notaciones diferentes (cuálquiera de ellas sería igualmente válida):

a) En pseudocódigo:

```

clase <nombre-de-clase> es
    superclase <espec-super>,
    <atributos-de-clase>
fin
```

b) Mediante lenguajes de manipulación simbólica:

```

(nombre-marcos *sin-valor*
  (generalización
    (marco-1 marco-2 ... marco-n)
  (especialización
    (marco-1 marco-2 ... marco-n)
  (instancias (inst-1 inst-2 ... inst-n)
  (slot-1 *sin-valor*
    (slot-11 *sin-valor*))
```

```
(slot-12 *sin-valor*
(slot-121 *sin-valor*)
(slot-122 *sin-valor*
....)))
```

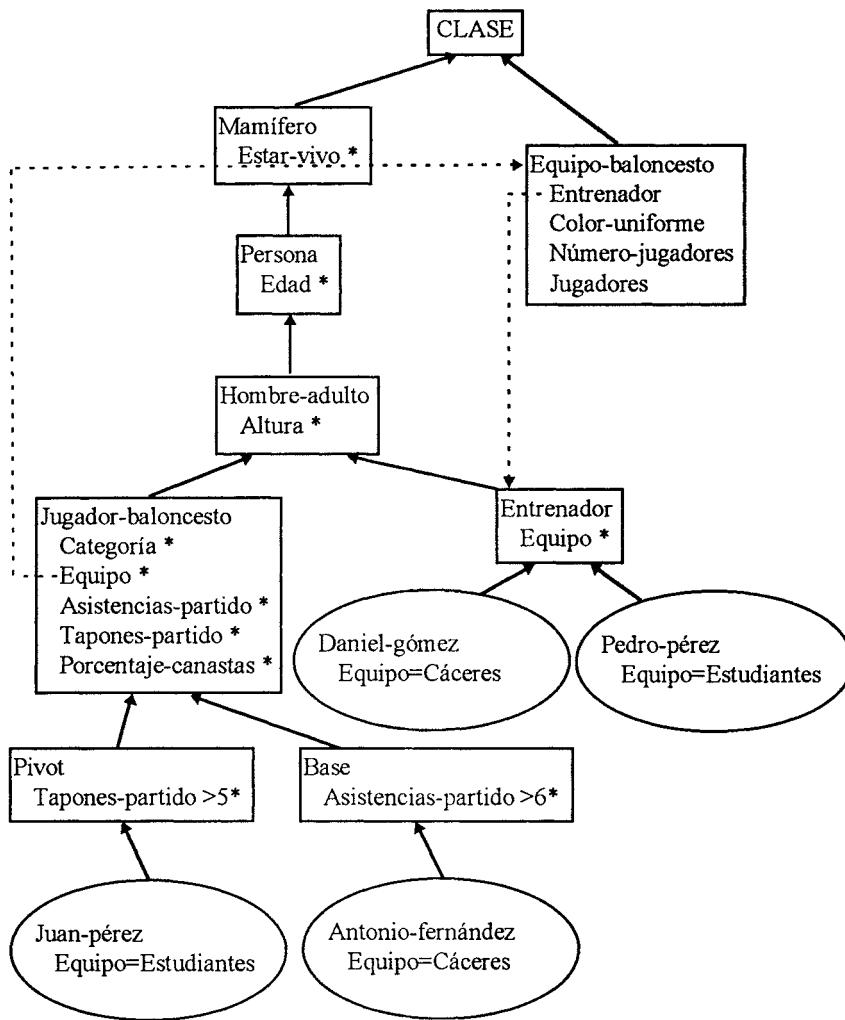


Figura 6-17

En este ejercicio se va a introducir una nueva notación, bastante cercana a los lenguajes de programación orientados a objetos. Estos lenguajes, al ser fuertemente

tipados, son más eficientes que los basados en listas (como LISP), pero son menos flexibles.

```
{ CLASE jugador-baloncesto
  SUPERCLASE hombre-adulto
  ATRIBUTOS:
    categoria {excelente, no-excelente}
    DEMONIOS {NECESIDAD tipo-categoría};
    equipo {caceres, estudiantes, caja-cantabria...};
    ENTERO asistencias-partido;
    ENTERO tapones-partido;
    REAL porcentaje-canastas }

{ CLASE pivot
  SUPERCLASE jugador-baloncesto
  ATRIBUTOS:
    ENTERO tapones-partido
    DEMONIOS {ASIGNACION tapones-minimos }

{ CLASE base
  SUPERCLASE jugador-baloncesto
  ATRIBUTOS:
    ENTERO asistencias-partido
    DEMONIOS {ASIGNACION asistencias-minimas }

{ CLASE equipo-baloncesto
  SUPERCLASE Clase
  ATRIBUTOS:
    SIMBOLO entrenador;
    color-uniforme {blanco, verdinegro, azulgrana...};
    ENTERO numero-jugadores;
    LISTA jugadores }

{ CLASE entrenador-baloncesto
  SUPERCLASE hombre-adulto
  ATRIBUTOS:
    SIMBOLO equipo }

{ CLASE hombre-adulto
  SUPERCLASE persona
  ATRIBUTOS:
    REAL altura }
```

```
{ CLASE persona
  SUPERCLASE mamiferos
  ATRIBUTOS:
    ENTERO edad }

{ CLASE mamifero
  SUPERCLASE clase
  ATRIBUTOS:
    ENTERO altura }
```

Dado que Juan Pérez es un pivot, se sabe que debe superar los cinco tapones por partido. Por tanto:

```
{ INSTANCIA juan-perez
  CLASE pivot
  ATRIBUTOS:
    equipo = estudiantes;
    entrenador = pedro-perez
    tapones-partido = 6 } /*6 es un valor supuesto*/
```

Dado que Antonio Fernández es un base, se sabe que debe superar las seis asistencias por partido. Por tanto:

```
{ INSTANCIA antonio-fernandez
  CLASE base
  ATRIBUTOS:
    equipo = caceres;
    entrenador = daniel-gomez;
    asistencia-partido = 7 } /*7 es un valor supuesto*/

{ INSTANCIA pedro-perez
  CLASE entrenador-baloncesto
  ATRIBUTOS:
    equipo = estudiantes }

{ INSTANCIA daniel-gomez
  CLASE entrenador-baloncesto
  ATRIBUTOS:
    equipo = caceres }
```

Quedan por definir los demonios mencionados previamente:

```

{ jugador-baloncesto.categoría::tipo-categoría
( ARGUMENTOS:
  ?jugador )
( CUERPO:
  SI
    (?jugador.tapones-partido +
     ?jugador.asistencias-partido > 10) ∧
     (?jugador.porcentaje-canastas > 70)
ENTONCES
  (?jugador.categoría = excelente) ) }

{ pivot.tapones-partido::tapones-minimos
( ARGUMENTOS:
  ?jugador
  ?tapones )
( CUERPO:
  SI
    (?tapones > 5)
ENTONCES
  (?jugador.tapones-partido = ?tapones) ) }

{ base.asistencias-partido::asistencias-minimas
( ARGUMENTOS:
  ?jugador
  ?asistencias )
( CUERPO:
  SI
    (?asistencias > 6)
ENTONCES
  (?jugador.asistencias-partido = ?asistencias) ) }

```

Obsérvese que los dos últimos demonios pueden ser perfectamente sustituidos definiendo un par de restricciones en los campos correspondientes.

b) En el caso de que Juan Pérez fuera, además, jugador de balonmano, podría haberse utilizado el mecanismo conocido como "puntos de vista", que básicamente consiste en proporcionar de forma expresa los diferentes valores de un campo según sea la perspectiva bajo la que se estén analizando. Por consiguiente, la modificación sobre la instancia que representa al jugador Juan Pérez puede ser la siguiente:

```
{ INSTANCIA juan-perez
CLASE pivot
ATRIBUTOS:
    equipo = estudiantes;
    entrenador = pedro-perez;
    asistencias-partido
    VALOR:
        (baloncesto = 6)
        (balonmano = 9) } /*6 y 9 valores supuestos*/
```

Por supuesto, para ello habría que modificar la definición de la clase correspondiente:

```
{ CLASE jugador-baloncesto
SUPERCLASE hombre-adulto
ATRIBUTOS:
    categoria {excelente, no-excelente}
    DEMONIOS {NECESIDAD tipo-categoría};
    equipo {caceres, estudiantes, caja-cantabria...};
    ENTERO asistencias-partido
    VALOR:
        baloncesto
        balonmano;
    ENTERO tapones-partido;
    REAL porcentaje-canastas }
```

- **PROBLEMA 6.9:** (adaptado de [GoldWorks III, 1989])

Se va a considerar a continuación un problema que describe y muestra un ejemplo práctico de utilización de la herramienta comercial *GoldWorks*, empleada en el desarrollo de sistemas expertos y que permite la utilización de marcos para la representación de conocimiento.

SOLUCIÓN:

¿Qué pretende este problema? En el presente capítulo se han estudiado diferentes características relacionadas con la representación de conocimiento e inferencia mediante marcos. Este problema y el siguiente pretenden resaltar que dichas características, lejos de quedarse en la categoría de meros conceptos teóricos, han sido incorporadas en las herramientas comerciales actuales para la construcción de sistemas basados en conocimiento.

GoldWorks es una herramienta comercial utilizada en el desarrollo de *sistemas basados en conocimiento*, que permite el empleo de *reglas*, así como de *marcos*. Estos sistemas se denominan *sistemas expertos* cuando el conocimiento refleja la pericia

requerida para actuar como un experto en la resolución de una tarea en un dominio concreto. En estos casos se realiza una labor bastante ardua para conseguir extraer el conocimiento del experto.

La *base de conocimientos* y el *motor de inferencia* forman el núcleo de los sistemas basados en conocimiento que permiten la combinación de reglas y marcos. Más adelante se describirá cómo están implementados estos dos elementos en *GoldWorks*. Conviene recordar que esta formulación, la que combina ambos mecanismos de representación del conocimiento, ha resultado ser la más utilizada por su flexibilidad y adecuación a muchos problemas del mundo real; no obstante, sobre todo en los primeros sistemas basados en conocimiento, se utilizaban sólo reglas o sólo marcos como alternativas hasta cierto punto contrapuestas.

BASE DE CONOCIMIENTOS

La información del experto se puede considerar como conocimiento activo o pasivo. Un hecho que el experto sabe que es verdadero se considera conocimiento pasivo, mientras que un método que un experto usa para hacer deducciones o inferencias se considera conocimiento activo, dado que necesita que se ejecuten una serie de acciones para llevarlo a cabo.

GoldWorks usa las *afirmaciones* para representar el conocimiento sobre hechos o conocimiento pasivo obtenido del experto. Los hechos se declaran mediante aserciones o afirmaciones que son consideradas por el sistema como declaraciones ciertas y probadas en el dominio en cuestión y son almacenadas en la base de conocimientos. Esta herramienta posee métodos para controlar las afirmaciones, ordenarlas y dar un grado de confianza en la verdad de las mismas. *GoldWorks* soporta dos tipos de afirmaciones:

- a) **Afirmaciones no estructuradas:** son las introducidas directamente en la base de conocimientos.
- b) **Afirmaciones estructuradas:** son afirmaciones estructuradas por la red de marcos.

- **Información de dependencia**

La información de dependencia de una afirmación se relaciona con cómo y por qué ésta existe en la *base de afirmaciones*. La información de dependencia tiene dos componentes:

- a) **Una justificación** para las afirmaciones.

Esto es un registro de *por qué* se cree que la afirmación es verdadera o de qué depende la afirmación lógicamente.

b) La derivación de una afirmación.

Representa *cómo* una afirmación fue metida en la base de conocimientos. Es un registro del camino inferencial que llevó a la creación de la misma.

La **red de dependencia** del sistema es la red completa con información de dependencia para cada afirmación en la base de conocimientos. La red de dependencia se usa para controlar los contenidos de la base de conocimientos. En particular, esta red permite a *GoldWorks* determinar cómo el estado de una afirmación depende de otras afirmaciones. Hay dos casos particulares:

a) Una afirmación debe cambiar si determinadas afirmaciones cambian. En este caso, la afirmación es lógicamente dependiente de las otras afirmaciones.

b) Una afirmación no tiene por qué cambiar si otras afirmaciones cambian. En este caso, la afirmación no es lógicamente dependiente de las otras afirmaciones.

Se controla la justificación de una afirmación usando los estados de dependencia de la regla o reglas que llevan a la afirmación. Si la dependencia de una regla se pone a **t** (true), entonces las afirmaciones hechas a partir de la misma son lógicamente dependientes de las afirmaciones y valores que hacían que se cumpliera el antecedente de la regla. Por defecto, la dependencia de las reglas es **nil**.

• **Certeza**

Se puede expresar el grado de confianza en la exactitud de una afirmación asignando un valor de certeza a la misma. Así, a las afirmaciones que son probables se les puede asignar un nivel de certeza más bajo que a una afirmación que es absolutamente cierta. *GoldWorks* usa un sistema de certeza numérico, con valores que oscilan entre 0 y 1. Se puede asignar certeza a afirmaciones y reglas. El sistema utiliza estas certezas para asignarlas a nuevas afirmaciones generadas cuando una regla se ejecuta.

Reglas

Además de una estructura para organizar datos, se necesita una manera de manipularlos para alcanzar objetivos. *GoldWorks* utiliza las reglas para representar el conocimiento activo o inferencial en una aplicación. Las reglas en *GoldWorks* tienen un *antecedente* y un *consecuente*. También tienen *dirección*, que puede ser **hacia adelante, hacia atrás o bidireccional**. La dirección de la regla determina cómo y por qué ésta se *ejecuta*. Se distinguen tres mecanismos de inferencia:

- a) Encadenamiento hacia adelante.
- b) Encadenamiento hacia atrás.
- c) Encadenamiento hacia adelante dirigido por las metas.

Las *reglas hacia adelante* son reglas que se usan en *encadenamiento hacia adelante*. Esto significa que para que una regla se ejecute, todas las condiciones del antecedente deben ser satisfechas, haciendo verdadero el consecuente.

Las *reglas hacia atrás* se usan en *encadenamiento hacia atrás*. El encadenamiento hacia atrás persigue generar valores y afirmaciones que satisfagan un objetivo concreto. En este método sólo se disparan las reglas que sirven para alcanzar el objetivo. Por tanto, se restringe el espacio de búsqueda con lo que mejora la eficiencia del proceso.

El *encadenamiento hacia adelante dirigido por las metas* integra los dos tipos de encadenamiento ya vistos. Este tipo de encadenamiento usa *conjuntos de reglas* (*rule sets*) para agrupar reglas hacia adelante que tienen un mismo objetivo. Cuando se tiene un patrón objetivo que se equipara con el consecuente de un conjunto de reglas, este conjunto se activa y el sistema inicia un encadenamiento hacia adelante con las reglas del mismo. Si una vez procesadas todas las reglas hacia adelante activas no se alcanza el objetivo, el encadenamiento hacia atrás se reanuda.

Las *reglas bidireccionales* pueden ser usadas en encadenamiento hacia adelante o encadenamiento hacia atrás.

Las reglas pueden ser agrupadas en los llamados *conjuntos de reglas* (ya mencionados anteriormente) que pueden ser usados en encadenamiento hacia adelante o hacia atrás. Una posible razón para el empleo de conjuntos de reglas es tener en cada momento activas sólo aquellas reglas que sean necesarias para el funcionamiento de una aplicación, pues normalmente no es necesario considerarlas todas en todo momento. Las reglas hacia adelante individuales o agrupadas en conjuntos de reglas pueden ser asignadas a un determinado *patrocinador* cuya función es controlar cuándo dichas reglas pueden ser o no ejecutadas. Un patrocinador puede ser activado o desactivado mediante un conjunto de reglas que dirijan la estrategia de la tarea que se esté llevando a cabo. Si se desactiva no podrá ejecutarse ninguna de las reglas que gestiona. Se podría dar prioridad a las reglas de un patrocinador de manera que la última regla en ejecutarse permitiera la activación de un nuevo patrocinador. De este modo se puede controlar cuidadosamente la secuencia de eventos en una aplicación. La prioridad para reglas se asigna en *GoldWorks* dentro de una escala entre -1000 y 1000. Las reglas con más alta prioridad se ejecutan antes. El número concreto empleado no importa, excepto en relación a otras asignaciones de prioridad. Por defecto, la prioridad de una regla es

0. Cuando un patrocinador está activo, las reglas hacia adelante que gestiona pasan a ser candidatas para ejecución si su antecedente se cumple. Todas las instancias de reglas que se formen se introducirán en una *agenda* asociada al patrocinador. Esta agenda ordena las instancias según la prioridad que tengan asociada. Para instancias con la misma prioridad se puede definir la agenda para que actúe como pila o cola. La primera instancia que pasa a ejecutarse es siempre la que figura en la parte alta de la agenda. Los patrocinadores están organizados en una jerarquía estricta con un patrocinador superior llamado **top-sponsor**. Cualquier regla que no tenga patrocinador, es colocada por defecto en el **top-sponsor**. Sin embargo, se pueden crear subpatrocinadores debajo del **top-sponsor**. La jerarquía que se forma está diseñada para que si se desactiva un patrocinador, todos sus subpatrocinadores sean desactivados.

Marcos

En *GoldWorks* se pueden definir *clases* para describir tipos generales de objetos. Dentro de cada clase aparecen campos que representan los atributos que la definen. En esta herramienta los campos que forman las clases no pueden almacenar valores, aunque sí pueden tener asociadas otro tipo de facetas: valores por defecto, restricciones... Para introducir valores en el sistema correspondientes a objetos específicos del dominio, se crean las llamadas *instancias*. En *GoldWorks* cada instancia sólo puede tener campos heredados y no campos propios.

Los marcos (clases e instancias) están conectados en una red. La red referida no será más que un conjunto de nodos o marcos conectados en una jerarquía que soporta *herencia de propiedades*. La red de marcos proporciona una poderosa y fácil manera de estructurar la información. Se pueden expresar relaciones entre marcos mediante enlaces padre-hijo, que son los que permiten ubicar a cada marco en la red. Si no se fija una posición específica para un marco dentro de la red, el sistema le asigna como marco padre el marco **top-frame**, que se encuentra en el nivel más alto de la jerarquía y no tiene campos. Por tanto, todo marco en la jerarquía tiene un nodo padre, haciendo de la red de *GoldWorks* una red con raíz. La red *GoldWorks* soporta herencia múltiple, de manera que las clases puedan heredar información de más de una clase padre. De cualquier modo, el comportamiento de las instancias es diferente, ya que sólo se permite a cada una de ellas tener un único marco padre. Mediante la palabra clave **with** se especifica un campo y su valor para una instancia determinada. Cuando se escribe un valor en un campo, el sistema crea una *afirmación campo-valor* que pasa a formar parte de la base de afirmaciones. Otros objetos *GoldWorks*, tales como las reglas, podrán entonces *equiparar* dichos valores con los que aparecen en sus antecedentes para hacer inferencia en una aplicación.

- **Facetas de campo**

Las *facetas* en *GoldWorks* pueden ser usadas para ampliar la definición de un campo de un marco. Sirven para controlar qué valores puede aceptar un campo, para asociar una función Lisp a un campo, para documentar un campo... Algunas de las facetas de campo en *GoldWorks* son:

- **DOCUMENTATION**: guarda una cadena de texto que describe el campo.
- **PRINT-NAME**: asigna un nombre “bonito” al campo.
- **MULTIVALUED**: especifica si el campo puede aceptar más de un valor.
- **DEFAULT-VALUE**: asigna un valor por defecto a un campo.
- **DEFAULT-CERTAINTY**: dicta el nivel de confianza en el valor asignado a un campo. Tiene un rango de valores entre 0 y 1, siendo 1 el valor que se asigna por defecto.
- **WHEN-ACCESSED**: llama a funciones Lisp cuando se accede a un campo. Las funciones se ejecutan en el orden en que aparecen en una lista. Esta faceta difiere de la faceta **WHEN-MODIFIED** en que las funciones especificadas se llaman cada vez que se accede al valor del campo, de modo que el valor del mismo no tiene por qué cambiar para que las funciones sean llamadas. Además, la función debe devolver un valor que se convertirá en el nuevo valor del campo.
- **WHEN-MODIFIED**: asocia una lista de funciones Lisp al campo, de manera que siempre que el valor del campo cambie (sea afirmado, borrado o modificado), el sistema evalúe las funciones Lisp.
- **CONSTRAINTS**: controla el tipo de valores que pueden ser introducidos en el campo. Algunos ejemplos de estos tipos de restricciones son:
 - one-of**: especifica una lista de valores legales que se pueden elegir para el campo.
 - range**: especifica un rango numérico en el que se puede encontrar el valor.
 - child-frame-of**: especifica que el valor de campo debe ser un marco que tenga el marco reseñado como padre.
- **NO-INFERENCE**: si esta faceta tiene valor **t**, el campo no sería considerado durante el proceso de inferencia. Por defecto adopta el valor **nil**.
- **QUERY-FORM**: solicita al usuario un valor durante el encadenamiento hacia atrás cuando no hay otras reglas que sirvan para obtener dicho valor.

- **NO-SAVE**: si esta faceta tiene valor *t*, el campo se graba con el objeto marco, pero el valor del campo no se graba con la instancia. Se puede fijar esta faceta para instancias individuales. Por defecto adopta el valor *nil*.
- **GRAPHIC**: Se usa para asociar objetos gráficos que visualicen los valores de un campo.

Las facetas se definen y editan en los marcos y se heredan en la red. Algunas facetas, tal como **when-modified** y **when-accessed** se pueden establecer también para instancias.

EL MOTOR DE INFERENCIA

El motor de inferencia de *GoldWorks* se encarga de realizar la inferencia sobre los datos. Aplica reglas sobre datos cuando se busca una solución en una aplicación. Hay varias técnicas inferenciales disponibles: encadenamiento hacia adelante, encadenamiento hacia atrás, encadenamiento hacia adelante dirigido por las metas o una combinación de los tres. Además, se puede controlar el proceso de inferencia para que se evalúen unas reglas antes que otras y se puede expresar la confianza en el valor de las reglas y las afirmaciones generadas cuando se ejecutan esas reglas, asignando factores de certeza.

EJEMPLO PRÁCTICO

A continuación se desarrolla un ejemplo que puede ayudar a la comprensión de algunas de las características del sistema descritas hasta el momento.

Se pretende crear un pequeño sistema que ayude a seleccionar el medio de transporte más económico para realizar un envío. Partiendo de la información sobre la carga que se desea transportar, el sistema determina la mejor forma de hacerlo basada en el criterio que se especifique. Las reglas introducidas en este ejemplo (se empleará encadenamiento hacia adelante) usan las clases, instancias y valores de campos de los objetos del dominio para decidir el tipo de transporte más adecuado. El objetivo es seleccionar el modo más efectivo en cuanto a coste para transportar un producto dentro de un periodo de tiempo dado y con ciertas restricciones respecto a la entrega. El sistema basa su salida en información sobre el producto que está siendo transportado y las diversas compañías entre las que escoger. La aplicación debe tener en cuenta la naturaleza del producto para determinar si es necesario un tipo especial de transporte. También debe considerar la cantidad que será transportada y el origen y destino del envío. La decisión final estará basada en qué compañía cumple todos los requiri-

mientos y puede efectuar el envío a menor precio. La información para este ejemplo se puede estructurar usando cuatro marcos:

producto: registra toda la información relacionada con el producto a enviar: su peso, nivel de seguridad requerido... Esta categoría será posteriormente dividida para tener en cuenta los diferentes tipos de productos que pueden ser enviados.

compañía-de-envíos: información relacionada con las compañías disponibles para el envío, tal como los precios que ofrecen, capacidad para cumplir las necesidades especiales del envío, lugares cubiertos y restricciones de tiempo de entrega.

orden-de-envío: todos los criterios especificados por el cliente, tales como el origen, destino, urgencia del envío...

posibles-compañías: usado por el sistema para guardar información sobre la capacidad de una compañía para ser elegida a la hora de realizar la entrega a partir de los criterios especificados.

En la clase producto se definen tres subclases que se corresponden con los tres tipos de productos que se pueden enviar: normal (que no requiere tratamiento especial), perecedero y frágil.

La red de clases del sistema quedaría del siguiente modo:

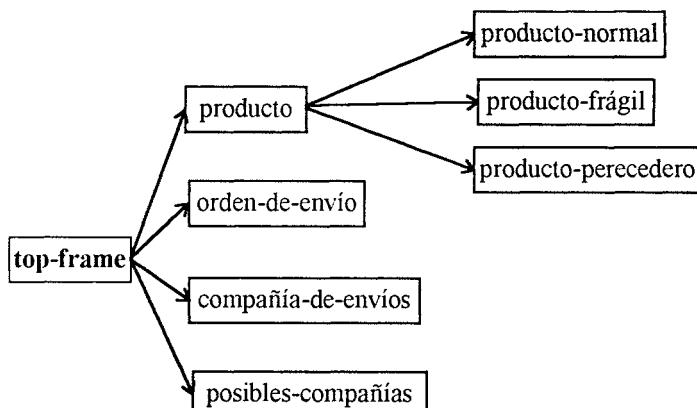


Figura 6-18

Las clases producto-perecedero y producto-frágil tienen cada una campos locales relacionados con el tipo particular de producto que representan. Por ejemplo, la clase producto-frágil tiene un campo “fragilidad” que describe el grado de fragilidad del

producto. La clase producto-normal no tiene campos, pero se usa para mantener la estructura jerárquica del sistema.

La clase producto

Tiene los siguientes campos:

peso (en kilogramos): calculado para una unidad de producto enviado. Ese valor es multiplicado por la cantidad de unidades enviadas para calcular el peso total del envío. Este peso se utiliza para seleccionar un transporte adecuado.

unidades-por-caja: número de unidades de producto enviadas en cada paquete-caja.

embalaje-por-unidad: calculado para una unidad de producto enviado.

peso-por-caja: calculado multiplicando el valor del campo embalaje-por-unidad por el valor del campo unidades-por-caja.

En cuanto a la clase producto-frágil:

fragilidad: determina si el producto requiere manipulación especial. Se necesita definir este campo aún más, usando la faceta “one-of” para limitar la elección del cliente a “alto” o “bajo” cuando se rellene este campo.

Finalmente, la clase producto-perecedero tiene los siguientes campos:

refrigeración-requerida: acepta un valor de “sí” o “no” para indicar si el producto requiere refrigeración. De nuevo se utiliza “one-of”.

sensible-a-la-luz: acepta “sí” o “no”. También usado para medir los requerimientos de manipulación del producto.

La clase compañía-de-envíos

Ahora se necesita estructurar el conocimiento referente al medio a través del cual se envía el producto. El marco compañía-de-envíos estructura toda la información relacionada con las compañías disponibles para el envío. Sus campos son:

nombre: permite introducir el nombre completo de la compañía usando espacios u otros caracteres no aceptados como nombre de objeto.

coste-por-km: su valor es usado para calcular el coste total del envío a partir de la distancia del trayecto y la cantidad del producto transportado.

localizaciones: contiene los nombres de las zonas que la compañía cubre o sirve (los valores deben ser introducidos sin blancos).

restricción-de-peso: representa el peso más alto que una compañía transportará en un envío.

refrigeración: según sea “sí” o “no” indica si la compañía puede enviar un producto que requiere refrigeración.

manipulación-especial: el valor “sí” o “no” determina si la compañía tiene los medios para proporcionar entregas con manipulación especial.

tipo-transporte: admite dos valores posibles: “aire” o “tierra” .

La clase orden-de-envío

Estructura toda la información que el transportista considera para el envío de un producto. Los campos para esta clase y su propósito son:

id-encargo: guarda el código de identificación asignado a la orden de envío.

urgencia: (“alta” o “baja”) la urgencia del envío afectará al medio de envío.

nombre-del-producto: acepta el nombre del producto enviado.

cantidad: acepta el número de unidades del producto enviadas. Este valor es usado para calcular el peso total del envío, que a su vez es usado para calcular el coste total del mismo.

origen: especifica la ciudad desde la que el envío será hecho. Una compañía debe cubrir esta ciudad para ser candidata a realizar el envío.

destino: especifica la ciudad destino.

distancia: entre origen y destino. Se utiliza para calcular el coste del envío.

especial: (“sí” o “no”) el valor por defecto de este campo sería “no”. Este valor indica si el producto que se envía requiere manipulación especial.

peso-del-encargo: almacena el peso total del envío.

modo-de-envío: el sistema introduce un valor en este campo después de que ha determinado el método más económico de envío.

compañía-de-envío: el sistema se encarga de llenar este campo una vez que se selecciona una compañía concreta.

coste-de-envío: después de que el sistema ha seleccionado el mejor tipo de transporte, se introduce el coste total de envío en este campo.

La clase *posibles compañías*

Este marco es usado exclusivamente para guardar valores relacionados con los criterios de elegibilidad que una compañía debe cumplir para poder ser considerada como candidata a realizar el envío. Únicamente el sistema irá creando instancias de esta clase. El valor en el campo *id-encargo* de una instancia de *orden-de-envío* es usado como el nombre para una instancia de *posibles-compañías*. Haciendo lo anterior, se asegura una instancia de *posibles-compañías* para cada instancia de *orden-de-envío*. El sistema introduce valores en los campos de las instancias de acuerdo con la capacidad de las compañías para enviar pedidos. Aquellos valores son entonces usados para determinar la elegibilidad final de las compañías.

Los campos de esta clase son:

cumple-requerimiento-lugar: si una compañía cumple los criterios de cobertura del origen y destino especificados por la instancia que se hace del marco *orden-de-envío*, el nombre de la compañía se introduce en este campo.

cumple-requerimiento-peso: si una compañía cumple el criterio de peso, su nombre es introducido en este campo.

cumple-requerimiento-especial: si una compañía cumple el criterio de manipulación especial, su nombre es introducido en este campo.

compañias-candidatas: acepta los nombres de compañías candidatas a realizar el envío del producto. El sistema determina la menos cara de estas compañías y muestra en pantalla los resultados. Este campo acepta más de un valor.

DEFINICIÓN DE LAS REGLAS DEL EJEMPLO:

Se definirán tres tipos de reglas:

1) Reglas de criterios: usan información del producto y del tipo de encargo hecho para determinar las condiciones que debe cumplir una compañía para poder realizar el envío.

2) Reglas de compañías-candidatas: determina las compañías que pueden realizar el envío, es decir, aquéllas que cumplen los criterios reseñados en el apartado anterior.

3) Reglas de costes: calculan el coste del envío para aquellas compañías candidatas y determina qué compañía es la más barata.

1) Reglas de criterios

Los criterios usados son los de peso, distancia, localización y manipulación especial.

a) Reglas de pesos.

Estas reglas calculan el peso total que debe enviarse y determinan si una compañía está capacitada para transportar ese peso.

Regla PESO1:

“Encontrar el nombre del producto y la cantidad que está siendo transportada a partir de la orden de envío; encontrar también el peso del producto. Entonces, multiplicar la cantidad transportada por el peso del producto y registrar el peso total hallado en el campo peso-del-encargo de la instancia de la clase orden-de-envío.”

Esta regla podría ser introducida en *GoldWorks* del siguiente modo:

```
IF
  INSTANCE ?orden IS orden-de-envío
    WITH nombre-del-producto ?nombre
    WITH cantidad ?cantidad
  INSTANCE ?nombre IS producto
    WITH peso ?peso
THEN
  INSTANCE ?orden IS orden-de-envío
    WITH peso-del-encargo (EVALUATE (* ?peso ?cantidad))
```

En esta regla existe un AND implícito entre las dos cláusulas del antecedente.

Regla PESO2:

“Si el peso de un envío (calculado por la regla anterior) es menor que la restricción en cuanto a peso de una compañía, entonces esa compañía pasa a ser candidata para realizar el envío.”

```
IF
  INSTANCE ?orden IS orden-de-envío
    WITH peso-del-encargo ?peso-del-encargo
    WITH id-encargo ?id-encargo
  INSTANCE ?compañía IS compañía-de-envíos
    WITH restricción-de-peso ?rp
  RELATION <= ?peso-del-encargo ?rp
```

THEN

INSTANCE ?id-encargo IS posibles-compañías
 WITH cumple-requerimiento-peso ?compañía

b) Reglas de distancias.

Se creará una regla para envíos a una distancia de menos de 500 kms. y otra para el resto.

Regla TRANSPORTE-TERRESTRE:

“Si un envío no es urgente o el envío se hace a menos de 500 kms., entonces realizar el envío por tierra.”

IF

OR

INSTANCE ?orden IS orden-de-envío (1)
 WITH urgencia baja
 AND (2)
 INSTANCE ?orden IS orden-de-envío (3)
 WITH urgencia alta
 WITH distancia ?distancia
 RELATION < ?distancia 500 (4)

THEN

INSTANCE ?orden IS ?orden-de-envío
 WITH modo-de-envío tierra

En esta regla hay un OR que afecta a las cláusulas (1) y (2) y un AND que lo hace sobre las cláusulas (3) y (4).

Regla TRANSPORTE-AÉREO:

“Si un envío es urgente y se hace a más de 500 kms. de distancia, entonces realizar el envío por aire.”

IF

INSTANCE ?orden IS orden-de-envío
 WITH urgencia alta
 WITH distancia ?distancia
 RELATION >= ?distancia 500

THEN

INSTANCE ?orden IS orden-de-envío
 WITH modo-de-envío aire

c) Reglas de localizaciones.

Determina si una compañía cubre el origen y destino del envío.

“Si una orden de envío especifica un origen, destino, modo de envío y código de encargo para un envío, y si una compañía cubre el origen y el destino, entonces encontrar o crear una instancia usando el código de encargo como nombre de instancia y almacenar en esta instancia la información de que la compañía cumple los requerimientos de localización.”

IF

INSTANCE ?orden IS orden-de-envío
 WITH origen ?origen
 WITH destino ?destino
 WITH modo-de-envío ?modo
 WITH id-encargo ?id-encargo
 INSTANCE ?compañía IS compañía-de-envíos
 WITH localizaciones ?origen
 WITH localizaciones ?destino
 WITH tipo-transporte ?modo

THEN

INSTANCE ?id-encargo IS posibles-compañías
 WITH cumple-requerimiento-lugar ?compañía

d) Reglas de manipulación especial.

Regla MANIPULACIÓN-ESPECIAL1:

“Si un envío requiere una temperatura de menos de 5 grados centígrados y una compañía cumple los requerimientos de refrigeración, entonces encontrar o crear una instancia de posibles-compañías que se corresponda con el código de encargo y afirmar que la compañía cumple el requerimiento de manipulación especial en cuanto a refrigeración. El sistema también afirma el valor sí en el campo especial de la instancia de orden-de-envío.”

IF

INSTANCE ?orden IS orden-de-envío
 WITH nombre-del-producto ?nombre
 WITH id-encargo ?id-encargo

INSTANCE ?nombre IS producto-perecedero
WITH refrigeración-requerida sí
INSTANCE ?compañía IS compañía-de-envíos
WITH refrigeración sí
THEN
INSTANCE ?id-encargo IS posibles-compañías
WITH cumple-requerimiento-especial ?compañía
INSTANCE ?orden IS orden-de-envío
WITH especial sí

Regla MANIPULACIÓN-ESPECIAL2:

“Si un envío es frágil y una compañía cumple los requerimientos de manipulación especial, entonces encontrar o crear una instancia de posibles-compañías que se corresponde con el código de encargo y afirmar que la compañía reúne el requerimiento de manipulación especial en cuanto a fragilidad. El sistema también afirma el valor sí en el campo especial de la instancia de orden-de-envío.”

IF
INSTANCE ?orden IS ?orden-de-envío
WITH id-encargo ?id-encargo
WITH nombre-del-producto ?nombre
INSTANCE ?nombre IS producto-frágil
WITH fragilidad alta
INSTANCE ?compañía IS compañía-de-envíos
WITH manipulación-especial sí
THEN
INSTANCE ?id-encargo IS posibles-compañías
WITH cumple-requerimiento-especial ?compañía
INSTANCE ?orden IS orden-de-envío
WITH especial sí

2) Reglas de compañías candidatas

Habrá dos reglas de este tipo. La primera es para envíos que requieren manipulación especial. La segunda es para el caso en que este requerimiento no sea necesario.

Regla COMPAÑÍA-ELEGIBLE1:

“Si una compañía reúne los requerimientos de manipulación especial, peso, lugar de recogida y entrega de un envío, entonces la compañía pasa a ser candidata para realizar el encargo o envío.”

IF

```

INSTANCE ?orden IS orden-de-envío
    WITH modo-de-envío ?modo
    WITH id-encargo ?id-encargo
    WITH especial sí
INSTANCE ?compañía IS compañía-de-envíos
    WITH tipo-transporte ?modo
INSTANCE ?id-encargo IS posibles-compañías
    WITH cumple-requerimiento-lugar ?compañía
    WITH cumple-requerimiento-peso ?compañía
    WITH cumple-requerimiento-especial ?compañía

```

THEN

```

INSTANCE ?id-encargo IS posibles-compañías
    WITH compañías-candidatas ?compañía

```

Regla COMPAÑÍA-ELEGIBLE2:

“Si una compañía cumple los requerimientos en cuanto al peso y lugares de recogida y entrega del envío, entonces la compañía pasa a ser candidata para realizar el envío.”

IF

```

INSTANCE ?orden IS orden-de-envío
    WITH modo-de-envío ?modo
    WITH id-encargo ?id-encargo
    WITH especial no
INSTANCE ?compañía IS compañía-de-envíos
    WITH tipo-transporte ?modo
INSTANCE ?id-encargo IS posibles-compañías
    WITH cumple-requerimiento-lugar ?compañía
    WITH cumple-requerimiento-peso ?compañía

```

THEN

```

INSTANCE ?id-encargo IS posibles-compañías
    WITH compañías-candidatas ?compañía

```

3) Reglas de costes

Estas reglas calculan el menor precio para un envío, muestran en pantalla las compañías candidatas al envío y el precio, recomiendan la compañía más barata y afirman el nombre de la compañía preferida en el campo compañía-de-envíos de la instancia de orden-de-envío.

Regla PRECIO-TOTAL:

“Si un encargo especifica la distancia, el peso y el precio por kilómetro para cada compañía candidata, entonces se muestra en pantalla el nombre de cada compañía y el precio total del envío.”

```

IF
  INSTANCE ?orden IS orden-de-envío
    WITH id-encargo ?id-encargo
    WITH distancia ?distancia
    WITH peso-del-encargo ?peso-del-encargo
  INSTANCE ?compañía IS compañía-de-envíos
    WITH coste-por-km ?coste
  INSTANCE ?id-encargo IS posibles-compañías
    WITH compañías-candidatas ?compañía
  BIND ?calc
    (TRUNCATE
      (+ (* ?distancia ?coste) (* .01 ?peso-del-encargo)))
THEN
  RELATION PRINT-OUT
    :RETURN “Vd. puede efectuar el envío por la compañía ”
      ?compañía “ a un precio de” ?calc “ pts.”
  UNSTRUCTURED-ASSERTION coste ?compañía ?calc

```

Las dos reglas siguientes calculan el precio mínimo de envío. Estas reglas pueden ser usadas como un algoritmo general para el cálculo de mínimos.

Regla CALC-COST1:

```

IF
  UNSTRUCTURED-ASSERTION coste ?compañía1 ?coste1
  UNSTRUCTURED-ASSERTION coste ?compañía2 ?coste2
  RELATION < ?coste1 ?coste2
THEN
  RETRACT coste ?compañía2 ?coste2

```

Regla CALC-COST2:

```

IF
  INSTANCE ?orden IS orden-de-envío
  INSTANCE ?id-encargo IS posibles-compañías
  UNSTRUCTURED-ASSERTION coste ?compañía ?coste
THEN
  RELATION PRINT-OUT
    :RETURN "RECOMENDACIÓN: compañía " ?compañía
           " a un precio de " ?coste " pts."
  INSTANCE ?orden IS ?orden-de-envío
    WITH coste-de-envío ?coste
    WITH compañía-de-envío ?compañía
  RETRACT INSTANCE ?id-encargo IS posibles-compañías

```

Evidentemente, la segunda regla deberá ejecutarse siempre después que la primera con el fin de que se eliminen las compañías más caras. Para ello, se le puede asignar a calc-cost2 una prioridad de -10, por ejemplo.

CASO PRÁCTICO

Supóngase que se tuvieran las siguientes instancias en el sistema (Obsérvese que muchos valores numéricos introducidos no guardan relación con la realidad al ser el objetivo del presente ejercicio puramente didáctico.).

<u>producto-perecedero</u>	<u>almejas-vivas</u>
peso	2
unidades-por-caja	
peso-por-caja	
embalaje-por-unidad	
refrigeración-requerida	sí
sensible-a-la-luz	sí

<u>orden-de-envío</u>	<u>orden-de-almejas</u>
coste-de-envío	
destino	Cádiz
distancia	1500
id-encargo	L1
peso-del-encargo	
origen	Barcelona
urgencia	alta
nombre-del-producto	almejas-vivas
cantidad	50
modo-de-envío	
compañía-de-envío	
especial	

<u>compañía-de-envíos</u>	<u>trainsa</u>
coste-por-km	0.05
localizaciones	Barcelona Zamora Cuenca Cádiz
nombre	“Transportes Internacionales S.A.”
refrigeración	sí
manipulación-especial	sí
tipo-transporte	tierra
restricción-de-peso	10000

<u>compañía-de-envíos</u>	<u>eninsa</u>
coste-por-km	0.06
localizaciones	Barcelona Zamora Palencia Cádiz
nombre	“Envíos Internacionales S.A.”
refrigeración	no
manipulación-especial	sí
tipo-transporte	tierra
restricción-de-peso	10000

<u>compañía-de-envíos</u>	<u>enur</u>
coste-por-km	0.09
localizaciones	Barcelona Zamora Madrid Cádiz
nombre	“Envíos urgentes”
refrigeración	sí
manipulación-especial	sí
tipo-transporte	aire
restricción-de-peso	1000

compañía-de-envíos	tranur
coste-por-km	0.1
localizaciones	Barcelona Zamora Segovia Cádiz
nombre	“Transporte urgente”
refrigeración	sí
manipulación-especial	sí
tipo-transporte	aire
restricción-de-peso	1000

Se tendría, por tanto, la siguiente red de marcos:

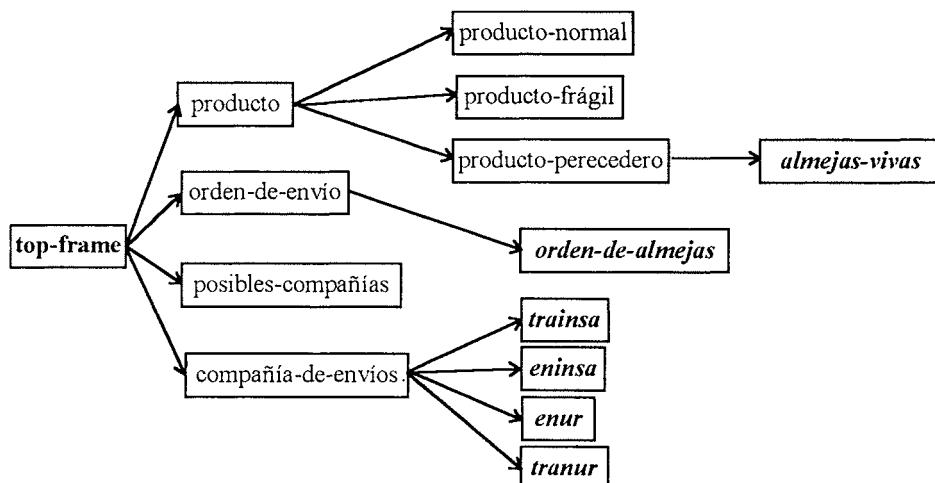


Figura 6-19

Finalmente, la salida que daría *GoldWorks* sería:

Vd. puede efectuar el envío por la compañía TRANUR a un precio de 151 pesetas.

Vd. puede efectuar el envío por la compañía ENUR a un precio de 136 pesetas.

RECOMENDACIÓN: compañía ENUR a un precio de 136 pesetas.

- **PROBLEMA 6.10:** (adaptado de [Rules Element 4.0, 1996])

El siguiente problema examina algunas de las características del entorno para el desarrollo de sistemas expertos *Nexpert Object* (también conocido como *Rules Element*) y las compara con las que posee *GoldWorks*.

SOLUCIÓN:

Campos y valores

En *Nexpert Object*, al contrario que en *GoldWorks*, los campos de las clases tienen asociada la faceta valor. Además, en cada instancia pueden definirse campos propios que no provengan de la herencia de propiedades en la jerarquía.

Subobjetos

En cuanto a las capacidades para la representación del conocimiento, en *Nexpert Object* se puede hacer uso de una relación del tipo “es parte de” entre dos objetos específicos o instancias. En este caso, un *subobjeto* o subinstancia, que es también considerada como una instancia, representaría una parte de otro objeto o instancia. Dos instancias unidas por este enlace, usualmente no comparten muchas características o campos. Finalmente, una subinstancia, al ser también una instancia, puede pertenecer también a una clase.

En *GoldWorks* no existe el concepto de subinstancia o subobjeto.

Herencia ascendente

Nexpert Object permite los dos tipos de herencia: descendente y ascendente. De hecho, existe un metacampo o faceta llamado *heredabilidad* que determina si un campo puede ser heredado o no y en qué dirección (ascendente o descendente). En la siguiente figura aparece cuál es el comportamiento por defecto para campos de instancias:

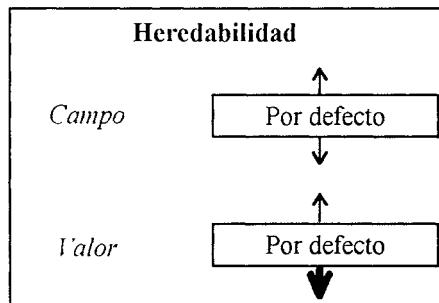


Figura 6-20 Capacidad de herencia desde instancias.

Es decir, por defecto sólo se permite que una subinstancia de la instancia en consideración pueda heredar el valor del campo. El comportamiento por defecto para clases es el siguiente:

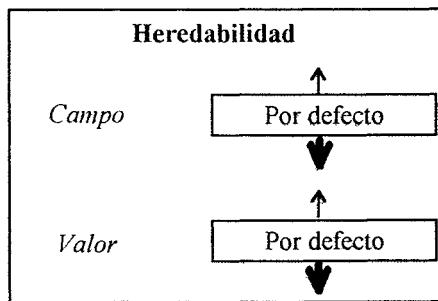


Figura 6-21 Capacidad de herencia desde clases.

de manera que, en principio, se permite sólo herencia descendente hacia subclases e instancias, de valores y campos.

Los valores por defecto de heredabilidad pueden modificarse de manera que se permita la herencia ascendente, que no se contempla en *GoldWorks*.

Resolución de conflictos en herencia múltiple

En *GoldWorks* cada instancia sólo puede tener una clase padre. Como consecuencia de lo anterior, los únicos conflictos que se pueden dar se limitan a la herencia entre clases de facetas de campos con el mismo nombre. *GoldWorks* resuelve estos conflictos de dos formas posibles, dependiendo del tipo de faceta que se esté considerando: o se da un simple mensaje de error y no se permite la herencia de las facetas, o se selecciona únicamente la faceta del primer parente para ser heredada (los padres de un marco figuran en una lista).

En *Nexpert Object* se pueden seguir diferentes estrategias que realizan una búsqueda del espacio de herencia, bien en amplitud o bien en profundidad, para deshacer el conflicto y decantarse por una de entre todas las fuentes de información disponibles. Considerese, por ejemplo, la jerarquía de la figura 6-22.

Si el sistema necesitara determinar el valor del campo *objeto.prop*, los resultados que producirían las diferentes estrategias de búsqueda son los siguientes:

1) Clase primero, amplitud primero:

Clase_1, clase_2, objeto_1, objeto_2, clase_3, clase_4, clase_5, objeto_3, objeto_4, clase_6, clase_7, objeto_5, objeto_6.

2) Clase primero, profundidad primero:

Clase_1, clase_3, clase_6, clase_7, clase_4, clase_2, clase_5, objeto_1, objeto_3, objeto_5, objeto_6, objeto_2, objeto_4.

3) Objeto primero, amplitud primero:

Objeto_1, objeto_2, clase_1, clase_2, objeto_3, objeto_4, clase_3, clase_4, clase_5, objeto_5, objeto_6, clase_6, clase_7.

4) Objeto primero, profundidad primero:

Objeto_1, objeto_3, objeto_5, objeto_6, objeto_2, objeto_4, clase_1, clase_3, clase_6, clase_7, clase_4, clase_2, clase_5.

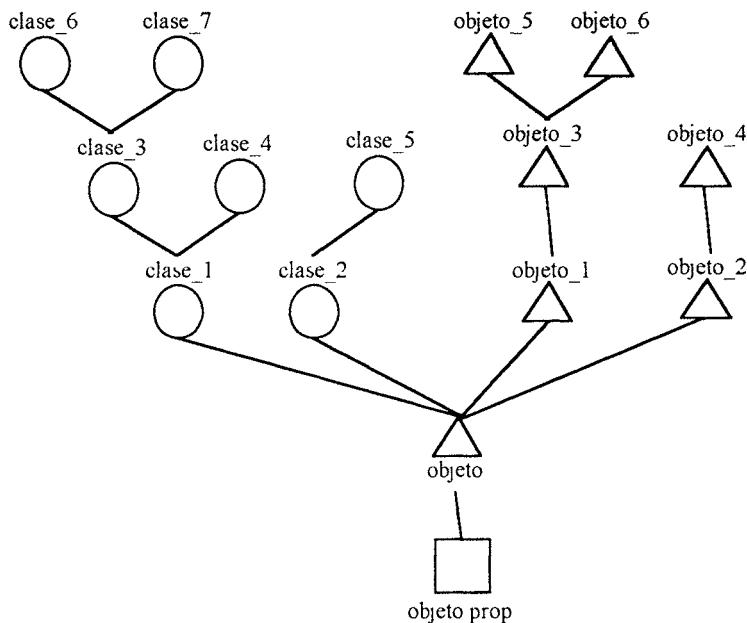


Figura 6-22

Si la búsqueda es en amplitud, se intenta encontrar todas las posibles fuentes de información dentro de cada nivel. Si en un mismo nivel aparece más de una fuente, el “empate” se puede deshacer teniendo en cuenta las facetas *prioridad de herencia* de cada parente o hijo. Por defecto, todos los campos tienen una prioridad de herencia de 1. El valor de esta prioridad puede oscilar entre -32000 y 32000.

Campos privados

Mientras que en *GoldWorks* todos los campos son públicos, los campos que se crean en *Nexpert Object* pueden ser públicos o privados. Si un campo es privado, se restringe el acceso al mismo desde diferentes entidades del sistema ajenas a aquélla a la que pertenece el campo.

Tratamiento de la incertidumbre

GoldWorks aborda la cuestión de la incertidumbre asociada a muchos hechos o reglas, aunque de una manera algo rudimentaria en lo que se refiere a la propagación de estas incertidumbres cuando nuevos hechos son inferidos.

Por otra parte, en *Nexpert Object* no existe incorporado un mecanismo de tratamiento de la incertidumbre.

Reglas bidireccionales

Por último, mencionar que, mientras que en *GoldWorks* se diferencia entre reglas hacia adelante, reglas hacia atrás y reglas bidireccionales, en *Nexpert Object* todas las reglas son bidireccionales.

• PROBLEMA 6.11:

Diseñar un sistema simple y general de diagnóstico médico utilizando el formalismo de *marcos* ampliado con la capacidad de empleo de *métodos* asociados a *objetos* de la jerarquía, que se activan mediante el paso de mensajes.

SOLUCIÓN:

En primer lugar, es necesario introducir una clase *PACIENTE* que represente al enfermo. La interacción entre el sistema de diagnóstico médico y el propio enfermo se lleva a cabo a través del paso de mensajes a éste último. Cada posible mensaje que el enfermo pueda recibir tendrá asociado un método o procedimiento almacenado localmente en la clase *PACIENTE*. Por ejemplo, es conveniente poder disponer de un mensaje con el que solicitar cuáles son los síntomas que presenta el paciente (ver figura 6-23).

Cada uno de los síntomas que puede presentar cualquier enfermo se representa como una clase (ver figura 6-24). Como ejemplo puede citarse la clase *Fiebre* que dispondrá de un campo “temperatura” y otro “duración” que den idea de la temperatura corporal del enfermo y cuál es el periodo desde el que persiste la misma. La temperatura de cualquier paciente tendrá un valor por defecto de 36,5 °C y su duración otro de 1 día.

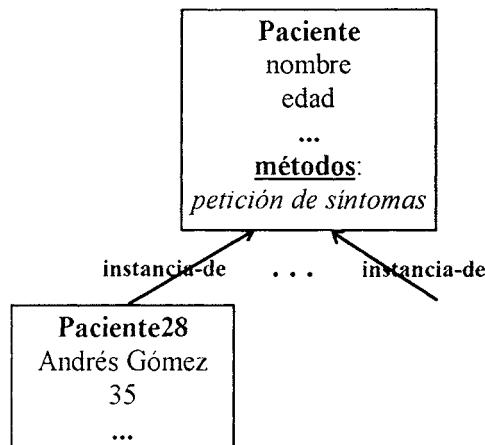


Figura 6-23

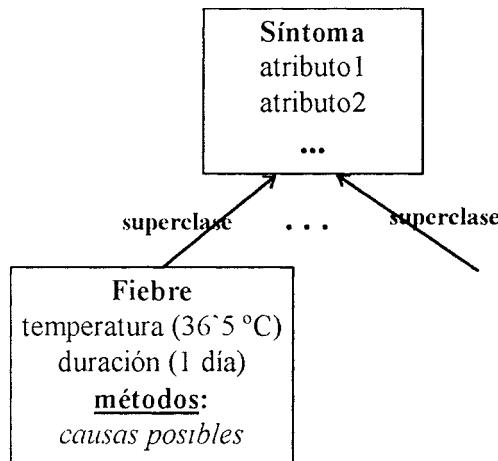


Figura 6-24

Una estrategia para el establecimiento de un diagnóstico podría ser, dado el conjunto de síntomas del paciente, determinar las posibles causas o enfermedades que han dado origen a los mismos (éste es el fin del método *causas posibles* del síntoma **Fiebre**) e investigar cuál es la causa más probable.

El método *causas posibles* asociado a **Fiebre** podría tener la forma siguiente:

Si la temperatura del paciente es mayor de 38 °C y su duración es mayor de 3 días, la causa de la fiebre es una infección grave; si no, la causa es una gripe.

Es necesario que el sistema disponga de información sobre las distintas enfermedades que pueden ser diagnosticadas. Para cada enfermedad se tiene una lista con sus síntomas más probables (ver figura 6-25).

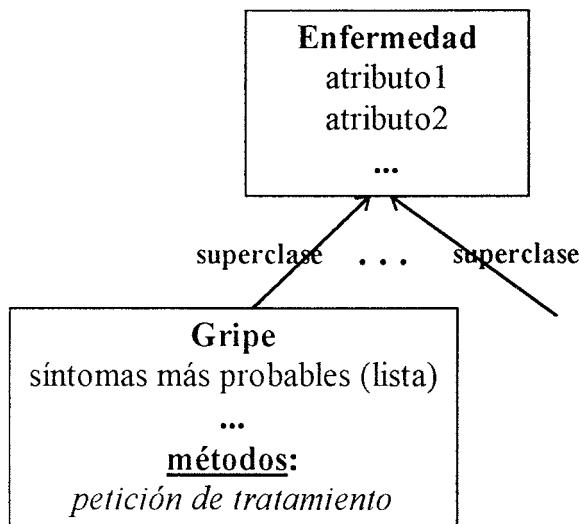


Figura 6-25

La estructura de control del proceso global de diagnóstico consistiría en:

1. Pedir los síntomas del paciente.
2. Escoger un síntoma y solicitar su causa más probable.
3. Pedir cuáles son los síntomas más probables de la enfermedad hallada en 2. y comprobar si el paciente sufre tales síntomas. En caso afirmativo, pedir el tratamiento para dicha enfermedad.

Los tratamientos a aplicar al enfermo constituyen nuevos objetos del dominio que deben ser representados (ver figura 6-26).

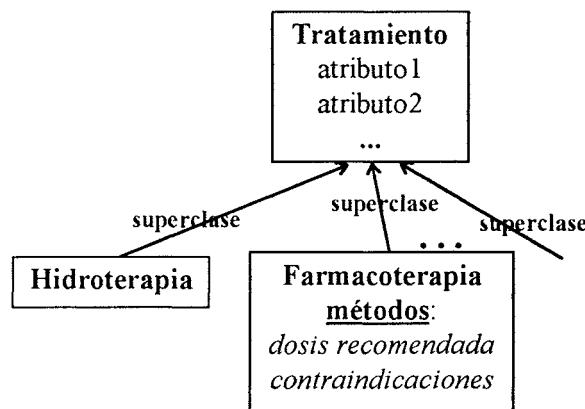


Figura 6-26

Finalmente, para completar la información que debe poseer el sistema, se organizan todos los datos disponibles sobre los diferentes medicamentos en una nueva jerarquía, una parte de la cual podría ser:

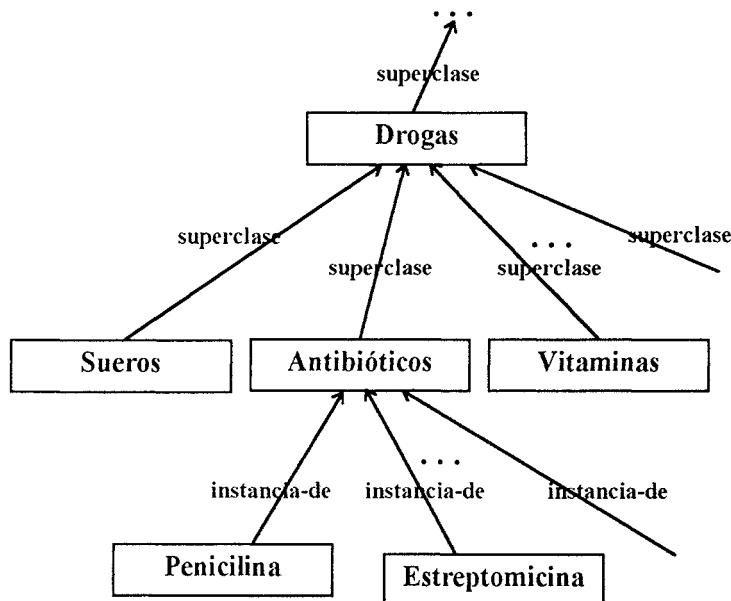


Figura 6-27

• PROBLEMA 6.12:

Diseñar un guión que dé cuenta de las acciones que tienen lugar cuando se hace uso de los servicios ofrecidos por un restaurante.

SOLUCIÓN:

El guión \$RESTAURANTE va a tener los siguientes *roles* o, dicho de otra forma, van a intervenir los siguientes personajes:

C: cliente

M: camarero

N: cocinero

P: propietario

Para cada situación real que se pueda estar considerando, las variables *C*, *M*, *N* y *P* adoptarán unos valores concretos determinados.

Como *objetos* típicos del guión \$RESTAURANTE se tienen:

mesas

menú

D: comida

cuenta

dinero

La comida se representa mediante una variable, ya que no se sabe a priori qué tipo de alimentos va a tomar el cliente.

Como *condiciones* previas para una posible activación de este guión, se pueden citar:

C tiene hambre.

C tiene dinero.

Los *resultados* surgidos una vez completada la última escena podrían ser:

C tiene menos dinero.

P tiene más dinero.

C no tiene hambre.

Finalmente, el guión \$RESTAURANTE estaría formado por las siguientes *escenas*:

Escena 1: Entrada en el restaurante

C entra en el restaurante.

C busca una mesa donde sentarse.

C se sienta.

Escena 2: Petición de la comida

C lee el menú.

C decide qué comida tomar.

C comunica a *M* su elección.

M comunica a *N* la elección de *C*.

N prepara *D*.

Escena 3: El cliente come

N entrega *D* a *M*.

M lleva *D* a *C*.

C toma *D*.

Una vez finalizada la escena 3, el cliente podría pedir más comida si así lo desea. En tal situación se volvería a la escena 2. Si éste no fuera el caso, se finalizaría con la siguiente escena:

Escena 4: Salida del restaurante

M escribe la cuenta.

M entrega la cuenta a *C*.

C paga.

C se va del restaurante.

Cada una de las acciones especificadas en las escenas de un guión se representan internamente mediante grafos de dependencia conceptual de Schank. Fueron en realidad Schank y sus colaboradores quienes crearon los guiones como continuación de su trabajo sobre representación de dependencia conceptual. A modo de ejemplo, la acción en la que el cliente toma la comida que le ha sido servida se representaría de la siguiente forma:

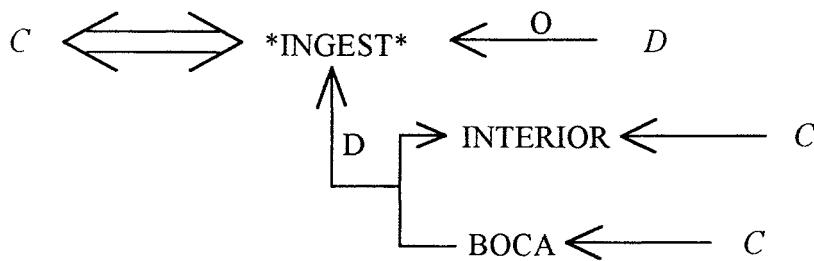


Figura 6-28

• PROBLEMA 6.13:

Establecer un guión que refleje los eventos producidos durante el acto de entrada a un cine.

SOLUCIÓN:

El guión \$ENTRADA_A_UN_CINE constaría de los siguientes *roles* o *personajes*:

P: aquella persona o grupo de personas que van a ver la película

T: persona que está en las taquillas

E: persona que controla la entrada al cine

A: acomodador

Cada situación real particular dará lugar a que las variables anteriores acepten unos valores concretos diferentes.

Los *objetos* típicos de este guión son:

dinero

entradas

butacas

pantalla de cine

Como *condiciones* previas para una posible activación del guión, se tendría:

P tiene dinero

P desea ver una película.

Los *resultados* que se pueden esperar una vez completada la última escena serían:

P tiene menos dinero.

T tiene más dinero.

P ha sacado una determinada impresión de la película.

Finalmente, el guión \$ENTRADA_A_UN_CINE estaría formado por las siguientes *escenas*:

Escena 1: Compra de la entrada (o entradas)

P hace cola frente a las taquillas del cine.

P compra las entradas a *T*.

P se dirige hacia la entrada del cine.

Escena 2: Entrada al cine

P entrega su entrada a *E*.

E marca la entrada.

E indica a *P* la entrada a la sala de proyección.

Escena 3: Tomar asiento

P entrega su entrada a *A*.

A indica a *P* qué asiento debe tomar.

P se sienta

Escena 4: Proyección de la película

P ve la película.

P sale del cine.

Como ya ha quedado reseñado en un problema previo, cada una de las acciones especificadas en las escenas de un guión se representan internamente mediante grafos de dependencia conceptual de Schank. A modo de ejemplo, la acción en la que *P* compra la entrada se representaría de la siguiente forma:

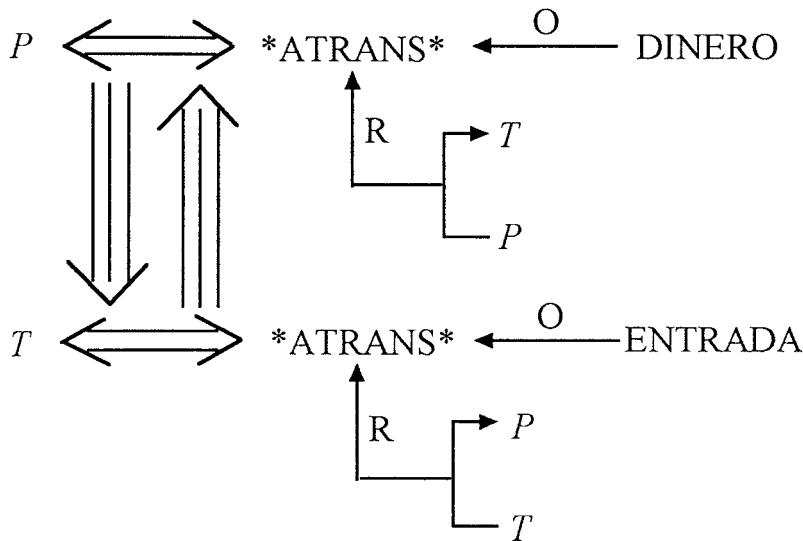


Figura 6-29

6.4 Contexto adicional

6.4.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

- Hoy en día las redes jerárquicas de marcos dotadas de un mecanismo de herencia de propiedades constituyen un método de estructuración y modelado de conocimiento cada vez más usual.
- Se suelen añadir a la red anterior procedimientos que se encargan de mantener la consistencia del sistema (demonios).
- La unión de marcos con otros métodos ya estudiados de representación de conocimiento, principalmente reglas, forma la estructura base de las herramientas comerciales actuales para la construcción de sistemas basados en conocimiento.
- Los guiones son adecuados en la representación de situaciones dinámicas estereotipadas

6.4.2 Pistas de autoevaluación

- Se propone al lector la construcción e implementación de un sistema de marcos a partir del conocimiento de un dominio que se preste a una representación de tipo jerárquico (ver apartado de software de apoyo). Añadir a tal representación nuevo conocimiento del dominio en forma de reglas o lógica. Realizar inferencia en el sistema completo.
- Diseñar el guión correspondiente a la realización de una llamada telefónica desde una cabina.

7 PERSPECTIVA INTEGRADA

7.1 Contexto previo

7.1.1 ¿Por qué está aquí este capítulo?

Hasta ahora se han presentado en este texto de forma aislada una serie de mecanismos alternativos de inferencia y/o representación para abordar la solución de problemas diferentes. Es más, se ha hecho hincapié en distinguir la conveniencia o no de aplicar distintos métodos para cada tipo de problema. No obstante, las situaciones reales distan mucho de poder ser afrontadas bajo una sola perspectiva, siendo necesario generalmente combinar distintas técnicas. Por consiguiente, este capítulo surge de una forma natural para satisfacer un doble objetivo: por un lado, pretende ilustrar la solución de problemas reales en dominios concretos introduciendo problemas de una cierta entidad que son algo más que meros ejercicios, y, por otra parte, intenta proporcionar una perspectiva integradora de las distintas técnicas que forman los contenidos de la materia.

En concreto, este capítulo amplía los contenidos del tema sexto del presente texto. Aprovechamos la concepción genérica de marco, en lo referido a su capacidad expresiva (de representación: red de entidades con relaciones entre ellas) y operativa (de inferencia: herencia de propiedades y valores a lo largo de la red), para plantear bajo un único formalismo de representación, suficientemente flexible y homogéneo, la aplicación combinada de los distintos métodos estudiados. El formalismo elegido está asociado a una opción de implementación específica.

Conviene insistir en que los ejercicios del presente tema no son enunciados concretos y perfectamente definidos, sino **descripciones relativamente amplias que pretenden ilustrar el conocimiento del dominio**. Este planteamiento tiene por objeto destacar la complejidad asociada a la tarea de representación que caracteriza las situaciones reales.

Como final de esta perspectiva integrada se introduce, como un método más disponible, la resolución de problemas mediante técnicas de aprendizaje. Analizaremos exclusivamente la utilización de estas técnicas y su integración en el forma-

lismo adoptado, no así su funcionamiento, que queda fuera de los objetivos del presente temario.

7.1.2 ¿Dónde hay conocimiento teórico del campo?

Bibliografía Básica

- **Teoría:**

[Mira *et al.*, 1995] En especial el capítulo 8 dedicado a los marcos.

- **Ejercicios:**

Problemas resueltos en el capítulo presente.

Bibliografía Complementaria

- **Textos generales:**

[Russell & Norvig, 1995] Este libro proporciona una visión global de la IA desde la perspectiva de los agentes.

[Rich & Knight, 1991] [Ginsberg, 1993]

- **Textos específicos:**

[Boticario, 1994] En el último capítulo de esta tesis se describe el agente aprendiz, haciendo especial hincapié en los procesos de aprendizaje.

[Dent *et al.*, 1992] Artículo de la AAAI-92 en el que se describe un agente aprendiz.

[Fah-Chun, 1996] Un libro dedicado por entero a analizar los agentes en el contexto de la red Internet.

[Hayes-Roth, 1993] Revisión de *la teoría unificada de la cognición* de Newell desde la perspectiva de la construcción de agentes.

[Newell, 1990] Descripción de *la teoría unificada de la cognición*.

[VanLehn, 1991] En VanLehn, K. (ed.) se describen las tres arquitecturas: THEO, PRODIGY y SOAR. (véase [Mitchell *et al.*, 1991])

7.1.3 ¿Qué conocimientos previos se suponen necesarios para comprender este capítulo?

Debido al carácter integrador del presente capítulo, con el fin de facilitar su comprensión, además de recomendar la natural lectura de los capítulos precedentes, nos gustaría insistir en los fundamentos en los que se apoyan las soluciones de los problemas abordados.

1. Importancia del **conocimiento de control** en la resolución eficiente de problemas. Los fundamentos en los que se apoya este concepto se reflejan especialmente en las siguientes secciones del texto base recomendado [Mira et al., 1995]: 3.2.2, 3.2.3, 4.1.1, 5.4.3, 6.3.2, 6.4, 6.7.2, 7.3.3, 7.3.4, 7.4.3, 8.2 y 8.4.2. Por otro lado, cualquiera de los ejercicios de los capítulos precedentes es el resultado de la aplicación de una estrategia de control concreta. En especial, se recomiendan los ejercicios 2.2, 2.6, 2.7 y 2.10, dado que en ellos se puede apreciar la influencia del conocimiento del dominio en la mejora de la solución obtenida. El control del razonamiento en reglas se trata en los ejercicios 4.1 a 4.10. Destaca el algoritmo RETE (Apéndice 4.A) como muestra de la realización efectiva de una estrategia de control.
2. La **capacidad de expresión** de los distintos formalismos, con una mayor atención a los aspectos básicos de los siguientes: lógica de proposiciones y de predicados, estructura de las reglas, jerarquías de conceptos, concepto de marco y lenguajes para su representación. El desarrollo teórico de estos temas aparece respectivamente en las siguientes secciones del mencionado texto base: 5.1, 5.2, 5.3, 6.2, 7.3.2 y 8.1. Los ejercicios de los temas 3, 4, 5 y 6 son todos ellos muestras de las capacidades expresivas de la lógica, reglas, redes y marcos respectivamente. No obstante, resaltamos los problemas 3.1, 3.25 y los comprendidos entre el 6.1 y el 6.13.
3. Análisis del **contraste y/o combinación de los distintos formalismos**. Las secciones 6.7, 7.4.3, 7.5 y 8.4 del texto base son las que hacen un mayor hincapié en este aspecto. En los ejercicios 3.3, 3.14, 3.16, 5.6, 5.7, 5.8, 6.3 y 6.4 se ilustran diferentes aplicaciones relacionadas. Por otro lado, dado que en algunos problemas del presente capítulo se van a utilizar negaciones en los antecedentes de reglas de estilo prolog, recomendamos la revisión de los ejercicios 3.23 y 3.24, en los que se aclara el tratamiento de la negación en Prolog frente al utilizado en la lógica de primer orden.

7.1.4 ¿Qué software de apoyo está disponible?

Destacamos el siguiente software de libre distribución:

1. Arquitecturas Integradas

THEO está disponible en

<http://www.cs.cmu.edu/afs/cs.cmu.edu/user/stork/mosaic/theo-dist.html>

PRODIGY está disponible en

<http://www.cs.cmu.edu/~prodigy/>

SOAR está disponible en

<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/soar/public/www/home-page.html>

2. Herramientas de desarrollo genéricas

Se utilizaran 4 programas alternativos dependiendo del tipo de problemas:

- a) Una versión de PROLOG (Prolog-2) estándar, compatible con el Prolog de Edimburgo, disponible en la Open University del Reino Unido (puede obtenerse en <http://www.dia.uned.es/~jgb/>), que puede utilizarse para realizar encadenamiento hacia atrás con reglas exclusivamente.
- b) MIKE: Una herramienta para construir sistemas expertos diseñada para fines docentes por la Open University del Reino Unido (se puede obtener tambien en <http://www.dia.uned.es/~jgb/util/mike.html>). Incluye: encadenamiento hacia atrás y hacia adelante, estrategias de resolución de conflictos definibles por el usuario, y un lenguaje de representación de marcos con mecanismos de herencia y demonios que permite que el usuario especifique sus propias estrategias de herencia.
- c) ES: Una herramienta para desarrollar sistemas expertos de muy fácil manejo que incorpora encadenamiento hacia adelante, hacia atrás, relaciones de conjuntos borrosos y métodos de explicación del razonamiento (se puede obtener via ftp en “unix.hensa.ac.uk” dentro del directorio “/mirrors/uunet-/pub/ai/”).
- d) CBR-Works: Una herramienta para desarrollar sistemas utilizando Razonamiento Basado en Casos a través de la definición y manipulación de los

mops correspondientes (se puede obtener en <http://www.dia.uned.es/~jgb/util/cbr.html>).

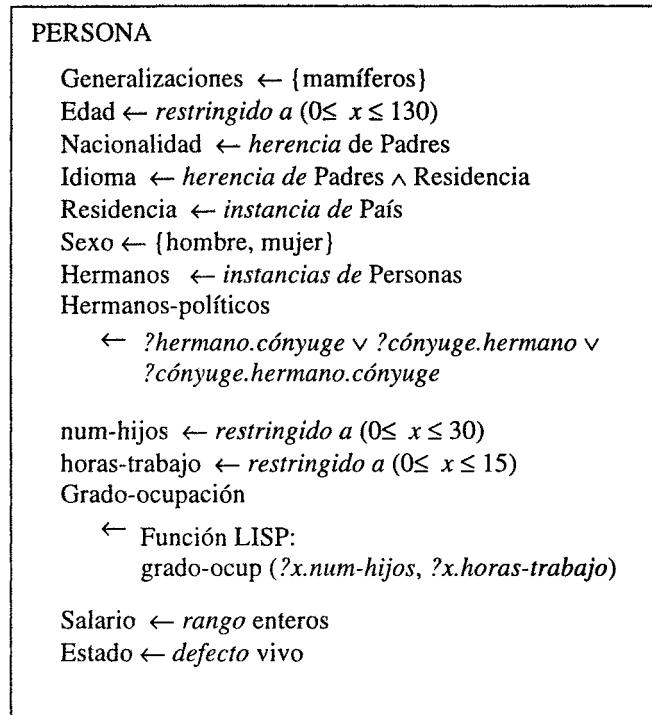


Figura 7-1 Marco genérico de Persona con diferentes formas de inferencia.

7.2 Motivación

En la resolución de cualquier problema sería deseable disponer de un esquema de representación que nos permitiera conjugar de una forma estándar y homogénea diferentes técnicas (razonamientos lógicos, aplicación de funciones escritas en algún lenguaje simbólico como LISP, mecanismos de herencia diversos, restricciones varias, etc.) para calcular los valores de los diversos atributos que poseen los objetos de un determinado dominio. Por ejemplo, en la figura 7-1 se representan algunos de estos mecanismos de inferencia asociados a una serie de propiedades de un marco genérico de *persona*.

En este planteamiento convendría que, al preguntar por el valor de un campo, en caso de no tener un valor previamente asignado o calculado, se pudiera activar el mecanismo de inferencia que sirviera para interpretar el código asociado a dicho campo. Así, si se solicitan los hermanos-políticos de una persona, se debería aplicar el método *prolog* para tratar el conjunto de reglas lógicas correspondientes (cláusulas escritas en estilo Prolog). En otro caso, como es el del atributo grado-ocupación, se tendría que llamar a la función Lisp grado-ocup que devolvería el grado de ocupación de una persona según sea su número de hijos y las horas trabajadas.

Por otro lado, para garantizar el proceso de inferencia de cualquier atributo, sería útil poder fijar una serie de mecanismos de *inferencia por defecto*¹ que fueran aplicados en el caso de no haber sido asociado ningún método concreto a dicho campo o en el caso de que el método indicado no haya producido el éxito deseado. Otra cuestión significativa relacionada consiste en determinar si se desea que todo valor inferido sea guardado o bien sea descartado para evitar la sobrecarga de la memoria del sistema.

En definitiva, tres de las características básicas que serían deseables en un formalismo de representación e inferencia flexible y eficiente son:

1. *Representación uniforme de todos sus elementos.*
2. *Selección de diferentes métodos de inferencia* (con la posibilidad de definir nuevos métodos y de aplicar mecanismos de aprendizaje alternativos).
3. *Organización eficiente del conocimiento adquirido.*

Estos tres principios, tal y como veremos enseguida, son precisamente los que persiguen un cierto tipo de sistemas denominados *Arquitecturas Integradas*. En lo que se refiere al **aprendizaje**, debe entenderse como aquella capacidad que poseen algunos sistemas de solucionar mejor la tarea encomendada según se vaya adquiriendo experiencia en su aplicación. La posibilidad de aprender es una cualidad innata del ser humano y un objetivo deseable en cualquier aplicación de IA (*el lector interesado en introducirse en el tema de aprendizaje puede acudir a los siguientes textos en castellano: [Mira et al., 1995, Moreno et al., 1994; Rich & Knight, 1991]*). No obstante, el conocimiento de dichas técnicas queda fuera de los objetivos de este texto, siendo exclusivamente nuestro interés ilustrar la ventaja de su utilización en el contexto de la solución de problemas (véase 7.7.3.5).

¹ Utilizaremos el término *inferencia por defecto* ya que es el que se utiliza en la mayoría de los libros. No obstante, lo correcto sería denominarlo *inferencia por omisión*.

7.3 Estructura y contenidos

En este capítulo se pretende ilustrar la conveniencia de combinar diferentes técnicas de representación e inferencia en la solución de problemas que requieren una formalización de una cantidad considerable del conocimiento del dominio y que, por tanto, son algo más que meros ejercicios.

Conviene recordar al lector que esta estrategia de combinar representación, inferencia e implementación es, de alguna forma, complementaria de otras perspectivas metodológicas tales como CommonKADS², en las que se pretende establecer una distinción clara entre las distintas fases de desarrollo de un sistema: análisis (modelo conceptual), especificación de distintas capas en el modelo del dominio (estrategia, tarea, inferencia y dominio) y, finalmente, obtención de una descripción formal y reescritura de los términos del modelo usando las primitivas de un lenguaje de implementación concreto (por ejemplo, el Lisp). Lo que queremos hacer notar es que, el involucrarse desde el comienzo en la implementación, aquí ha sido intencional; hemos pretendido llevar al lector desde la concepción a la implementación, comprometiéndole desde el principio con unas determinadas entidades y unas opciones de control específicas, en este caso simbólicas (*slots* de un tipo concreto en el que se asume una determinada forma de inferencia); por que, frente a las evidentes ventajas de las metodologías convencionales (CommonKADS, PROTÉGÉ, etc.), también es justo señalar el riesgo del desinterés en la implementación.

El discurso del tema lo dividiremos en tres partes claramente diferenciadas. La primera, de **introducción**, se divide en dos apartados: los motivos por los que los contenidos abordados tienen un interés considerable y las herramientas actualmente disponibles que pretenden satisfacer dichos objetivos, las denominadas *arquitecturas integradas*. En la segunda, de **fundamentación**, se concretan los diferentes métodos de inferencia disponibles en el esquema de una arquitectura concreta y se ilustra su utilización en un ejemplo introductorio que es representativo de dominios con relaciones jerárquicas. En la tercera, de **aplicación**, se utiliza dicho esquema para resolver algunos problemas tipo. El primero, denominado *apilar bloques*, es característico de los dominios en los que predominan los objetos estructurados con muchas relaciones entre ellos. El segundo, denominado *agenda personalizada*, comprende un conjunto de problemas existentes en un sistema real que se utiliza para

² La *modelización del conocimiento* es un tema que requiere un análisis detallado que excede los objetivos del presente texto. Por ejemplo, [Mira et al., 1995] contiene una introducción en castellano a este tema, en la que se incluye la metodología CommonKADS.

ayudar a gestionar la organización de las actividades en las que participa el usuario potencial del sistema; en este caso, un profesor de universidad con un alto grado de ocupación. Los problemas asociados a la agenda son, en realidad, instancias de una misma tarea, la *clasificación*. Debido a que la solución conocida de dichos problemas no aporta la respuesta deseada, en el último ejercicio del capítulo se introduce la posibilidad de **combinar estrategias de aprendizaje** con los mecanismos de inferencia estudiados previamente.

7.4 Arquitecturas integradas

En los *textos genéricos de IA* ([Nilsson, 1980; Pazos, 1987; Rich & Knight, 1991; Ginsberg, 1993; Borrajo *et al.*, 1993; Mira *et al.*, 1995, Russell & Norvig, 1995]) se pueden estudiar diversos mecanismos de representación del conocimiento e inferencia y en el presente texto se ha verificado su aplicabilidad. Pero, ¿cómo podríamos integrar todos estos elementos en un mismo sistema de una forma coherente? El objetivo es el desarrollo de una arquitectura³ capaz de proporcionar un entorno genérico de programación en el cual puedan combinarse dichos elementos (búsqueda, deducciones lógicas, distintos tipos de encadenamientos y de herencia, etc.). La integración y el diseño de estos macrosistemas tiene una serie de cuestiones asociadas, entre las que destacan: ¿qué métodos de representación, inferencia, etc., deben elegirse para cada tipo de problemas?, ¿cuáles son los elementos del diseño que deben representarse explícitamente?, ¿cuáles son las partes más difíciles de solventar y cuáles no son abordables?, ¿cuándo y qué debe aprenderse?, ¿qué decisiones de compromiso deben adaptarse? (por ejemplo, flexibilidad frente a homogeneidad y eficiencia), etc. Por tanto, parece claro que hay un espacio propio para la investigación de los problemas de diseño de arquitecturas, sobre el cual todavía quedan muchas asuntos por resolver (plantearemos algunos en esta sección).

En definitiva, se pretende proporcionar una fácil adaptabilidad a problemas diferentes. Por ello, tal y como refleja la tabla 7-1, las arquitecturas existentes proporcionan métodos genéricos de solucionar problemas y mecanismos alternativos de representación del conocimiento y de aprendizaje. Otra razón de ser de estos sistemas es el facilitar el desarrollo rápido de aplicaciones; fundamentalmente en el campo de los *sistemas basados en el conocimiento*. El mundo real es muy cambiante y se necesitan muchos sistemas fácilmente adaptables.

³ El término *arquitectura* proviene de la analogía existente entre este enfoque y las arquitecturas "hardware", donde hay ciertos componentes fijos de los cuales se parte y otros ajustables en el diseño.

	SOAR	PRODIGY	THEO
Inferencia	Encadenamiento hacia adelante	Análisis de Medios-fines	Encadenamiento hacia atrás
Representación	LTM: Reglas de Producción STM <obj atrib. valor>	Lógica de Primer Orden	<Marco slot valor>

Tabla 7-1 Comparación de diferentes arquitecturas integradas.

En la tabla 7-1 señalamos, por orden cronológico del comienzo de su desarrollo, tres de las arquitecturas existentes más conocidas: PRODIGY⁴ [Carbonell *et al.*, 1990], SOAR⁵ [Laird & Rosenbloom, 1990] y THEO⁶ [Mitchell *et al.*, 1991]. LTM son las iniciales del término inglés *Long Term Memory* (que señala el concepto informático de *memoria secundaria*). La memoria queda dividida lógicamente en dos espacios diferentes de almacenamiento. El ya señalado y el STM, o sea, *Short Term Memory*, que se refiere al término informático de *memoria principal*.

En dicha tabla puede observarse la coincidencia en utilizar como esquema básico de representación la terna <entidad atributo valor>. Ésta es una forma genérica de representación del conocimiento lo suficientemente flexible y general como para haber sido elegida el esquema básico de un gran número de aplicaciones.

Esquema básico de representación: <Entidad Atributo Valor>

Cada una de estas arquitecturas refleja un planteamiento general para abordar la solución de problemas. En SOAR se realiza una búsqueda hacia adelante en un espacio de estados. Cuando en dicho proceso de búsqueda surge un estado para el que no existe una decisión preestablecida se genera un *impás o punto de parada*, entonces se lanza una nueva búsqueda (dentro de la anterior) y, al ser resuelta, se crea una nueva regla de decisión que resume el proceso de búsqueda realizado. El almacenamiento de estas reglas de decisión provoca un aprendizaje paulatino en el dominio de aplicación. Esta forma básica de aprendizaje se denomina *chunking* o *encapsulamiento*, ya que consiste en recoger en una sola regla la esencia que explica la búsqueda realizada.

⁴ PRODIGY está disponible en <http://www.cs.cmu.edu/~prodigy/>

⁵ SOAR está disponible en <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/soar/public/www/home-page.html>

⁶ THEO está disponible en <http://www.cs.cmu.edu/afs/cs.cmu.edu/user/stork/mosaic/theo-dist.html>

En lo que se refiere a PRODIGY, es un sistema bastante complejo con muchas decisiones implicadas: elegir un elemento en el espacio de búsqueda (un nodo en este caso), elegir una meta para dicho nodo, elegir algún operador aplicable y determinar una asociación de variables para el operador seleccionado (recordar los procesos de equiparación y unificación de la lógica y la equiparación de patrones realizada por los sistemas basados en reglas). En este caso, el funcionamiento se basa en aprender reglas de control sobre cada una de estas decisiones realizadas. En concreto, en el contexto de solucionar problemas, lo que se realiza es un aprendizaje a posteriori de los caminos del espacio de búsqueda que llevan a la solución de un problema. Así, cuando desde un estado se puede alcanzar la solución, se aprenden las condiciones que han permitido aplicar el operador utilizado en dicho estado. En otras palabras, se aprende cuándo es útil aplicar un operador o, si se prefiere, cuándo un estado es una instancia de los estados a los que conviene aplicar un cierto operador. En este sistema se ha investigado el concepto de **utilidad del conocimiento aprendido**, de forma que sólo se guardan las reglas que producen un ahorro —en tiempo de cómputo— superior al coste implicado en la comprobación de su posible aplicación. Dado que toda regla tiene una serie de antecedentes, cuantos más antecedentes tenga, y más reglas haya, más lento será comprobar qué regla debe aplicarse en cada momento.

En cuanto al sistema THEO [Mitchell *et al.*, 1991], cuya estructura se refleja en el tipo de arquitectura que vamos a utilizar en los problemas del presente capítulo, podemos resaltar que está construido en el lenguaje Lisp, con una representación uniforme basada en marcos. Tanto los hechos, (también conocidos con el nombre de *creencias*⁷) como los problemas y los métodos para solucionarlos, se representan a través de los valores asignados a las características de los marcos (a las que también denominaremos *slots*, con el fin de evitar ambigüedades en su interpretación). El mecanismo de mantenimiento de verdad (*truth maintenance system*), es decir, de garantizar la coherencia entre todos los hechos existentes en cada momento, está basado en la continua creación de explicaciones para todas las creencias inferidas. Esto es, cada vez que se infiere el valor de un *slot*, se anota en un *subslot* todos aquellos *slots* de los cuales depende el valor obtenido. Por ejemplo, tal y como veremos en un problema de relaciones familiares (véase la figura 7-2), si el valor del *slot* hijos de Ana López depende del *slot* padres de sus hijos respectivos, entonces se anotaría dicha dependencia.

⁷ El término creencias —o suposiciones— proviene de la descripción del *Nivel del Conocimiento* [Newell, 1981]. La abstracción clave de dicho nivel es la noción de un *agente racional idealizado*, utilizado para describir el llamado *nivel del conocimiento* de los sistemas informáticos. Para explicar su comportamiento, se le atribuyen al agente *objetivos* y *creencias*, de forma que su comportamiento pueda considerarse racional (por ejemplo, [Mira *et al.*, 1995] incluye una introducción al Nivel del Conocimiento).

El aprendizaje en THEO se puede realizar a través de la activación de diversos métodos: mediante el mero almacenamiento de la información obtenida (llamado *caching*, o *recogelotodo*); a través de la creación de macrométodos (al igual que en SOAR es un resumen que explica lo ocurrido); aplicando un proceso de obtención de reglas aprendidas estilo prolog, generadas a partir de árboles de decisión (el aprendizaje de este tipo de árboles ha dado muy buenos resultados en las *tareas de clasificación*, véanse los apartados 0 y 7.7.3.5) y utilizando un modelo básico de aprendizaje conexionista (el clásico Backpropagation explicado en: [Rich & Knight, 1991; Mira *et al.*, 1995; Russell & Norvig, 1995]).

Si realizamos un análisis comparativo que resalte los aspectos comunes de las arquitecturas de la tabla 7-1, encontraremos que, en cuanto al tipo de representación uniforme adoptada, SOAR y PRODIGY utilizan espacios de problemas, mientras que THEO tiene como elemento básico de representación los *slots*. En cuanto a lo que podría denominarse "ejecución del agente guiada por impases"⁸, SOAR está dirigido por "impases" del espacio del problema, mientras que la acción de THEO se basa en encontrar el valor del *slot* requerido (inferencia del valor *por demanda*). En lo que se refiere a los métodos generales utilizados en la solución de los problemas planteados, SOAR busca en espacios de problemas, PRODIGY utiliza además *analogía* y THEO puede emplear varios métodos —fundamentalmente búsqueda, valores por defecto y herencia— (ver más adelante el apartado 7.5.2). Por último, en lo que se refiere al aprendizaje provocado por "impases", SOAR aprende nuevas producciones por cada punto de parada, PRODIGY utiliza *aprendizaje basado en la explicación* en las situaciones seleccionadas y THEO por defecto aplica *caching* para todos los *slots* cuyos valores se hayan inferido (la bibliografía recomendada sobre aprendizaje se ha referenciado previamente al comienzo del apartado 7.2).

7.5 Esquema de una arquitectura

Una vez aclarado el contexto, vamos a centrarnos en comentar cómo podría funcionar una de estas arquitecturas y vamos a utilizarla para resolver una serie de problemas. Esta arquitectura, inspirada en Theo [Mitchell *et al.*, 1991] y en la aplicación del Calendario [Dent *et al.*, 1992] (véase el apartado 0), pretende ser una herramienta para el desarrollo de sistemas capaces de solucionar problemas con capacidad autónoma de mejorar su comportamiento. Para ello proporciona un marco general y uniforme de representación del conocimiento, de tal forma que, tanto los

⁸ Situaciones en las que el "agente" no tiene ninguna regla aplicable y se debe iniciar un proceso de búsqueda para solventar la decisión implicada.

métodos seguidos para solucionar los problemas como los utilizados por los diversos mecanismos de aprendizaje, puedan describirse mediante las mismas estructuras de datos (véase 7.5.1). Por tanto, intenta ser un entorno plural en el que se pueden describir y estudiar problemas de integración de técnicas de inferencia y representación diferentes.

Una arquitectura debería considerarse como una herramienta de desarrollo que proporciona, para ser estudiados, mecanismos de:

- Solución general de problemas (aplicando cualquier método disponible: búsqueda, inferencias lógicas, reglas, etc.).
- Aprendizaje.
- Representación del conocimiento.
- Interacciones entre los anteriores.

Tal y como ocurre en THEO, para que una arquitectura pueda ser lo suficientemente genérica y flexible como para llegar a ser útil en tareas tan diversas como el control de un robot [Mitchell, 1990] o la gestión personalizada de una agenda [Dent *et al.*, 1992; Mitchell *et al.*, 1994], **debe ser capaz de acceder y modificar la mayoría de las estructuras que la forman**. Por tanto, el aprendizaje es inseparable de los problemas relativos a la representación del propio sistema y de las tareas del dominio que quieran resolverse, así como del acceso —a través de un sistema de indexación (comentado en 7.5.1)— a estos elementos en el momento adecuado —dentro del funcionamiento general de la aplicación específica que se esté llevando a cabo—.

Lo que queremos resaltar en un libro como éste, dedicado a búsqueda y representación del conocimiento, es que hay al menos dos *formas de abordar el problema de la representación*:

1. Considerando exclusivamente la programación de la tarea.
2. Teniendo como objetivo modelar y programar el aprendizaje, es necesario acceder y modificar las estructuras que constituyen la tarea.

El estar involucrado en la implementación permite seleccionar entre opciones concretas diferentes. Unas no podrán soportar después el aprendizaje, en cambio, otras dejan en la estructura la posibilidad de acceder a ella y modificarla.

Para facilitar la reutilización y ampliación del conocimiento, podemos utilizar una única estructura elemental, los *slots*, que sirven tanto para representar nuevas características de elementos del dominio como para ampliar y/o modificar los métodos de inferencia y aprendizaje inicialmente provistos. Esta forma unívoca de

representación facilita la modularización de los desarrollos, a la vez que normaliza el acceso a la información relevante existente y a su actualización.

7.5.1 Características estructurales

Las entidades que sirven para describir cualquier objeto o elemento en el sistema son: *marcos*, *slots* y sucesivos niveles de *subslots* —tantos como se consideren necesarios—. Conviene precisar que, aunque también utilicemos los términos *atributos* y *características* para referirnos a los *slots* de un marco, seguiremos empleando *slots* y *subslots*, con el fin de resaltar que son los componentes que describen las propiedades de un objeto en un formalismo de representación determinado, que se conoce con el nombre de *marco* (*frame*). Por tanto, la terna que refleja el mecanismo de representación básico del sistema es: <marco slot valor> y en torno a esta se estructura la representación de cualquier problema.

Esquema básico de representación: <marco slot valor>

En concreto, la organización del sistema es la siguiente:

1. Un **problema** se describe con la pareja (<entidad> <slot>), siendo la solución del problema el valor inferido: "(<entidad> <slot>) = <valor>". Por ejemplo, según la figura 7-2, para averiguar los hermanos de Elena Rodríguez habría que preguntar por (elena hermanos). Dicho de otra forma, el elemento <slot> es el nombre de la relación y cada problema es la pregunta por el rango (los valores) de una instancia de dicha relación.

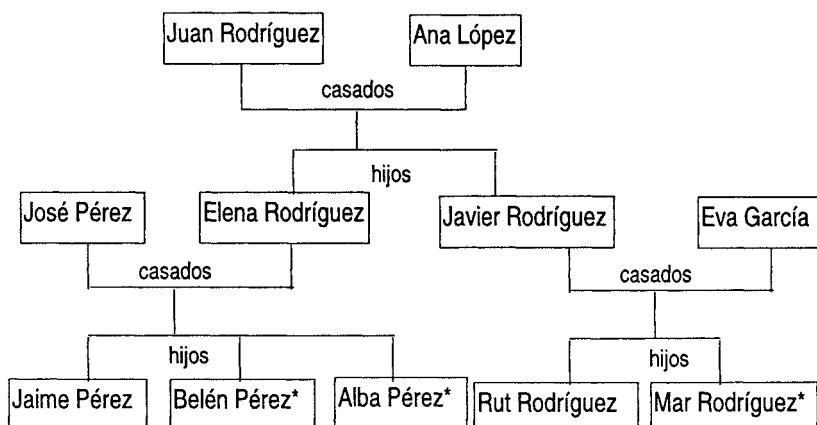


Figura 7-2 Jerarquía de relaciones familiares.

2. También existen **clases de problemas** —o conjunto de instancias de problemas— representados por ($?x <slot>$), o simplemente $<slot>$; y señala la relación $<slot>$ restringida al subdominio especificado por el tipo de entidad especificado en $?x$. Por ejemplo, el slot padres representa la clase de instancias de problemas de la forma ($?x padres$), donde $?x$ es cualquier miembro del dominio del slot, tal como (belen padres). De esta forma, (mujeres padres) representaría la clase de instancias de problemas ($?m padres$), donde $?m$ es algún miembro de la clase mujeres. Si sólo se indica el nombre de la relación, no se impone ningún tipo de restricción y se considera como problemas relacionados todas aquellas *direcciones* cuyo último elemento sea $<slot>$. Se considera que la *dirección de un slot* es el nombre de la entidad representado por dicho slot. Es decir; la secuencia de slots desde el nivel superior hasta alcanzar el slot en cuestión (esto se entenderá mejor con la lectura del siguiente punto). Por ejemplo, si se habla del slot conyuge, una posible dirección asociada sería (eva conyuge).
3. Las **instancias de problemas** y las **clases de problemas** son a su vez **entidades**. Así, se pueden expresar *creencias* sobre problemas y problemas sobre otros problemas, como si de otra entidad se tratase. Por ejemplo, la creencia ((ana hijos) existen?) = t denota que la solución del problema que consiste en averiguar los hijos de Ana, va a tener como resultado algún elemento distinto de NIL (en Lisp T significa *cierto* y NIL denota *falso* y *nulo*). En estos casos, para facilitar la notación, se pueden suprimir los paréntesis intermedios, quedando expresiones como (ana hijos existen?) = t. Por tanto (según lo indicado en 2), (ana hijos existen?) sería una posible dirección asociada al slot existen?
4. El **conocimiento de control** del sistema se almacena siguiendo las mismas convenciones descritas previamente (1, 2 y 3). De esta forma, todo lo que se refiere a los métodos recomendados para responder a una *clase de problemas* se almacena como *creencias* sobre dicha clase. Por ejemplo, (persona conyuge métodos) = (herencia valor-por-defecto) indica que los métodos para solucionar el problema (eva conyuge) son: "herencia" y "valor-por-defecto" (serán aplicados en dicho orden, según se explica en 7.5.2). Así, cuando se pregunte por el valor del slot conyuge de una instancia de persona, primero se comprueba si ya hay un valor asignado, y en caso negativo, se verifica si se han señalado expresamente métodos de inferencia para dicho slot, en cuyo caso se aplican uno a uno dichos métodos hasta que alguno consiga inferir algún valor. De esta forma, se consigue un esquema de representación basado en la homogeneidad, tanto en la representación de los

objetos del dominio como en los mecanismos de inferencia que se les pueden aplicar a dichos objetos (y a otros de otros dominios).

Representación uniforme:

conocimiento de control \equiv conocimiento del dominio

5. Una vez que se haya solucionado un problema, el sistema puede almacenar su resultado como una creencia asociada al nombre del problema. Así, si se infiere que el cónyuge de Eva es Javier, el sistema establecería la creencia (eva conyuge) = javier. Además, cada vez que se infiere una solución se almacena una explicación de cómo se ha alcanzado dicho valor. Con ello, el sistema puede, por un lado, **mantener la consistencia** con el resto de los valores de los *slots* de los cuales depende y, por otro, aplicar métodos de aprendizaje que consideran la explicación de dicho proceso de inferencia.

Resumiendo, del análisis previo podemos resaltar:

- Los hechos o creencias se almacenan como valores de los atributos de las entidades.
- Los problemas son preguntas sobre el valor de un atributo de una entidad.
- Los atributos son a su vez entidades que representan el problema de inferir dicho slot. Como entidades, estos *slots* pueden tener subslots que, por ejemplo, indiquen los métodos de inferencia para calcularlos.

7.5.2 Características funcionales

Para plantear un problema en esta arquitectura, o lo que es lo mismo, para invocar cualquier método de inferencia asignado a un slot, se debe preguntar por su valor asociado mediante la función predefinida `dame-valor`.

Obtención del valor de un slot: (`dame-valor <slot> <entidad>`)

Para generar el valor de un *slot* se pueden definir métodos de inferencia propios (adaptados a cada dominio) o se pueden utilizar una serie de métodos predefinidos. Cuando no se especifica ningún método en concreto sobre un *slot* determinado se aplican en un cierto orden una serie de ellos (predefinidos). El proceso consiste en probar uno tras otro hasta que alguno infiera un valor para el *slot* requerido. De los métodos utilizables nos vamos a quedar con los siguientes (están numerados según el orden en que son probados):

1. HERENCIA (en el sistema recibe el nombre herencia): Consiste en inferir los valores de un *slot* a través de la relación jerárquica definida mediante los *slots* generalizaciones y especializaciones. En concreto, consiste en sustituir en la dirección del *slot* en cuestión su dominio por el indicado en generalizaciones. Por ejemplo, si se pregunta por (ana herederos), se intentaría inferir el valor de la dirección (persona herederos), suponiendo que la generalización de Ana López fuera persona (tal y como veremos en 7.6). Si ésta tampoco diera resultado, se volvería a sustituir persona por su generalización; por ejemplo, se preguntaría por (animal-racional herederos). Este proceso se repetiría tantas veces como fuera necesario en la jerarquía hasta dar con el valor buscado o llegar a la raíz de la misma.
2. PÉRDIDA DE CONTEXTO (recibe el nombre perdida-contexto): Devuelve el valor de "la cola" de la dirección en la que se halla inmerso el *slot* en cuestión. Es decir, la dirección que resulta de omitir el primer elemento de la misma. Por ejemplo, la pérdida de contexto aplicada sobre la dirección (persona herederos metodos) del *slot* metodos devolvería el valor contenido en la dirección (herederos metodos).
3. VALOR POR DEFECTO (valor-por-defecto): Accede al valor del *subslot* valor-por-defecto del *slot* dado. Por ejemplo, de (rut padres) se pregunta por (rut padres valor-por-defecto).
4. ESPECIALIZACIÓN (especializacion): Obtiene el valor de alguna especialización del slot. Por ejemplo, a partir de (jose hijos) buscaría en (jose hijos-varones).

Además de estos métodos, que suponemos que se aplican siempre que no se indique nada sobre el valor de un campo, existen otra serie de métodos que pueden ser invocados directamente, ya que también están predefinidos. Entre éstos, destacamos el Prolog por su importancia. Este método permite utilizar reglas lógicas que indican cómo puede inferirse el valor de un campo determinado. El razonamiento realizado es perfectamente deductivo en el sentido de las lógicas clásicas. Es decir, dados una serie de antecedentes de una regla se comprueba si existen hechos que los confirmán para poder deducir (utilizando el clásico *modus ponendo ponens*) el consecuente de la misma.

- PROLOG: determina el valor de un campo aplicando un conjunto de reglas lógicas hasta que alguna infiera el valor requerido.

Sintaxis: tendremos que definir una estructura de expresiones válidas para las reglas de estilo prolog. La figura 7-3 resume (utilizando la notación BNF) las formas básicas en que pueden generarse expresiones bien formadas en prolog.

Para comprender esta sintaxis vamos a repasar brevemente los ejemplos de la figura 7-4. La afirmación (padres elena juan) se cumple cuando el *slot* padres de la entidad elena tiene entre sus valores el elemento juan. La variable ?cualquiera en el término (hijos dios ?cualquiera) permite establecer una asociación entre dicha variable y cualquier valor del *slot* hijos de la entidad dios. Es decir, todas las expresiones que tengan variables se presuponen cuantificadas universalmente, por lo que puede haber tantas instancias de una regla concreta como entidades que cumplan la relación señalada. Este factor condiciona, como veremos más adelante (sec. 7.7.1.1), las sucesivas aplicaciones de una regla para cada uno de los valores de un *slot* que sea multivalorado.

```

{clausulas prolog} ::= ( {regla}* )
{regla}      ::= ( {literal} :- {condición}+ )
{condición}   ::= {literal} | ! |
                  ( eval {termino} {termino} ) |
                  fallo |
                  ( not {literal}+ )
{literal}     ::= ( {identificador} {termino} {termino} )
{termino}     ::= {atom} | ( {termino}* )
{variable}    ::= ?{identificador}
{atom}        ::= {constante} | {variable}
{constante}   ::= {identificador} | {número}
{identificador} ::= {slot}

-----
Nota:      {...}* significa 0 o más instancias de {...}
          {...}+ significa 1 o más instancias de {...}

```

Figura 7-3 Sintaxis de prolog.

Al referirnos al valor ?m del atributo métodos (figura 7-4) de la clase de problemas denominada grado-ocupacion en la sentencia (grado-ocupacion ?persona ?tipo), estamos permitiendo que ?m pueda tomar el valor de cualquiera de los métodos de inferencia definidos sobre dicho atributo, independientemente de cuál sea el dominio de dicho atributo (?persona) y de cuál sea el rango de valores del mismo (?tipo). Finalmente, las reglas permiten establecer un nuevo hecho o creencia en el sistema cuando

se cumplen los antecedentes de la misma. Así, según el ejemplo de la figura, si se puede encontrar una `?persona` que tenga en el atributo padres algún valor (asignado en el proceso de equiparación a la variable `?padres`) entonces, si se cumple a su vez (segunda cláusula) que dicho valor (`?padres`) tiene entre sus padres algún otro valor, este último será asignado al atributo abuelos de dicha `?persona`. Nótese la presencia de unos paréntesis externos que encierran a todas las cláusulas que componen una regla.

```

término:
(padres elena juan)

(hijos dios ?cualquiera)

(metodos (grado-ocupacion ?persona ?tipo) ?m)

regla :
((abuelos ?persona ?abuelos) :-
  (padres ?persona ?padres)
  (padres ?padres ?abuelos))

```

Figura 7-4 Ejemplos de prolog.

Funcionalidad: las reglas de estilo prolog se pueden asignar a un atributo concreto indicando, por un lado, que el método prolog es el que debe utilizarse para inferir el valor del *slot* y, por otro, estableciendo de forma explícita el contenido de dichas reglas. Para lo primero se utiliza el atributo metodos y, para lo segundo, se debe definir el *subslot* denominado clausulas-prolog. Siguiendo estas convenciones, en la figura 7-5 se muestra la definición del *slot* abuelos en la entidad persona.

Se pueden asignar tantas reglas prolog como se deseé, encerrándolas entre paréntesis. **Las reglas se aplican siguiendo el orden en que están escritas** empezando por la que se encuentre más cerca del *subslot* clausulas-prolog. Para que alguna de ellas pueda aplicarse **se requiere que se cumplan todos sus antecedentes**. Además, tal y como se puede ver en la notación de la figura 7-3, se pueden invocar funciones escritas en algún lenguaje de programación directamente en el antecedente de las reglas. Aquí vamos a suponer que dichas funciones están escritas en el lenguaje de

manipulación símbolica Lisp (su utilización se comenta en el punto siguiente).

```
(persona *sin-valor*
  (generalizacion
    (animal-racional mamifero ser-vivo))
  (especialización
    (hombre mujer))
  (nombre *sin-valor*)
  (apellido *sin-valor*)
  (conyuge *sin-valor*)
  (abuelos *sin-valor*
    (metodos (prolog))
    (clausulas-prolog (
      (abuelos ?persona ?abuelos) :-
        (padres ?persona ?padres)
        (padres ?padres ?abuelos))))))
```

Nota: () son los paréntesis que encierran cada una de las reglas prolog.

Figura 7-5 Asignación de cláusulas prolog.

```
(objeto-fisico *sin-valor*
  (generalización
    (marco))
  (volumen *sin-valor*)
  (densidad *sin-valor*)
  (peso *sin-valor*
    (metodos (prolog))
    (clausulas-prolog (
      ((peso ?objeto ?peso) :-
        (volumen ?objeto ?vol)
        (densidad ?objeto ?den)
        (eval (* ?vol ?den) ?peso))))))
```

Figura 7-6 Evaluación de funciones en cláusulas.

FUNCIONES: en algunas ocasiones es necesario evaluar la aplicación de funciones dentro de las cláusulas de una regla. ¿Cómo si no podría determinarse el peso de un objeto a partir de su densidad y su volumen? Para ello se utiliza eval, que se encarga de asignar el valor devuelto por su

primer argumento a su segundo argumento. Este caso queda reflejado en la definición del *slot* peso de la entidad objeto-físico de la figura 7-6.

La función eval se encarga de asignar el valor devuelto por su primer argumento a su segundo argumento.

En otros casos, la definición explícita mediante cláusulas prolog de todos los pasos que hay que realizar para calcular el valor de un *slot* puede ser demasiado prolífica e ineficiente. En su lugar, se puede definir una función Lisp que calcule directamente dicho valor. Por ejemplo, supongamos que queremos averiguar el grado de ocupación de una persona, al cual vamos a considerar alto, medio o bajo, según el número de horas trabajadas. En la figura 7-7 se define mediante cláusulas prolog el valor de dicho *slot*.

```
(persona *sin-valor*
  (... 
    (horas-trabajadas *sin-valor*
      (valores-possibles numero)
      (valor-minimo 0)
      (valor-maximo 15))
    (... 
      (grado-ocupacion *sin-valor*
        (metodos (prolog))
        (clausulas-prolog (
          ((grado-ocupacion ?persona ?valor) :-
             (horas-trabajadas ?persona ?horas)
             (eval (>= ?horas 0))
             (eval (<= ?horas 6))
             (eval 'bajo ?valor))
          ((grado-ocupacion ?persona ?valor) :-
             (horas-trabajadas ?persona ?horas)
             (eval (> ?horas 6))
             (eval (<= ?horas 8))
             (eval 'medio ?valor))
          ((grado-ocupacion ?persona ?valor) :-
             (horas-trabajadas ?persona ?horas)
             (eval (> ?horas 8))
             (eval (<= ?horas 15))
             (eval 'alto ?valor)))))))
```

Figura 7-7 Cálculo explícito del valor de un *slot*.

En lugar de tener que repetir sucesivas veces la búsqueda de los hechos que constituyen los antecedentes de las tres reglas sobre el *slot* grado-

ocupacion, podría haberse definido una función calculo-horas que devolviera el valor alto, medio o bajo de una sola vez; tal y como queda indicado en la figura 7-8. La selección de una u otra alternativa es un caso concreto en el que se puede contrastar el dilema genérico entre *utilizar técnicas declarativas o procedimentales* de representación del conocimiento. En el sistema que estamos describiendo esto dependerá de si queremos dejar constancia o no del camino de inferencia seguido. Por ejemplo, sería mejor utilizar la definición explícita si se van a aplicar posteriormente *técnicas de aprendizaje basadas en la explicación* del camino de inferencia (la bibliografía recomendada sobre aprendizaje se ha referenciado previamente al comienzo del apartado 7.2).

```
(persona *sin-valor*
  ...
  (horas-trabajadas *sin-valor*
    (valores-posibles numero)
    (valor-minimo 0)
    (valor-maximo 15)
  ...
  (grado-ocupacion *sin-valor*
    (metodos (prolog))
    (clausulas-prolog (
      ((grado-ocupacion ?persona ?valor) :-
         (horas-trabajadas ?persona ?horas)
         (eval (calculo-horas ?horas) ?valor)))))))
```

Figura 7-8 Cálculo implícito del valor de un slot.

```
(persona *sin-valor*
  ...
  (horas-trabajadas *sin-valor*
    (valores-posibles numero)
    (valor-minimo 0)
    (valor-maximo 15)
  ...
  (grado-ocupacion *sin-valor*
    (metodos (lisp))
    (lisp grado-ocupacion)))
```

Figura 7-9 Cálculo del valor de un slot aplicando una función Lisp.

- INVERSO: es otro de los métodos que se pueden invocar directamente. Su aplicación hace que se busque el valor en otro *slot* que tenga como valor de su propiedad inverso el nombre del *slot* buscado. Por ejemplo, si se establece que (marido inverso)= mujer y (mujer inverso)= marido. Si Juan y Ana estuvieran casados (véase la figura 7-2) y se preguntase por (marido ana) se accedería a ((marido inverso) ?marido) = ana; o sea, (mujer ?marido) = ana. En otras palabras, se accede al valor de aquella relación cuyo dominio asociado a la propiedad especificada en el *subslot* inverso sea el valor buscado y cuyo rango sea precisamente el dominio de la relación original. Para que este método sea probado debe ser asignado a la propiedad *métodos* del *slot* que corresponda.
- LISP: no vamos a entrar en detalle en la especificación de funciones en este lenguaje concreto, ya que requiere conocer sus rudimentos. No obstante, queremos dejar constancia de que declarando (<*slot*> métodos (lisp)) se busca en el *subslot* (<*slot*> lisp) el nombre de una función lisp que al ser aplicada recibe como argumento la *dirección* (sec. 7.5.1) del <*slot*> (véase la figura 7-9).

Como puede observarse, cualquier método de inferencia asignado a un *slot*, realmente lo único que hace es **señalar un conjunto de posibles direcciones** (*slots* de entidades del sistema) **en las que puede encontrarse el valor** de dicho *slot*. Esta idea tan simple es también interesante. Podríamos ver al sistema como una "tela de araña en construcción". Al principio, sólo hay unas pocas direcciones asignadas, que dicen por dónde se puede expandir el sistema (la tela de araña). Según el sistema (la araña) tenga interés en crecer en alguna dirección (se le pregunte), éste empieza a crecer en la misma. La idea de la tela de araña cuadra igualmente con el mecanismo utilizado para el mantenimiento de la coherencia del sistema (en cuyo estudio no nos vamos a extender aquí). En dicha tela todos los hilos están unidos de alguna forma entre sí y, si ésta se rompe, hay que reconstruirla o dejar constancia del "agujero". Igual ocurre en el sistema, si un *slot* depende del valor de otro y éste deja de tener dicho valor, entonces el primero también deja de tener el suyo y habría que reconstruir el proceso de inferencia (este mecanismo nos recuerda la *dependencia reversible* de las reglas).

7.6 Ejemplo introductorio

Aunque a lo largo de los apartados previos ya hemos utilizado las relaciones familiares de la figura 7-2 para ejemplificar el esquema de representación de la arquitectura elegida, vamos a concretar ahora la estructura global que tendría dicho problema. Ya hemos comentado que lo esencial del modelo elegido es utilizar los

marcos como elemento básico de representación, cuya forma genérica se muestra en la figura 7-10. Todas las *creencias* (sec. 7.5.1) sobre un objeto se aglutan en torno al marco que lo define. Tanto los marcos como los *slots* se consideran entidades. Es decir, se pueden definir una serie de propiedades o relaciones asociadas a ellos. Se utiliza la notación *sin-valor* para indicar que, al crear una instancia del marco correspondiente, se podrá asignar el valor que se deseé.

```
(nombre-marco *sin-valor*
  (generalizaciones
    (marco-1 marco-2 ... marco-n)
  (especializaciones
    (marco-1 marco-2 ... marco-n)
  (slot-1 *sin-valor*
    (slot-11 *sin-valor*)
    (slot-12 *sin-valor*
      (slot-121 *sin-valor*)
      (slot-122 *sin-valor*
        ...
        ...
      )
    )
  )
)
```

Figura 7-10 Marco genérico.

En las siguientes figuras mostraremos, como un ejemplo de aplicación del marco genérico, la representación asociada a la entidad persona, que refleja la información del dominio sobre cada uno de los objetos de la relación jerárquica familiar de la figura 7-2. Para llevar a cabo esta representación hemos considerado el siguiente enunciado del problema:

- 1) Definir un marco genérico que contenga la siguiente información: nombre, apellido, cónyuge, padres, hijos y si está vivo o no. Para ello, se debe completar dicha entidad considerando las indicaciones siguientes:
 - a) Definir las funciones (*demonios*) que actualicen en caso de fallecimiento las relaciones: padres, abuelos y tíos de una determinada persona.
 - b) Definir el conjunto de reglas que determinen cuáles son los hermanos políticos de una determinada persona.
 - c) Definir el conjunto de reglas que determinen cuáles son los herederos de una persona. Para ello se utilizará la jerarquía de objetos previamente definida y habrá que tener en cuenta las siguientes circunstancias:
 - i) Si una persona tiene un cónyuge que está vivo, entre sus herederos estará el cónyuge.

- ii) Si una persona tiene hijos que están vivos, éstos estarán entre sus herederos.
- iii) Si una persona tiene un descendiente (hijo) que tiene hijos, los hijos del descendiente serán sus herederos siempre que el cónyuge y sus hijos hayan fallecido.
- d) Definir, de las instancias de la figura 7-2, una para cada uno de los tipos de entidades distintos en el dominio.

```
(persona *sin-valor*
  (generalizaciones
    (animal-racional mamifero ser-vivo))
  (especializaciones
    (hombre mujer))
  (nombre *sin-valor*)
  (apellido *sin-valor*)
  (conyuge *sin-valor*)
  (vivo *sin-valor*
    (valores-possibles (si no)))
  (padres *sin-valor*
    (metodos (prolog)
      (clausulas-prolog *sin-valor*
        (((padres ?x ?padre) :-
          (hijos ?padre ?x)
          (vivo ?padre si)))))

    (hijos *sin-valor*
      (metodos (prolog))
      (clausulas-prolog *sin-valor*
        (((hijos ?x ?hijo) :-
          (padres ?hijo ?x)
          (vivo ?hijo si)))))))
```

Figura 7-11 Definición del marco asociado a la entidad persona.

En este ejercicio utilizaremos prolog como principal método de inferencia de las relaciones familiares mencionadas. Para ello, todos los *slots* que identifiquen alguna relación deberían incluir la sentencia: (metodos (prolog)). Más adelante veremos otros ejercicios en los que se emplearán el resto de los métodos disponibles (apartado 7.5.2).

```

(persona *sin-valor*
(.....
(tios *sin-valor*
(metodos (prolog))
(clausulas-prolog *sin-valor* (
((tios ?x ?tio) :- (
(hijos ?padre ?x)
(hijos ?abuelo ?padre)
(hijos ?abuelo ?tío)
(vivo ?tío si)))))

(abuelos *sin-valor*
(metodos (prolog))
(clausulas-prolog *sin-valor* (
((abuelos ?x ?abuelo) :- (
(hijos ?padre ?x)
(hijos ?abuelo ?padre)
(vivo ?abuelo si)))))))

```

Figura 7-12 Definición de los tíos y los abuelos.

Vamos a describir paso a paso la definición de cada uno de los *slots* necesarios para representar la información del problema. En primer lugar, definimos los atributos que contienen la descripción mínima que nos permita establecer las relaciones familiares de la figura 7-2. Para ello, tenemos en cuenta el apartado a) del enunciado. Las definiciones de la figura 7-11 se han realizado considerando sólo los *slots* básicos de las entidades personas. El único aspecto resaltable de estas definiciones es que se supone que las relaciones familiares sólo se mantienen entre personas que están vivas. En realidad, los *slots* deberían llamarse padres-vivos, hijos-vivos, etc. Por tanto, se utiliza el atributo vivo para garantizar el valor de una relación. Por esta razón, se escriben reglas que determinan que alguien sólo tiene padres o hijos que estén vivos. En lugar de haber incluido la condición "estar vivo" en el valor de dichas relaciones, se podría haber incluido las cláusulas correspondientes en el campo vivo que actualizaran dichas relaciones en caso de que alguien falleciese (*?persona vivo no*). Ambas soluciones son factibles y su aplicación depende del contexto (en la realización de los *demonios* se debe optar por una u otra). La aquí adoptada es más "económica", ya que *no se actualizan todas las relaciones que dependen de otra hasta que no se pregunta por aquéllas*. En este caso, dado que no se parte del supuesto de que alguien pueda resucitar, ambas opciones son igualmente válidas.

```
(persona *sin-valor*
(.....
(hermanos *sin-valor*
(metodos (prolog))
(clausulas-prolog *sin-valor* (
((hermanos ?x ?hermano) :-
(hijos ?padre ?x)
(hijos ?padre ?hermano))
(vivo ?hermano si)))))

(tios *sin-valor*
(metodos (prolog))
(clausulas-prolog *sin-valor* (
((tios ?x ?tio) :-
(hijos ?padre ?x)
(hermanos ?padre ?tio)
(vivo ?tío si))))
```

Figura 7-13 Definición de tios considerando la relación hermanos.

```
(persona *sin-valor*
(.....
(hermanos-politicos *sin-valor*
(metodos (prolog))
(clausulas-prolog *sin-valor* (
((hermanos-politicos ?x ?hermano-p) :-
(conyuge ?x ?conyuge)
(hermanos ?conyuge ?hermano-p))
((hermanos-politicos ?x ?hermano-p) :-
(hermanos ?x ?hermano)
(conyuge ?hermano ?hermano-p))
((hermanos-politicos ?x ?hermano-p) :-
(hermanos ?x ?hermano)
(conyuge ?hermano ?hermano-p))
((hermanos-politicos ?x ?hermano-p) :-
(conyuge ?x ?conyuge)
(hermanos ?conyuge ?hermano-p1)
(conyuge ?hermano-p1 ?hermano-p))))))
```

Figura 7-14 Definición de los hermanos-políticos.

Para aquellas situaciones de dependencia mutua en las que es frecuente que uno de los valores sea muy volátil; por ejemplo, si (habitacion luz encendida) entonces (robot ver si), no es necesario cambiar el valor de (robot ver no) cada vez que

se apague la luz, a no ser de que sea expresamente preguntado (se supone que esto ocurre cuando se quiere activar dicho robot).

Además de los *slots* básicos del enunciado, nos piden que definamos las relaciones tíos y abuelos (figura 7-12). Como ya hemos indicado anteriormente (sec. 7.5.1), las relaciones se definen creando nuevos *slots* para la entidad persona.

```
(persona *sin-valor*
(.....
(herederos *sin-valor*
  (clausulas-prolog *sin-valor* (
    ((herederos ?x ?heredero) :-
      (conyuge ?x ?heredero)
      (vivo ?heredero si))
    ((herederos ?x ?heredero) :-
      (hijos ?x ?hijo)
      (vivo ?hijo si))
    ((herederos ?x ?heredero) :-
      (conyuge ?x ?conyuge)
      (vivo ?conyuge ?no)
      (hijos ?x ?hijo)
      (not (vivo ?hijo si))
      (hijos ?hijo ?heredero)
      (vivo ?hereredero si)))
    ((herederos ?x ?heredero) :-
      (conyuge ?x ?conyuge)
      (vivo ?conyuge ?no))
      (hijos ?x ?hijo)
      (not (vivo ?hijo si))
      (padres ?x ?padre)
      (not (vivo ?padre si))
      (herederos ?padre ?heredero))))))
```

Figura 7-15 Definición de los herederos.

Aunque no se mencione en el enunciado del problema, existen una serie de relaciones que podrían simplificar y clarificar los procesos de inferencia en el esquema de relaciones familiares dado. Por ejemplo, en la figura 7-13 se observa la simplificación realizada en la relación *tios* cuando se tiene en cuenta la definición de una nueva relación *hermanos*.

Respondiendo al apartado b), para definir los hermanos políticos de una persona tendremos en cuenta, según aparece reflejado en la **figura 7-14**, todas las relaciones

transitivas que puede abarcar dicha relación. Hemos utilizado la propiedad hermanos para simplificar la solución del problema.

```
(persona juan
  (nombre Juan)
  (apellido Ródriguez)
  (conyuge ana)
  (vivo si)
  (hijos (elena javier))
  (herederos (ana elena javier)))

(persona elena
  (nombre Elena)
  (apellido Ródriguez)
  (conyuge jose)
  (vivo si)
  (padres (juan ana))
  (hijos (jaime belen alba))
  (hermanos (jose))
  (herederos (jose jaime belen alba))
  (hermanos-políticos (eva)))

(persona jaime
  (nombre Jaime)
  (apellido Pérez)
  (vivo si)
  (padres (jose elena))
  (hermanos (belen alba))
  (tios (javier eva))
  (abuelos (juan ana))
  (herederos (jose elena belen alba)))
```

Figura 7-16 Definición de instancias de personas.

En lo que se refiere a la respuesta del apartado c) vamos a introducir una regla que satisfaga cada una de las cuatro circunstancias señaladas. La solución de este problema se muestra en la figura 7-15. Las dos primeras reglas son inmediatas. En la tercera se observa que, dado que las variables de las cláusulas se aplican como si estuvieran cuantificadas universalmente (sec. 7.5.2), la cláusula (not (vivo ?hijo si)) permite asegurarse de que no existe ningún hijo que esté vivo (recordemos la equivalencia lógica entre $\forall\neg$ y $\neg\exists$). En esta misma regla, la variable ?heredero es en realidad un nieto de ?x. Aunque no es necesario, también se podría haber definido

la relación nieto para simplificar el problema. En la última regla se vuelve a utilizar el predicado negación para garantizar la inexistencia de hijos o padres vivos.

Finalmente, nos queda por definir algunas de las instancias de la figura 7-2 que representen a cada uno de los tipos de personas existentes. Para ello, partiremos del supuesto de que todas las personas están vivas. Mostraremos una sola instancia para representar a cada uno de los tipos existentes (ver la figura 7-16).

Aunque no aparece de forma expresa en el enunciado, hemos supuesto que los padres y hermanos están entre los herederos de una persona cuando ésta no tiene una familia propia. De igual manera, tampoco hemos considerado herederos de una persona a sus padres cuando ésta tiene cónyuge e hijos vivos.

7.7 Problemas asociados

7.7.1 Apilar bloques

Vamos a describir uno de los dominios más utilizados en los problemas de planificación y de aprendizaje. En este tipo de problemas se parte de la existencia de un conjunto reducido de objetos con alguna propiedad que les permite relacionarse. Algunos ejemplos son: una serie de bloques y pilares que pueden acoplarse formando un arco, piezas con cavidades y salientes que pueden unirse para formar distintas figuras geométricas, cubos y objetos varios que pueden apilarse, etc. Nosotros hemos elegido dos tipos de objetos: cubos y mesas, con una única propiedad relevante, la posibilidad de ser apilados unos encima de otros.

7.7.1.1 Representación

Partimos de una serie de enunciados que reflejan el conocimiento sobre los objetos del dominio y sus relaciones. Un objeto físico x se puede apilar sobre otro cuando este último no es frágil o cuando el primero es menos pesado. Un objeto es más ligero que otro cuando el peso del primero es inferior al del segundo. Un objeto se considera frágil si su peso es inferior o igual a 5.

Un cubo es un objeto físico y tiene entre sus propiedades: el peso, la longitud, la altura y la anchura. El peso de un objeto físico es igual a la multiplicación de su volumen por su densidad. El volumen de un cubo es igual a su altura por su anchura y por su longitud. Por otro lado, se sabe que el peso de una mesa plegable es 5 (no se van a considerar las unidades de medida en ninguno de los valores del ejercicio).

En este dominio se parte de la existencia de los siguientes objetos: una mesa plegable, mesa1, de color marrón; un cubo de color verde, al que llamaremos cubo1, con una densidad de 5 y con una longitud, altura y anchura igual a 10; un segundo cubo de color azul, cubo2, con un peso de 4 que se encuentra inicialmente encima del cubo1.

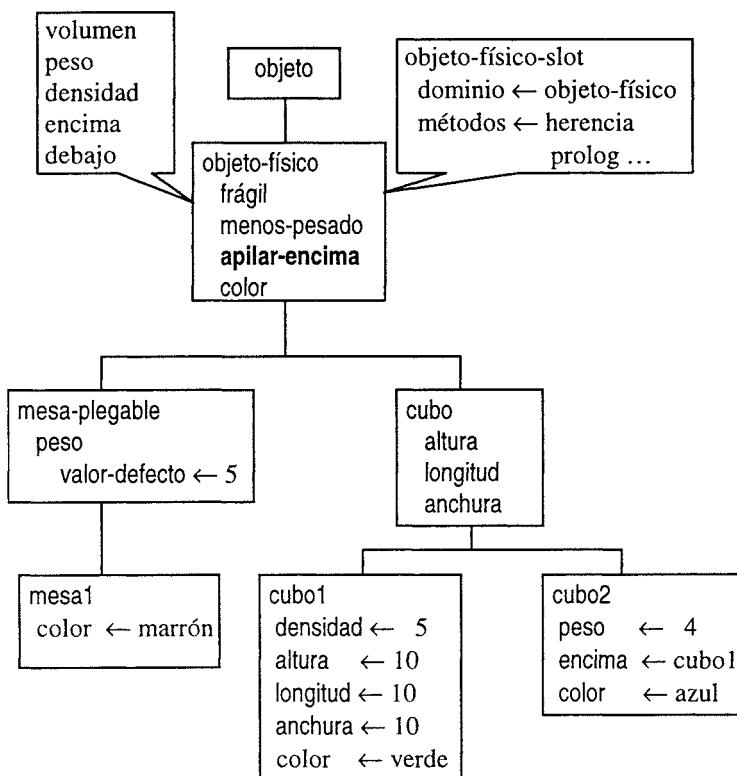


Figura 7-17 Jerarquía de relaciones entre objetos.

Cuando no se especifique un método concreto, cualquier atributo de un objeto físico se intenta inferir mediante la aplicación en secuencia de los siguientes métodos: herencia, prolog, pérdida de contexto y valor por defecto e inverso (explicados en 7.5.2).

Considerando la información disponible se pide representar todos los objetos y relaciones del dominio.

```
(objeto-fisico *sin-valor*
  (generalizaciones (objeto))
  (especializaciones
    (cubo mesa-plegable))
  (fragil *sin-valor*
    (generalizaciones (objeto-fisico-slot)
      (valores-possibles (t NIL))
      (metodos (prolog))
      (clausulas-prolog *sin-valor* (
        ((fragil?x ?valor) :-
          (peso ?x ?peso1)
          (eval (<= ?peso1 5) t))))))
  (menos-pesados *sin-valor*
    (generalizaciones (objeto-fisico-slot)
      (numero-valores n)
      (metodos (prolog))
      (clausulas-prolog *sin-valor* (
        ((menos-pesado ?x ?y) :-
          (peso ?x ?peso1)
          (peso ?y ?peso2)
          (eval (< ?peso1 ?peso2) t))))))
  (apilar-encima *sin-valor*
    (generalizaciones (objeto-fisico-slot))
    (numero-valores n)
    (metodos (prolog))
    (clausulas-prolog *sin-valor* (
      ((apilar-encima ?x ?y) :-
        (menos-pesado ?y ?x))
      ((apilar-encima ?x ?y) :-
        (not (fragil ?x)))))))
  (color *sin-valor*))
(objeto-fisico-slot *sin-valor*
  (generalizaciones (marco-slot))
  (especializaciones (fragil menos-pesado
    color apilar-encima altura volumen peso
    anchura longitud densidad encima debajo))
  (dominio (objeto-fisico))
  (metodos (herencia prolog perdida-contexto
    valor-por-defecto inverso))))
```

Figura 7-18 Definición de objeto-fisico.

- **Solución**

Como advertencia inicial queremos recalcar que en este ejercicio, al contrario de lo especificado en el apartado 7.6, no se utilizará la declaración (métodos (prolog)) para algunos de los *slots* con reglas prolog, con el fin de poder ilustrar la utilización conjunta de varios métodos de inferencia. En concreto, se ha dejado prolog como el segundo de los métodos aplicados por defecto. En primera instancia se aplicará herencia.

A partir del enunciado podemos completar un cuadro de relaciones entre los distintos objetos del dominio (véase la figura 7-17). Para completar la definición de los objetos y relaciones del dominio comenzaremos describiendo los marcos que definen los elementos más generales mencionados en el enunciado.

```
(mesa-plegable *sin-valor*
  (generalizaciones (objeto-fisico))
  (peso *sin-valor* (valor-por-defecto 5)))

(cubo *sin-valor*
  (generalizaciones (objeto-fisico))
  (especializaciones (cubo1 cubo2))
  (altura *sin-valor*
    (generalizaciones (objeto-fisico-slot)
      (valores-posibles numero)))
  (longitud *sin-valor*
    (generalizaciones (objeto-fisico-slot)
      (valores-posibles numero)))
  (anchura *sin-valor*
    (generalizaciones (objeto-fisico-slot)
      (valores-posibles numero))))
```

Figura 7-19 Definición de subclases de objetos-físico.

La entidad *objeto-físico* (figura 7-18) tiene la particularidad de tener un *slot*, *apilar-encima* (indica que el objeto que ocupa el valor del *slot* se puede apilar sobre el objeto que tiene dicho *slot*), que puede tener un *numero-valores n* (*indefinido*); es decir, es un **slot multivaluado**. Este *slot* se obtiene aplicando una regla con una disyunción (OR) en su antecedente (responde a la primera frase del enunciado). En lugar de representar explícitamente la disyunción OR en el antecedente de la regla, en nuestro formalismo repetimos la definición de dicha regla para cada una de las condiciones del OR. Esto es, si la disyunción de la regla la representamos

como $A \vee B \rightarrow C$, nosotros definiremos las reglas $A \rightarrow C$ y $B \rightarrow C$. El efecto es el deseado, ya que las reglas se aplican una a una hasta que se consiga inferir el valor requerido.

Para que una regla prolog se ejecute, se tienen que cumplir todos sus antecedentes.

También hemos definido una nueva entidad objeto-fisico-slot con el fin de asegurar el proceso de inferencia descrito en el enunciado. Tanto en las propiedades menos-pesados (multivaluado) como fragil se realiza una evaluación directa de una expresión Lisp que, en caso de ser cierta, devolvería el segundo argumento, en ambos casos la t indica cierto, con lo cual se cumpliría la cláusula correspondiente dentro de la regla.

```
(desidad *sin-valor*
  (generalizaciones (objeto-fisico-slot))

(volumen *sin-valor*
  (generalizaciones (objeto-fisico-slot))
  (dominio cubo)
  (clausulas-prolog *sin-valor* (
    ((volumen ?x ?vol) :-
       (altura ?x ?altura)
       (longitud ?x ?longitud)
       (anchura ?x ?anchura)
       (eval (* ?altura ?longitud ?anchura) ?vol)))))

(peso *sin-valor*
  (generalizaciones (objeto-fisico-slot))
  (clausulas-prolog *sin-valor* (
    ((peso ?x ?peso) :-
       (densidad ?x ?densidad)
       (volumen ?x ?volumen)
       (eval (* ?volumen ?densidad) ?peso)))))

(encima *sin-valor*
  (generalizaciones (objeto-fisico-slot))
  (inverso debajo))

(deabajo *sin-valor*
  (generalizaciones (objeto-fisico-slot))
  (inverso encima))
```

Figura 7-20 Definición independiente de slots de objetos-fisicos.

Una vez descrito el contexto más general de los objetos físicos, pasamos a definir dos de sus subclases: los cubos y las mesas plegables (véase la figura 7-19). Para ello, hemos definido las propiedades indicadas en el enunciado y les hemos asignado la generalización **objeto-fisico-slot** para que en todas ellas se lleve a efecto el mismo proceso de inferencia.

Por otro lado, en la figura 7-20 introducimos una nueva técnica de definir *slots* de entidades. En lugar de tener que describir cada propiedad (por ejemplo, frágil en la figura 7-18) dentro de la entidad a la cual describe (la entidad descrita es el dominio y el rango serían los valores que pudiera tomar) se pueden **definir los slots separadamente**. Esto permite, por un lado, descargar y hacer más eficiente la representación de los objetos del dominio que, como se puede comprobar en este y otros ejercicios, puede llegar a ser demasiado extensa, y, por otro, asignar la definición de un *slot* a diversas entidades sin tener que repetir su definición.

```
(cubo1 *sin-valor*
  (generalizaciones (cubo))
  (densidad 5)
  (altura 10)
  (longitud 10)
  (anchura 10)
  (color verde))

(cubo2 *sin-valor*
  (generalizaciones (cubo))
  (encima cubo1)
  (peso 4)
  (color azul))

(mesa1 *sin-valor*
  (generalizaciones (mesa-plegable))
  (color marron))
```

Figura 7-21 Definición de los objetos del dominio.

Los *slots* son también entidades que pueden definirse independientemente.

Para garantizar el funcionamiento correcto cuando el *slot* se define separadamente es necesario asociar dicho *slot* a la entidad o entidades que corresponda mediante la utilización del *subslot* predefinido **dominio** (en negrita en la figura 7-20). También se requiere emplear el *subslot* **generalizaciones** (igualmente resaltado en la figura)

para enmarcar convenientemente el *slot* en la jerarquía de *slots*. Conviene resaltar que, siguiendo las indicaciones del enunciado, la definición dada de volumen está asociada a la entidad cubo, mientras que la propiedad peso es más general y está asociada a cualquier objeto-físico.

Finalmente, sólo nos queda por definir los objetos más concretos del dominio, cubo1 y cubo2. El primero tiene una descripción bastante completa y del segundo destaca la cualidad de estar situado debajo del primero (tal y como se muestra en la figura 7-21).

7.7.1.2 Inferencia

A partir del problema enunciado en la sección anterior se pide describir el proceso de inferencia realizado para responder a las siguientes cuestiones: ¿cuál es el peso del cubo1? y ¿qué objetos se pueden apilar sobre la mesa?

Para responder a estas cuestiones se puede optar por dos alternativas. La primera, resaltar exclusivamente las preguntas principales generadas indicando su orden. La segunda, que es la que aquí vamos a utilizar, mostrar la mayoría de las direcciones generadas en el proceso de inferencia.

- **Solución**

Responderemos a las preguntas ordenando en secuencias las direcciones a las que se accederá en cada caso. Para ello, optaremos por describir en niveles de profundidad sucesivos las preguntas realizadas, de tal forma que los problemas que sirvan para responder a otros problemas aparezcan enmarcados dentro de aquéllos. Para realizar el desarrollo del proceso de inferencia seguiremos las indicaciones de las reglas y los métodos descritos en el problema anterior (sec. 7.7.1.1).

En lo que se refiere a la cuestión ¿cuál es el peso del cubo1?, tal y como se ha explicado anteriormente (sec. 7.5.2), lo primero que hace el sistema es preguntar por el valor del *slot* correspondiente y, en caso de no tener ningún valor asignado, invocar secuencialmente los métodos de inferencia indicados para dicho *slot*. El proceso de inferencia se describe globalmente en la figura 7-22; pero, antes de abordar su desarrollo, conviene precisar algunas cuestiones puntuales sobre la notación empleada. Las líneas que contienen expresiones como ">5 dar (cubo densidad)" indican que en un determinado nivel, en este caso el 5, se establece una nueva submeta para intentar avanzar en el cálculo del valor de un *slot* de un nivel anterior (inferior a 5). De forma análoga, las líneas que tienen la forma "<5 dar (10(((cubo1 altura))))" indican que el valor 10 es la respuesta a la pregunta hecha en el nivel 5 sobre el *slot* (cubo1 altura).

```
(> (dame-valor peso cubo1)
>1 dar (cubo1 peso)
>3 dar (cubo peso)
>5 dar (objeto-fisico peso)
>7 dar (objeto-fisico densidad)
<7 dar (*infer-sin-valor* (((objeto-fisico densidad))))
<5 dar (*infer-sin-valor* (((objeto-fisico peso))))
>5 dar (cubo densidad)
<5 dar (*infer-sin-valor* (((cubo densidad))))
<3 dar (*infer-sin-valor* (((cubo peso))))
>3 dar (cubo1 densidad)
<3 dar (5 (((cubo1 densidad))))
>3 dar (cubo1 volumen)
>5 dar (cubo volumen)
>7 dar (cubo altura)
<7 dar (*infer-sin-valor* (((cubo altura))))
<5 dar (*infer-sin-valor* (((cubo volumen))))
>5 dar (cubo1 altura)
<5 dar (10 (((cubo1 altura))))
>5 dar (cubo1 longitud)
<5 dar (10 (((cubo1 longitud))))
>5 dar (cubo1 anchura)
<5 dar (10 (((cubo1 anchura))))
<3 dar (1000 (((cubo1 volumen))))
<1 dar (5000 (((cubo1 peso))))
5000
```

Figura 7-22 Inferencia del peso del cubo1.

Al preguntar por (cubo1 peso), al igual que ocurre con cualquier otro slot cuyo valor no existe de antemano, lo primero que se hace es preguntar por el *subslot* (cubo1 peso **métodos**), que en este caso es heredado del slot (objeto-fisico-slot métodos), entre cuyos valores se encuentra aplicar las reglas prolog existentes. La dirección (objeto-fisico-slot métodos) se genera debido a que, ante cualquier pregunta, si no hay ningún valor ni método concreto asociado, se aplica la secuencia de métodos de inferencia que por omisión se han definido en el sistema (recordar de nuevo la sección 7.5.2), siendo herencia el primer método aplicado. Además, en este caso se ha declarado la secuencia de métodos aplicables en la propiedad métodos de la entidad objeto-fisico-slot, tal y como se resalta en la figura 7-18. Por tanto, tal y como queda resultado en la figura 7-22, se llegan a probar dos métodos. En primera instancia se aplica herencia, pero no da ningún resultado positivo (denotado por *infer-sin-valor*). En segundo término se prueba prolog

(figura 7-18), produciéndose entonces el valor deseado, tal y como vamos a describir a continuación. La aplicación de ambos métodos se indica mediante las líneas en negrita de la figura 7-22.

La descripción de la figura 7-22 contiene las preguntas más destacadas del proceso de inferencia. Al aplicar herencia al problema (cubo1 peso) se genera la dirección (cubo peso), para la cual no existe un valor concreto; por tanto, siguiendo con la aplicación de herencia, se pregunta de nuevo por (objeto-físico peso). Ahora se da por concluido el proceso de herencia ya que el dominio de la relación peso es objeto-físico. En este caso, aunque tampoco existe un valor asignado para esta dirección, aplicando de nuevo los métodos de inferencia por defecto (esta secuencia no se ha incluido para no complicar aún más la figura) se pregunta por la dirección (peso cláusulas-prolog). Utilizando las cláusulas correspondientes, se comienza preguntando por (objeto-físico densidad) y se termina de aplicar herencia sin poder inferir ningún valor, tal y como se indica en la línea "**<3 dar (*infer-sin-valor* (((cubo peso))))**".

A partir de ese momento se busca en las direcciones asociadas a las cláusulas prolog que definen la solución del problema (cubo1 peso), comenzando entonces por preguntar por (cubo1 densidad). El resto del proceso consiste en seguir buscando las demás cláusulas de dicha regla. Después de preguntar por su densidad se lanza el proceso de inferencia correspondiente a la propiedad volumen. Para ello, siguiendo un proceso análogo al descrito para el peso, se terminan aplicando las reglas de prolog correspondientes, lo que culmina con la recuperación de las propiedades: altura, longitud y anchura. Devolviendo finalmente el valor asociado al peso del cubo1, a la vez que se establece el nuevo hecho (peso cubo1 5000) y se crea una explicación del proceso realizado en la dirección (peso cubo1 explicacion). Esta explicación contiene todos los *slots* de los cuales depende el valor inferido (no vamos a entrar aquí en analizar el contenido de esta dirección).

Para responder a la pregunta ¿qué objetos se pueden apilar sobre la mesa1?, hay que plantear el problema (apilar-encima mesa1). En la figura 7-23 se presenta un resumen del proceso de inferencia realizado. Básicamente, se trata de obtener, según se indica en las cláusulas prolog del slot apilar-encima (ya que aparece explícitamente en la figura 7-18 el método prolog asignado a este slot), todos aquellos objetos que sean menos-pesados que mesa1. Para ello, siguiendo el orden de aplicación de métodos establecido, se accede en primer lugar a la definición prolog del slot peso. Al no poderse inferir ningún valor a través de dicha definición (comprobar cómo se realiza la ascensión en las entidades siguiendo el método herencia), se termina aplicando el método valor-por-defecto (aparece resaltado

en la figura) que busca el valor por omisión que poseen todas las mesas plegables. Una vez concluido este proceso, la obtención del peso del resto de los objetos del dominio es casi inmediata, dado que el peso del cubo1 ya se había inferido previamente (figura 7-22). Lo más interesante es fijarse que al preguntar por (*menos-pesados ?y ?x*) (recordar la figura 7-18) el sistema busca todos aquellos objetos (dominio de la relación) cuyo *slot* sea *menos-pesados*; es decir, todos aquéllos que sean *objeto-físico*.

```
> (dame-valor apilar-encima mesa1)
>2 dar (mesa1 apilar-encima prolog)
  >4 dar (mesa1 menos-pesados prolog)
    >6 dar (mesa1 peso prolog)
    >8 dar (mesa1 volumen prolog)
    <8 dar (*infer-sin-valor* (((mesa1 volumen prolog))))
      >10 dar (mesa-plegable volumen prolog)
      <10 dar (*infer-sin-valor*
                    (((mesa-plegable volumen prolog))))
      >12 dar (objeto-fisico volumen prolog)
      <12 dar (*infer-sin-valor*
                    (((objeto-fisico volumen prolog))))
    <6 dar (*infer-sin-valor* (((mesa1 peso prolog))))
      >8 dar (mesa-plegable peso prolog)
      <8 dar (*infer-sin-valor*
                    (((mesa-plegable peso prolog))))
      >10 dar (objeto-fisico peso prolog)
      <10 dar (*infer-sin-valor*
                    (((objeto-fisico peso prolog))))
    >8 dar (mesa-plegable peso valor-por-defecto)
      <8 dar (5 (((mesa-plegable peso valor-por-defecto))))
    >6 dar (cubo2 peso prolog)
    <6 dar (4 (((cubo2 peso prolog))))
    >6 dar (cubo1 peso prolog)
    <6 dar (5000 (((cubo1 peso prolog))))
    <4 dar ((cubo2) (((mesa1 menos-pesados prolog))))
    <2 dar ((cubo2) (((mesa1 apilar-encima prolog))))
(cubo2)
```

Figura 7-23 Inferencia de los objetos apilables sobre el mesa1.

7.7.2 Revisión del problema "apilar bloques"

7.7.2.1 ¿Qué pretende este problema?

Al igual que en el resto de los problemas del presente capítulo, los objetivos de este ejercicio tienen una doble naturaleza. Por un lado, reflejan una tarea concreta en un dominio dado; en este caso: *representar objetos físicos estructurados* —como las mesas y los cubos— con relaciones entre ellos e *ilustrar el proceso de inferencia* asociado a la tarea "apilar un objeto sobre otro". Por otro parte, se pretende exemplificar una serie de problemas genéricos que se presentan en la mayoría de las situaciones reales. En concreto, este ejercicio incluye los siguientes:

1. **El problema de la representación:** Partiendo de una descripción del conocimiento del dominio se requiere:
 - a) Extraer las entidades relevantes en función de los objetivos o, lo que es lo mismo, eliminar descriptores innecesarios.
 - b) Detectar inconsistencias y omisiones.
 - c) Seleccionar el formalismo de representación más adecuado y las opciones concretas dentro de éste. Por ejemplo, en el problema de "apilar bloques" se insiste en la conveniencia de separar la definición de *slots* que vayan a ser usados por diferentes entidades.
 - d) Representar el conocimiento dado —la denominada *base de conocimientos inicial*, constituida por las distintas entidades y las relaciones entre éstas.
 - e) Definir y asignar los métodos de inferencia alternativos —o su combinación— para cada problema. En el formalismo adoptado se distinguen métodos aplicados *por omisión* (si no se asigna explícitamente un método para una entidad) y *por petición* (deben ser expresamente asignados).
2. **El problema de elegir la notación más oportuna:** Dentro de los distintos formalismos de representación (lógica, reglas, redes y marcos) se pueden adoptar diferentes notaciones, tal y como se puede comprobar en los problemas del presente texto. La elección de una u otra depende de su adecuación al problema, de la propia experiencia del interesado y, sobre todo, de la *herramienta utilizada para su implementación*. Por ejemplo, es muy ilustrativa la comparación de la representación adoptada en el ejercicio 6.8 —más propia de lenguajes de programación orientados a objetos— con la

utilizada en los problemas de este capítulo que es característica de lenguajes de manipulación simbólica.

3. **El problema de integrar métodos de inferencia distintos:** En el esquema de representación que se propone en este capítulo cualquiera de los métodos de inferencia utilizados tiene por objeto señalar las direcciones donde se puede encontrar el valor del *slot* solicitado. Por tanto, su integración es inmediata; no importa la secuencia de pasos asociados a cada método (reglas, funciones, conocimiento aprendido, ...) todos ellos revierten en accesos y asignaciones de valores a *slots* de entidades. La integración de métodos se realiza a través de la definición del *subslot* *metodos*. Por ejemplo, en el proceso de inferencia descrito en la figura 7-22 se aprecia la combinación de herencia y prolog para inferir el valor de un *slot*.
4. **El problema del acceso al conocimiento utilizado:** Aunque separemos este aspecto para destacarlo, es en realidad una de las claves del problema de la representación del conocimiento. En distintos formalismos caben dos opciones: representar el conocimiento necesario para resolver un problema o compilarlo y utilizarlo directamente. En la presente exposición veremos la posibilidad de, bien utilizar funciones definidas aparte (por ejemplo, escritas en Lisp) o bien representar las acciones de dichas funciones en las cláusulas de las reglas asociadas a los distintos *slots* (tal y como se refleja al contrastar la figura 7-7 con la figura 7-8). Esta segunda opción permite, por ejemplo, comparar el conocimiento aprendido con el descrito inicialmente (véanse las diferencias entre la figura 7-41 y la figura 7-45). Una u otra opción depende de las necesidades concretas del problema. En cualquier caso, la compilación del conocimiento garantiza una mayor eficiencia.
5. **El problema de la cantidad de conocimiento almacenado:** Otro problema derivado de la representación adoptada es la cantidad de entidades que van a ser definidas inicialmente y almacenadas como resultado de la ejecución del sistema. En este caso, cuantos más *slots* se definan y almacenen, menor número de inferencias habrá que realizar en el cálculo de nuevos *slots*. Suponiendo que viéramos al sistema como una "tela de araña en construcción". Al principio, sólo habría unas pocas direcciones asignadas, que indican por dónde se puede expandir el sistema (la tela de araña). Según el sistema (la araña) vaya teniendo interés en crecer en alguna dirección (se le pregunte), éste empieza a crecer en la misma. Si no se almacena el camino recorrido (la

tela construida) habría que volverlo a recorrer en sucesivas demandas. Por tanto, *la conveniencia o no de guardar una determinada entidad depende del número de peticiones que la soliciten y de la eficiencia requerida tanto para su almacenamiento (cantidad de memoria disponible) como para su gestión* (cuanto más conocimiento haya almacenado más se tardará en comprobar cuál es el pertinente). Por ejemplo, en el problema 5.4 la definición de la propiedad hermanos puede simplificar el cálculo de otras relaciones del dominio.

7.7.2.2 ¿De qué tipo de problemas es representativo?

El dominio de este ejercicio es uno de los más utilizados en los problemas de planificación y de aprendizaje. En este tipo de problemas se parte de la existencia de un conjunto reducido de objetos con alguna propiedad que les permite relacionarse. Algunos ejemplos son: una serie de bloques y pilares que pueden acoplarse formando un arco, piezas con cavidades y salientes que pueden unirse para formar distintas figuras geométricas, cubos y objetos varios que pueden apilarse, etc. Nosotros hemos elegido dos tipos de objetos: cubos y mesas, con una única propiedad relevante, la posibilidad de ser apilados unos encima de otros.

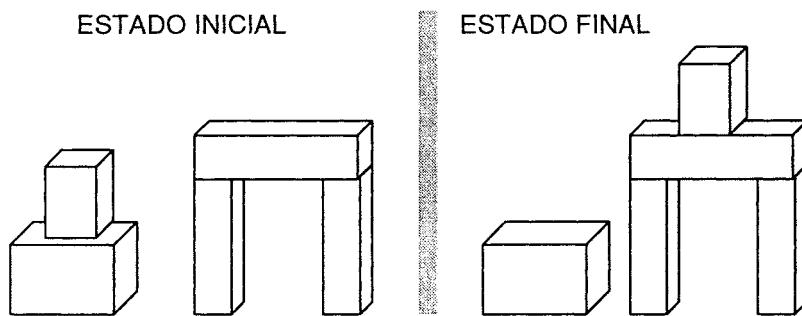


Figura 7-24 Apilar bloques.

7.7.2.3 ¿Qué dificultades puede haber encontrado?

A. En el enunciado

Aunque en este caso resulte demasiado obvio (véase la figura 7-24), con el fin de aclarar la situación descrita en el enunciado, muchas veces resulta útil representar gráficamente los objetos y las situaciones descritas en el dominio y, cuando su escenificación no sea tan inmediata, se puede sustituir por diagramas que representen

el flujo de acciones y entidades que participen en el proceso (como se muestra en la figura 7-28).

En cuanto al formato del enunciado en sí, éste coincide con el resto de los problemas del capítulo, en los que, al contrario de lo que ocurre en la mayoría de los ejercicios de capítulos precedentes, no se especifica una descripción breve de una situación dada, sino que se describen un conjunto de enunciados que conforman el conocimiento del dominio. Al igual que ocurre generalmente en las situaciones reales, se ha introducido algún elemento superfluo que no influye en la realización de la tarea objeto (por ejemplo, el color de los distintos tipos de bloques).

B. En el planteamiento previo ante el problema

Antes de representar los objetos y relaciones del dominio en el formalismo dado, convendría representar en un grafo dichas entidades, tal y como se muestra en la figura 7-17. En este punto, y considerando la organización del capítulo, el lector debería ser capaz de representar, utilizando el formalismo descrito en el apartado 5.3 ya aplicado en 5.4, el objeto mesa-plegable con el atributo color y el objeto cubo con los atributos: altura, longitud, anchura y densidad, ya que su asignación se declara expresamente en el planteamiento. En el enunciado aparecen otros atributos significativos, como el peso y densidad y el volumen, asociados a la entidad objeto-físico. En principio, dado que no se dan definiciones específicas para el peso y la densidad, no parece haber ningún dato significativo que haga dudar sobre la conveniencia de definir objeto-físico con dichos atributos. Por el contrario, la definición de volumen dada es la que caracteriza a los cubos, pero no a cualquier objeto. De igual forma, el resto de las propiedades atribuidas a los cubos no pueden extrapolarse al resto de los objetos físicos.

También se especifica que "un objeto físico x se puede apilar sobre otro cuando este último no es frágil o cuando el primero es menos pesado". Esta disyunción es importante y puede ocasionar que inicialmente se plantee alguna duda sobre su realización. Sin embargo, al explicar el método prolog en el apartado 7.5.2 se resaltó el hecho de que las reglas de este tipo se aplican en el orden en que están escritas siempre y cuando se cumplan todos los antecedentes de las mismas. Por tanto, la disyunción requerida se puede traducir en la definición de dos reglas consecutivas, cada una con un componente de la misma.

Por otro lado, se declara que "un objeto es más ligero que otro cuando el peso del primero es inferior al del segundo y un objeto se considera frágil si su peso es inferior o igual a 5". Es decir, se dan las dos definiciones requeridas para poder inferir el valor del slot requerido, al que hemos denominado apilar-encima. Este slot es una

relación (definida en la figura 7-18). Quizá convenga recordar la naturaleza de cualquier relación.

Las relaciones se representan mediante slots que asocian los valores del conjunto del dominio con el conjunto del rango.

Una relación está formada por el conjunto de pares ordenados que la verifica. En este problema la relación apilar-encima representa las parejas de **pares ordenados** (x , y) en las que y se puede colocar sobre x .

Finalmente, se incluye una descripción de tres objetos concretos del dominio denominados: cubo1, cubo2 y mesa1, que serán las instancias de las entidades definidas previamente, por lo que simplemente tendrán valores asignados para algunas de las propiedades ya declaradas.

C. En las opciones de representación adoptadas

Quizá pueda sorprender en un principio la estructuración excesiva de las entidades del dominio. Las razones de **separar la definición de ciertos slots** (por ejemplo, densidad, volumen y peso) ya han sido explicadas en la solución propuesta. No obstante, dada la importancia de esta técnica, queremos insistir en su conveniencia cuando se cumplan algunas de las siguientes condiciones:

1. *Evitar aglutinar en una sola descripción simbólica toda la información referida a una entidad compleja, descargando y haciendo más eficiente la representación de los objetos del dominio.*
2. *Asignar a una serie de entidades un conjunto de métodos de inferencia o de propiedades compartidas.*
3. *Utilizar la definición del mismo slot en diversas entidades, evitando así su repetición en todas ellas.*

La situación descrita en el segundo apartado se ilustra claramente con el conjunto de métodos aplicados por omisión para todos los atributos de objeto-físico (asignados a la propiedad metodos en la figura 7-18); es decir, todas las especializaciones de objeto-físico-slot.

Con respecto al tercer punto conviene precisar que, cuando las entidades tienen un mismo antecesor en una jerarquía de objetos, puede omitirse la formulación separada de la propiedad compartida. En este caso se podría optar por incluir la definición en dicho antecesor.

En cualquier caso, tal y como se muestra en la figura 7-20, se utilizan los *subslots* generalizaciones y dominio para determinar el rango de aplicabilidad de los *slots* separados. Así, al especificar que el dominio de volumen es cubo se asegura, por ejemplo (véase la figura 7-22), que el proceso de herencia de (cubo1 volumen) accede a la definición de volumen en cuanto se alcance (cubo volumen). La forma genérica de indicar la separación de *subslots* se representa en la figura 7-25 (tal y como se realiza con las propiedades de la figura 7-20).

```
(subslot *sin-valor*
  (dominio (slot-de-origen))
  (.....))
```

Figura 7-25 Definición de *slots* separados.

En lo que se refiere a la propiedad apilar-encima (véase la figura 7-18), se ha introducido el *subslot* numero-valores para especificar si es o no un *slot multivaluado*. En los casos en que dicho número sea n y haya varias reglas prolog asignadas, el sistema intenta obtener todos los valores posibles aplicando todas las reglas en lugar de parar tan pronto como se infiere un valor del *slot*. En otros casos, en los que el *slot* no fuera multivaluado habría que tener en cuenta el orden de ejecución de las reglas (*se ejecutan primero las que se encuentran más cerca del consecuente*).

D. En la notación utilizada

En la aplicación de las reglas prolog del *slot* apilar-encima se introduce una cláusula negativa (not(fragil ?x)). Debido al funcionamiento del proceso de equiparación de la variable ?x con los objetos de la base de conocimientos que le puedan ser asignados (los procesos de equiparación o comparación de patrones se explican en la mayoría de los textos genéricos referenciados al comienzo del apartado 7.4), se debe entender que la expresión negada está cuantificada universalmente ($\forall x (\text{not}(\text{fragil } x))$). Esto quiere decir que se va a intentar equiparar dicha variable con todos los objetos posibles. Por tanto, si el resultado de la equiparación de (fragil x) fuera nulo, entonces el valor de la expresión (not(fragil x)) sería cierto. O lo que es lo mismo, se estaría comprobando que no hay ningún objeto que sea frágil. De hecho, existe una equivalencia lógica entre las expresiones $\forall x (\text{not}(\text{fragil } x))$ y $(\text{not}(\exists x (\text{fragil } x)))$ (la lógica, como formalismo de representación e inferencia, también se introduce en los textos genéricos del campo, destacando su utilización en [Ginsberg, 1993]).

Una cuestión de notación que también puede llamar la atención es la inclusión de cláusulas que, al igual que realizan los predicados de distintos lenguajes (por ejemplo, Lisp), devuelven un valor "cierto" si se cumple una determinada condición. Este es el caso de `(eval (< ?peso1 ?peso2) t)` en la **figura 7-18**. De esta forma, como se explica en la sección 7.7.1.1, si se cumple que el peso del primer objeto es menor que el del segundo, se devuelve "t" —que en el formalismo elegido significa "cierto", evaluándose entonces el resto de las cláusulas que componen la regla correspondiente.

La comprobación asociada a las cláusulas que incluyen la expresión `eval` nos permitiría, por ejemplo, inicializar los valores de un *slot* a un valor elegido de antemano. Bastaría con crear una regla con una sola cláusula que se cumpla siempre. Supongamos que queremos dar, tal y como se muestra en la Figura 7-26, un valor 10 a la propiedad altura de los cubos grandes en el dominio.

```
(cubo-grande *sin-valor*
  (generalizaciones (cubo))
  (altura *sin-valor*
    (generalizaciones (objeto-fisico-slot)
      (metodos (prolog))
      (clausulas-prolog *sin-valor* (
        ((altura ?x 10) :-
          (eval t t)))))))
```

Figura 7-26 Definición de *subslots* separados.

Esta misma técnica se ha aplicado en un caso concreto para hacer que `reunion` sea el valor por omisión asignado al tipo de eventos del calendario (véase la figura 7-30).

E. En una etapa concreta del desarrollo

Las principales dificultades se pueden encontrar en los procesos de inferencia descritos en la figura 7-22 y en la figura 7-23. En la primera se muestra como la combinación de los métodos de `herencia` y de `prolog` pueden servir para hallar el valor buscado. Si se intenta seguir el proceso atendiendo a todos los *slots* utilizados, el análisis resulta un tanto complejo. En su lugar, resulta mucho más sencillo realizar el seguimiento en niveles de complejidad creciente. Así, la pregunta inicial se refiere al cálculo de `(cubo1 peso)`. Una vez se ha comprobado que dicha entidad no tiene un valor asociado, se verifica —al igual que se hace siempre en este formalismo— si existen métodos asignados para su cálculo. Se aplican entonces los métodos asociados al *slot* `peso`. El resto del proceso consiste en aplicar en secuencia dichos métodos. Se

comienza utilizando herencia y, dado que no se alcanza un valor concreto (representado por **infer-sin-valor**), se utiliza prolog con la consiguiente ejecución de las reglas existentes, que en este caso indican que el peso es igual al producto del volumen por la densidad. Este primer nivel de inferencia se resalta en negrita en la figura 7-22. En el siguiente nivel se vuelven a considerar las posibilidades asociadas a cada *slot* demandado. Por ejemplo, para determinar la definición de peso es necesario, aplicando el método por defecto herencia, acceder a la entidad (*objeto-fisico peso*). Por otro lado, para calcular el volumen de *cubo1* se aplican en secuencia herencia sin éxito y prolog con el valor 1000 finalmente inferido.

El proceso de inferencia que describe cómo se hallan los objetos que se pueden apilar sobre la mesa plegable *mesa1* sigue el mismo tipo de planteamiento (véase la figura 7-23). En este caso, al aplicar la regla prolog que lleva asociada la cláusula (*menos-pesados ?y ?x*) se accede a la definición de *menos-pesado* y se devuelven asociados a la variable *?y* todos aquellos objetos en los que la relación (*< ?peso-y ?peso-x*) sea cierta. Sin embargo, este proceso no es tan inmediato y antes de esto requiere un segundo nivel de comprobaciones en las que se intenta inferir por herencia, siguiendo el camino de la jerarquía existente (*mesa1, mesa-plegable, objeto-fisico*) el valor asociado a (*mesa1 menos-pesado*), siendo fallido este intento inicial (**infer-sin-valor**).

7.7.3 Agenda personalizada: CAP

Situémonos en el contexto de un profesor de universidad muy ocupado, que diariamente tenga que realizar el siguiente tipo de tareas: organizar los cursos que debe impartir o aquéllos a los que desea asistir, reorganizar su agenda debido a la celebración de una reunión extraordinaria del comité de admisión de alumnos graduados, organizar en las fechas y en las salas (o aulas) disponibles las próximas reuniones de carácter periódico, atender a una visita inesperada de un grupo de empresas influyentes, atender una petición de una reunión con el decano de su facultad, enviar mensajes de correo electrónico para recordar a los asistentes a alguna reunión próxima el lugar y la fecha de su celebración, etc.

Consideremos que se ha construido un sistema que realiza las funciones de un agente⁹ basado en el conocimiento que sirve para anotar en una agenda con forma de

⁹ Entendiendo por *agentes* a los sistemas de software inteligentes que combinan algunas de las capacidades siguientes: razonar basándose en casos previos, aprender del propio comportamiento y del entorno, entender las intenciones del usuario y adaptarse a sus preferencias, interactuar en un espacio físico concreto, gestionar espacios virtuales de recursos (por ejemplo, la red Internet), etc.

Calendario todos los eventos (reuniones, cursos, visitas, etc.) a los cuales tenga que asistir nuestro usuario potencial. Imaginemos además que dicho sistema, **denominado CAP** (proviene de Aprendiz del Calendario, o *Calendar Apprentice*), fuera capaz de agilizar la gestión de tales eventos, sugiriendo lugares y fechas de celebración, valorando adecuadamente su urgencia relativa, el tipo de asistentes (alumnos de doctorado, profesores, personas ajena a la universidad, etc.), la disponibilidad de una determinada sala de reuniones, el grado de ocupación asociado al período del año en que se encuentra enmarcado el evento, etc. Supongamos igualmente que el sistema fuera capaz de reorganizar las reuniones previstas en función de una situación inesperada —por ejemplo, una situación familiar que requiera la asistencia urgente del interesado— y vayamos más allá considerando la posibilidad de que se pudiera recordar automáticamente —a través del correo electrónico— a los asistentes a una reunión próxima el lugar y la fecha de celebración.

Finalmente, supongamos que dicho sistema se ha construido incluyendo técnicas de aprendizaje que le permiten aprender a adaptarse a los gustos de su dueño (o usuario), de tal forma que termina adquiriendo el conocimiento que subyace en las decisiones tomadas previamente por dicho usuario. Los sistemas así construidos se denominan *sistemas aprendices*.

Sistemas aprendices: Consejeros interactivos basados en el conocimiento, que pueden extraer nuevo conocimiento a través de la observación directa y el análisis de los pasos en la búsqueda de la solución proporcionados por el usuario a lo largo del funcionamiento regular del sistema [Mitchell *et al.*, 1985]

Para el desarrollo de este ejercicio, después de concretar el planteamiento del problema, vamos a formular una serie de enunciados que permitan solucionar los aspectos diferentes en los que se puede dividir esta aplicación. Cada uno de estos enunciados servirá para concretar las estructuras de representación de los elementos estáticos (entidades y atributos) y dinámicos (reglas y procedimientos de inferencia) del dominio. Finalmente, para ilustrar la utilidad de plantear problemas bajo la perspectiva del aprendizaje frente a la formulación requerida por las técnicas básicas de inferencia descritas previamente (sec. 7.5.2), introduciremos el planteamiento asociado a la resolución de alguna de las tareas del Calendario mediante técnicas de aprendizaje (sec. 7.7.3.5). Como es lógico, la explicación de los rudimentos de las técnicas de aprendizaje quedan fuera de los objetivos del presente capítulo (el lector interesado puede acudir a la bibliografía recomendada sobre este tema al comienzo del apartado 7.2). Nuestro único objetivo aquí es ejemplificar su integración en el esquema general de inferencia del sistema.

7.7.3.1 Planteamiento del problema

En este apartado nos vamos a centrar en describir los elementos del dominio del Calendario [Dent *et al.*, 1992]. Para ello, antes de plantear enunciados concretos, vamos a comentar las situaciones que pudieran surgir y de ahí sacar los elementos más significativos.

Partiremos de una situación concreta que nos puede ayudar a comprender el funcionamiento del sistema. Supongamos que Ángel quiere reunirse con Tomás (el dueño de la agenda) esta semana. Para ello, llama por teléfono a la secretaria de este último. En el transcurso de la conversación la secretaria, con ayuda del sistema, tiene que decidir el siguiente conjunto de cuestiones:

- ¿Cuándo debería celebrarse la reunión?
- ¿En qué sala sería conveniente celebrarla?
- ¿Cuál es su duración estimada?
- ¿Debería hacerse hueco en la presente semana para esta reunión?
- ¿Sería mejor retrasarla una semana o acortarla para poder atenderla lo antes posible?
- ¿Debería confirmar con Tomás esta reunión?
- ¿Convendría reservar la sala donde tendrá lugar?
- ¿Debería enviar un mensaje de correo electrónico que le recordara a Ángel la reunión?

La función de CAP es ayudar a responder a estas cuestiones, o incluso automatizar alguna de ellas (por ejemplo, mandar una advertencia a los participantes en una reunión). Para ello, el sistema debería considerar el conocimiento sobre la situación actual (época del año, grado de ocupación, fiestas venideras, etc.) y sobre la forma en que se han resuelto conflictos parecidos en ocasiones precedentes. Nuestro objetivo será representar los elementos mínimos necesarios para resolver algunos de estos problemas.

En la práctica, partimos de una situación en la que una secretaria está utilizando el Calendario para atender las demandas de reuniones, o eventos de otros tipos, en los que tiene que participar el "dueño" de la agenda.

04/29/1994 Reunion del Claustro de la UNED

HORA	Lunes 4-25	Martes 4-26	Miercoles 4-27	Jueves 4-28	Viernes 4-29
8:00		* Carolina Fc235	* Marron Fc235	* Carolina Fc235	* Fuera!
8:30					
9:00	* Mira Fc226		* Lisp-Cur Fc09		V
9:30		* Angel Fc235	ISP: boticario		V
10:00	* Dormido Fc233		V	* Diez Fc235	V
10:30			V		V
11:00	V		V		V
11:30	V		V		V
12:00		* Angeles Fc235	V		V
12:30			V		V
1:00					V
1:30					V
2:00	* Redes-Sem Fc09				V
2:30	ISP: mira				V
3:00	V				V
3:30	V				V
4:00	V			* Mira Fc226	V
4:30	V				V
5:00	V			V	V
5:30	V			V	V
6:00					

HORA	4-25	4-26	4-27	4-28	4-29
------	------	------	------	------	------

Tipo solicitud: C-A [reunion]

Figura 7-27 Interfaz de usuario de CAP.

La interfaz gráfica del sistema consiste en una pantalla de edición de la agenda, tal y como muestra la figura 7-27, en la que prima la visualización de los elementos más representativos de cada una de las anotaciones (seleccionándolas se puede ver la anotación en su totalidad). Como ejemplo, se muestra la activación del comando C-a (también se puede activar pulsando simplemente la tecla <a>), que está asociado a la acción añadir una nueva solicitud. El valor por defecto aconsejado para el tipo de solicitud es reunion. Presionando la <barra de espacios> se visualizaría el resto de los tipos de eventos posibles.

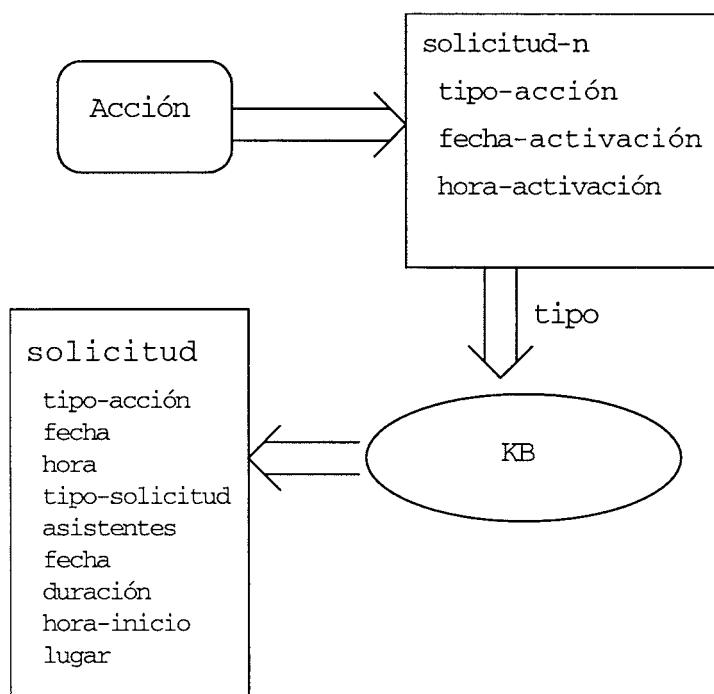


Figura 7-28 Flujo de acciones en CAP.

Algunos de los comandos más característicos del interfaz son (véase la figura 7-27); referidos a los eventos posibles: añadir, borrar, mover, cambiar la información, copiar a otro día, ampliar la información, visualizar el contenido, añadir alguna nota; referidos a los acontecimientos próximos: introducir uno nuevo, listar los existentes; referidos a la visualización de pantallas: mostrar una determinada semana (crear su ventana o mostrar la existente), abandonar la semana mostrada; referidos a la propia gestión de los datos introducidos: salvar la pantalla de una

semana, almacenar el estado actual del Calendario, guardar la sesión de comandos como ejemplos de entrenamiento (para las técnicas de aprendizaje aplicadas más adelante). Este último tiene una gran trascendencia para facilitar el diseño de los experimentos de aprendizaje realizados.

En la figura 7-28 se representa el flujo del programa. El ciclo básico de actuación consiste en esperar a que se invoque una acción (por ejemplo, pulsando la tecla «a» para añadir una solicitud). Como resultado de esta acción se crea un nuevo elemento en el sistema (una solicitud genérica con muy poca información inicial) que aglutinará toda la información que se vaya creando a su alrededor. En un principio, la única información disponible es la que identifica la propia acción; es decir, su nombre y el instante de su activación. Con esta información de partida se acude a los métodos de inferencia del sistema para completar el resto de sus atributos. En realidad, lo que ocurre es que para completar cada uno de los *slots* que definen al tipo de solicitud en construcción, se infiere un valor que se le muestra al usuario como **valor aconsejado** y, finalmente, el usuario decide el **valor asignado**.

El esquema de funcionamiento se puede reducir a un bloque de cuestiones que se realizan cada vez que se intenta introducir algún nuevo evento en el sistema. En la tabla 7-2 se muestran los tipos de eventos que pueden ocurrir. *Cada uno de los atributos de cada uno de estos eventos puede ser objeto de una tarea de inferencia o de aprendizaje en el sistema.*

Reunión	Salida	Seminario	Curso
Asistentes [?]	Fecha [nn/nn/nn]	Tipo [valor]	Nombre [valor]
Fecha [nn/nn/nn]	Duración[nn]	Ponentes [?]	Ponentes [?]
Duración [nn]	Inicio [h:mm]	Fecha [nn/nn/nn]	Fecha [nn/nn/nn]
Inicio: [h:mm]	Lugar [defecto]	Duración[nn]	Duración[nn]
Lugar [valor]		Hora-inicio [n:nn]	Inicio [h:mm]
		Lugar [defecto]	Lugar [defecto]

Tabla 7-2 Formularios alternativos del Calendario.

Un caso concreto podría ser el siguiente: Ángel solicita una reunión con Tomás (el dueño de la agenda). La secretaria activa la acción añadir evento y empieza a llenar las cuestiones asociadas al tipo de evento `reunion` (que coincide con el tipo propuesto por defecto). En primer lugar, inserta el valor del slot `sol-asistentes`; al ser completado, el resto de los atributos que todavía quedan por designar pueden utilizar dicha información para sugerir valores acordes con dicho asistente. A continuación, la secretaria llega a un acuerdo con Ángel sobre la fecha de la reunión

(*slot* sol-fecha), teniendo en cuenta el valor previamente aconsejado por el sistema sobre dicho *slot*. Si ese día no pudiera asistir Ángel, podría asignarse otro valor alternativo que le conviniera a ambos. Es decir, siempre existen dos valores para estos atributos, uno, el aconsejado, y otro, el asignado. Este proceso se repetiría para el resto de los campos del formulario correspondiente al tipo de solicitud reunion. Se realizarían procesos análogos para el resto de los tipos de solicitudes del Calendario (los formularios de cada tipo de solicitud se muestran en la tabla 7-2).

La diferenciación de dos valores para cada uno de los atributos asignables en el interfaz es uno de los aspectos más singulares de este planteamiento. El primero, **valor-aconsejado**, es el que resulta de aplicar las reglas conocidas o aprendidas sobre dicho *slot*. El segundo, **valor-asignado**, contiene el valor finalmente introducido en la interacción con el sistema.

La diferenciación entre el valor-aconsejado y el valor-asignado es análoga a los **puntos de vista** (valores distintos de un campo según sea la perspectiva utilizada; por ejemplo, la propiedad volar de la entidad ave debería incluir un valor-general sí y un valor-excepcional no para los pingüinos y los avestruces).

Base de Conocimientos		
Tipos de elementos		Atributos significativos
Muy volátiles	solicitudes	fecha, hora-inicio, duración, localización, asistentes, tipo
	días	fecha, sucesos-del-día
	sucesos	duración, localización, fecha, creador, hora-inicio
Poco volátiles	trabajos	participantes
	personas	puesto, institución, tutor

Tabla 7-3 Tipos de entidades y atributos.

Los formularios asociados a cada uno de los tipos de solicitudes de la tabla 7-2 señalan los elementos del dominio que más directamente van a ser objeto de modificación por el usuario. Además de éstos, existen otra serie de entidades que necesitan ser representadas. La base de conocimientos que aglutina todos estos elementos puede dividirse en dos grandes grupos: aquéllos que cambian continuamente por la propia actividad del sistema y aquéllos que permanecen estables durante un cierto tiempo (tal y como se señala en la tabla 7-3). Entre estos últimos, por ejemplo, las personas que asisten a los distintos eventos podrían ser extraídos de

algunas bases de datos que, previsiblemente, ya los contuvieran. Esta circunstancia, la existencia de un soporte informático en el cual residen muchos de los datos que pueden formar parte del conocimiento del dominio, suele ser bastante frecuente en aplicaciones reales. Por otro lado, muchos programas de IA no son hechos aislados, sino más bien al contrario, suelen estar diseñados como módulos que resuelven tareas concretas dentro de un sistema más amplio.

Después de haber concretado los elementos del dominio y algunos de los atributos más significativos, vamos a especificar en un conjunto de enunciados todos estos elementos y las reglas previamente conocidas sobre el funcionamiento del sistema.

7.7.3.2 Elementos básicos

Del planteamiento general del Calendario (sec. 7.7.3.1) hemos extraído como elementos básicos del dominio las entidades y atributos de la tabla 7-2. En este primer enunciado se pide representar dichos elementos utilizando el formalismo de representación descrito en la sección 7.5.

Por otro lado, entre los elementos básicos consideraremos especialmente la diferenciación ya comentada en el apartado previo sobre los dos tipos de valores existentes para los atributos de dicha tabla: valor-asignado y valor-aconsejado. Para garantizar que el primero no va a ser inferido por ninguno de los métodos del sistema (ya que debe recuperarse el valor dado por el usuario), introducimos un nuevo método que denominaremos *dar-loquehay*. Este método debe, simplemente, acceder a la dirección del *slot* correspondiente y devolver su valor. Por ejemplo, devolvería **sin-valor** cuando todavía no hubiera un valor asignado. Sólo se pide representar su utilización.

Además, se conoce la siguiente información sobre los elementos menos volátiles del sistema:

- Existen cuatro tipos de solicitudes: reuniones, seminarios, cursos y salidas (cualquier visita o acontecimiento que ocurre fuera del campus universitario).
- Los días de la semana tienen asociados los eventos producidos para dicho día. Se distinguen dos tipos de días: laborables y festivos.
- Todas las personas del Calendario pueden tener un valor asignado para los siguientes atributos: puesto, grado de ocupación, si es o no persona de la UNED, proyectos asignados, oficina, jefe, institución, departamento, tutor, ¿reuniones regulares? (si las tiene o no) y última reunión asistida (señala la última reunión de ese tipo, es decir, con los mismos asistentes y las mismas características).

- Se distinguen una serie de grupos de personas concretas. Así, se puede decir que alguien pertenece o no a los siguientes grupos: UNED, No-UNED, Departamento (con dos valores posibles: Inteligencia Artificial e Informática y Automática) y Dueño (consideramos que Tomás es el dueño de la agenda). El Grupo Calendario, el Grupo Aprendizaje y la Secretaría están dentro del Grupo Dueño. El Grupo Robot y el Grupo Redes están dentro del Grupo Departamento. El Grupo Profesores y el Grupo Alumnos están dentro del Grupo UNED. El Grupo Investigadores Visitantes se considera dentro de los grupos adscritos al Dueño.
- Existen inicialmente las siguientes personas: Jaime y Andrés son Profesores. El primero es del Departamento de Inteligencia Artificial (despacho 06 del edificio de Ciencias) y Andrés (despacho 232 de Ciencias) pertenece al de Informática y Automática. Elena es una alumna del Grupo Dueño y del de Aprendizaje. Pedro es un investigador visitante. El decano está en el Grupo UNED (su despacho es el 04 de Ciencias) y el investigador Pedro está en el Grupo Redes y Calendario. La secretaria se llama Andrea y su dirección de correo es secretar@dep-ia.

• Solución

Para representar las entidades y atributos de la tabla 7-2, dado que hay una serie de propiedades que se repiten en todas las columnas de la misma, optamos por definir una sola entidad, *solicitud*, que contiene los campos necesarios para dar cabida a todas las particularidades de cada tipo de solicitud. La solución propuesta y el nombre de todos los *slots* necesarios se muestra en la figura 7-29. Los *slots* se denominan *sol-xxx* para distinguir las propiedades de las solicitudes del resto de los elementos del dominio.

Por otro lado, para aquellos atributos que están directamente asociados a los valores asignados por el usuario, se establece la diferenciación de dos valores por atributo: *valor-asignado* y *valor-aconsejado*. El primero contiene el valor finalmente dado por el usuario y el segundo el valor inicialmente sugerido por el sistema. Las definiciones que permiten realizar esta diferenciación se muestran en la figura 7-31. En esta misma figura incluimos la utilización del método *dar-loquehay*, así como los objetos básicos de la jerarquía que forman todas las entidades del Calendario. Hemos introducido la entidad *slotslot* para designar a los *subslots* de cualquier dominio.

```
(solicitudes *sin-valor*
  (generalizaciones (calendar-objeto))
  ;; slots de entrada
  (sol-evento-tipo *sin-valor*)
  (sol-fecha *sin-valor*)
  (sol-inicio *sin-valor*)
  (sol-fin *sin-valor*)
  (sol-duracion *sin-valor*)
  (sol-lugar *sin-valor*)
  ;; para reuniones
  (sol-asistentes *sin-valor*)
  ;; para seminarios
  (sol-seminar-tipo *sin-valor*)
  ;; para cursos
  (sol-curso-nombre *sin-valor*)
  ;; para cursos y seminarios
  (sol-ponentes *sin-valor*)
  ;; slots inferidos
  (numero-de-asistentes *sin-valor*)
  (persona-unica? *sin-valor*)
  (dia *sin-valor*)
  (sol-ano *sin-valor*)
  (sol-dia *sin-valor*)
  (sol-mes *sin-valor*)
  (sol-dia-en-semana *sin-valor*)
  (urgencia *sin-valor*)
  (importancia *sin-valor*)
  (ultima-reunion-asistida *sin-valor*)
  (siguiente-reunion-asistida *sin-valor*))
```

Figura 7-29 Definición de solicitudes, la entidad básica del Calendario.

En lo que se refiere a los tipos de solicitudes, sabemos que existen cuatro variantes, expresamente declaradas en la definición de la figura 7-30. Además, también se ha comentado en el planteamiento del problema (sec. 7.7.3.1) que la solicitud inicialmente aconsejada es del tipo reunión. Por tanto, tal y como puede observarse en la misma figura, el consecuente de la única regla asociada al slot sol-evento-tipo tiene como valor asignado reunion. Para que dicho valor sea devuelto siempre, se incluye como única cláusula antecedente (eval t t), que se cumple en todo momento. Por otro lado, dado que valor-aconsejado es un *subslot* del slot sol-evento-tipo, para que dicho *subslot* sea convenientemente asignado, es necesario acceder al mismo mediante la sentencia (valor-aconsejado (?x sol-evento-tipo) ?valor). Esto tipo de asignaciones se repite en todos los atributos cuyo valor sea modificado por el usuario y previamente aconsejado por el sistema.

```
(sol-evento-tipo *sin-valor*
  (generalizaciones (solicitud-slot))
  (dominio solicitudes)
  (valores-possibles (reunion seminario curso salida))
  (numero-valores 1)
  (metodos (valor-asignado valor-aconsejado))
  (valor-aconsejado *sin-valor*
    (numero-valores 1)
    (clausulas-prolog *sin-valor*
      (((valor-aconsejado (?x sol-evento-tipo) reunion) :-
        (eval t t))))))
```

Figura 7-30 Definición de los tipos de solicitud.

```
(calendar-objeto *sin-valor*
  (generalizaciones (frame)))

(calendar-slot *sin-valor*
  (generalizaciones (slot calendar-objeto)))

(sol-slot *sin-valor*
  (generalizaciones (calendar-slot)))
  -
  (valor-aconsejado *sin-valor*
    (generalizaciones (slotslot))
    (metodos (prolog))
    (clausulas-prolog *sin-valor*))

  (valor-asignado *sin-valor*
    (generalizaciones (slotslot))
    (metodos (dar-loquehay))
    (dar dar-loquehay)))
```

Figura 7-31 Entidades primarias del dominio.

Cada vez que se pregunta por *sol-evento-tipo* se aplican los métodos asociados. En primer lugar, *valor-asignado* no puede emplearse dado que todavía no hay ningún valor asignado sobre dicho *slot* (se supone que es la primera vez que se pregunta sobre el tipo de la solicitud en curso). En segundo término, *valor-aconsejado* obliga a que se busque en el *subslot* también denominado *valor-aconsejado* del *slot* en cuestión. La única regla disponible hace que se devuelva el *valor reunion* que finalmente será ofertado al usuario.

```
(personas *sin-valor*
  (generalizaciones (calendar-objeto))
  (persona-uned? *sin-valor*)
  (puesto *sin-valor*)
  (ocupacion *sin-valor*)
  (proyectos *sin-valor*)
  (oficina *sin-valor*)
  (jefe *sin-valor*)
  (institucion *sin-valor*)
  (departamento *sin-valor*)
  (tutor *sin-valor*)
  (reuniones-regulares? *sin-valor*)
  (ultima-reunion *sin-valor*)))
```

Figura 7-32 Descripción de las personas.

En la figura 7-32 se definen a las personas con sus atributos correspondientes. Los grupos de personas y de proyectos forman una jerarquía de entidades relacionadas entre sí, tal y como se muestra en la figura 7-35. Finalmente, con toda esta información podemos crear las instancias de las personas conocidas asignándoles los grupos de trabajo en los que participan (figura 7-36). Por otro lado, las estructuras que representan los días de la semana se muestran en la figura 7-34.

```
(puesto *sin-valor*
  (generalizaciones (persona-slot))
  (dominio persona)
  (valores-possibles (profesor alumno investigador
    secretaria visitante))
  (metodos (prolog))
  (clausulas-prolog *sin-valor*
    ((puesto ?perona ?puesto) :-
      (generalizaciones ?persona ?puesto-p)
      (eval (member ?puesto-p '(profesor alumno
        investigador secretaria visitante)
        ?puesto-existe)
      (eval (if ?puesto-existe ?puesto-p
        *sin-valor*)
        ?puesto))))))
```

Figura 7-33 Definición del puesto de las personas.

```
(dias *sin-valor*
  (generalizaciones (calendar-objeto))
  (especializaciones (laborables festivos)))

(laborables *sin-valor*
  (generalizaciones (dias))
  (especializaciones
    (lunes martes miercoles jueves viernes)))

(festivos *sin-valor*
  (generalizaciones (dias))
  (especializaciones
    (sabado domingo)))
```

Figura 7-34 Definición de los días de la semana.

(uned *sin-valor*	(robot *sin-valor*
(generalizaciones (personas)))	(generalizaciones (departamento)))
(no-uned *sin-valor*	(redes *sin-valor*
(generalizaciones (personas)))	(generalizaciones (departamento)))
(departamento *sin-valor*	(profesor *sin-valor
(generalizaciones (uned))	(generalizaciones (uned)))
(tutor tomas)	
(dueño *sin-valor*	(investigador *sin-valor*
(generalizaciones (uned)))	(generalizaciones (dueño))
(calendario *sin-valor*	(jefe (tomas))
(generalizaciones (dueño))	
(proyecto (cap))	(secretaria *sin-valor*
(jefe (tomas))	(generalizaciones (dueño))
(aprendizaje *sin-valor*	(jefe (tomas))
(generalizaciones (dueño))	
(proyecto (aprendizaje))	(int-art *sin-valor*
(jefe (tomas))	(generalizaciones (uned))
(alumno *sin-valor*	(proyecto (int-art-trabajo)))
(generalizaciones (uned)))	
	(inf-aut *sin-valor*
	(generalizaciones (uned))
	(proyecto (inf-aut-trabajo)))

Figura 7-35 Grupos de personas.

Cada uno de los atributos de las personas tiene una definición asociada. Por ejemplo, en la figura 7-33 se muestra la del puesto. El puesto se puede obtener del atributo generalizaciones asociado a las personas. El problema es que no todas

las generalizaciones de personas son un puesto válido. Por ello, se ha utilizado la función Lisp `member` para comprobar que el puesto es un valor admisible y, en caso contrario (aplicando la función Lisp `if`), se le asigna `*sin-valor*`. Lo importante en este caso no es saber utilizar diferentes funciones Lisp, sino *comprobar la facilidad con que pueden intercalarse las llamadas a funciones dentro de las cláusulas estilo prolog* (esta es una posibilidad que flexibiliza la rigidez del formalismo lógico y que, por otro lado, está presente en la mayoría de las aplicaciones reales).

(jaime *sin-valor*	(pedro *sin-valor*
(generalizaciones (profesor))	(generalizaciones
(puesto titular)	(investigador
(institucion uned)	(calendario))
(departamento int-art)	(tutor dueño)
(oficina ciencias06)	(institucion uned)
	(proyectos (cap redes))
(andres *sin-valor*	
(generalizaciones (profesor))	(decano *sin-valor*
(puesto titular)	(generalizaciones (uned))
(institucion uned)	(oficina ciencias04))
(departamento info-auto)	
(oficina ciencias232))	(andrea *sin-valor*
	(generalizaciones
(elena *sin-valor*	(secretaria))
(generalizaciones	(oficina ciencias237)
(alumno aprendizaje))	(e-mail secretar@dep-ia)
(tutor dueño)	
(institucion uned)	
(departamento int-art))	

Figura 7-36 Personas concretas.

7.7.3.3 Duración de una reunión

De todos los atributos del interfaz (tabla 7-2), en este ejercicio nos centramos en uno de los parámetros más importantes a la hora de "encajar" una determinada reunión en la agenda: el cálculo de su duración. En este enunciado se parte de la existencia de una serie de funciones predefinidas que podrán utilizarse en el cálculo de los valores de los *slots* correspondientes. En concreto, se pide representar *sól-duracion* considerando los siguientes datos:

- Se supone definida una función `grado-ocupacion` que devuelve el número de horas ocupadas (expresadas en minutos) de una semana concreta.

- También se parte de la existencia de una función `fecha-actual` que devuelve el día en curso. Existe igualmente una función `de-fecha-a-dias` para calcular el número de días con respecto a una fecha base, de forma que puedan restarse dos fechas entre sí, pudiendo entonces obtenerse el número de días transcurridos entre ambas. Por ejemplo, se podría evaluar la expresión `(eval (- (de-fecha-a-dias (fecha-actual)) (de-fecha-a-dias '(3 11 1997))) ?diferencia-dias)`.
- El cálculo de la duración de una reunión se subdivide en dos procesos. Primero, se determina la **duración deseada** del evento solicitado; es decir, aquélla en que se *asigna un tiempo sin considerar posibles conflictos* con otros eventos. En segundo lugar, para calcular el valor definitivo de la **duración asignada** se considera necesario evitar que la semana en la que en principio quiere ubicarse dicho evento pase a estar muy *ocupada*. El cálculo de ambos valores, el deseado y el definitivo, se realiza siguiendo las indicaciones de los siguientes enunciados.
- Los valores de la *duración deseada* de una reunión se calculan como sigue:
 - Si se trata de un seminario, su duración deseada es 60 (la unidad base de tiempo son minutos).
 - Si es un curso, su duración deseada es 90.
 - Si es una salida (fuera de la Universidad), se le supone una duración deseada de 600.
 - Si la solicitud es una reunión, sólo asiste una persona y su puesto es alumno de doctorado, entonces su duración deseada es de 30.
 - Si la solicitud es una reunión, sólo asiste una persona y su tutor es Tomás, entonces su duración deseada es de 60.
 - Si la solicitud es una reunión, sólo asiste una persona y su puesto es estudiante, entonces su duración deseada es de 30, si han pasado más de 7 días desde la última reunión.
 - La duración deseada para personas ajenas a la Universidad es de 60 minutos.
 - Los valores de duraciones deseadas posibles son: 30, 60, 90, 120, 150 y 180.
- Si la solicitud tiene una duración deseada que supera 30 y el grado de ocupación del día de la solicitud supera las 4 horas diarias durante todos los días laborables, entonces se le asigna una duración definitiva a la solicitud que es 60 minutos inferior a la inicialmente deseada.

- Si no se ha podido inferir ningún valor deseado para la duración de un evento, se le asigna una duración de 30 minutos.
- Los valores de duraciones válidos son: 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 360, 420, 480, 540 y 600.

En realidad, la tarea que se pide resolver es una **tarea de clasificación**. Esto es, dados una serie de objetos descritos mediante una serie de atributos (en este caso solicitudes y sus *slots*), se desea asignar la clase a la que pertenecen (un valor de duración, que pertenezca al conjunto de valores posibles dados). Para ello, se define un modelo que relaciona cada objeto con su clase (véase la figura 7-37). En este caso, el modelo es el conjunto de reglas anteriormente descritas. El conocimiento de dichas reglas no siempre es una tarea fácil. De hecho, las que aquí se han enunciado son sólo una primera aproximación a las realmente existentes. Por ello, como veremos más adelante (sec. 7.7.3.5), las reglas de asignación pueden ser el objeto de una tarea de aprendizaje.

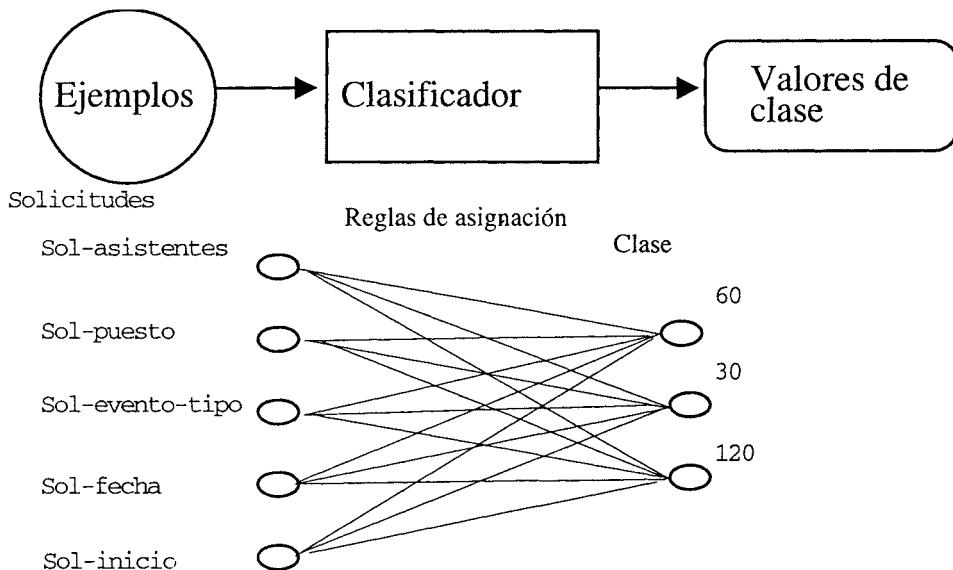


Figura 7-37 Tarea de clasificación.

Para los problemas en que se pueden identificar atributos relevantes para la clasificación, pero resulta difícil llegar a establecer una función que permita enlazar

correctamente objetos y clases, se aplican métodos de aprendizaje que alcanzan cotas altas de fiabilidad.

El modelado de tareas es un tema de creciente interés en IA. En concreto, la *tarea de clasificación* es una de las que ha recibido una mayor atención dada su utilidad en problemas diversos. Por ejemplo, una *tarea de diagnóstico* puede formularse como una de clasificación. El problema se podría resolver estableciendo una clasificación que asigne los *síntomas* presentados con el *diagnóstico* más adecuado. En concreto, un método aplicado para resolver esta tarea podría desglosarse en un conjunto de acciones: primero, *abstraer* los datos asociados a los síntomas (por ejemplo, una temperatura de 40 grados es una termeratura *alta*); segundo, *seleccionar* los diagnósticos que incluyen dichos síntomas; y tercero, *evaluar* estos diagnósticos con el fin de decidir el diagnóstico definitivo. Este último proceso, que también recibe el nombre de *refinamiento*, consiste en comprobar el resto de los síntomas asociados a cada uno de los diagnósticos inicialmente seleccionados, con el fin de ir descartando aquéllos que no cumplan el resto de los síntomas presentados. Esta descomposición en subtareas coincide con la denominada *clasificación heurística* [Clancey, 1985].

- **Solución**

Tal y como se pide en el enunciado, dividiremos el valor de la duración de una reunión en un primer valor intermedio, al que denominaremos *duracion-deseada*, y en otro definitivo (aquél que finalmente será aconsejado), que se corresponde con el valor utilizado en el interfaz, al que hemos denominado *sol-duracion* (véase la figura 7-29). La duración deseada se define en la figura 7-38 considerando los datos del enunciado. Siguiendo las recomendaciones del problema, nos hemos aprovechado de las funciones predefinidas en el sistema.

Con el valor de la *duracion-deseada* se comprueba si puede haber conflictos causados por el grado de ocupación de la semana en la que se intenta situar la solicitud en curso. En dichos casos se corrige el valor deseado y se le restan 60 minutos. Para realizar este cálculo se definen una serie de reglas en la figura 7-39. Dado el funcionamiento de las reglas de Prolog, la segunda regla se cumplirá cuando no se haya podido aplicar la primera. De esta forma nos garantizamos devolver el valor deseado cuando no existan problemas en la asignación de la reunión. Finalmente, la tercera regla permite devolver un valor de 30 minutos para el resto de los casos.

```
(duracion-deseada *sin-valor*
  (personal-kb? nil)
  (generalizaciones (request-slot))
  (dominio requests)
  (valores-possibles (30 60 90 120 150 180))
  (metodos (prolog))
  (numero-valores 1)
  (clausulas-prolog
    (((duracion-deseada ?s 60) :-
       (sol-evento-tipo ?s seminario))
     ((duracion-deseada ?s 90) :-
       (sol-evento-tipo ?s curso))
     ((duracion-deseada ?s 600) :-
       (sol-evento-tipo ?s salida))
     ((duracion-deseada ?s 30) :-
       (sol-evento-tipo ?s reunion)
       (unica-persona? ?s si)
       (sol-asistentes ?s ?a)
       (puesto ?a alumno-doctorado))
     ((duracion-deseada ?s 60) :-
       (sol-evento-tipo ?s reunion)
       (unica-persona? ?s si)
       (sol-asistentes ?s ?a)
       (tutor ?a tomás))
     ((duracion-deseada ?s 30) :-
       (sol-evento-tipo ?s reunion)
       (unica-persona? ?s si)
       (sol-asistentes ?s ?a)
       (puesto ?a estudiante)
       (ultima-reunion-asistida ?s ?ultima)
       (sol-fecha ?ultima ?ultimo-d)
       (eval (de-fecha-a-dias ?ultimo-d) ?fin)
       (eval (fecha-actual) ?ini-d)
       (eval (de-fecha-a-dias ?ini-d) ?inicio)
       (eval (> (- ?fin ?inicio) 7) t)))
     ((duracion-deseada ?s 60) :-
       (sol-evento-tipo ?s reunion)
       (sol-asistentes ?s ?a)
       (persona-uned? ?a no))))))
```

Figura 7-38 Determinación de la duración deseada de una solicitud.

En todas las reglas de la figura 7-39 llama la atención la utilización de la expresión (valor-aconsejado (?r sol-duracion) ?duracion). Esto permite acceder al valor del *subslot* valor-aconsejado del slot sol-duracion de la solicitud ?s y asignarle el valor asociado a la variable ?duracion.

```
(sol-duracion *sin-valor*
  (generalizaciones (sol-slot))
  (dominio solicitudes)
  (valores-possibles (30 60 90 120 150 180 210 240 270 300
                      360 420 480 540 600))
  (metodos (valor-asignado valor-aconsejado))
  (valor-aconsejado *sin-valor*
    (numero-valores 1)
    (clausulas-prolog
      (((valor-aconsejado (?s sol-duracion) ?duracion) :- 
         (duracion-deseada ?s ?deseada)
         (eval (>= ?deseada 30) t)
         (eval (fecha-actual) ?fecha)
         (eval (grado-ocupacion ?fecha) ?grado-oc)
         (eval (> ?grado-oc (* 20 60)) t)
         (eval (- ?deseada 60) ?duracion)))
      ((valor-aconsejado (?s sol-duracion) ?duracion) :- 
         (deseada-duracion ?s ?duracion)))
      ((valor-aconsejado (?s sol-duracion) 30) :- 
         (not (sol-deseada ?s ?d))))))))
```

Figura 7-39 Determinación de la duración aconsejada de una solicitud.

7.7.3.4 Fecha de una reunión

Otro de los atributos de mayor relevancia en el sistema (**tabla 7-2**) es la fecha en que es aconsejable celebrar un determinado evento. El valor de este atributo tiene una dificultad muy particular. Dado que puede tomar un elevado número de valores, es imposible plantearlo directamente como un problema de clasificación, tal y como hicimos en el ejercicio sobre la duración (sección previa). Los valores de clase serían todas las fechas posibles. En lugar de resolver este problema directamente, se resuelven otras cuestiones de menor dificultad, quedando patente una vez más la utilidad de dividir problemas complejos en otros más simples.

Para simplificar el cálculo de la fecha assignable a un determinado evento se aconseja la creación de dos nuevos atributos asociados a la entidad solicitud, a los que denominaremos: *días-entre* y *dia-semana*. El primero representa el número de días generalmente transcurridos entre dos eventos del mismo tipo. Este valor es significativo ya que suele ocurrir que, por ejemplo, Tomás tenga reuniones todos las semanas —cada 7 días— con su alumna Elena. En este, y en otros muchos casos (tanto en este contexto como en otros dominios), es muy frecuente que transcurra un número concreto de días (7, 15, 20, etc.) entre dos reuniones consecutivas. De igual

forma, se ha comprobado que las reuniones de un determinado tipo suelen celebrarse el mismo día de la semana. Por ejemplo, Tomás tiene una reunión con el grupo que está trabajando en el Calendario todos los viernes. Por tanto, el día de la semana en que ocurre un evento también es un valor constante que puede ser utilizado para determinar en última instancia la fecha que debe ser asignada.

Finalmente, se tiene la convicción de que ambos parámetros, los **días-entre** y el **dia-semana**, se pueden conjugar teniendo en cuenta la siguiente heurística: **es más fiable el valor devuelto por dia-semana que por días-entre**. Este criterio, basado en el conocimiento del dominio, se concreta, como veremos enseguida, en una ponderación de los valores devueltos por ambos atributos. El siguiente conjunto de enunciados, diferenciados para cada uno de los dos nuevos atributos, determina el valor finalmente seleccionado para las fechas de las reuniones.

- Días transcurridos entre eventos:
 - Los investigadores tienen reuniones cada 7 días.
 - Si sólo hay un asistente a la reunión entonces se devuelve el número de días que más se repite transcurrido entre dos reuniones consecutivas de dicho tipo.
 - Si el tipo de solicitud es un seminario y su ponente es x entonces se devuelve el número de días más frecuente transcurridos entre dos seminarios sucesivos de dicho tipo.
 - Si el tipo de solicitud es un curso y el ponente es x entonces se devuelve el número de días más frecuente transcurridos entre dos cursos sucesivos de dicho tipo.
 - Si lo que se solicita es una salida fuera del campus, se busca el valor más frecuente entre dos salidas consecutivas.
- Día de la semana en que es más probable que ocurra un cierto evento:
 - Elena suele tener las reuniones los miércoles.
 - Andrés tiene reuniones unipersonales los martes y el resto de las reuniones los viernes.
 - Las reuniones con Pedro suelen celebrarse los viernes.
 - Si la solicitud es una reunión con una única persona, se devuelve los días más frecuentes en que se celebran las reuniones unipersonales con dicho solicitante.
 - Si la solicitud es de un seminario con un ponente concreto, se devuelve el día más frecuente en que dicho ponente imparte el seminario.

- Si la solicitud es de un curso impartido por alguien, se devuelve el día de la semana en que suele impartir dicho curso dicho ponente.
- Si se trata de una salida fuera del campus, se devuelve el día de la semana en que suelen realizarse dichas salidas.
- Ponderación heurística entre `dia-semana` y `dias-entre`, considerando que la variable `?s` señala la solicitud en curso y que `?dias-entre` y `?dia-semana` contienen los valores inferidos aplicando los enunciados previos para los *slots* correspondientes:
 - 1) Suponiendo que `?fecha-previa` es la fecha en que ocurrió el último evento asociado a la solicitud `?s`, se debe sumar a `?fecha-previa` el valor de `?dias-entre` para obtener una `?fecha-dias-entre` que señala la fecha en que debería ocurrir el evento considerando únicamente los días entre fechas de solicitudes consecutivas del mismo tipo.
 - 2) Si el valor de `?fecha-dias-entre` supera la fecha actual (fecha de la solicitud) y el día de la semana de dicha fecha coincide con el obtenido a partir de `dia-semana` (`?dia-semana`), entonces se devuelve la fecha indicada.
 - 3) Si el valor de `?fecha-dias-entre` no supera la fecha actual, se devuelve la fecha del siguiente día de la semana —desde la fecha actual— cuyo día de la semana coincida con el de `?dia-semana`.
 - 4) Si el valor de `?fecha-dias-entre` supera la fecha actual (fecha de la solicitud) y el día de la semana de dicha fecha no coincide con el obtenido a partir de `dia-semana` (`?dia-semana`), entonces se devuelve la primera desde `?fecha-dias-entre` cuyo día de la semana coincide con el `?dia-semana`.
 - 5) Para el resto de los casos, dado que se sabe que, excepto en casos anómalos, una reunión nunca se concierta para el día en que se solicita, se devuelve como fecha el tercer día siguiente a la fecha actual.

Para poder realizar este ejercicio es necesario considerar los siguientes funciones y *slots* previamente definidos.

- Existe un *slot* de cualquier solicitud denominado `sol-fecha-previa` que devuelve la fecha en que se celebró la última solicitud del tipo dado. Así, por ejemplo, si se pregunta por `(sol-fecha-previa ?s ?fecha-previa)` y la solicitud `?s` es de tipo reunión con el único asistente Elena, se devolvería la fecha de la última reunión unipersonal de Elena en la variable `?fecha-previa`.

- La función `suma-a-fecha` recibe como argumentos una fecha (26 10 1997) y una cantidad (por ejemplo, 10), y devuelve la fecha resultante de sumarle a la fecha el número indicado (5 11 1997).
- La función `dia-de-fecha` recibe como argumento una fecha y devuelve el día de la semana de dicha fecha.
- La función `encuentra-reuniones-previas` recibe como argumentos una solicitud, el asistente, la fecha actual y el número de meses hacia atrás en los que se quiere buscar eventos de un tipo y devuelve todos los eventos encontrados.
- La función `encuentra-reuniones-previas` recibe como argumentos una solicitud, el asistente, la fecha actual y el número de meses hacia atrás en los que se quiere buscar eventos de un tipo, y devuelve todos los eventos encontrados.
- La función `encuentra-seminarios-previos` recibe como argumentos una solicitud, el ponente, la fecha actual y el número de meses hacia atrás en los que se quiere buscar eventos de un tipo y devuelve todos los eventos encontrados.
- La función `encuentra-cursos-previos` recibe como argumentos una solicitud, el ponente, la fecha actual y el número de meses hacia atrás en los que se quiere buscar eventos de un tipo y devuelve todos los eventos encontrados.
- La función `encuentra-salidas-previas` recibe como argumentos una solicitud, la fecha actual y el número de meses hacia atrás en los que se quiere buscar eventos de un tipo y devuelve todos los eventos encontrados.
- La función `dias-entre-eventos` recibe como argumentos una lista de eventos y devuelve el número de días más frecuente transcurridos entre dichos eventos. Para ello, llama internamente a la función `valor-mas-frecuente`.
- La función `dia-eventos` recibe como argumentos una lista de eventos y devuelve el día de la semana en que más frecuentemente ocurren dichos eventos. Para ello, llama internamente a la función `valor-mas-frecuente`.
- La función `valor-mas-frecuente` recibe como argumentos una lista de valores y devuelve el que más se repite.
- La función `coincidir-fecha-dia` recibe como argumentos una fecha y un día de la semana (lunes, martes, ...) y devuelve una nueva fecha que será la

primera fecha desde la fecha dada cuyo día de la semana coincide con el dado. La fecha que recibe como argumento puede venir expresada en formato de fecha o en el formato estandarizado del número de días desde una cierta fecha. Es decir, puede recibir como argumento el valor devuelto por aplicación de la función fecha-actual.

• Solución

En primer lugar, concretamos la definición del *slot* dias-entre asociado a las solicitudes del Calendario. Cada una de las reglas de prolog de la figura 7-40 reflejan el contenido de cada uno de los hechos enunciados previamente en el planteamiento del problema. Se colocan delante las reglas más específicas para casos concretos conocidos de antemano. Como puede observarse, la estructura de todas las reglas es la misma. Primero se concreta el tipo de evento en cuestión (reunión, seminario, curso o salida) y luego se recuperan de la memoria de eventos (en este caso se recogen los ocurridos en los dos meses previos) aquéllos que sean del tipo especificado.

En segundo lugar, concretamos la definición del *slot* dia-semana asociado a las solicitudes del Calendario. Al igual que en el caso de la inferencia de dias-entre, cada una de las reglas de prolog de la figura 7-41 reflejan el contenido de cada uno de los hechos enunciados previamente en el planteamiento del problema. Como puede observarse, la estructura de todas las reglas es la misma y coincide con la del *slot* dias-entre. Primero se concreta el tipo de evento en cuestión (reunión, seminario, curso o salida) y luego se recuperan de la memoria de eventos (en este caso se recogen los ocurridos en los dos meses previos) aquéllos que sean del tipo especificado. Como en el caso previo, se colocan delante las reglas más específicas para las circunstancias concretas conocidas de antemano.

Finalmente (véase la figura 7-42), tomando como base los valores inferidos para dias-entre y dia-semana, se determina el valor aconsejado para sol-fecha. Para ello, se siguen fielmente los enunciados asociados a la heurística que prima el valor deducido desde el *slot* dia-semana frente al obtenido desde dias-entre. El cálculo realizado se reduce a sumar a la fecha de la última reunión el número especificado en dias-entre y luego, comprobar si el día de la semana coincide con dia-semana y además es mayor que la fecha actual. Para el resto de los casos se aconseja realizar la reunión tres días después de la fecha de la solicitud. Esta heurística viene determinada por el conocimiento de los ficheros históricos de ejemplos del dominio, en los que se refleja que siempre que se solicita una reunión, incluso las que se solicitan por primera vez, se atiende pasados unos días desde la fecha en que se genera.

```
(dias-entre *sin-valor*
  (generalizaciones (sol-slot))
  (dominio solicitudes)
  (valores-posibles numero)
  (metodos (valor-asignado valor-aconsejado))
  (valor-aconsejado *sin-valor*
    (numero-valores 1)
    (clausulas-prolog
      (((valor-aconsejado (?s dias-entre) 7) :-
         (unica-persona ?s si)
         (sol-asistente ?s ?a)
         (puesto ?a investigador))
       ((valor-aconsejado (?s dias-entre) ?dias-entre) :-
         (unica-persona ?s si)
         (sol-asistente ?s ?a)
         (eval (encuentra-reuniones-previas ?s ?a
                                           (fecha-actual) 2) ?previos)
         (eval (dias-entre-eventos ?previos)
               ?dias-entre))
       ((valor-aconsejado (?s dias-entre) ?dias-entre) :-
         (sol-seminar-tipo ?s ?seminario)
         (sol-ponentes ?s ?ponente)
         (eval (encuentra-seminarios-previos ?s
                                           ?ponente (fecha-actual) 2) ?previos)
         (eval (dias-entre-eventos ?previos)
               ?dias-entre))
       ((valor-aconsejado (?s dias-entre) ?dias-entre) :-
         (sol-curso-nombre ?s ?curso)
         (sol-ponentes ?s ?ponente)
         (eval (encuentra-cursos-previos ?s ?ponente
                                           (fecha-actual) 2) ?previos)
         (eval (dias-entre-eventos ?previos)
               ?dias-entre))
       ((valor-aconsejado (?s dias-entre) ?dias-entre) :-
         (sol-evento-tipo ?s salida)
         (eval (encuentra-salidas-previas ?s
                                           (fecha-actual) 2) ?previos)
         (eval (dias-entre-eventos ?previos)
               ?dias-entre))))))
```

Figura 7-40 Determinación de los días entre solicitudes del mismo tipo.

```

(dia-semana *sin-valor*
  (generalizaciones (sol-slot))
  (dominio solicitudes)
  (valores-possibles (lunes martes miercoles jueves viernes
    sabado domingo))
  (metodos (valor-asignado valor-aconsejado))
  (valor-aconsejado *sin-valor*
    (numero-valores 1)
    (clausulas-prolog
      (((valor-aconsejado (?s dia-semana) viernes) :-
        (unica-persona ?s si)
        (sol-asistente ?s elena))
       ((valor-aconsejado (?s dia-semana) martes) :-
        (unica-persona ?s si)
        (sol-asistente ?s andres))
       ((valor-aconsejado (?s dia-semana) viernes) :-
        (unica-persona ?s no)
        (sol-asistente ?s andres))
       ((valor-aconsejado (?s dia-semana) viernes) :-
        (sol-asistente ?s pedro))
       ((valor-aconsejado (?s dia-semana) ?dia-semana) :-
        (unica-persona ?s si)
        (sol-asistente ?s ?a)
        (eval (encuentra-reuniones-previas ?s ?a
          (fecha-actual) 2) ?previas)
        (eval (dia-eventos ?previas) ?dia-semana)))
       ((valor-aconsejado (?s dia-semana) ?dia-semana) :-
        (sol-seminar-tipo ?s ?seminario)
        (sol-ponentes ?s ?ponente)
        (eval (encuentra-seminarios-previos ?s
          ?ponente (fecha-actual) 2) ?previos)
        (eval (dia-eventos ?previos) ?dia-semana)))
       ((valor-aconsejado (?s dia-semana) ?dia-semana) :-
        (sol-curso-nombre ?s ?curso)
        (sol-ponentes ?s ?ponente)
        (eval (encuentra-cursos-previos ?s ?ponente
          (fecha-actual) 2) ?previos)
        (eval (dia-eventos ?previos) ?dia-semana)))
       ((valor-aconsejado (?s dia-semana) ?dia-semana) :-
        (sol-evento-tipo ?s salida)
        (eval (encuentra-salidas-previas ?s
          (fecha-actual) 2) ?previos)
        (eval (dia-eventos ?previos) ?dia-semana)))))))

```

Figura 7-41 Día de la semana en que suelen ocurrir las solicitudes del mismo tipo.

```

(sol-fecha *sin-valor*
  (generalizaciones (sol-slot)) (dominio solicitudes)
  (valores-possibles fechas)
  (metodos (valor-asignado valor-aconsejado))
  (valor-aconsejado *sin-valor* (numero-valores 1)
    (clausulas-prolog
      ((valor-aconsejado (?s sol-fecha) ?fecha) :-
        (dias-entre ?s ?num-dias)
        (dia-semana ?s ?dia-semana)
        (sol-fecha-previa ?s ?f-previa)
        (eval (fecha-actual) ?hoy)
        (eval (de-fecha-a-dias ?hoy) ?dias-de-hoy)
        (eval (suma-a-fecha ?f-previa ?num-dias) ?fecha)
        (eval (de-fecha-a-dias ?fecha) ?dias-entre)
        (eval (> (- ?dias-entre ?dias-de-hoy) 0) t)
        (eval (dia-de-fecha ?fecha) ?d-entre)
        (eval (equal ?d-entre ?dia-semana) t))
      ((valor-aconsejado (?s sol-fecha) ?fecha) :-
        (dias-entre ?s ?num-dias)
        (dia-semana ?s ?dia-semana)
        (sol-fecha-previa ?s ?f-previa)
        (eval (fecha-actual) ?hoy)
        (eval (de-fecha-a-dias ?hoy) ?dias-de-hoy)
        (eval (suma-a-fecha ?f-previa ?num-dias) ?f-entre)
        (eval (de-fecha-a-dias ?f-entre) ?dias-entre)
        (eval (> (- ?dias-entre ?dias-de-hoy) 0) t)
        (eval (dia-de-fecha ?f-entre) ?d-entre)
        (eval (not (equal ?d-entre ?dia-semana)) t)
        (eval (coincidir-fecha-dia ?f-entre
          ?dia-semana) ?fecha)))
      ((valor-aconsejado (?s sol-fecha) ?fecha) :-
        (dias-entre ?s ?num-dias)
        (dia-semana ?s ?d-semana)
        (sol-fecha-previa ?s ?f-previa)
        (eval (fecha-actual) ?hoy)
        (eval (de-fecha-a-dias ?hoy) ?dias-de-hoy)
        (eval (suma-a-fecha ?f-previa ?num-dias) ?f-entre)
        (eval (de-fecha-a-dias ?f-entre) ?dias-entre)
        (eval (> (- ?dias-de-hoy ?dias-entre) 0) t)
        (eval (coincidir-fecha-dia ?hoy ?d-semana) ?fecha)))
      ((valor-aconsejado (?s sol-fecha) ?fecha) :-
        (eval (fecha-actual) ?dias-hoy)
        (eval (suma-a-fecha ?dias-hoy 3) ?fecha)))))))

```

Figura 7-42 Fecha aconsejada para la solicitud en curso.

7.7.3.5 Fecha de una reunión con aprendizaje

La propuesta elaborada en el ejercicio anterior para inferir la fecha de las reuniones, además de tener una cierta complejidad (las consideraciones enunciadas son sólo un subconjunto de las que realmente ocurren), no aporta una solución aceptable del problema. El porcentaje de aciertos, es decir, de consejos coincidentes con el valor finalmente asignado, no supera el 30%, mientras que, por ejemplo, para predecir el parámetro *dia-semana*, el porcentaje de aciertos de las reglas aprendidas llega a ser del 70% para los casos en que existe alguna regla aplicable.

Las técnicas de aprendizaje que vamos a utilizar son especialmente adecuadas para los problemas cuya solución conocida no aporte la respuesta deseada, siempre y cuando se pueda identificar un conjunto de atributos relevantes de una colección suficiente de ejemplos observados en el dominio.

Dado que no es nuestro objetivo profundizar en el contenido de la estrategia de aprendizaje aplicada (los interesados pueden acudir a la bibliografía referenciada previamente al comienzo del apartado 7.2), vamos simplemente a ilustrar su utilización con el fin de comparar el planteamiento asociado a algunos problemas de aprendizaje. Para poder hacerlo, vamos a comentar ciertas consideraciones que deben realizarse al respecto.

La tarea que se quiere aprender es la de **clasificación** (tal y como se ha ilustrado en la figura 7-37). El nuevo planteamiento consiste en suponer que el enlace entre los valores de atributo de los objetos del dominio y los valores de clase a la que pertenecen no aparece explícito en un conjunto de reglas, como las vistas en la sección previa, sino que *se construye de forma automática mediante la aplicación de técnicas de aprendizaje*. En concreto, se va a utilizar una estrategia que construye dicho conjunto de reglas a partir de un árbol de decisión sobre los atributos previamente seleccionados.

Al igual que en la sección 7.7.3.4, la tarea asociada a la clasificación de las solicitudes según la fecha asignada, se subdivide en dos tareas, también de clasificación, asociadas a los parámetros *días-entre* y *dia-semana*. La decisión final sobre la fecha aconsejada puede seguir resolviéndose según lo indicado en la figura 7-42. La ventaja de subdividir el problema es indudable. El número de clases es menor y la desviación de dicha clasificación disminuye. Ésta es otra de las razones (recordar la heurística del enunciado del problema anterior) que a priori hacen suponer que es más fiable el aprendizaje del valor de *dia-semana* (7 valores posibles) frente al de *días-entre* (en el peor de los casos llega a tener hasta 30 valores).

El método de aprendizaje que vamos a utilizar (una evolución del conocido ID3 [Quinlan, 1986]) requiere, al igual que el resto de los enmarcados dentro del aprendizaje inductivo basado en ejemplos¹⁰, que se disponga de un número suficiente de ejemplos. Además, este método considera todos los ejemplos simultáneamente. Dado que en CAP los ejemplos se generan secuencialmente, solicitud tras solicitud, es necesario definir algún procedimiento que sea capaz de guardar, reunir y recrear todas las acciones realizadas al dar de alta cada petición considerada. La solución a estos problemas no es el objetivo de esta exposición.

	asistente	tipo	individual?	ponente	dia-semana
1	elena	reunión	sí	*sin-valor*	viernes
2	andres	reunión	sí	*sin-valor*	martes
3	andres	reunión	no	*sin-valor*	viernes
4	pedro	reunión	sí	*sin-valor*	viernes
5	pedro	reunión	no	*sin-valor*	viernes
6	andrés	seminario	no	tomás	lunes
7	pedro	seminario	no	tomás	lunes
8	elena	seminario	no	tomás	lunes
9	jaime	seminario	no	tomás	lunes
10	elena	seminario	no	andrés	miércoles

Tabla 7-4 Muestra de un fichero histórico de ejemplos.

Se supone que existen una serie de ficheros históricos con ejemplos suficientes (una porción reducida de estos se ilustra en la **tabla 7-4**) como para poder aprender reglas fiables para los parámetros `días-entre` y `día-semana`. La estrategia de aprendizaje genera un árbol de decisión a partir del conjunto de ejemplos guardados que permite:

1. Clasificar correctamente los ejemplos observados.
2. Predecir la clasificación correcta de ejemplos todavía no presentados.

Los *árboles de decisión* (en inglés, “Decision Trees” o, más exactamente, “Top-Down Inductive Decision Trees”, TDIDT) se construyen de forma inductiva siguiendo un proceso iterativo que comienza en la raíz del árbol y termina en sus hojas. Son aquéllos en los que los nodos no extremos son atributos de los ejemplos presentados, las ramas representan los valores de dichos atributos y las hojas son los valores de la *clase*. Para determinar la clasificación de un objeto se pregunta

¹⁰ Una introducción a este tipo de técnicas de aprendizaje se resume en [Mira *et al.*, 1995].

sucesivamente en cada nodo del árbol, empezando por la raíz, cuál es el valor del atributo indicado por el nodo en el objeto (cada uno de los ejemplos considerados); se elige entonces la rama de dicho valor hasta alcanzar una hoja que indica uno de los valores de la *clase*.

En la figura 7-43 se muestra un árbol generado para los ejemplos de la tabla 7-4. Como puede apreciarse, el árbol no incluye todas las preguntas posibles, es decir, no se ajusta exactamente a los ejemplos tratados, sino que genera una estructura de preguntas y respuestas más simple que permitirá clasificar nuevos ejemplos no analizados todavía.

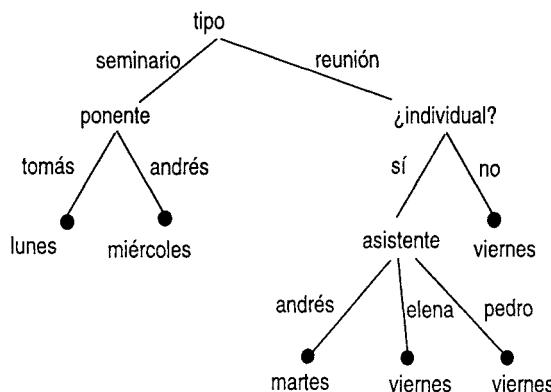


Figura 7-43 Ejemplo de árbol de decisión aprendido.

Otra de las ventajas de los árboles de decisión es que se pueden extraer las reglas de clasificación a partir del árbol obtenido. Aunque el proceso es más complejo [Quinlan, 1987], en principio sería tan sencillo como convertir cada rama del árbol generado en una regla. Por ejemplo:

Si $(\text{tipo} \text{ reunión}) \wedge (\text{individual? no}) \Rightarrow (\text{dia-semana viernes})$

El enunciado del problema de aprendizaje sólo requiere concretar cuáles son los atributos considerados relevantes para describir los ejemplos presentados y cuál es el atributo asociado al valor de clase buscado (véase el esquema de la figura 7-44). Generalmente, se suelen definir diferentes conjuntos de atributos relevantes y se hacen pruebas con éstos para determinar cuál es el que da mejor resultado. Como ejemplo, en la **tabla 7-5** se muestran dos conjuntos diferentes de atributos significativos para la clasificación del parámetro *dia-semana*. Se ha comprobado

empíricamente que un número de atributos entre 10 y 20 suele ser el que da mejores resultados.

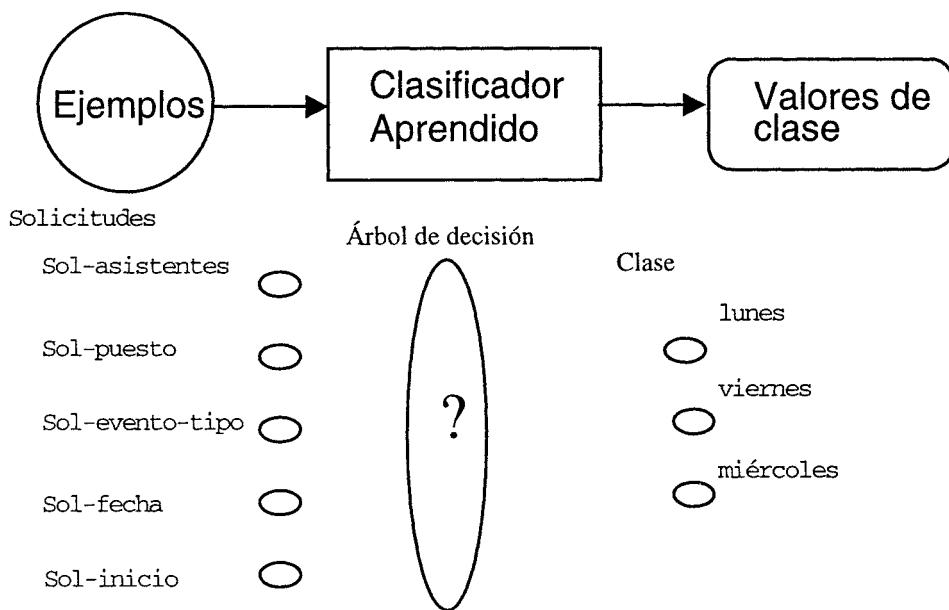


Figura 7-44 Aprendizaje de la tarea de clasificación.

Conjunto Atributos 1	Conjunto Atributos 2
sol-tipo	sol-tipo
sol-asistentes	sol-asistentes
sol-seminar-tipo	sol-seminar-tipo
sol-curso-nombre	sol-curso-nombre
puesto-asistente	
unico-asistente?	
sol-ponentes	

Tabla 7-5 Conjuntos de atributos relevantes

Para que el sistema sea capaz de responder con el valor aprendido antes que con el valor inferido, es necesario definir un nuevo *subslot* del atributo correspondiente al que denominaremos reglas-prolog-aprendidas.

Teniendo presente todo lo expuesto hasta ahora, se pide representar las reglas aprendidas para el atributo dia-semana considerando el árbol de decisión de la figura 7-43.

- **Solución**

Con el fin de que el sistema sea capaz de responder con el valor aprendido antes que con el valor inferido, es necesario definir un nuevo *subslot* al que denominaremos reglas-prolog-aprendidas (véase la figura 7-45). Para que esta solución funcione correctamente, suponemos que el método de inferencia valor-aconsejado tiene en cuenta la existencia del *subslot* que contiene las reglas aprendidas.

```
(dia-semana *sin-valor*
  (generalizaciones (sol-slot))
  (dominio solicitudes)
  (valores-possibles (lunes martes miercoles jueves viernes
                      sabado domingo))
  (metodos (valor-asignado valor-aconsejado))
  (reglas-prolog-aprendidas
    ((dia-semana ?s viernes) :-
       (unica-persona ?s si)
       (sol-asistente ?s elena))
    ((dia-semana ?s martes) :-
       (unica-persona ?s si)
       (sol-asistente ?s andres))
    ((dia-semana ?s viernes) :-
       (unica-persona ?s no))
    ((dia-semana ?s viernes) :-
       (unica-persona ?s si)
       (sol-asistente ?s pedro))
    ((dia-semana ?s lunes) :-
       (sol-seminar-tipo ?s ?seminario)
       (sol-ponentes ?s tomas))
    ((dia-semana ?s miercoles) :-
       (sol-seminar-tipo ?s ?seminario)
       (sol-ponentes ?s andres)))))
```

Figura 7-45 Reglas aprendidas para el parámetro dia-semana

Como puede apreciarse, las reglas aprendidas para dia-semana no necesitan incluir el *slot* sol-evento-tipo dado que sol-ponentes sólo puede estar asociado a las reuniones que son seminarios y sol-asistente es un atributo propio de las que son reuniones.

Para concluir, queremos resaltar el hecho de que las reglas aprendidas en parámetros como sol-lugar, con una distribución mucho más sesgada que sol-fecha, llegan a superar el 90% de éxito en sus predicciones cuando sólo se consideran aquellos ejemplos para los que ya existe algún valor aprendido (los que tienen que ver con la asignación de dicho *slot*). Es decir, para ciertas tareas, se puede llegar a automatizar completamente su gestión. Dejando que los casos para los que se sabe que hay un conocimiento muy fiable aprendido sean clasificados sin necesidad de que intervenga directamente el usuario.

7.8 Contexto adicional

7.8.1 ¿Qué conocimientos nuevos se supone que debe haber aprendido el lector en este capítulo?

1. La integración de técnicas de inferencia de distinta naturaleza no sólo es posible, sino que resulta beneficiosa en los problemas del mundo real.
2. Las arquitecturas integradas proporcionan distintos mecanismos de inferencia combinados bajo un mismo formalismo de representación.
3. El planteamiento de problemas mediante técnicas de aprendizaje también puede realizarse de una forma integrada. Su aplicación es muy útil en ciertas tareas para las que no existe una solución efectiva a priori. En concreto, se han utilizado ampliamente en tareas de clasificación.

7.8.2 Pistas de autoevaluación

1. Una recomendación genérica es que, teniendo presente la perspectiva introducida en el capítulo presente, repase los ejercicios planteados en temas precedentes, intentando justificar la técnica aplicada en la resolución de cada uno de éstos. Para descubrir la utilidad de la técnica aplicada se recomienda aplicar otra alternativa sobre el enunciado en cuestión. Por ejemplo, podría intentarse representar mediante marcos las entidades del ejercicio 5.13. En este caso, como contrapartida a las redes bayesianas, sería útil definir los demonios

que garantizaran las dependencias existentes entre los diversos atributos del dominio.

2. Modificar la solución del ejercicio 7.6 de tal forma que prime el mantenimiento de la coherencia entre los valores de las propiedades utilizadas. Para ello, realizar los cambios que permitan aplicar demonios de actualización inmediata de las relaciones: padres, abuelos y tíos a partir del cambio en la propiedad vivo.
3. Si la relación herederos del problema 7.6 dejara de ser un *slot* multivaluado y pasara a tener un único valor, ¿cómo afectaría este cambio al planteamiento del problema?, ¿en qué orden escribiría las reglas asociadas al mismo? Justifique su decisión.
4. ¿Cuáles serían los herederos de Ana López si Elena Rodríguez y Javier Rodríguez fallecieran? Describir el proceso de inferencia correspondiente según el enunciado del problema 7.6.
5. Modifique el esquema de representación del problema 7.6 y adapte las reglas de herencia para que se pudieran inferir correctamente los herederos de una persona suponiendo que el contexto social fuera un patriarcado en el que los primeros herederos serían siempre los varones, pasando a ser herederos las mujeres sólo en el caso de que no hubiera varones vivos.
6. Realizar los cambios necesarios en el problema 7.7.1 para que el sistema funcionara sin considerar la entidad objeto-fisico-slot. ¿Qué opción de representación le parece la más acertada y por qué?
7. En el problema 7.7.1 la mesa plegable tiene un peso 5 por defecto. Suponga que deja de existir el *subslot* valor-por-defecto. Realice los cambios necesarios para que dicho valor sea inferido siempre que la mesa no sea una mesa de comedor.
8. Suponga que en el problema 7.7.3.2 no tuviera en cuenta las coincidencias en los formularios alternativos del Calendario (véase la tabla 7-2) ¿Cómo representaría los diferentes tipos de solicitudes? ¿Qué opción es más eficiente? Justifique su respuesta.
9. Suponga que los eventos del Calendario en el problema 0 pudieran durar 30, 60, 90, 120 y 600 minutos. Las reuniones con becarios y alumnos duran 30

minutos. Las reuniones con los investigadores y profesores duran 60 minutos. Las reuniones con el Decano duran 90 minutos. Las salidas de la Universidad suelen durar el día entero, es decir, 600 minutos. Los seminarios duran 90 minutos. Los cursos duran 60 minutos. Considerando toda esta información, representar los elementos del sistema necesarios para poder inferir la duración de un evento.

10. Suponga que los eventos del Calendario en el problema 7.7.3.2 pudieran realizarse en Decanato, Despacho06, Despacho235, y en el Salón de Grados. Representar estos elementos junto con las reglas del dominio que permitieran inferir el lugar de celebración de los distintos tipos de eventos. Para ello considerar que los alumnos, investigadores, profesores y secretaría se reúnen en el Despacho06, los becarios en el Despacho235, el Decano en Decanato y los visitantes en el Salón de Grados.

Agradecimientos

Quisiéramos expresar nuestra más profunda gratitud al profesor Tom Mitchell, de la Universidad de Carnegie Mellon (CMU), Pittsburg, Pennsylvania (EEUU) y a todo el equipo de trabajo del “Calendar group”, por darnos la oportunidad de participar en el desarrollo de los llamados *sistemas aprendices*. Gracias también al profesor Jaime Carbonell y a Tom por apoyar y supervisar personalmente la participación en dicho trabajo.

REFERENCIAS

- [Aranda *et al.*, 1993] Aranda, J., Fdez. Marrón, J.L. y Morilla, F., *Lógica Matemática*. Sanz y Torres, 1993.
- [Borrajo *et al.*, 1993] Borrajo D., Juristo N., Martínez V, y Pazos J. *Inteligencia Artificial: Métodos y Técnicas*. Centro de Estudios Ramón Areces. Madrid, 1993.
- [Boticario, 1994] Boticario, J.G. *Contribuciones a la teoría computacional del aprendizaje: ejemplos de aplicación*. Tesis doctoral, Dpto. Informática y Automática. UNED, Madrid, 1994.
- [Brewka, 1991] Brewka, G., *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, 1991.
- [Carbonell *et al.*, 1990] Carbonell, J.G., Knoblock, C., y Minton, S., *PRODIGY: An integrated architecture for planning and learning*. En: VanLehn, K. (Ed.), *Architectures for intelligence*. Earlbaum, Hillsdale, NJ, 1990.
- [Clancey, 1985] Clancey, W.J., *Heuristic Classification*. Artificial Intelligence, 27(1985) 289-350.
- [Dent *et al.*, 1992] Dent, L., Boticario, J.G., McDermott, J., Mitchell T., y Zabowski, D., *A personal learning apprentice*. En: *Proceeding of the National Conference on Artificial Intelligence*. AAAI Press/MIT Press, San Jose, CA, 1992, pp. 96-103.
- [Díez, 1994] Díez Vegas, F.J., *Sistema Experto Bayesiano para Ecocardiografía*. Tesis Doctoral, Dpto. Informática y Automática, U.N.E.D., 1994.
- [Fah-Chun, 1996] Fah-Chun, C., *Internet agents: spiders, wanderers, brokers, and bots*. New Riders Publishing, Indianapolis, IN, 1996.
- [Gensler, 1990] Gensler, H.J., *Symbolic Logic, Classical and Advanced Systems*. Prentice-Hall, Inc., 1990.

- [Ginsberg, 1993] Ginsberg, M., *Essentials of artificial intelligence*. Morgan Kaufmann Publishers, San Francisco, CA, 1993.
- [GoldWorks III, 1989] *GoldWorks III Reference Manual*. Gold Hill Computers, Inc., 1989.
- [Hayes-Roth, 1993] Hayes-Roth, B., *On building integrated cognitive agents: A review of Newell's Unified Theory of Cognition*. Artificial Intelligence, **59** (1993) 329-341. Reimpreso en W. Clancey, S. Smoliar, M. Stefik (Eds.), *Contemplating Mind*, MIT Press, 1994.
- [Jackson, 1990] Jackson, P., *Introduction to Expert Systems*. Addison-Wesley Publishing Co., 1990. Second Edition.
- [Jackson et al., 1989] Jackson, P., Reichgelt, H., y Van Harmelen, F., *Logic-Based Knowledge Representation*. MIT Press, 1989.
- [Klir & Yuan, 1995] Klir, G.J., y Yuan, B., *Fuzzy Sets and Fuzzy Logic, Theory and Applications*. Prentice-Hall, Inc., 1995.
- [Laird & Rosenbloom, 1990] Laird, J.E., y Rosenbloom, P.S., *Integrating planning, execution, and learning in SOAR for external environments*. En: *Proceedings of the Eighth National Conference on Artificial Intelligence*, Vol 2. AAAI Press, Menlo Park, CA, 1990, pp. 1022-1029.
- [Lucas & Van der Gaag, 1991] Lucas, P., y Van der Gaag, L., *Principles of Expert Systems*. Addison-Wesley Publishing Co., 1991.
- [Mira, 1997] Mira, J., *Modelar Conocimiento como Tarea Básica en Inteligencia Artificial y en Redes Neuronales Artificiales*. VIII Cursos de Verano de la UNED sobre Aspectos Básicos de la Inteligencia Artificial y las Redes Neuronales. Tema 1, I-1 a I-25.
- [Mira et al., 1995] Mira J., Delgado A.E., Boticario J.G. y Díez J. *Aspectos Básicos de la Inteligencia Artificial*. Sanz y Torres. Madrid, 1995.
- [Mitchell et al., 1985] Mitchell, T. M., Mahadevan, S. y Steinberg, L. *LEAP: a learning apprentice system for VLSI design*. En: *Proceedings of the IJCAI-85 (International Joint Conference on Artificial Intelligence)*. Los Angeles 1985, pp. 573-580.
- [Mitchell, 1990] Mitchell, T. M., *Becoming increasingly reactive*. En: *Proceedings of the Eight National Conference on Artificial Intelligence*, AAAI-90. Boston, MA, 1990. MIT Press, pp. 1051-1058.
- [Mitchell et al., 1991] Mitchell, T.M., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., y Schlimmer, J., *THEO: A framework for self-improving systems*.

- In: VanLehn, K. (Ed.), *Architectures for Intelligence*, Erlbaum, Hillsdale, NJ, 1991.
- [Mitchell *et al.*, 1994] Mitchell, T.M., Caruana, R., Freigat, D., McDermott, J., y Zabowski, D., *Experience with a learning personal assistant*. Communications of the ACM 37, 80-91, 1994.
- [Moreno *et al.*, 1994] Moreno Ribas A. y otros. *Aprendizaje Automático*. Ediciones UPC (Universidad Politécnica de Cataluña). Barcelona, 1994.
- [Newell, 1981] Newell, A., *The Knowledge Level*. AI Magazine, verano (1981) 1-20.
- [Newell, 1990] Newell, A., *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press, 1990.
- [Nilsson, 1980] Nilsson, N.J., *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980. Traducción al castellano: *Principios de Inteligencia Artificial*. Díaz de Santos, Madrid, 1987.
- [Norvig, 1992] Norvig, P., *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann Publishers, Inc., 1992.
- [Pazos, 1987] Pazos J. *Inteligencia Artificial*. Paraninfo, Madrid, 1987.
- [Pearl, 1988] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988. Reimpreso con correcciones en 1991.
- [Quinlan, 1986] Quinlan J.R., *Induction of Decision Trees*. Machine Learning Vol. I, pp. 81-106, 1986.
- [Quinlan, 1987] Quinlan J.R., *Generating Production Rules from Decision Trees*. En: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-87)*. Milan, Italia, pp. 304-307, 1987.
- [Rich & Knight, 1991] Rich, E., y Knight, K., *Artificial Intelligence*. McGraw Hill, 1991. Traducción al castellano: *Inteligencia Artificial*. Segunda Edición. McGraw-Hill, Madrid, 1994.
- [Rules Element 4.0, 1996] *Intelligent Rules Element (version 4.0) Language Programmer's Guide*. Neuron Data, Inc., 1996.
- [Russell & Norvig, 1995] Russell, S., y Norvig, P., *Artificial intelligence: A modern approach*. Prentice-Hall International, New Jersey, 1995.
- [Shapiro, 1992] Shapiro, S.C. (Ed.), *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, New York, 1992. Second Edition.

- [VanLehn, 1991] VanLehn, K. (Ed.), *Architectures for Intelligence*, Erlbaum, Hillsdale, NJ, 1991.
- [Winston, 1992] Winston, P.H., *Artificial Intelligence*. Addison-Wesley Publishing Co., 1992. Third Edition. Traducción al castellano: *Inteligencia Artificial*. Tercera Edición. Addison-Wesley Iberoamericana, S.A., 1994.

LISTA DE ERRATAS del libro:

“Problemas Resueltos de Inteligencia Artificial Aplicada: Búsqueda y Representación”

Autores: Severino Fernández Galán, Jesús González Boticario y José Mira Mira

Editorial: Addison-Wesley

Año: 1998

(NOTA IMPORTANTE: Este libro fue descatalogado el 1 de Junio de 2015. Si desea informar a los autores sobre una errata en el mismo, puede hacerlo mandando un correo electrónico a seve@dia.uned.es.)

Versión: 7 de Septiembre de 2015

- En la portada hay que sustituir “Severiano” por “Severino”.
- En la página 34 falta un estado en la búsqueda ascendente de la figura 1-7, que permite obtener el estado (2, 3) a partir del operador 6. El estado mencionado es:
(3, 2) y (4,1) Casos particulares de...
 $(x, y) \ x+y=5$
- En la página 44 hay que sustituir:
“Como consecuencia de ello, deja de ser necesario revisar la existencia de posibles reorientaciones de enlaces.”
por:
“Como consecuencia de ello, deja de ser necesario revisar la existencia de posibles reorientaciones de enlaces **desde cualquier nodo previamente expandido en el proceso de búsqueda. (Nótese que un nodo expandido es aquel cuyos hijos ya han sido generados.)**”
- En la página 77 hay que intercambiar en la figura 2-37 el orden de expansión de Santander y Madrid: 2↔3.
- En la página 135 hay que sustituir en el apartado c) del Problema 3.15:
 $c(F \cap G) = c(F) \cup c(G)$
por:
 $c(F \cup G) = c(F) \cap c(G)$
- En la página 197 hay que realizar las siguientes sustituciones:
x por x_1
y por x_2
z por x_3
- En la página 199 hay que sustituir en la memoria de trabajo el elemento (**DE**) por (**ED**).
- En las páginas 238 y 239 hay que realizar las siguientes sustituciones:
 $P(e)$ por $P(s)$
 $P(s | e)$ por $P(e | s)$
 $P(n | t)$ por $P(n | t, s)$
- En la página 408 hay que sustituir traducción por traducción.