Sistemas basados en el conocimiento

PID_00267995

Vicenç Torra i Reventós

Revisión a cargo de
Jasmina Casals Terré

Tiempo mínimo de dedicación recomendado: 5 horas

Universitat Oberta de Catalunya

Vicenç Torra i Reventós **Jasmina Casals Terré**

La revisión de este recurso de aprendizaje UOC ha sido coordinada por el profesor: Carles Ventura Royo (2019)

Tercera edición: septiembre 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción..... 5

Objetivos..... 6

1. Introducción a los sistemas basados en el conocimiento..... 7
- 1.1. Apuntes sobre la construcción de un sistema basado en el conocimiento 9
  - 1.1.1. La construcción del modelo 9
  - 1.1.2. Validación y verificación 13

2. La representación del conocimiento..... 15
- 2.1. Niveles de un formalismo de representación del conocimiento 17
- 2.2. Aspecto formal y aspecto inferencial 19
- 2.3. Tipo de conocimiento 20
- 2.4. El aspecto inferencial y la búsqueda 21
- 2.5. Otros aspectos sobre los formalismos de representación del conocimiento 23

3. Sistemas basados en reglas..... 26
- 3.1. Aspecto inferencial en un sistema basado en reglas 28
- 3.2. Análisis de los sistemas basados en reglas 32

4. Sistemas con representación estructurada..... 34
- 4.1. Aspecto formal 35
- 4.2. Aspecto inferencial 39
- 4.3. Análisis de los sistemas de marcos 47

5. Sistema de razonamiento basado en casos..... 49
- 5.1. Aspecto inferencial 51

6. Sistemas de razonamiento basado en modelos..... 54
- 6.1. Aspecto formal 55
- 6.2. Aspecto inferencial 56

Actividades..... 59

Glosario..... 63

Bibliografía..... 64

Introducción

> *Heuristic Search Hypothesis*. The solutions to problems are represented as symbol structures. A physical symbol system exercises its intelligence in problem solving by search – that is, by generating and progressively modifying symbol structures until it produces a solution structure.
>
> A. Newell; H.A. Simon (1976). "Computer Science as Empirical Inquiry: Symbols and Search". *Communications of the ACM* (vol. 3, núm. 19, págs. 113-126).

El éxito de los sistemas basados en el conocimiento y, en particular, de los sistemas basados en reglas en los años ochenta fue una de las claves del auge de la inteligencia artificial. Varias aplicaciones mostraron la utilidad de estos sistemas cuando se trataba de problemas con un dominio limitado.

En este módulo, presentaremos los elementos principales para entender los sistemas basados en el conocimiento. El módulo está dividido en seis apartados.

El primer apartado constituye una introducción a los sistemas basados en el conocimiento. Se ven sus componentes principales y las fases más importantes que se deben tener en cuenta en su desarrollo.

A continuación se pasa a un segundo apartado en el que se plantea la representación del conocimiento. Empezamos definiendo qué es el conocimiento, para pasar después a definir los conceptos más importantes. También se describe aquí la cuestión de la búsqueda, y su papel en los sistemas de representación del conocimiento. Esta cuestión aparecerá más adelante al hablar de formas concretas de representación. El apartado acaba con algunos de los problemas relativos a la representación.

El apartado tercero está dedicado a los sistemas en los que el conocimiento está representado mediante reglas. Se describe el funcionamiento de este tipo de sistema.

El cuarto apartado introduce los mecanismos de representación estructurada centrándonos en los sistemas de marcos. Veremos aquí las características de los sistemas que utilizan esta representación.

El módulo sigue con dos apartados dedicados al razonamiento basado en casos y al razonamiento basado en modelos. Se describe el funcionamiento de estos sistemas, y se comparan sus características.

Objetivos

Este módulo didáctico quiere alcanzar los objetivos siguientes:

1. Obtener una visión general de lo que son los sistemas basados en el conocimiento.

2. Entender los sistemas basado en el conocimiento como herramientas para resolver problemas concretos.

3. Estudiar diferentes herramientas de representación del conocimiento.

4. Conocer los pros y contras de los métodos de representación.

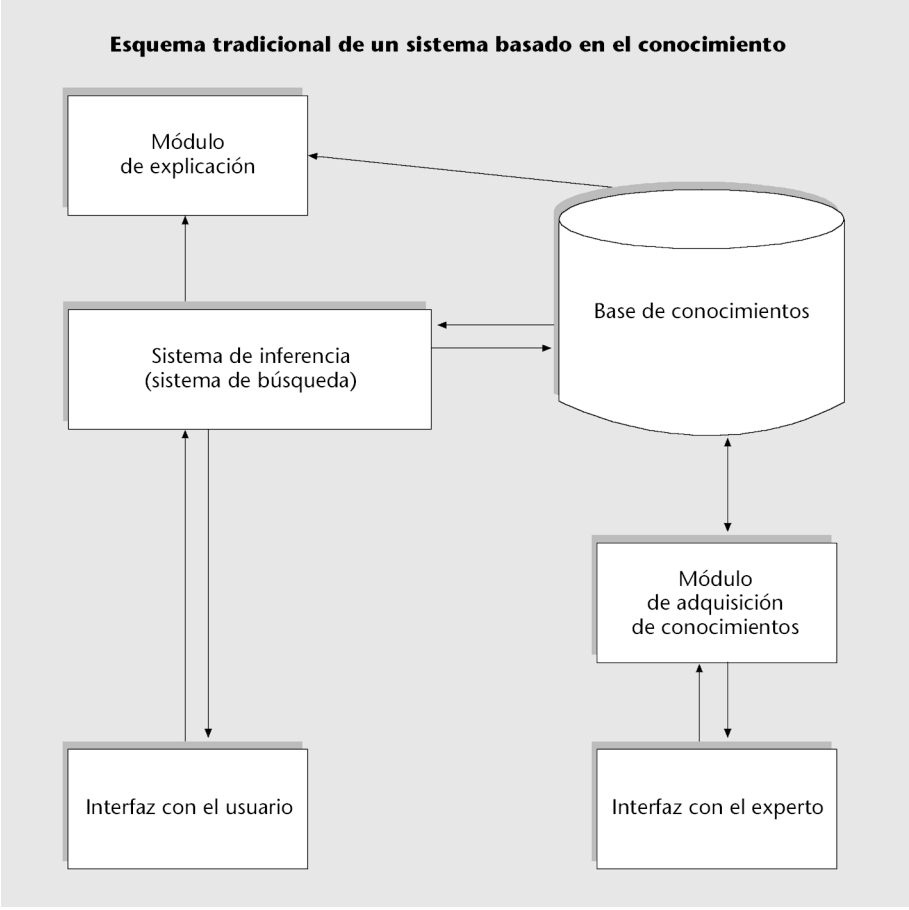
5. Valorar el papel que tiene la búsqueda en los sistemas basados en el conocimiento y en los mecanismos de representación del conocimiento.

1. Introducción a los sistemas basados en el conocimiento

Como su nombre indica, los sistemas basados en el conocimiento o SBC (*knowledge-based systems* o *knowledge systems*) resuelven los problemas utilizando de manera intensiva conocimiento del campo de aplicación.

Por ejemplo, en un sistema de diagnosis médica habrá información sobre las relaciones entre los síntomas y las enfermedades. Esta información es la que guiará la búsqueda cuando se plantee un problema concreto. Por ejemplo, cuál es la enfermedad que padece un determinado paciente.

Figura 1



Con el fin de tener en cuenta tanto el conocimiento como los mecanismos de búsqueda, un sistema basado en el conocimiento se ve tradicionalmente de acuerdo con la arquitectura de la figura 1. Esta figura subraya los principales módulos que lo integran:

a) Base de conocimientos: corresponde al almacén de conocimiento del sistema. Se guarda todo lo que resulta necesario para guiar el proceso dirigido a encontrar las soluciones.

b) Subsistema de inferencia: constituye la parte que razona sobre las soluciones de los problemas. El razonamiento se activa a partir, por ejemplo, de la pregunta de un usuario y todo el proceso de búsqueda está dirigido por la base de conocimientos a responder esta pregunta. A este subsistema a veces se le denomina *motor de inferencia*.

c) Interfaz con el usuario: es la que permite que los usuarios finales puedan interactuar con el sistema.

Además, a veces hay módulos de explicación y de adquisición del conocimiento. El primero permite construir una descripción de cómo se construye la solución. Para hacerla se basan en lo que hace el sistema de inferencia y en el conocimiento de la base de conocimientos. El módulo de adquisición del conocimiento ayuda en todo el proceso de construcción de la base de conocimientos.

Aunque la descomposición en módulos presentada permite subrayar los diferentes elementos de un sistema de estas características, las implementaciones reales de estos sistemas no mantienen siempre esta estructura. De todos modos, al hablar de las formas concretas de representación nos aparecerá un aspecto formal y un aspecto inferencial que más o menos corresponden a la base de conocimientos y al sistema de inferencia. También se verá que los dos aspectos se encuentran muy ligados y que a menudo no resultan independientes y, por lo tanto, no pueden formar módulos separados.

La ingeniería del conocimiento

El área de la inteligencia artificial que estudia el proceso de construcción de sistemas basados en el conocimiento es la llamada *ingeniería del conocimiento*, de manera análoga al hecho de que la ingeniería del software estudia la construcción de software.

En este módulo se describirán los elementos básicos de los sistemas basados en el conocimiento. Empezaremos con la descripción del proceso de construcción de estos sistemas. Una vez visto el proceso, veremos diferentes mecanismos de representación del conocimiento y describiremos el funcionamiento de algunos sistemas.

1.1. Apuntes sobre la construcción de un sistema basado en el conocimiento

La primera cuestión que se debe tratar antes de empezar la construcción de un sistema basado en el conocimiento es decidir si realmente necesitamos este sistema. A causa de que la construcción de un sistema de estas características requiere una buena inversión de tiempo y recursos, resulta conveniente evaluar si ésta constituye la opción mejor.

Lecturas complementarias

Para más detalles sobre la ingeniería del conocimiento podéis consultar las obras siguientes:

G. Guida; G. Tasso (1994). *Design and Development of Knowledge-Based Systems*. John Wiley and Sons.

M. Stefik (1995). *Introduction to Knowledge Systems*. Morgan Kauffman.

Algunas de las cuestiones que se deben tener en cuenta a la hora de tomar la decisión son éstas:

- Los problemas que se han de resolver no se pueden resolver con las técnicas de programación habitual.
- El dominio de la aplicación dispone de un conocimiento bien estructurado, y hay expertos que lo conocen.
- En los entornos en los que se necesita el conocimiento, no siempre se dispone de la experiencia humana.
- El coste del desarrollo del sistema basado en el conocimiento se debe poder justificar en términos de la utilidad posterior del sistema.

1.1.1. La construcción del modelo

Una vez se ha decidido que la construcción resulta conveniente, pasaremos a definir cuáles son los problemas que queremos resolver y a definir la base de conocimientos (BC)[1] convenientemente. La base de conocimientos deberá contener todo el conocimiento que el sistema necesita para resolver los problemas previstos. En general, existen dos maneras de conseguir este conocimiento: por medio de un experto (éste es el caso que se ha ilustrado en la figura 1) o mediante técnicas de aprendizaje.

[(1)]Abreviamos *base de conocimientos* con la sigla BC.

En el primer caso, durante el proceso denominado *adquisición del conocimiento*, un experto junto con el mencionado ingeniero del conocimiento formulan el conocimiento.

El proceso de adquisición de conocimiento consiste en la construcción de un modelo del campo donde se ha de aplicar el sistema (el dominio de aplicación) sobre la base de la experiencia del experto.

El experto es el que sabe del campo. Sabe qué problemas se plantean, y las técnicas apropiadas para resolverlos. Sabe cómo evaluar los datos y cómo tratar casos mal definidos o con incertidumbre. El ingeniero del conocimiento, en cambio, es el experto en las técnicas de inteligencia artificial y de representación del conocimiento. Las tareas principales de este último son seleccionar las herramientas adecuadas para la representación, ayudar al experto a formular el conocimiento necesario e implementarlo en una base de conocimientos de manera eficiente.

Para la construcción del modelo se pueden tener en cuenta las tres funciones que ha de satisfacer:

1) Un modelo es una abstracción que se utiliza para unos objetivos concretos (se utiliza en un marco concreto para llevar a cabo una determinada tarea). Cuando sabemos el objetivo del modelo, podemos realizar una aproximación del fenómeno que queremos modelizar, consiguiendo sólo la información que resulta relevante para conseguir el cumplimiento de la tarea. De esta manera, la complejidad del problema se reduce.

2) Un modelo representa una manera estructurada de comprender las entidades y los procesos que permiten construir la solución de una tarea en el mundo real.

3) Un modelo debe permitir comprender mejor el proceso así como también predecir.

El proceso de construcción de un sistema basado en el conocimiento y, en particular, del modelo no resulta una tarea fácil. Por una parte, porque la comunicación entre el experto y el ingeniero del conocimiento es difícil a causa de las diferencias de formación. Por otra, el modelo debe explicitar (y hacerlo de manera consistente) todo lo necesario para resolver el problema. También se han de tener en cuenta las dificultades inherentes a los formalismos de representación (por ejemplo, dificultades para expresar conocimiento incierto, relaciones espaciales, el problema de representar el cambio, etc.). Para facilitar este proceso se han desarrollado algunas metodologías. Algunas de estas metodologías son, por ejemplo, las llamadas KADS, Components of Expertise y Generic Tasks.

Representación del cambio: el problema del marco

El problema del marco (*frame problem*) aparece cuando intentamos representar los resultados de acciones y acontecimientos. El problema corresponde a la necesidad aparente de representar una gran cantidad de hechos obvios sobre cosas que no cambian cuando se efectúa una acción. Cuando un sistema utiliza estos hechos para hacer un razonamiento, resultará poco eficiente.

Aprendizaje

Hemos comentado antes dos alternativas para la construcción de la base de conocimientos. La primera consistía en que un experto y un ingeniero del conocimiento construyeran un modelo que permitiera resolver los problemas. La otra alternativa es la de utilizar técnicas de aprendizaje. Ahora, pasemos a comentar esta segunda posibilidad.

Las técnicas de aprendizaje permiten construir modelos de un dominio a partir de un conjunto de ejemplos.

Por ejemplo, los sistemas de aprendizaje inductivo formulan hipótesis que representan el conjunto de ejemplos que se le pasan. Estas formulaciones pueden ser usadas como una parte del conocimiento de una base de conocimientos.

Ejemplo de un sistema de aprendizaje

Supongamos que disponemos de cinco registros en un fichero descritos por cuatro variables ($v1$, $v2$, $v3$ y $v4$), y necesitamos expresar la cuarta variable en términos de las tres primeras. Entonces, un sistema de aprendizaje inductivo nos podrá ofrecer varias hipótesis sobre la relación (diferentes expresiones que nos relacionan las variables de acuerdo con los ejemplos). Por ejemplo, si tenemos los ejemplos:

Tabla 1

v1	*v2*	*v3*	*v4*
1	1	1	1
1	1	0	1
1	0	1	1
0	0	0	0
0	1	0	0

El sistema podría plantear que $v4$ es equivalente a $v1$, o que $v4$ es equivalente a $v1 \wedge (v2 \vee v3)$. Entonces, si nos encontramos con un nuevo registro con los valores correspondientes a las variables $v1$, $v2$ y $v3$ pero sin el de la variable $v4$, podemos conjeturar el valor de $v4$ con alguna de nuestras hipótesis.

Los sistemas de aprendizaje pueden ser aplicados a la construcción de un sistema basado en el conocimiento. En este caso, en lugar de necesitar que un experto nos construya un modelo (nos defina una relación entre las variables), podemos construir este modelo a partir de unos ejemplos.

Por ejemplo, considerad el caso de un sistema para la diagnosis de un circuito digital que dadas tres entradas binarias $v1$, $v2$ y $v3$, nos calcula una salida también binaria $v4$. Entonces, para construir el sistema de diagnosis podemos pedir al experto que nos dé la descripción del circuito (cómo se expresa $v4$ en términos de $v1$, $v2$ y $v3$). Sin embargo, si esto no resulta posible, podemos realizar unas pruebas iniciales con un circuito que funcione correctamente. Probamos qué es lo que da para unas cuantas combinaciones de valores de $v1$, $v2$ y $v3$. A partir de los resultados construimos un modelo (como hemos hecho antes) y utilizamos este modelo en el sistema de diagnosis.

Construcción de un sistema basado en el conocimiento

Queremos construir un sistema basado en el conocimiento para controlar la temperatura de un calorímetro. Consideraremos la temperatura a la que queremos el aparato y la temperatura a la cual está en este momento. Con esta información, nuestro sistema ha de decidir si quiere calentar o enfriar y cuánto para que la temperatura tienda hacia la temperatura deseada. Queremos conseguir llegar a la temperatura deseada lo más pronto posible y que continúe (que no se produzcan oscilaciones).

De acuerdo con lo que hemos visto, una manera de realizar esto es que un experto nos diga qué es lo que debemos hacer en cada situación. La otra alternativa que podemos considerar consiste en utilizar un método de aprendizaje. En este caso, lo que efectuamos es pedir al experto que nos enseñe qué hace en unas cuantas situaciones. Le planteamos unos objetivos: poner el calorímetro a la temperatura A, después ponerlo a la temperatura B. En este caso, si conectamos un ordenador al sistema y registramos los datos del sistema y de lo que aplica el experto, podemos aplicar un método de aprendizaje que nos determine qué hemos de hacer en cada situación.

Cabe subrayar que, aunque los métodos de aprendizaje construyen modelos que podemos utilizar en sistemas basados en el conocimiento, estos modelos presentan limitaciones. Los sistemas construidos deberán tener presentes estas limitaciones. Un aspecto importante que hay que tener en cuenta es que los ejemplos que utilizamos en el aprendizaje son un subconjunto de las situaciones en las que el sistema se encontrará cuando opere realmente. Esto hará que si el modelo seleccionado no resulta del todo adecuado, el sistema basado en el conocimiento no funcionará correctamente en todas las situaciones.

Limitaciones de los modelos

En el ejemplo de las cuatro variables booleanas, hemos considerado dos equivalencias para la variable $v4$. Si el sistema basado en el conocimiento toma como modelo que la variable $v4$ es equivalente a $v1$, cuando el circuito digital da como resultado $v4 = 0$ en la entrada $v1 = 1$, $v2 = 0$ y $v3 = 0$, tendremos que señala un error del comportamiento del sistema. En este caso, si hubiéramos tomado la hipótesis de que la variable $v4$ es igual a $v1 \wedge (v2 \vee v3)$, entonces el modelo nos diría que el sistema funciona correctamente.

Aquí hemos hablado del aprendizaje sin entrar en detalles de los métodos existentes. De hecho, dado que existen muchos formalismos de representación del conocimiento, se han desarrollado diferentes algoritmos de aprendizaje para generar hipótesis en los diferentes formalismos. Además, para un mismo formalismo hay algoritmos que difieren en las propiedades del modelo que construyen. En el módulo 5 se introducen varias técnicas de aprendizaje automático que son aplicables en la construcción de un modelo basado en conocimiento.

Compartir el conocimiento

Hemos considerado aquí el problema de la definición de los sistemas basados en el conocimiento suponiendo que el conocimiento que necesita el sistema es suministrado por un experto o es aprendido a partir de ejemplos. Hemos supuesto que para cada sistema partimos de cero con independencia de que existan otras aplicaciones en el mismo dominio.

Esta manera –que resulta frecuente– de definir los sistemas basados en el conocimiento, provoca duplicación del trabajo y que aumenten los costes de construcción a medida que los sistemas son mayores.

Con el nombre de compartir el conocimiento[2] se engloban todos los aspectos relativos al hecho de compartir el conocimiento entre diferentes sistemas de manera que no resulte necesario recodificarlo cuando se empiece a construir un nuevo sistema.

(2)En inglés, *knowledge sharing*.

Dentro de este campo, se han estudiado los impedimentos existentes para compartir el conocimiento, y se consideran varios campos de acción:

1) Mecanismos para traducir el conocimiento formulado mediante diferentes formalismos de representación: a causa de que existen muchas familias de lenguajes de representación, y hay muchos lenguajes en cada familia, es impracticable construir traductores para todos los pares de lenguajes. El número de traductores es cuadrático en relación con el número de lenguajes. La alternativa más adecuada es definir un lenguaje intermedio (una *interlingua*) de manera que la representación del conocimiento común (o del conocimiento que se comunican) se lleve a cabo por medio de este lenguaje. De esta manera, el número de traductores es lineal en relación con el número de lenguajes.

2) Especificación de sistemas de representación del conocimiento: dentro de una misma familia de lenguajes existen diferencias en la semántica. Estas diferencias se deben principalmente a la necesidad de resolver los conflictos entre el poder de expresión, la compleción en la inferencia y los recursos disponibles. Actualmente se estudian definiciones de especificaciones comunes a todos los lenguajes de una misma familia.

Ved también

Los sistemas multiagentes se estudian con más profundidad en la asignatura optativa *Aprendizaje computacional*.

El tema de compartir conocimiento no resulta sólo importante cuándo se quieren construir diferentes agentes autónomos (independientes) en un mismo dominio para no haber de recodificar el conocimiento, sino también cuando se construyen sistemas distribuidos que trabajarán de manera coordinada y simultáneamente. Esto es, en el caso de construir sistemas multiagentes.

1.1.2. Validación y verificación

Un aspecto ligado a los sistemas basados en el conocimiento y al proceso de construcción lo constituye la validación y verificación. Esto, como en el caso de la ingeniería del software, consiste en comprobar que el programa satisface las necesidades del usuario. Sin embargo, las técnicas que se utilizan en aquella área no son útiles aquí porque los dos entornos no son equivalentes. Una de las diferencias es que los procedimientos de inferencia de un sistema basado en el conocimiento son simples, pero el comportamiento real depende del co-

nocimiento escondido en la base de conocimiento. Esto provoca que resulte difícil separar el software y el modelo. Otra de las diferencias es que los requerimientos de usuario están mal definidos (la especificación es incompleta).

Por ello, ha sido necesario desarrollar herramientas propias. Estas herramientas las podemos dividir en dos grupos: las que analizan la estructura interna del sistema y las que analizan su conducta externa. Entre las primeras, se encuentran las que se utilizan para comprobar que la base de conocimientos no es inconsistente y que la representación del conocimiento no permite que el proceso deductivo genere ciclos. Entre las segundas, se sitúan las que analizan la conducta del sistema mediante, por ejemplo, técnicas de test. Es decir, experimentar con un conjunto de casos cuáles son las conclusiones del sistema. Una vez se tienen las respuestas que da el sistema, se compararán con las soluciones correctas o con aquellas soluciones que un experto da para los mismos casos.

Bases de conocimientos inconsistente y cíclica

Una base de conocimientos inconsistente es aquélla en la que podemos deducir una propiedad a y también su negación $\neg a$.

Una base de conocimientos cíclica es la que para deducir un hecho necesita deducir otro que necesita después el primero. Un ejemplo es el de las dos implicaciones siguientes a $\leftarrow b$ y $b \leftarrow$ a. Para decidir a necesitamos deducir b y para deducir b necesitamos a.

2. La representación del conocimiento

Una cuestión fundamental en los sistemas basados en el conocimiento es la representación del conocimiento, ya que es lo que permite a los sistemas saber cosas sobre el mundo que los rodea (sobre el dominio de aplicación). Por ello, se han desarrollado formalismos para representar la información que necesitan y herramientas para manipularla. La representación del conocimiento es el campo de la inteligencia artificial que estudia estos formalismos y herramientas.

Antes de pasar a ver los diferentes formalismos, se detallarán algunos conceptos generales de este campo. En primer lugar, consideramos qué es la representación del conocimiento y qué factores toman parte en ella. Daremos para ello dos definiciones:

1) La primera nos permite distinguir conocimiento de información y de datos:

> "One usually makes a distinction between "data" and "information". Data consists of raw figures, measurements, and files that do not necessarily answer the questions that its users may have. Information, on the other hand, is somewhat more refined. It is often the result of processing crude data, giving useful statistics for the data, or answering specific questions posed by users. In AI we usually distinguish a third kind of order in memory: "knowledge". We think of knowledge as a refined kind of information, often more general than that found in conventional databases. But it may be incomplete or fuzzy as well. We may think of knowledge as a collection of related facts, procedures, models and heuristics than can be used in problem solving or inference systems. Knowledge may be regarded as information in context, as information organized so that it can be readily applied to solving problems, perception and learning."
>
> S. L. Tanimoto (1987). *The elements of artificial intelligence* (pág. 89). Computer Science Press.

2) La segunda definición nos describe los factores que se deben tener en cuenta en la representación del conocimiento:

"What is a knowledge representation? We argue that the notion can best be understood in terms of five distinctive roles it plays, each crucial to the task at hand:

- A knowledge representation (KR) is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it.
- It is a set of ontological commitments, i.e., an answer to the question: In what terms should I think about the world? (...) The commitments are in effect a strong pair of glasses that determine what we can see, bringing some part of the world into sharp focus, at the expense of blurring other parts.
- It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (i) the representation's fundamental conception of intelligent reasoning; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends (...).
- It is a medium for pragmatically efficient computation, i.e., the computational environment in which thinking is accomplished. One contribution to this pragmatic efficiency is supplied by the guidance a representation provides for organizing information so as to facilitate making the recommended inferences.
- It is a medium of human expression, i.e., a language in which we say things about the world."

R. Davis; H. Shrobe; P. Szolovits (1993). "What is a Knowledge Representation?" *AI Magazine* (vol. 1, núm. 14, págs. 17-33).

Lectura complementaria

Podéis encontrar el trabajo de Davies, Schrobe y Szolovits en la dirección siguiente:

http://www.medg.lcs.mit.edu/ftp/psz/k-rep.html

Como se afirma en esta última definición, la representación (como herramienta de cálculo), afecta a la eficiencia del sistema (el espacio de búsqueda se puede reducir: menos número de estados o menos número de operadores) y nos determina aquello que se puede inferir (qué es lo que se puede calcular y qué es lo que no).

Una determinada representación nos puede simplificar enormemente el proceso de encontrar la solución a un problema, o bien nos puede impedir encontrarle una solución.

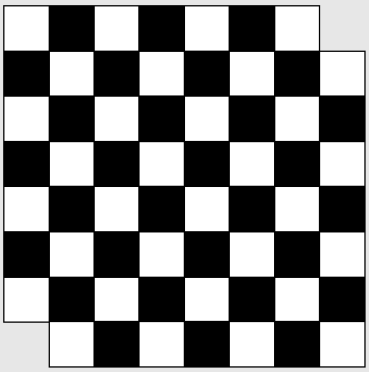
El tablero de ajedrez recortado

Un ejemplo bien conocido de que una buena representación puede simplificar el problema lo constituye el que planteamos a continuación.

Se considera el tablero de la figura siguiente y se quiere recubrir con unas fichas de dominó que ocupan exactamente dos posiciones del tablero. El recubrimiento se debe llevar a cabo de manera que no quede ninguna posición vacía y sin que se produzca solapamiento entre las piezas. La cuestión es, ¿tiene solución este problema?

Figura 2

El tablero de ajedrez recortado



Si sólo consideramos la forma del tablero y la de las piezas, deberemos ir pensando posibles ubicaciones de piezas en el tablero hasta que el problema esté resuelto, o bien que se hayan probado todas las combinaciones y sepamos que ninguna recubre el tablero.

Otra alternativa consiste en considerar también el color de las posiciones del tablero. En este caso, si contamos el número de posiciones blancas y de posiciones negras en el tablero (hay treinta y dos blancas y treinta negras) y se tiene en cuenta que una ficha de dominó ocupará siempre una posición blanca y una negra, se hace evidente la respuesta.

Como se ha señalado antes, en este módulo se describirán algunos de los formalismos de representación del conocimiento existentes. En particular, se presentarán las reglas y los marcos (como uno de los formalismos de representación estructurada). Además de éstos, también podemos utilizar algún tipo de lógica. Por ejemplo, podemos utilizar una lógica de primer orden. En este caso, el conocimiento vendrá expresado mediante fórmulas en la lógica elegida.

Ved también

Para más detalles sobre los diferentes tipos de lógica, podéis ver el módulo de lógica de predicados de la asignatura *Lógica*.

2.1. Niveles de un formalismo de representación del conocimiento

Un formalismo de representación del conocimiento se puede ver en tres niveles diferentes de abstracción: de conocimiento, lógico y de implementación.

1) Nivel de conocimiento (o epistemológico): este nivel es el que utiliza un agente para interaccionar con su formalismo de representación del conocimiento. Desde un punto de vista abstracto, nos interesa saber aquello que sabe (que se puede deducir) sin tener en cuenta cómo está implementado.

2) Nivel lógico: en este nivel tenemos la codificación del conocimiento en sentencias concretas. Aquí nos interesan las propiedades lógicas del formalismo. Esto es, cuál es su significado, y su capacidad de expresión. Por ejemplo, nos podemos plantear si podemos expresar que un determinado hecho se cumple para todos los individuos de una determinada población. Si queremos utilizar lógica de orden cero, esto no se podrá representar porque no hay cuantificadores. En cambio, con la lógica de primer orden sí que se podrá. Si utilizamos una lógica como un formalismo de representación, también consideraremos en este nivel las propiedades de los procedimientos de inferencia lógica. Por ejemplo, si el método es sólido o completo.

Métodos sólidos y completos

Recordad que un método es sólido cuando sólo se generan oraciones implicadas.

Recordad que un método es completo cuando se pueden deducir todas las consecuencias de un sistema formal.

3) Nivel de símbolo (o implementación): en este nivel se trata de cómo representar el conocimiento en un ordenador. Esto es, qué estructuras de datos son adecuadas para representar lo que ha de saber el sistema y qué algoritmos resultan adecuados para la inferencia. Por ejemplo, en el caso de la lógica se debería elegir una representación para los predicados y otra para las fórmulas. También se deberían implementar los algoritmos para efectuar la inferencia. Por ejemplo, la resolución.

Esta división en niveles de abstracción nos permite considerar los requerimientos de un formalismo de la representación de la misma manera.

1) Nivel de conocimiento: desde el punto de vista de que el sistema debe interaccionar con el usuario, ha de ser posible expresar el conocimiento en niveles de detalle diferentes. Por ejemplo, para un sistema de razonamiento general sobre el universo, se debe poder representar que los planetas dan vueltas en torno al sol. En un sistema para físicos sobre la misma cuestión, será necesario considerar las ecuaciones de los movimientos de los cuerpos en el espacio.

2) Nivel lógico: un requerimiento es que en una representación la semántica (el significado) de las expresiones ha de resultar clara. Las expresiones bien formadas deben tener un significado comprensible. Se aplica aquí el llamado *principio de composicionalidad* o *principio de Gottlob Frege* que dice:

> "El significado de una frase o una sentencia es una función del significado de sus partes y de la manera en la que se combinan."

En el caso de la lógica, una fórmula de la forma $p(a) \rightarrow q(b)$ tiene significado en términos del significado de $p(a)$, de $q(b)$ y de $\rightarrow$. Otro requerimiento será que las conclusiones del sistema sean sólidas en relación con el conocimiento. También se pide que el conocimiento se pueda expresar de manera modular, de modo que no haya de ser considerado todo a la vez. Esto facilitará las tareas de corrección y actualización de la base de conocimientos.

3) Nivel de implementación: los requerimientos para el nivel de implementación corresponden a la eficiencia de los procesos. Necesitaremos que los mecanismos resulten eficientes con respecto al tiempo (tanto el acceso, como la

(3)En inglés, *hash tables*.

inferencia) y en cuanto al espacio. Por ejemplo, se deberán considerar mecanismos de indexación (los mecanismos que permiten encontrar un predicado en la base de conocimientos) que resulten eficientes a fin de que cuando el número de predicados sea grande se encuentre uno rápidamente. Típicamente, utilizaremos para la indexación tablas de dispersión[3] o árboles (binarios) de búsqueda.

Aparte de estos requerimientos, también deberemos tener en cuenta que se dé buena sintonía entre los niveles. Por ejemplo, que la implementación prevea las propiedades que exigimos para el nivel lógico.

El número de niveles de un sistema de representación del conocimiento

El número de niveles de un sistema de representación del conocimiento y de cuáles son estos niveles no es una cuestión resuelta. No todos los autores coinciden. Por ejemplo, hay quien siguiendo a Newell (A. Newell [1982]. "The Knowledge Level". *Artificial Intelligence* (núm. 18, págs. 87-127) considera sólo el nivel de conocimiento y el nivel de símbolo. En relación con el nivel lógico considerado aquí Uschold afirma:

"It is not clear how to view the logical level. From a programming perspective, it seems far from implementation concerns, and thus at the knowledge level. However, from the perspective of building a knowledge base, it is not unreasonable to view a formal representation at the logical level as an implementation of structures identified at the conceptual, ontological, and epistemological levels."

M. Uschold (1998). *Knowledge level modelling: concepts and terminology*.

2.2. Aspecto formal y aspecto inferencial

En todo formalismo de representación del conocimiento se pueden distinguir dos aspectos: uno formal y uno inferencial.

El aspecto formal corresponde a cómo se almacena la información cuando la tenemos representada de manera explícita: cuál es el formalismo que se utiliza.

De hecho, en este aspecto caben todas las cuestiones relacionadas con la sintaxis y la semántica del formalismo.

El aspecto inferencial corresponde a cómo se obtiene la información que se encuentra en el sistema pero sólo de manera implícita.

En este caso, habrá una serie de reglas abstractas (que no dependerán del conocimiento almacenado) que permitirán obtener la información a partir de la que es explícita. Resulta importante subrayar que aunque el término *inferencial* está sacado de la lógica, no nos restringimos a estos tipos de sistemas.

En el caso de la lógica, tendremos que el aspecto formal corresponde a cómo se forman las expresiones: cuáles son las expresiones primitivas (el vocabulario: constantes, variables, predicados) y cómo se construyen las fórmulas. El aspecto inferencial tratará los métodos que permiten demostrar un predicado como, por ejemplo, la deducción natural o la resolución.

Estos dos aspectos corresponden aproximadamente a dos de los módulos que habían aparecido en la arquitectura de la figura 1: la base de conocimientos y el subsistema de inferencia. Al describir la arquitectura de la figura se ha comentado que esta división en módulos no siempre resulta posible a causa de la estrecha relación que existe entre la base de conocimientos y los procedimientos de inferencia. Evidentemente, esta relación también se da aquí entre los dos aspectos de un formalismo de representación. El aspecto formal define las expresiones primitivas y cómo se construyen las expresiones complejas a partir de éstas. Sobre la base de éstas primitivas y de cómo se construyen las expresiones complejas se definirá la inferencia.

De todos modos, resulta importante diferenciar entre los dos aspectos porque a la hora de comparar mecanismos de representación del conocimiento cada aspecto tiene asociadas propiedades diferentes. Así, algunos de los requerimientos considerados en el nivel lógico se pueden dividir por aspectos. Por ejemplo, la expresividad recaerá sobre el aspecto formal y la solidez (o la completitud) sobre lo que es inferencial.

2.3. Tipo de conocimiento

Un sistema puede incluir diferentes tipos de conocimiento. Una primera división es la que nos permite distinguir entre el conocimiento de dominio y el conocimiento estratégico. Estos tipos de conocimiento los podemos formular así:

a) Conocimiento de dominio: lo que corresponde al conocimiento general sobre el campo de aplicación del sistema.

b) Conocimiento estratégico: lo que describe cómo utilizar el conocimiento de dominio para resolver un determinado problema.

En un sistema de diagnosis médica, el conocimiento de dominio corresponderá a todo lo relativo a las enfermedades que trata el sistema y el conocimiento estratégico a cómo se debe enfocar el caso concreto de un enfermo para encontrar una diagnosis para su enfermedad.

Otra división es la que distingue entre conocimiento declarativo y conocimiento procedimental.

a) El conocimiento procedimental: es el que se encuentra ligado a una utilización concreta. Por ejemplo, podemos definir una función para calcular el factorial de un número. Vista la función como un conocimiento hecho explícito, tenemos que este conocimiento sólo se puede utilizar para calcular el factorial de un número. La función no nos permite saber cuál es el número que tiene como factorial el 150, o si el factorial de 3 es 54.

b) El conocimiento declarativo: es el que es independiente de su uso. Si expresamos la relación entre un número y su factorial mediante lógica proposicional, entonces podemos utilizar el conocimiento de diferentes maneras. En particular, podemos razonar sobre qué número tiene como factorial el 150 y si el factorial de 3 es 54. En este caso, el conocimiento nos permite resolver diferentes tipos de preguntas. Algunas de las cuales quizá no habían sido consideradas en el diseño inicial de la base de conocimientos.

PROLOG

El lenguaje lógico PROLOG, que se llama declarativo, nos permite definir el factorial de manera que todas estas cuestiones se pueden formular y resolver.

Ejemplo de conocimiento declarativo

Un ejemplo de este tipo de conocimiento lo tenemos cuando consideramos el modelo del ejemplo de aprendizaje del subapartado 1.1.1. Si tenemos como modelo $v4 = v1 \wedge (v2 \vee v3)$ podremos razonar sobre el valor de $v4$ según el valor de las otras variables, pero también podemos razonar sobre el valor de las otras variables cuando sabemos el de $v4$. Por ejemplo, si sabemos que $v4$ es cero, $v1$ es 1, entonces podemos deducir que tanto $v2$ como $v3$ tienen que ser cero.

Si en lugar de este modelo tenemos una implementación de la función lógica en un lenguaje concreto, esto no será posible. Deberemos aplicar la función tal como la hemos diseñado.

Conocimiento declarativo:

$$v4 = v1 \wedge (v2 \vee v3)$$

Conocimiento estratégico:

```
funcion expresión (v1: booleano; v2; booleano; v3: booleano) retorna booleano es
            retorna (v1 and (v2 or v3))
ffuncion.
```

2.4. El aspecto inferencial y la búsqueda

Como se ha dicho, uno de los aspectos de un sistema de representación del conocimiento lo constituye el aspecto inferencial.

Ved también

Podéis ver los conceptos de *resolución de problemas* y de *búsqueda* en el módulo "Resolución de problemas y búsqueda" de esta asignatura.

En la implementación de los mecanismos de inferencia aparecen problemas de búsqueda. Esto es así porque la inferencia se hará para cumplir un objetivo y en el proceso de cumplirlo se nos presentarán a menudo diferentes alternativas de las cuales deberemos elegir una. Al entorno de la representación del conocimiento en el problema de la búsqueda se le denomina a veces el problema del control.

Por ejemplo, en el caso de utilizar la lógica de enunciados como herramienta de representación del conocimiento y la deducción natural como forma de inferencia, tendremos que el objetivo puede ser la demostración de un enunciado. Para realizar la demostración llevaremos a cabo un conjunto de pasos, y en cada uno de ellos seleccionaremos una regla de entre las que podemos aplicar. Como en todo problema de búsqueda, no todas las alternativas que podemos aplicar serán buenas. Habrá reglas que nos llevarán hacia la demostración del enunciado y habrá otras que no. Aquí interesará, como en los problemas de búsqueda, seleccionar las reglas que nos llevan hacia el objetivo lo antes posible. Más adelante veremos que esto también ocurre en los otros mecanismos de representación.

Los diferentes tipos de reglas

Recordemos que existen dos tipos de reglas en la lógica de enunciados: las reglas de introducción y las reglas de eliminación. Por ejemplo, regla de introducción de la conjunción, de la disyunción... y reglas de eliminación de la conjunción, de la disyunción...

Se ha comentado antes de que el conocimiento estratégico describe cómo utilizar el conocimiento de dominio para resolver un determinado problema. Esto corresponde, de hecho, a guiar la búsqueda a fin de que nos acerquemos hacia el objetivo lo más rápidamente posible.

En ocasiones, este conocimiento se encontrará de manera implícita en el aspecto inferencial del formalismo de representación del conocimiento, pero a veces aparece de manera explícita. Por ejemplo, en el caso de la lógica y la deducción natural podemos tener una implementación que para decidir qué regla se debe aplicar siga un esquema prefijado. Por ejemplo, primero se intenta introducir una conjunción, después si esto no va bien, se intenta introducir una disyunción... Esto corresponde a un conocimiento estratégico expresado de manera procedimental porque para cambiar cómo se realiza la elección se ha de rescribir de nuevo el mecanismo de inferencia. Otra alternativa es que el sistema a la hora de decidir qué regla aplica utilice un conjunto de recomendaciones de manera explícita. Por ejemplo, recomendaciones en la línea de los consejos prácticos que se dan en la asignatura de lógica para demostrar un enunciado:

PROLOG

El lenguaje PROLOG, visto desde este punto de vista, utiliza un esquema prefijado o la forma de realizar la inferencia sigue siempre un mismo esquema.

Ved también

Podéis ver los consejos prácticos que se dan en el módulo "Lógica de enunciados" de la asignatura *Lógica*.

Consejo 4.a) Si en las premisas aparecen implicaciones, puede ir bien intentar obtener sus consecuentes aplicando la regla de eliminación de la implicación. Para esto puede ser necesario resolver antes el subproblema de obtener el antecedente.

Consejo 4.b) Si en las premisas aparecen disyunciones, puede ser apropiado plantear todo el problema como una prueba por casos, aplicando la regla de eliminación de la disyunción.

En este caso, un cambio en las recomendaciones provocará un cambio en la forma de selección de las reglas, sin que sea necesario cambiar el mecanismo de inferencia.

El hecho de tener el conocimiento estratégico de manera explícita presenta algunas ventajas:

a) Se puede utilizar conocimiento estratégico específico de un dominio, de manera que al cambiar el dominio podemos cambiar la forma de afrontar un problema. Por ejemplo, en diagnosis médica podemos considerar que si una enfermedad resulta más frecuente que otra, empezamos comprobando la enfermedad más frecuente. Si pasamos a un problema de diagnosis de averías de coches, podemos utilizar otras heurísticas. Cuando el conocimiento estratégico es explícito, podremos cambiar de dominio sin cambiar el mecanismo de inferencia: será suficiente cambiar este conocimiento.

b) Se puede reutilizar el conocimiento de dominio en aplicaciones diferentes. Cuando cambia la aplicación pero el dominio se mantiene, cambiará la parte del conocimiento estratégico. El conocimiento del dominio se puede mantener. Por ejemplo, un sistema de diagnosis médica y un sistema para ayudar a que los estudiantes de medicina aprendan sobre los síntomas de una enfermedad puede utilizar el mismo conocimiento sobre las relaciones síntomas-enfermedades. La manera de realizar las deducciones deberá ser diferente al deducir las primeras sobre enfermedades y las segundas sobre síntomas.

c) Uno de los usos de los sistemas basados en el conocimiento es dar, además de la conclusión, una explicación del por qué aquella solución es la seleccionada. El conocimiento estratégico ayuda en la construcción de la explicación. Nos dice cómo debemos realizar las selecciones.

2.5. Otros aspectos sobre los formalismos de representación del conocimiento

El desarrollo de sistemas basados en el conocimiento ha obligado a desarrollar métodos de razonamiento para tratar los diferentes tipos de situaciones en los que se pueden encontrar. Por ejemplo, algunos sistemas han de poder razonar sobre el tiempo (sobre procesos que se alargan en el tiempo, que son anteriores o simultáneos); otros deben tratar con falta de conocimiento (no se conoce toda la información que sería deseable para resolver un problema). También los hay que deben tener en cuenta lo que saben los demás con el fin de tomar decisiones. A continuación se repasan algunas técnicas que resultan adecuadas para algunos tipos de problemas:

a) Razonamiento temporal: estas técnicas resultan adecuadas cuando un sistema ha de considerar procesos con una extensión temporal, y debe ser capaz de razonar sobre éstos. Para tratar estos aspectos existen en la actualidad diferentes formalismos. Algunos de los problemas a los que se dirigen son la necesidad de representar información temporal parcial o de razonar sobre posibles pasados o futuros.

b) Razonamiento espacial: el espacio constituye otro aspecto que es necesario considerar en algunos sistemas. Por ejemplo, han de poder razonar sobre la posición de los objetos o la forma. Algunos deben permitir tratar representaciones a diferente niveles.

c) Metarrazonamiento: a veces no es suficiente razonar sobre aspectos externos, sino que resulta también necesario razonar sobre el conocimiento. Tanto sobre el propio conocimiento, como sobre el conocimiento que poseen los demás. También deben tener en cuenta el llamado conocimiento común (el conocimiento que comparten de manera implícita un grupo de agentes). La lógica epistémica trata este tipo de razonamiento.

d) Razonamiento con información incompleta: la información deseable para resolver un problema no se encuentra siempre disponible. Hay ocasiones en las que algunos hechos se desconocen pero aun así resulta necesario razonar y tomar decisiones. Para tratar estas situaciones, se han desarrollado diferentes herramientas. Podemos subrayar los métodos de razonamiento monótono, de razonamiento bajo incertidumbre y de razonamiento aproximado. Los métodos de razonamiento bajo incertidumbre y de razonamiento aproximado se explican con detalle en el módulo 4 de esta asignatura.

Los métodos de razonamiento no monótono permiten realizar inferencias no sólidas (inferencias que extraen conocimiento nuevo) a partir del conocimiento incorporado en el sistema. Este conocimiento se expresa utilizando reglas no estándar, por ejemplo, las reglas por defecto[4]. Este tipo se denomina *no monótono* porque la incorporación de nuevo conocimiento al sistema no implica que se pueda deducir más de lo que se podía antes. Recordemos que en la lógica, cuando se incorporan nuevos hechos, siempre se pueden obtener nuevas consecuencias (que se añaden a las anteriores). Por lo tanto, existe monotonía al añadir información al sistema.

(4)En inglés, *default rules*.

En los métodos en los que utilizamos reglas por defecto, esto no es de esta manera porque se pueden haber aplicado inferencias no sólidas. Entonces, cuando se obtiene una nueva información, puede ser que ésta se encuentre en contradicción con las suposiciones que se han realizado. Por lo tanto, las conclusiones obtenidas previamente se deberán revisar. Por esta razón se denomina *razonamiento no monótono*. Los mecanismos de razonamiento no monótono están explícitos en algunos de los lenguajes de programación declarativos para la programación lógica y de bases de datos deductivas.

La necesidad de que la información de un sistema sea consistente aunque la inferencia no sea sólida o se utilicen reglas por defecto llevó al diseño de los llamados *sistemas de mantenimiento de la verdad* (TMS)[5]. Estos sistemas tienen constancia de los pasos de deducción y, cuando algunas proposiciones dejan de ser válidas, las abandonan. Es necesario tener en cuenta que si un sistema ha deducido A a partir de una regla por defecto, y en un paso posterior hemos

(5)TMS es la sigla de la expresión inglesa *Truth Maintenance Systems*.

deducido B a partir de A y de otra información, cuando nuevo conocimiento nos hace reconsiderar A no es suficiente con retirar A del conjunto de hechos conocidos, sino que también deberemos retirar B. Un sistema de mantenimiento de la verdad al retirar A también retiraría B.

El problema del sentido común

Se dice que los sistemas basados en el conocimiento son frágiles porque fuera de su dominio de aplicación no funcionan bien y además no saben cuál es la frontera de su conocimiento. Esto es, que no pueden detectar cuándo una situación se encuentra fuera del conocimiento que tienen incorporado y que por lo tanto sus inferencias no resultan útiles. Esto también provoca la llamada *falta de sentido común*: en algunas ocasiones, las respuestas del sistema son absurdas porque falta conocimiento que podríamos denominar elemental. Hechos y relaciones que todo el mundo tiene asumidos: falta de sentido común.

Hay investigadores que creen que este problema se puede resolver elaborando bases de conocimiento mayores. La construcción de bases de conocimiento a gran escala sigue esta línea para conseguir comportamientos inteligentes.

Éste es el caso del proyecto CYC que dirige Lenat. El sistema pretende capturar todo el conocimiento –tanto explícito, como implícito– que aparece en un conjunto de entradas de la Enciclopedia Británica. El interés principal reside en formular la información subyacente que los autores de los artículos suponen a sus lectores. Para conseguirlo, a finales de 1997 se habían incluido ya más de 4.000.000 aserciones. Se espera que la formulación del conocimiento implícito permita al sistema leer el resto de la enciclopedia y capturar el sentido sin intervención humana. CYC constituye un proyecto ambicioso, que cuando empezó (en el año 1984) debía durar diez años y costar cincuenta millones de dólares. Sin embargo, todavía hoy no se cumplen los objetivos marcados al inicio del proyecto.

Lectura complementaria

Encontraréis información sobre el proyecto CYC en la dirección siguiente:

http://www.cyc.com/

Los resultados obtenidos hasta hoy han motivado que esta aproximación a la inteligencia artificial haya sido fuertemente criticada. Esta crítica ha venido especialmente de entre aquellos que no creen que la inteligencia artificial se pueda conseguir basándose únicamente en la hipótesis de Newell y Simon.

Ved también

Podéis ver la hipótesis de Newell y Simon en el apartado 2 del módulo "Qué es la inteligencia artificial" de esta asignatura.

De todos modos, con independencia de la utilidad práctica real de CYC, el proyecto ha tenido que afrontar los problemas que aparecen en cualquier sistema basado en el conocimiento que utiliza bases de conocimiento de grandes dimensiones. En particular, se han tenido que encontrar algoritmos de inferencia eficientes, y encontrar maneras de realizar mantenimiento y actualización de la base de conocimientos.

3. Sistemas basados en reglas

Los sistemas expertos o sistemas basados en reglas son uno de los métodos más conocidos de sistemas basados en el conocimiento y constituyen la base de muchas de las aplicaciones existentes. Se aplican sobre todo a problemas de clasificación (identificación) y diagnosis.

En estos sistemas el conocimiento viene representado mediante hechos y reglas. Los hechos se corresponden a conocimiento sobre los objetos, y las reglas permiten establecer relaciones entre estos objetos. Las reglas tienen la forma:

```
si <premisa> entonces <conclusión>
```

donde tanto la premisa, como la conclusión son afirmaciones sobre hechos. A partir de los hechos y de las reglas, el sistema de inferencia concluirá sobre nuevos hechos (o corroborará la certeza de unos hechos). Las reglas a menudo incluyen elementos para representar la incertidumbre sobre las relaciones.

Como ejemplo de reglas, a continuación se ofrecen dos del sistema Spong-ia para clasificación de esponjas marinas.

El sistema Spong-ia

El sistema Spong-ia se describe en la obra siguiente:

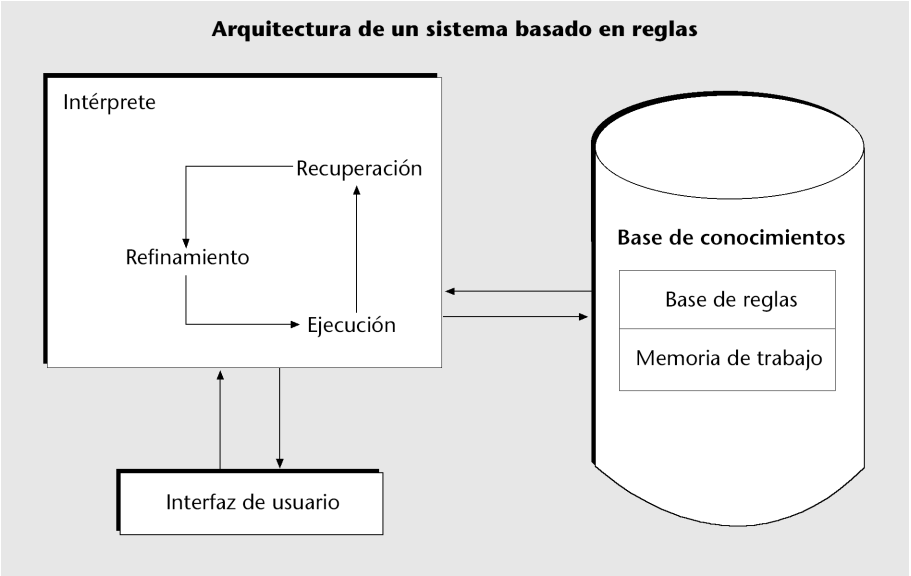
M. Domingo (1995). "An Expert System Architecture for Identification in Biology". Monografías del IIIA.

Es un sistema que sirve para clasificar esponjas marinas. Esto es, dado un ejemplar determinar a qué clase pertenece.

```
R002 If skeleton/pres and skeleton/chemical = (silica)
    then conclude Demospongiae is possible
R003 If skeleton/fibre
    then conclude Demospongiae is sure
```

En estas reglas, la conclusión utiliza los términos `possible` y `sure` para representar la incertidumbre existente entre los antecedentes y la conclusión:

Figura 3



Los sistemas basados en reglas presentan una arquitectura muy parecida a la proporcionada en la figura 1 como general para todos los sistemas basados en el conocimiento (comparad la arquitectura de la figura 3 con la de la figura 1). Disponen de una base de conocimiento formada por una base de reglas y una memoria de trabajo, además de un intérprete que corresponde al sistema de inferencia. A continuación se detallan un poco más estos elementos.

a) Base de reglas: la base de reglas corresponde a una parte de lo que hemos denominado el *aspecto formal del formalismo de representación del conocimiento*. Se guarda toda la información general relativa al dominio de aplicación. Por lo tanto, estarán todas las reglas necesarias para resolver los problemas que trata el sistema. En el caso de la clasificación de esponjas marinas, tendremos todas las reglas que relacionan las características de las esponjas, y las que nos permiten concluir su clasificación. La información de la base de reglas es independiente de la aplicación del sistema a casos concretos.

b) Memoria de trabajo: corresponde a la otra parte del aspecto formal de los sistemas basados en reglas. Se incluyen aquí todos aquellos hechos que pueden ser relevantes para el problema concreto que se quiere resolver. A diferencia del contenido de la base de reglas, aquí la información es temporal y corresponde al problema concreto que se quiere resolver. En el caso de la clasificación de esponjas, tendremos hechos (información) de la esponja que estamos clasificando en un momento dado (por ejemplo, datos sobre su esqueleto).

c) Intérprete: esto corresponde al aspecto inferencial del sistema. Permite deducir nuevos hechos a partir de la información existente en la memoria de trabajo y de las reglas de la base de reglas.

En esta descripción no hemos incluido ni el módulo de explicación, ni el de adquisición del conocimiento aunque a veces estos sistemas también los incluyen. En este caso, la función de estos módulos es la misma que la descrita en el apartado 1 de este módulo. A continuación describimos con un poco de detalle el procedimiento de inferencia en un sistema basado en reglas.

3.1. Aspecto inferencial en un sistema basado en reglas

Como se ha señalado, el intérprete corresponde al aspecto inferencial de un sistema basado en reglas. Muy a menudo, la inferencia se consigue mediante un esquema cíclico de recuperación –refinamiento– ejecución. Detallemos a continuación cada uno de estos pasos:

a) Recuperación: primero se seleccionan de la base de reglas las reglas que, sobre la base de la información de la que se dispone, se sabe que se pueden aplicar. A menudo en esta fase no se obtendrá una única regla, sino que se obtendrá un conjunto de éstas. Al conjunto de reglas seleccionadas se lo denomina *conjunto de conflicto*[6].

(6)En inglés, *conflict set*.

Els sistemes difusos

No sempre els sistemes basats en regles apliquen una sola regla cada vegada; els sistemes difusos acostumen a aplicar totes les regles a la vegada (els sistemes difusos s'estudien en el mòdul Incertesa i raonament aproximat).

b) Refinamiento: dado que los sistemas basados en reglas suelen aplicar sólo una regla cada vez, se deberá seleccionar una regla del conjunto.

c) Ejecución: una vez se ha seleccionado la regla, ésta se aplica.

Una vez se ha ejecutado una regla, se pasará otra vez a la recuperación. El ciclo acabará una vez se haya podido demostrar lo que se quería, o bien el conjunto de conflicto se encuentra vacío (por lo tanto no hay ninguna regla que se pueda aplicar). Observad que si se llega a un punto con un conjunto de conflicto vacío, como no habrá regla para aplicar, no habrá ningún cambio en la memoria de trabajo. Si no existe ningún cambio en la memoria de trabajo, el próximo conjunto de conflicto estará también vacío. Por lo tanto, de una situación con un conjunto de conflicto vacío no puede salir.

La descripción del funcionamiento del sistema que hemos presentado tiene el problema del control ya comentado anteriormente. En cada paso del proceso, debemos optar por una de entre las diferentes reglas que se pueden aplicar. Así, en general tenemos un objetivo (en el ejemplo que hemos considerado, determinar la clase a la que pertenece una esponja) y un problema de búsqueda ya que sólo un subconjunto de las reglas (y aplicadas en un cierto orden) nos lleva a la solución.

Para ayudar y dirigir la búsqueda, los sistemas consideran diferentes estrategias. Algunas de éstas son las siguientes:

a) Orden textual: el orden en el que aparecen las reglas en la base de reglas determina el orden de aplicación. En este caso, podemos controlar la estrategia de la búsqueda ordenando las reglas de manera adecuada. De hecho, el lenguaje PROLOG se puede ver como un sistema que sigue esta estrategia porque lo que hemos definido primero en el programa se aplica primero.

b) Obstinancia[7]: impide que una regla se aplique más de una vez. Cuando una regla ya ha sido aplicada, queda anulada. La idea que subyace detrás de la obstinancia es que cada vez que se aplica una regla a unos mismos hechos, se obtienen las mismas conclusiones. Se debe estar alerta con el uso de esta estrategia porque cuando una regla tiene variables, su aplicación puede dar diferentes resultados dado que podemos instanciarla con objetos diferentes. Esto resulta parecido a lo que ocurre con la lógica proposicional. Si tenemos $p(a)$, $p(b)$, $\forall x(p(\mathrm{x}) \rightarrow q(x)$ podemos obtener tanto $q(a)$ como $q(b)$. En este caso, para obtener $q(a)$ y $q(b)$ hemos de considerar $\forall x(p(x) \rightarrow q(x))$ dos veces, una con $p(a)$ y la otra con $p(b)$. Por otra parte, si el conocimiento no es monótono esta estrategia también puede provocar problemas.

(7)En inglés, *refractoriness*.

c) Recencia: se seleccionan las reglas que se refieren a los hechos más recientes de la memoria de trabajo. Para implementar esta estrategia es necesario que los predicados de la base de hechos incluyan una marca de tiempo[8] para poder resolver los conflictos. Cuando hay varias reglas que se pueden aplicar, se miran las marcas de tiempo de los predicados que aparecen en el antecedente. Se selecciona la regla donde los predicados tienen la última marca de tiempo.

(8)En inglés, *time tag*.

d) Especificidad: se seleccionan las reglas que corresponden a objetos más específicos. Aquí se considera un objeto A más específico que B cuando A es un tipo de los objetos B (por ejemplo, utilizaríamos antes una regla referida a *coche* , que una más general sobre *vehículos* cuando las dos son aplicables), o bien A es un elemento (un ejemplo particular) de B (por ejemplo, antes utilizaremos una regla referida al *coche del alcalde*, que una sobre *coches* si las dos son aplicables). También se pueden seleccionar aquellas reglas más específicas. Una de las maneras de evaluar la especificidad de las reglas es contando el número de conjuntandos (predicados en el antecedente que aparecen unidos con una conjunción) de la regla. Cuantos más conjuntandos, más específica será la regla. Así, una regla de la forma "si el suelo está mojado y no llueve entonces..." se aplicará antes que una de la forma "si el suelo está mojado entonces...".

Una alternativa diferente para controlar la búsqueda y conseguir que resulte más eficiente es utilizar conocimiento estratégico. Por ejemplo, el lenguaje Milord-II para desarrollo de sistemas expertos utiliza metarreglas (reglas de segundo nivel) para dirigir la búsqueda.

El lenguaje Milord-II

Milord-II es un lenguaje para la construcción de sistemas expertos desarrollado en el instituto de investigación en inteligencia artificial (http://www.iiia.csic.es/~puyol/Milord-II/).

Aparte del problema de seleccionar cuál es la regla que se debe aplicar en un momento dado, existen otras cuestiones relacionadas con la manera de realizar la inferencia. Una de éstas es que la inferencia se puede llevar a cabo di-

rigida por los datos, o bien dirigida por los objetivos. En el primer caso, las reglas se seleccionan y se aplican teniendo en cuenta lo que sabemos. En el segundo caso, el énfasis está en lo que queremos deducir. En el caso de sistemas basados en reglas, esto corresponde a lo que se denomina *razonamiento hacia adelante* y *razonamiento hacia atrás*. Esta nomenclatura se basa en la idea de que las reglas tienen un sentido: de los antecedentes a las conclusiones. A continuación detallamos estos dos tipos de razonamiento:

1) Razonamiento hacia adelante: partimos de la información disponible (el conocimiento que tenemos sobre el objeto que queremos clasificar o diagnosticar) y vamos aplicando reglas hasta que conseguimos deducir lo de que se quiere deducir (la clase o la diagnosis correspondiente). Se dice de *razonamiento hacia adelante* porque lo que miramos para saber si una regla se puede aplicar es si el antecedente se cumple. Si se cumple, añadiremos a la memoria de trabajo las conclusiones de la regla. Así, vamos de los antecedentes de las reglas a las conclusiones.

Así, por ejemplo, imaginamos un sistema en el que, según lo que hemos comentado anteriormente, podemos definir una base de reglas que esté formada por diez reglas:

R1. $A \wedge B \rightarrow E$
R2. $A \wedge C \rightarrow F$
R3. $\neg A \wedge B \rightarrow D$
R4. $\neg A \wedge \neg B \rightarrow E$
R5. $B \wedge C \rightarrow F$
R6. $\neg B \wedge \neg C \rightarrow \neg F$
R7. $D \wedge \neg F \rightarrow H$
R8. $D \wedge F \rightarrow G$
R9. $E \wedge F \rightarrow H$
R10. $E \wedge \neg F \rightarrow G$

De este sistema conocemos la memoria de trabajo inicial, formada por los hechos A, B y C, y se quiere averiguar si con esta informació se puede demostrar F. Con objeto de ayudar en el proceso de inferencia, se sabe que las reglas se ordenan según su numeración. Durante cada etapa realizada por el intérprete, las reglas que se han utilizado se eliminan de la base de reglas (lo que se conoce como obstinancia). Si hay más de una regla seleccionada, se ejecutará la de la numeración más baja.

Si para hacer la inferencia el sistema aplica un razonamiento hacia adelante, entonces nos encontraremos que partimos de una memoria de trabajo inicial: MT0 = {A, B, C}. Estos hechos, que en este momento son ciertos, convierten en aplicables tres reglas (R1, R2, R5) en la base de reglas. Esto genera el conjunto conflicto inicial CC0 = {R1, R2, R5}. Para resolver este conflicto en este ejemplo, se han establecido diferentes herramientas: las reglas se eliminan de la base de reglas una vez aplicadas, y se aplica inicialmente la de la numeración más

Nota

Podéis practicar con el ejercicio 1 de autoevaluación.

baja. Es decir, se selecciona R1 y se elimina de la base de reglas. En el ciclo de inferencia siguiente, la memoria de trabajo, una vez aplicada la regla 1, es MT1 = {A, B, C, E} y, por tanto, las reglas aplicables son R2 y R5, lo que genera un nuevo conjunto conflicto CC1 = {R2, R5}. Aplicando la misma estrategia de resolución de conflictos, se selecciona la regla R2 y se elimina de la base de reglas aplicables. En el tercer ciclo de inferencia, la memoria de trabajo consta de los siguientes hechos MT2 = {A, B, C, E, F} y, por tanto, el nuevo conjunto conflicto es CC2 = {R5, R9}. Se selecciona R5 y se elimina de la base de reglas. Dado que se quería demostrar F, en este momento, al mirar la memoria de trabajo, el sistema se pararía.

2) Razonamiento hacia atrás: en este caso se parte del objetivo (por ejemplo, de una hipótesis sobre la clase a la que pertenece un objeto o de su diagnosis) y seleccionamos las reglas que concluyen sobre este objetivo. La idea es que aquello que hay en la conclusión será cierto si se cumplen las condiciones que aparecen en el antecedente de una regla. Así, para poder demostrar la conclusión, debemos demostrar los antecedentes de las reglas. Entonces, se vuelve a repetir el mismo proceso planteando nuevos objetivos (los antecedentes de las reglas anteriores). De esta manera, consideramos, de hecho, las reglas de las conclusiones a los antecedentes.

Imaginemos que, a partir del sistema de reglas del ejemplo anterior, quisiéramos demostrar el hecho G aplicando el razonamiento hacia atrás en el proceso de inferencia si, en este caso, la base de hechos iniciales fuera {¬A, ¬B, ¬C}.

En este caso, para demostrar G, seleccionamos dos reglas que infieren este objetivo: R8 y R10. Si seguimos utilizando las mismas herramientas para ayudar en la búsqueda, es decir, que las reglas se aplican por la numeración y que, una vez aplicadas, se eliminan de la base de reglas, seleccionaríamos la regla R8 y, por tanto, como nuevos subobjetivos, tendríamos que poder incluir D y F en la base de hechos. Para incluir D, tendremos que aplicar R3, pero en nuestra base de hechos tenemos ¬B y esto haría que si ¬B es cierto, B no pueda serlo. De forma que el sistema repetiría el proceso pero ahora aplicando R10. Para inferir G, tendríamos que poder añadir E y ¬F a la base de hechos, que pasan a ser nuestros nuevos subobjetivos. Para demostrar E, podemos aplicar R1 y R4. R1 no se cumple, puesto que nuestra base de hechos iniciales contiene ¬A y ¬B, lo que invalida la aplicación de R1. En cambio, activa R4, que incorporaría {¬A, ¬B, ¬C, E} a la base de hechos. Finalmente, para demostrar el otro subobjetivo ¬F, habría que ver si R6 es aplicable, lo que se confirma porque ¬B y ¬C están en la base de hechos, y eso permite demostrar ¬F.

Nota

Podéis practicar con el ejercicio 2 de autoevaluación.

No todos los sistemas permiten los dos tipos de encadenamiento. En general, si podemos elegir, la elección se deberá hacer sobre la base de la información de la que disponemos y, como en cualquier problema de búsqueda, de cuál de las dos opciones presenta un factor de ramificación más pequeño. El control estratégico permite en algunos casos seleccionar entre las dos opciones.

3.2. Análisis de los sistemas basados en reglas

Las reglas constituyen una de las formas de representación del conocimiento más usadas para construir sistemas basados en el conocimiento. Algunas de las razones que se apuntan para este éxito es que son una manera sencilla de representar lo que el experto utiliza para resolver un problema. Por ejemplo, las relaciones entre síntomas y diagnosis, y, en el caso particular de la medicina, las relaciones entre síntomas y enfermedades.

Otra ventaja de los sistemas basados en reglas es su modularidad. El hecho de que la base de conocimientos sea independiente del proceso de inferencia permite modificar la base de datos sin haber de tocar nada relativo al aspecto inferencial. Además, el conocimiento temporal (la información que se encuentra almacenada en la memoria de trabajo) es independiente de la información permanente (la que se encuentra representada en las reglas).

También apunta en este sentido a la organización modular que algunos lenguajes de sistemas expertos[9] permiten para las reglas. Las reglas se pueden agrupar en módulos. Esto facilita la implementación de los sistemas y el posterior refinamiento o actualización. Esta modularización presenta una ventaja añadida que es que facilita los accesos a la información y a la inferencia. Al aplicar el proceso a un módulo, se aumenta la eficiencia de la recuperación y se disminuye el conjunto de conflicto.

(9)El lenguaje Milord-II mencionado antes permite esta organización.

Las reglas poseen una sintaxis restringida. Esto provoca que la capacidad expresiva resulte limitada y que, por lo tanto, existan cosas que no se puedan expresar. Sin embargo, esta restricción en la estructura del conocimiento permite la existencia de programas que operan con reglas. Por ejemplo, existen programas para tratar la validez de un conjunto de reglas (para ver si es consistente o si contiene ciclos) o para construir automáticamente reglas a partir de ejemplos.

Otra ventaja es la facilidad para construir una explicación del por qué las conclusiones se derivan de los hechos, dado que los sistemas expertos construyen un camino (la lista de reglas activadas) que liga lo que queremos demostrar con la información de la que disponemos. Una vez construido el camino, resulta posible reconstruir el razonamiento aplicado, y explicarlo.

De todos modos, no siempre es suficiente con la información que dan las reglas para construir la explicación. A veces, también resulta relevante el por qué se ha considerado una hipótesis antes que otra o por qué no se ha considerado una tercera. En estos casos, además de considerar el conocimiento de dominio, va bien el conocimiento estratégico para construir las explicaciones. Por lo tanto, convendrá que este tipo de conocimiento esté representado de manera explícita.

Para poder construir las explicaciones, además de los conocimientos de dominio y de control, necesitamos lo que se denomina *conocimiento de soporte*.

Este conocimiento explica el por qué las reglas son buenas. Además, permite razonar sobre por qué se toman unas determinadas decisiones. Esto facilita que en determinadas condiciones una regla se pueda saltar.

Ejemplo de explicación de una regla

Por ejemplo, podemos considerar la regla siguiente del sistema MYCIN:

```
si el paciente tiene menos de 8 años, entonces no recomendar tetracina
```

La explicación de esta regla está en el hecho de que aplicar la tetracina a niños a los que les están creciendo los huesos, les oscurece los dientes. Evidentemente, en casos de necesidad, esta regla se podría anular. Si el sistema conoce el por qué de la regla, la podrá anular.

4. Sistemas con representación estructurada

En los formalismos de representación considerados hasta ahora, el conocimiento se estructura en unidades independientes (predicados en la lógica de primer orden y reglas en los sistemas de reglas). No existen vínculos entre las diferentes unidades que definen la base de conocimiento.

Los sistemas de marcos[10], las redes semánticas[11] y los guiones[12] constituyen formalismos de representación alternativos que permiten relacionar las unidades a partir de las cuales se define el conocimiento. Estas relaciones corresponden a una modelización de las interconexiones entre los objetos del dominio.

[10] En inglés, *frame*.

[11] En inglés, *semantic networks*.

[12] En inglés, *scripts*.

Las diferencias entre los tres tipos de sistemas son pocas y la frontera entre éstos no es clara. Los tres poseen una estructura parecida: unos conceptos y relaciones entre sí. Mientras que en el caso de las redes semánticas se subrayan las relaciones, en los marcos se da más importancia a los conceptos. Por otra parte, los guiones corresponden a representaciones de secuencias de acontecimientos (para ser utilizados principalmente para la comprensión de historias).

Ejemplo de guión

En el trabajo inicial de Schank se utiliza como ejemplo de guión la ida a un restaurante. El guión es una descripción estereotipada (abstracta) del hecho de ir al restaurante y permite después entender un texto concreto describiendo la ida, así como también prever aquello que ocurrirá a continuación. Otro ejemplo del trabajo de Schank es el siguiente:

"II. We saw the Packers-Rams game yesterday. The Packers won on a dive play from the two with three seconds left. Afterwards they gave the game ball to the fullback.

(...) It is perhaps simplest to point out that the second paragraph is incomprehensible to someone who does not know football yet makes perfect sense to someone who does. In fact, football is never mentioned, yet questions like «What kind of ball was given to the fullback?» and «Why was it given?» are simple enough to answer, as long as the script is available."

R. Schank (1975). "The Structure of Episodes". A: D. G. Bobrow; A. Collins (ed.), *Memory, Representation and Understanding: Studies in Cognitive Science*. Nova York: Academic Press.

En este apartado, se describirán los sistemas de representación estructurados centrándonos en el caso de los marcos.

Los orígenes de los lenguajes de marcos se encuentran en los trabajos de M. Minsky sobre percepción en los sistemas de visión. Minsky, por su parte, estaba inspirado en algunas teorías de la Psicología como los esquemas de F.C. Bartlett: para analizar una situación nueva, un ser humano consulta las estruc-

turas elementales que integran las situaciones a las cuales ya se ha enfrentado antes. Esto es, se elige la estructura que le parece más próxima a la situación actual y, si es necesario, la modifica para adaptarla a la nueva situación.

Los marcos representan conceptos que definen estas situaciones o, incluso, representan las situaciones mismas. Permiten tener en cuenta los valores típicos (algunos lenguajes se basan en la idea del prototipo), las excepciones, la información incompleta o redundante. Además, con el fin de poder representar cómo cambia el entorno, permiten modificaciones, inclusiones o supresiones de sus propiedades. Estas modificaciones provocarán reajustes en la estructura de los marcos.

4.1. Aspecto formal

Desde el punto de vista del aspecto formal, un sistema de marcos es una red donde cada nodo representa un objeto (un concepto o un individuo) y los arcos corresponden a las relaciones que se pueden establecer entre los objetos. Los objetos se representan en marcos, que son un conjunto de campos[13] que lo definen.

(13)En inglés, *slots*.

Esta forma de representación estructura el conocimiento de manera parecida a como se estructura la información en un lenguaje orientado al objeto.

Los lenguajes de objetos

Visto de manera amplia, los lenguajes de objetos son todos los que llevan a una estructuración del universo de una aplicación en términos de objetos. Esta interpretación permite englobar los lenguajes de marcos en el paraguas de los orientados al objeto. En los lenguajes de objetos:

"Trois grandes families apparaisent naturellement, chacune privilégiant un point de vue sur la notion d'objet:

- Le point de vue *structurel*: l'objet est un type de donnés, qui définit un modèle pour la structure de ser représentants physiques et un ensemble d'opérations applicables à cette structure.
- Le point de vue *conceptuel*: l'object est une unité de connaissance, représentant le prototype d'un concept.
- Le point de vue *acteur*: l'objet est une entité autonome et active qui se repoduit par copie."

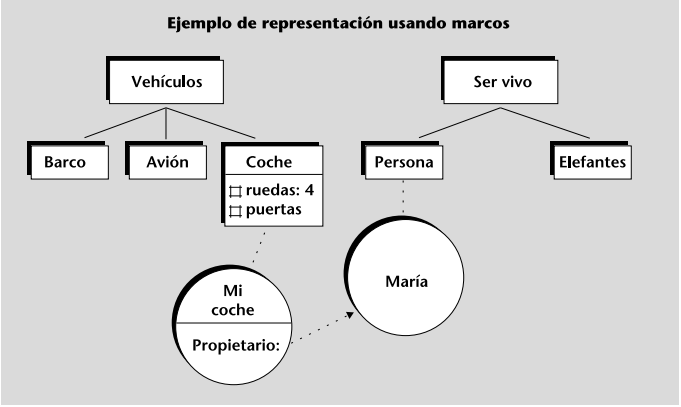
G. Masini; A. Napoli; D. Colnet; D. Léonard; K. Tombre (1989). *Les langages à objets*. París: InterEditions.

Los primeros son los lenguajes de clases, los segundos son los de marcos y los terceros, los de actores (o de agentes).

Unas de las relaciones que se establecen entre los objetos son las que también aparecen en estos lenguajes: relaciones de subclase, superclase y de instancia. La relación de subclase corresponde a cuando un concepto es un caso particular de otro. La relación de superclase constituye la relación inversa: cuando un concepto es más general que otro. La relación de instancia la encontramos cuando un marco corresponde a un individuo particular de una clase.

La figura 4 da un ejemplo de representación utilizando marcos. En este ejemplo, el concepto coche es una subclase del concepto vehículo, y vehículo es una superclase del concepto coche. Por otro lado, el marco correspondiente a mi coche representa una instancia del concepto coche (el mi es un caso particular de los coches). También la propietaria del coche, María, es una instancia de persona (un caso particular de persona).

Figura 4



A menudo, se encuentran campos de dos tipos: los de miembro[14] y los propios[15]. Los primeros son aquellos campos que se sitúan en cada instancia de un objeto (cada instancia –cada miembro de la clase– posee su propia copia y la modifica libremente), y los segundos son los que son propios del marco (y, por lo tanto, que comparten todas las instancias). Son como las variables de clase de un lenguaje orientado al objeto. Los campos pueden almacenar información de tipo muy diferente. Algunos de los elementos que típicamente se pueden incluir son éstos:

(14) En inglés, *member slot*

(15) En inglés, *own slot*.

- Valores de un dominio cualquiera.

- Referencia a otros marcos. En este caso, el valor indica un objeto del mismo sistema. Por ejemplo, en la figura 4 tenemos que el campo propietario del marco coche se refiere a una persona concreta: María. Este tipo de campo es el que nos permite ligar los conceptos entre sí para formar la estructura de los conceptos.

- Restricciones sobre valores. En lugar de representar el campo con un valor concreto, se dispone de una restricción sobre los valores posibles. Pueden ser restricciones sobre el tipo de valor (por ejemplo, el valor debe ser un entero), sobre qué tipo de instancia se puede ligar a un marco (por ejemplo, el propietario de un coche ha de ser una persona) o bien mediante conectivas lógicas (por ejemplo, el valor puede ser 30, 31 ó 32).

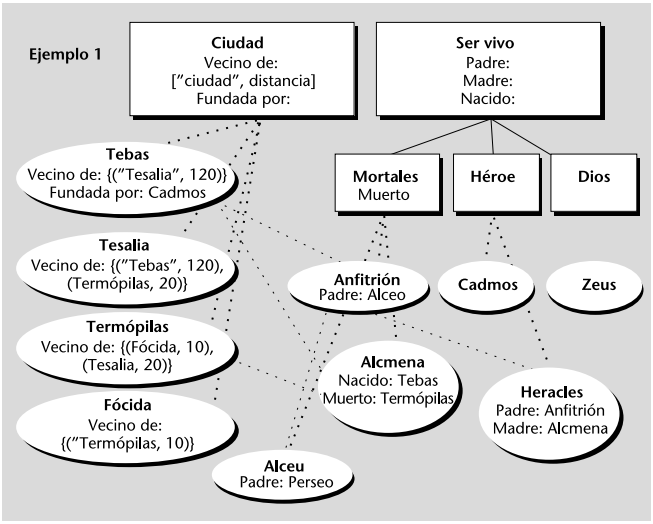
- Funciones. El campo tiene asociada una función a la que se acude cuando es necesario calcular el valor del campo. El hecho de ligar procedimientos a las estructuras constituye una característica importante de los sistemas

de marcos porque integran conocimiento procedimental que resulta muy difícil de representar de manera declarativa. Así, los marcos permiten incluir tanto conocimiento procedimental, como declarativo.

Ejemplo de marcos 1

Imaginemos que queremos modelizar los diferentes elementos de la mitología griega utilizando un sistema de marcos. Sabemos que en la antigua Grecia estaban las ciudades de Tebas, Tesalia, Termópilas y Fócida. De cada ciudad nos interesa conocer quién la fundó y sus comunicaciones. Para almacenar las comunicaciones, fijémonos que podemos definir un campo como una lista. Sabemos que Tebas fue fundada por Cadmos y que está a 120 kilómetros de Tesalia. De Tesalia sabemos que está a 20 kilómetros de Termópilas y a 120 kilómetros de Tebas. De Termópilas, que está a 10 kilómetros de Fócida y a 20 kilómetros de Tesalia, y de Fócida, que está a 10 kilómetros de Termópilas. La mitologia griega considera que los dioses, los héroes y los mortales son seres vivos, de los cuales nos interesa quiénes eran sus padres (padre y madre) y dónde nacieron. Algunos de los personajes de la mitologia griega son Zeus, un dios, y Cadmos, considerado un héroe, al igual que Heracles, o Hércules. De Heracles sabemos que su madre era Alcmena y su padre Anfitrión. Alcmena era mortal, nació en Tebas y murió en Termópilas. Anfitrión también era mortal y su padre era Alceo, también mortal, y cuyo padre fue Perseo. Así, pues, podríamos pensar en un sistema de marcos, con la representación gráfica siguiente:

Figura 5



Ejemplo de marcos 2

Imaginemos que queremos modelizar mediante marcos el famoso juego *Pokémon Go*. Dicho juego permite capturar, almacenar y entrenar para luchar diferentes *pokémons*. Éstos son almacenados por cada jugador particular en su *PokéDex* (almacén de todos los *pokémons*). De todos los diferentes tipos de *pokémons*, vamos a considerar tres: los *pokémons* de veneno (o *poisson*) (P), los *pokémons* de agua (o *water*) (W) y los *pokémons* de hierba (o *grass*) (G). Cualquier *pokémon* del juego se caracteriza por tener tres propiedades:

- El nivel de ataque: se supone que se refiere a la capacidad de destruir al contrario en una lucha y que puede ser mejorado entrenando al *pokémon*. Puede tomar los valores de bajo, moderado o alto (valor por defecto: bajo).

- La fuerza: medida con números enteros (unidad N).

- Si es un *pokémon* que proviene de otro (evolucionado) o es un tipo de *pokémon* en sí mismo (no evolucionado). Supondremos que puede tomar dos valores: No o Sí.

Suponemos que los *pokémon* de tipo veneno son siempre evolucionados y tienen una fuerza máxima de 10 N; los *pokémon* de agua y los de hierba son siempre no evolucionados. El nivel de ataque de un *pokémon* de agua es siempre alto; en cambio, los de hierba tienen un nivel de ataque medio por defecto. Finalmente, debido a restricciones en la fuerza, suponemos que este tipo de *pokémon* (hierba) tiene, por defecto, una fuerza de 3 N. Pero hay *pokémon* que combinan las capacidades de unos y otros: *pokémon* agua-veneno, no evolucionados, y *pokémon* hierba-veneno.

Una posible modelización de la información proporcionada mediante marcos podría estar formada por las clases y subclases siguientes:

- Pokémons: clase raíz de la jerarquía. Esta clase tiene los campos Evolucionado (No, Sí), Fuerza N (entero) y Nivel de ataque (bajo, moderado o medio, alto; valor por defecto: bajo). A partir de la clase se definirían tres subclases:
  - Pok_agua: subclase de pokémon. Tiene el valor propio No en Evolucionado y el valor propio Alto en Nivel de ataque. Por lo tanto, estos valores no se pueden cambiar en las subclases e instancias de esta clase.
  - Pok_veneno: subclase de *pokémon*. Tiene el valor, por defecto, Sí en Evolucionado, y la Fuerza está entre 0 y 10 N.
  - Pok_hierba: subclase de *pokémon*. Tiene el valor propio No en el campo Evolucionado, el valor por defecto Medio en Nivel de ataque y el valor por defecto de Fuerza es 3 N.

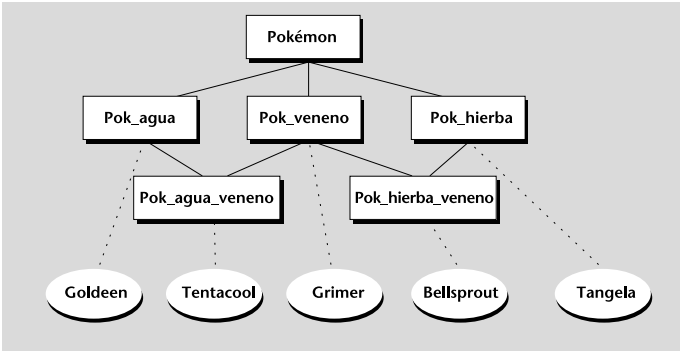
Finalmente, definiremos dos subclases que heredan de dos clases a la vez:

- Pok_agua_veneno: subclase de las clases Pok_agua y Pok_veneno. Tiene el valor por defecto No en el campo Evolucionado.
- Pok_hierba_veneno: subclase de las clases Pok_hierba y Pok_veneno.

Usando un sistema de marcos como éste podríamos representar la información que un jugador tiene en su *PokéDex* mediante instancias. Por ejemplo, un jugador tiene un Grimer, que es un *pokémon* de tipo veneno con 6 N de Fuerza; un Tentacool, que es un *pokémon* de tipo agua y veneno que tiene 2 N de Fuerza; un Goldeen, un *pokémon* de agua; un Bellsprout, un *pokémon* que tiene las características de los *pokémon* de hierba y también de veneno, y una Tangela, un *pokémon* de tipo hierba.

Así, pues, el sistema de marcos, en general, se podría representar de la manera siguiente:

Figura 6



Tendría las siguientes instancias:

- Goldeen: instancia de Pok_agua.
- Tentacool: instancia de Pok_agua_veneno. Fuerza: 2 N.
- Grimer: instancia de Pok_veneno. Fuerza: 6 N.
- Bellsprout: instancia de Pok_hierba_veneno.
- Tangela: instancia de Pok_hierba.

Todos los campos serían de tipo miembro, excepto aquellos en los que ya se ha especificado que son campos propios.

Los sistemas de marcos, además de los campos tienen los llamados procedimientos *demonios*[16]. Estos procedimientos se llaman como efecto secundario de alguna actuación relevante en la base de conocimientos. Los demonios se encuentran generalmente asociados a las descripciones de objetos y cuando el objeto cambia (o es creado o es eliminado), se activarán automáticamente. Por ejemplo, poder utilizar los demonios si al variar el valor de un campo se debe llevar a cabo una comprobación de su tipo, o se deben realizar pruebas para asegurar la consistencia de la nueva base de conocimientos. Éste sería el caso

(16)En inglés, *demons*.

de un sistema de ayuda a la conducción si quisiéramos controlar el nivel de gasolina del coche: tendríamos un procedimiento demonio asociado al nivel de gasolina del coche de manera que cuando la gasolina del coche estuviera por debajo de un umbral, se activara una cierta variable.

Por ejemplo, imaginad un sistema de marcos que modeltza los equipos y jugadores de una liga de fútbol. Se ha definido la clase:

Jugador_fútbol: superclase de hombre_adulto.

Campos:

- Categoría (Excelente, no Excelente), por defecto no Excelente.
  Demonio que modificará el valor.
- Equipo (todos los de la liga).
- Asistencias_partido: entero.
- Recuperaciones_partido: entero.
- Porcentaje_minutos: real.

Para calcular la categoría, se define un demonio que determina el valor cuando el número de recuperaciones de pelota más el de asistencias supera el valor de 10 y el *porcentaje de minutos de juego* por partido supera el 70% de la duración del partido, entonces el jugador se considera excelente.

```
{
    jugador_fútbol.categoría:: tipo
        (ARGUMENTOS: jugador)
        (CUERPO:
        Sí
            (jugador.Asistencias_partido + jugador.Recuperaciones_partido>10)
            ^(jugador.Porcentaje_minutos >70)
        ENTONCES
            (jugador.categoría=Excelente)
    )
}
```

4.2. Aspecto inferencial

La interrogación en un sistema de marcos suele realizarse para saber si un objeto satisface una determinada propiedad o para saber el valor que un determinado campo asocia a un objeto. En ambos casos, el problema se reduce a encontrar el campo que responde a la pregunta. Para encontrar este campo los sistemas de marcos se basan en la herencia, cuyo funcionamiento es igual que el que encontramos en los lenguajes orientados al objeto.

Así, el mecanismo general para encontrar el campo que puede responder a la interrogación consiste en mirar primero si el campo está definido en el mismo objeto y si no tiene ningún valor asociado. Cuando esto no ocurre, se consultarán otros marcos de acuerdo con las relaciones de herencia. La consulta se

efectúa repasando las relaciones establecidas entre objetos. Esto es, primero se mirará la superclase del objeto, después la superclase de la superclase, y así sucesivamente hasta que se encuentre el campo –la respuesta– que buscamos o ya no encontremos ningún otro objeto más general donde consultar.

Este mecanismo se complica cuando se permite, como es muy frecuente, la herencia múltiple.

Tenemos herencia múltiple cuando un objeto es instancia de más de una clase, o bien si es subclase de más de una clase. En estos casos, pueden presentarse conflictos para determinar la respuesta del sistema porque según el orden en el que se consideren las relaciones de superclase el sistema retornará una cosa u otra. Esto se debe al hecho de que podemos encontrar diferentes marcos con el campo por el que se interroga. Cuando los valores asociados a estos campos resultan diferentes, el resultado variará según el marco que se seleccione.

De hecho, el caso de tener herencia múltiple corresponde a un problema de búsqueda (un problema de control) porque para encontrar el resultado podemos considerar diferentes alternativas (diferentes relaciones clase-superclase), y no todas nos llevan a un resultado (y cuando llevan a uno no tiene por qué ser siempre el mismo).

Ved también

Podéis ver los aspectos de la búsqueda mencionados en el subapartado 2.4 de este módulo.

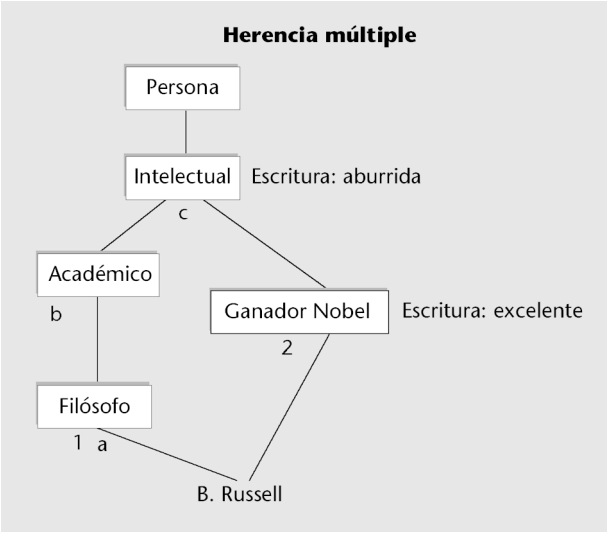
Desde un punto de vista abstracto, el problema de encontrar un determinado campo corresponde a realizar un recorrido por un grafo hasta que encontremos un nodo que satisface una determinada propiedad.

En este caso, tenemos que los nodos son objetos y los arcos son las relaciones con las que la herencia trabaja (es decir, relaciones de superclase y de instancia). Entonces, partiendo desde el nodo correspondiente al objeto por el cual se interroga debemos encontrar un nodo que contenga el campo. Cuando sólo hay herencia simple, el conjunto de marcos y relaciones corresponde a un árbol (o un bosque –un conjunto de árboles– si no hay una única raíz). Por lo tanto, hacia la raíz sólo existe un único camino. Así, al realizar la búsqueda, no se presentan nunca alternativas para considerar. En el caso de la herencia múltiple esto no es así. Existen nodos en los que hay más de un camino hacia la raíz.

Ved también

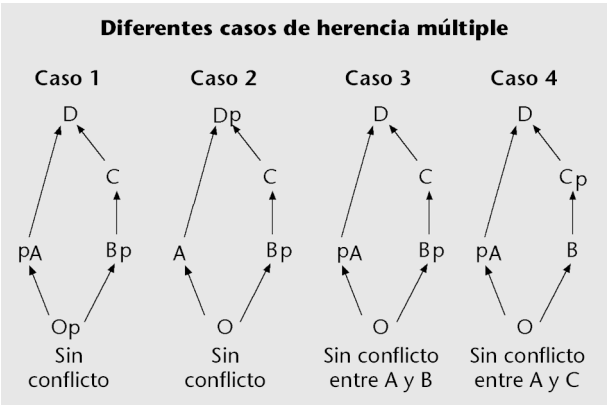
Podéis ver los algoritmos de búsqueda de los apartados 2 y 3 del módulo "Resolución de problemas y búsqueda" de esta asignatura.

Figura 7



Visto el problema como un recorrido en un grafo, podemos aplicar los algoritmos de búsqueda vistos en el módulo de búsqueda. Por ejemplo, podemos aplicar los algoritmos de anchura y profundidad. Sin embargo, se ha de tener en cuenta que métodos diferentes pueden dar soluciones diferentes. Por ejemplo, si consideramos el caso de la figura 5 y la pregunta "¿qué tipo de escritura tiene B. Russell?" obtendremos resultados diferentes cuando la búsqueda es en anchura, que cuando es en profundidad. En el primer caso, para encontrar el campo "tipo escritura" visitaremos los nodos 1 y 2 y obtendremos como resultado: excelente. En cambio, si la búsqueda es en profundidad obtendremos aburrida porque los nodos que visitaremos serán los marcados como *b* y *c*.

Figura 8



Casos con herencia múltiple

En la figura se presentan algunos casos de herencia múltiple que vale la pena considerar. Adaptado de H. Reighgelt (1991). *Knowledge Representation: An AI perspective.* Ablex Publishing Corporation.

De todas maneras, la manera de hacer el recorrido no es arbitraria y hay algunas condiciones que se han de tener en cuenta. En la figura 6 se presentan los casos que se deben considerar en la herencia múltiple.

En la figura 6, consideramos que estamos pidiendo si el objeto O satisface la propiedad p. Entonces habrá conflicto si la respuesta no se encuentra en el objeto O y la propiedad aparece en otros dos objetos O1 y O2 a los que se llega por dos caminos diferentes (y además no es posible llegar a un objeto desde el otro). Así, en el caso (1) no habrá conflicto porque la propiedad se encuentra

en el mismo objeto O, y en el caso (2) no lo habrá porque aunque se puede llegar al objeto D por un camino que no pasa por B, hay uno que sí que pasa. Tampoco habría conflicto en el caso (2) si la propiedad estuviera en C en lugar de D. En los casos (3) y (4) existe conflicto. En el (3) el conflicto es entre A y B, mientras que en el (4) lo hay entre A y C. El número de pasos que es necesario dar para encontrar un objeto no influye en el hecho de que haya conflicto.

De acuerdo con estos casos, los algoritmos deben tener en cuenta los cruces donde se encuentran varios caminos: antes de considerar lo que hay en el objeto asociado al cruce se deben haber tenido en cuenta los nodos que hay en los caminos que llevan desde el origen. En la figura 5, tenemos que el marco correspondiente a "intelectual" aparece en un cruce y, por esto, antes de decir que la escritura es aburrida, debemos haber visitado todas las subclases de "intelectual". En este caso, como el marco "ganador-de-nobel" también responde a la pregunta se deberá haber utilizado para responder a la pregunta sobre el tipo de escritura.

La ordenación topológica

La ordenación topológica es una ordenación de los vértices de un grafo dirigido acíclico, tal que si hay un arco de v_i a v_j, entonces v_j aparece después de v_i en la ordenación. Podéis encontrar más información sobre este concepto en la obra siguiente:

Weiss (1992). *Data Structures and Algorithm Analysis* (existe traducción en castellano). The Benjamin/Cummings Publishing Company, Inc.

Una manera de resolver este problema es mediante un algoritmo de ordenación topológica. Este algoritmo, dado un grafo no cíclico calcula una lista ordenada de nodos en la que se mantienen las relaciones que existen entre los nodos del grafo. Esta lista es la llamada *lista de precedencias*.

```
funcion ordenación topológica (graf) retorna lista de nodos es
    sean (a,b) los arcos del grafo si el nodo "a" precede al "b"
    lista de precedencia := lista vacía
    mientras queden arcos hacer
        Seleccionar un arco (a,b) tal que no exista ninguna pareja (*, a)
        ;; esto quiere decir que no hay ninguna dependencia sobre "a",
        ;; y si la había, las dependencias ya han sido visitadas
        lista de precedencia := lista de precedencia + "a"
        ;;añade el nodo "a" a la lista de precedencia
        eliminar todas las parejas de la forma (a, *)
        fmientras
    retorna la lista de precedencia
ffuncion
```

En este procedimiento, los pares (*a*, *b*) son los establecidos por el grafo. En nuestro caso, tendremos (*a*, *b*) si *b* constituye la superclase de *a* o si a es una instancia de *b*. Así, en el caso de la figura 7 tenemos que el conjunto de parejas es:

```
(B-Russell,), (Filósofo,), (B-Russell, Nobel),
(Académico, Intelectual), (Nobel, Intelectual),
(Intelectual, Persona).
```

El procedimiento construye la lista tomando cada vez un nodo que no tenga dependencias de ninguno otro nodo y después elimina todas las dependencias que dependen de dicho nodo. Esto es, debemos seleccionar un arco en el que el elemento de la izquierda no aparezca a la derecha en ningún par. En el ejemplo de la figura 5, empezamos seleccionando el arco (`B-Russell, Filósofo`) porque `B-Russell` no aparece nunca a la derecha de ningún par. A continuación, eliminamos todos los pares de la forma (`B-Russell, *`) –hay dos–, y la lista queda:

```
(Filósofo, Académico), (Académico, Intelectual), (Nobel, Intelectual),
    (Intelectual, Persona).
```

Se añade a la lista de precedencia `B-Russell`.

Pasamos a seleccionar otro arco. Ahora hay dos alternativas, porque podemos escoger tanto el arco (`Filósofo, Académico`) como el arco (`Nobel, Intelectual`). Si elegimos (`Filósofo, Académico`), suponemos que recorremos el árbol de izquierda a derecha, eliminamos todos los pares de la forma (`Filósofo, *`) y la lista de pares será:

```
(Académico, Intelectual), (Nobel, Intelectual), (Intelectual, Persona)
```

Y la lista de precedencia pasa a ser: `B-Russell, Filósofo`.

Pasamos a seleccionar otro arco, y de nuevo podríamos seleccionar (`Académico, Intelectual`) o (`Nobel, Intelectual`) pero estamos recorriendo el árbol de izquierda a derecha, por tanto, seleccionaremos (`Académico, Intelectual`). Y eliminamos los pares (`Académico, *`):

La lista de parejas será:

```
(Nobel, Intelectual), (Intelectual, Persona)
```

Y la lista de precedencias:

```
B-Russell, Filósofo, Académico
```

A continuación, sólo podemos escoger (`Nobel, Intelectual`) y la lista de precedencias nos queda:

```
B-Russell, Filósofo, Acadèmico, Nobel(Escritura_excelente)
```

Si recorremos el árbol de derecha a izquierda, después de añadir `B-Russell` a la lista de precedencias, escogemos el arco (`Nobel, Intelectual`) y entonces la lista de parejas nos quedará:

```
(Filósofo, Académico), (Académico, Intelectual), (Intelectual, Persona)
```

Y la lista de precedencia será:

```
B-Russell, Nobel
```

A continuación, seleccionaremos otro arco y sólo podemos seleccionar (`Filósofo, Académico`). Eliminamos (`Filósofo, *`) de la lista de parejas y añadimos `Filósofo` a la lista de precedencia.

Lista de parejas:

```
(Académico, Intelectual), (Intelectual, Persona)
```

Lista de precedencia:

```
B-Russell, Nobel, Filósofo
```

Seleccionamos la pareja (`Académico, Intelectual`), eliminamos (`Académico, *`) y añadimos `Académico` a la lista de precedencia.

Lista de parejas:

```
(Intelectual, Persona)
```

Lista de precedencia:

```
B-Russell, Nobel, Filósofo, Académico
```

Y, finalmente, visitamos el arco (`Intelectual, Persona`) y la lista de precedencia queda:

```
B-Russell, Nobel(Escritura_excelente), Filósofo, Académico,
    Intelectual(Escritura_aburrido)
```

Es importante subrayar que el algoritmo anterior no fija el resultado para cualquier grafo. Cuando en uno de los pasos de iteración hay más de un nodo que sólo aparece a la izquierda de una pareja, se tiene que seleccionar uno de los nodos. La manera de hacer la elección afectará al resultado. Este caso se da en nuestro ejemplo en el segundo paso, cuando se tiene que elegir entre los arcos (`Nobel, Intelectual`) y (`Filósofo, Académico`). Esta elección determina la ordenación relativa entre los nodos `Nobel` y `Filósofo`. Pero en ambas listas de precedencia la propiedad escritura excelente sale siempre antes. Por tanto, el conflicto es resoluble, puesto que ambos caminos conducen al mismo resultado.

Además de las operaciones que permiten evaluar las propiedades de un objeto, los sistemas de marcos disponen de algoritmos tanto para añadir nuevos conceptos, como para construir instancias de objetos ya existentes. La implementación de estas operaciones se puede realizar sobre la base de funciones asociadas a otros marcos de manera que el sistema sea completamente homogéneo. De esta manera, las operaciones estarán definidas en los marcos del mismo sistema. Esto resulta parecido a lo que ocurre en algunos lenguajes orientados al objeto (el caso paradigmático es Smalltalk) donde todo se puede definir de manera homogénea cuando todas las operaciones son definidas dentro de los mismos objetos del lenguaje.

Lectura complementaria

Una buena descripción de la estructura de clases enfatizando en la homogeneidad del sistema de objetos de Smalltalk se da en la obra siguiente:

T. Hopkins; B. Horan (1995). *Smalltalk: An introduction to application development using VisualWorks*. Prentice-Hall.

El ejemplo 2 de marcos que modelizaba el juego de *Pokémon Go* presentado anteriormente permitía conocer características de los *pokémons*. Pero, por ejemplo, para conocer el nivel de ataque del *pokémon* Bellsprout había que aplicar la ordenación topológica para resolver el conflicto que se produce entre el valor heredado de la clase *pokémon* (bajo) y el valor heredado de la subclase *Pok_hierba* (medio). Sin embargo, conviene señalar que este conflicto era resoluble mediante el algoritmo de ordenación topológica, puesto que aplicando el algoritmo, los pares (*a*, *b*) establecidos por el grafo son:

```
(Bellsprout, Pok_hierba_veneno), (Pok_hierba_veneno, Pok_veneno), (Pok_hierba_veneno,
```

```
    Pok_hierba), (Pok_veneno, Pokemon), (Pok_hierba, Pokemon)
```

De forma que si recorremos el grafo de izquierda a derecha, primero se selecciona el arco (`Bellsprout, Pok_hierba_veneno`) y se añade Bellsprout a la lista de precedencia. A continuación, la lista de pares queda:

```
(Pok_hierba_veneno, Pok_veneno), (Pok_hierba_veneno, Pok_hierba), (Pok_veneno, Pokemon),
            (Pok_hierba, Pokemon)
```

Después se selecciona entre la alternativa (`Pok_hierba_veneno, Pok_veneno`) o (`Pok_hierba_veneno, Pok_hierba`). Se elige (`Pok_hierba_veneno, Pok_veneno`) porque estamos recorriendo el árbol de izquierda a derecha, se añade `Pok_hierba_veneno` a la lista de precedencia y se eliminan las parejas (`Pok_hierba_veneno, *`). La lista de parejas resultante es:

```
(Pok_veneno, Pokemon), (Pok_hierba, Pokemon)
```

Finalmente, se selecciona un arco entre las alternativas `(Pok_veneno, Pokemon)` y `(Pok_hierba, Pokemon)`. Dado que lo estamos recorriendo de izquierda a derecha, se selecciona primero `(Pok_veneno, Pokemon)`. La lista de pares resultante es: `(Pok_hierba, Pokemon)` y se añade `Pok_veneno` a la lista de precedencias. Finalmente, se selecciona `(Pok_hierba, Pokemon)`. Por tanto, la lista de precedencia será:

```
Bellsprout, Pok_hierba_veneno, Pok_veneno, Pok_hierba
```

Cuando revisamos la lista de precedencia, encontramos la propiedad nivel de ataque definida en `Pok_hierba` con valor medio.

Si hubiéramos recorrido el grafo de derecha a izquierda, los pares establecidos por el grafo serían los mismos, pero después de haber añadido `Bellsprout` a la lista de precedencia, tendríamos la lista de pares siguiente:

```
(Pok_hierba_veneno, Pok_veneno), (Pok_hierba_veneno, Pok_hierba), (Pok_veneno, Pokemon),
    (Pok_hierba, Pokemon)
```

Seleccionaríamos `(Pok_hierba_veneno, Pok_hierba)` y añadiríamos `Pok_hierba_veneno` a la lista de precedencia eliminando las parejas `(Pok_hierba-veneno, *)`. A continuación, la lista de parejas resultante sería:

```
(Pok_veneno, Pokemon), (Pok_hierba, Pokemon)
```

En este caso, al recorrer el grafo de derecha a izquierda, escogeríamos primero `(Pok_hierba, Pokemon)`. La lista resultante sería `(Pok_veneno, Pokemon)` y añadiríamos `Pok_hierba` a la lista de precedencia, seleccionando finalmente `(Pok_veneno, Pokemon)`.

Por tanto, la lista de precedencia sería:

```
Bellsprout, Pok_hierba_veneno, Pok_hierba, Pok_veneno
```

Cuando revisamos la lista de precedencia, encontramos la propiedad nivel de ataque definido en `Pok_hierba` con valor medio, un nodo antes que en el caso anterior, pero con el mismo valor. Por tanto, se trata de un conflicto resoluble.

Nota

Revisad el problema 7 de autoevaluación para ver un ejemplo de conflicto no resoluble.

4.3. Análisis de los sistemas de marcos

Desde un punto de vista formal, los sistemas de marcos se pueden ver como una lógica de primer orden. De todos modos, hay dificultades en formalizar esta equivalencia cuando los campos de los marcos tienen asociados procedimientos o funciones. La equivalencia no impide que los marcos se utilicen a menudo por razones de eficiencia y claridad. Ello es así porque estos lenguajes permiten una representación gráfica del conocimiento y la visualización de la inferencia. Además, los lenguajes de consulta son simples, impidiendo que se realicen consultas complejas en la base de conocimientos (lo que facilita su tratamiento).

La restricción no aparece sólo a la hora de realizar las consultas, sino que también aparece en el conocimiento que se puede incluir en la base de conocimientos. Las restricciones en el lenguaje impiden expresar algunas de las proposiciones que podríamos escribir en lógica de primer orden. Por ejemplo, no se pueden traducir las sentencias “o bien el color del coche es azul, o bien el coche tiene más de cinco años” y “mi coche es azul o el tuyo es verde”. La primera no se puede representar porque relaciona dos campos de un mismo marco (el marco correspondiente al coche), y la segunda no se puede repre-

sentar porque relaciona dos marcos diferentes (tendremos un marco para cada uno de los coches). El lenguaje también presenta dificultades para representar algunos cuantificadores existenciales.

Las restricciones en el aspecto formal provocan que aquí no aparezca el problema de la semidecidibilidad porque o bien una propiedad está en la jerarquía, o bien no lo está. Esta limitación de la expresividad en favor de la eficiencia es un caso parecido al del lenguaje Prolog (que se utiliza más a menudo que los demostradores de teoremas de lógica de primer orden).

Por otra parte, la estructuración del conocimiento en jerarquías de objetos hace que sea muy fácil la representación de las excepciones y también del razonamiento por defecto. Cuando de un objeto no se conoce una propiedad, se tratará de encontrarla mediante la herencia. En el caso del ejemplo de la figura 5, tenemos que por defecto el estilo de escritura de cualquier Nobel resulta excelente. Si para una persona concreta descubrimos después el estilo, lo definiremos en el propio marco. Esto hará que a partir de aquel momento ya no se utilice el conocimiento por defecto, sino el propio y específico.

5. Sistema de razonamiento basado en casos

Una manera alternativa de resolución de problemas la constituye considerar el conjunto de experiencias pasadas y resolver las nuevas situaciones de manera parecida a como se resolvieron las anteriores. El razonamiento basado en casos[17] se fundamenta en esta idea.

[17]En inglés, *case-based reasoning*.

Para llevarla a cabo, el sistema almacena unas experiencias previas, y para cada una de éstas guarda la solución que se le ha aplicado. Entonces, cuando se le plantea una nueva situación, se buscará una situación parecida entre las que ya se han visto antes y se adaptará su solución a la nueva situación. En este tipo de sistemas, a cada experiencia o situación se la denomina caso, y al conjunto de casos, *base de casos*.

Estos sistemas se han aplicado a diferentes tipos de problemas. Algunas de las aplicaciones más características son el diseño, la diagnosis y la planificación. En el caso del diseño de un objeto, la experiencia previa sobre la construcción de otros objetos permite saber cómo tratar las restricciones (por ejemplo, que unos componentes no pueden ser adyacentes a otros, o que el peso total del artefacto debe estar por debajo de un determinado umbral). En diagnosis, los casos permiten evitar errores anteriores, reaprovechar decisiones ya tomadas o bien focalizar la búsqueda (en determinadas condiciones, sólo se ha de considerar un determinado conjunto de causas para unos síntomas). Existen aplicaciones en diagnosis de fallos de hardware y en diagnosis médica. Como se ha visto en el apartado 1.1.1 del módulo "Resolución de problemas y búsqueda", la planificación consiste en construir un plan a partir de operadores. Para realizarlo, se deben tener en cuenta los requisitos de cada operador. El razonamiento basado en casos permite usar planes ya construidos previamente o adaptarlos a la nueva situación.

La manera de definir el conocimiento de estos sistemas resulta muy diferente del conocimiento general que se representa mediante reglas o marcos. Aquí se dice que el conocimiento es operacional. En un caso, la información corresponde a cómo utilizar el conocimiento (cómo se aplica) en una situación concreta. La solución es para un caso concreto. Esta diferencia en el tipo de conocimiento hace que, para determinadas aplicaciones, los sistemas basados en casos resulten más adecuados que los sistemas que sólo utilizan conocimiento general.

Por ejemplo, en determinados entornos el conocimiento general puede ser demasiado abstracto y difícil de aplicar para situaciones concretas. Además, con este conocimiento operacional resulta más fácil tratar situaciones excepcionales (incluyendo el correspondiente conjunto de casos en la base de casos) o de dominios con conocimiento incompleto.

La representación del conocimiento partiendo de la base de casos no impide que el sistema incluya conocimiento general de dominio (por ejemplo, reglas) para resolver una determinada situación. Por ejemplo, se puede utilizar conocimiento general como dominio para adaptar la solución de un caso existente o bien para construir una solución cuando no se encuentra ningún caso similar.

Desde el punto de vista de la fase de adquisición del conocimiento, los sistemas de razonamiento basado en casos también presentan ventajas dado que se puede reducir todo el proceso de modelización. Ahora, al no necesitar conocimiento de dominio, es necesario que un experto lo defina con el subsiguiente gasto de recursos. En su lugar resulta suficiente con definir un conjunto de casos con sus soluciones: la descripción de un conjunto de experiencias. Suelen resultar más sencillas de obtener por parte de un experto estas experiencias, que el conocimiento general para resolver estas mismas experiencias.

Sistemas de razonamiento basado en casos

Existen sistemas de razonamiento basado en casos que sí que usan conocimiento para calcular soluciones de nuevos problemas o para determinar qué caso se parece al que vamos a resolver.

Por otra parte, estos sistemas pueden aprender de su propia experiencia al poder almacenar soluciones de nuevos casos. De esta manera, cuando un sistema vuelve a encontrar una determinada situación no resulta necesario recalcularla. El sistema puede reconocer esta situación y podrá aplicar la solución calculada anteriormente. También se puede almacenar información sobre los éxitos o los fracasos de soluciones dadas anteriormente. El uso de la información sobre las experiencias relevantes puede resultar de utilidad cuando el sistema se enfrente a nuevos problemas. La facilidad de integrar las fases de aprendizaje y razonamiento hace que a menudo se considere al razonamiento basado en casos como un subcampo del aprendizaje automático.

De todos modos, estos sistemas presentan también algunos inconvenientes en relación con los otros mecanismos de representación de conocimiento y de resolución de problemas. Se debe subrayar que un sistema de casos normalmente no explora todo el espacio de soluciones. Esto, por una parte, limita el campo de aplicación del sistema y, por otra, las soluciones que se obtienen no tienen por qué ser globalmente óptimas (pueden ser óptimos locales). Además, se ha de tener en cuenta que el sistema será sesgado según los casos que hay en la base de conocimientos.

5.1. Aspecto inferencial

La estructura de los sistemas de razonamiento basado en casos es variada aunque mantienen algunos rasgos comunes. En general, el sistema presenta una estructura cíclica en la que se consideran los pasos siguientes (podéis ver la figura 9):

1) Recuperación[18]: se recuperan los casos precedentes que son parecidos al caso que debemos resolver, y se selecciona lo mejor de éstos.

(18)En inglés, *retrieval*.

2) Reutilización[19]: se decide si la solución del caso seleccionado se puede aplicar directamente y si no es así, se la adapta a la nueva situación.

(19)En inglés, *reuse*.

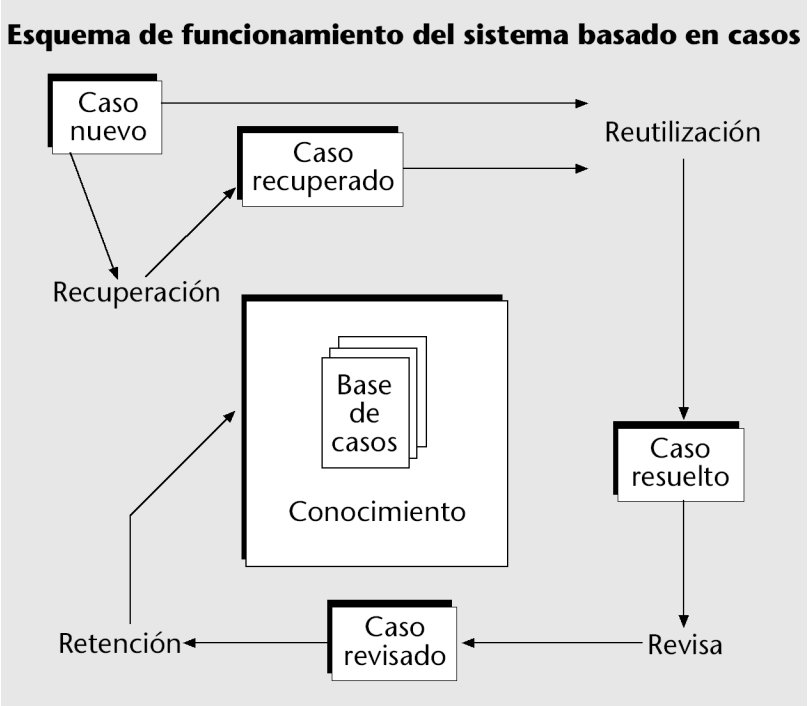
3) Revisión[20]: se aplica la solución al problema en curso y se evalúa. Puede ocurrir que la solución no se adapte completamente y se deba revisar.

(20)En inglés, *revision*.

4) Retención[21]: se almacena el caso y su solución para poder utilizarla más adelante.

(21)En inglés, *retain*.

Figura 9



Funcionamiento de un sistema basado en casos

Esta figura se ha adaptado de A. Aamodt; E. Plaza (1994). "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches". *AI Communications* (núm. 7, págs. 39-59).

Ahora comentaremos cada uno de estos pasos con un poco más de detalle:

1) Recuperación: en principio, la recuperación de un caso se ha de realizar sobre la base de la utilidad de su solución (para construir la nueva solución). De todos modos, como esto resulta difícil de saber, la recuperación se efectúa sobre la base de las características que describen un caso. Por ejemplo, en un sistema médico recuperaremos casos de pacientes con síntomas parecidos y un expediente médico similar. En general, calcularemos la similitud entre los

casos de la base de casos y el que queremos resolver y, de entre todos los considerados, devolveremos el que es más parecido. Para calcular esta semejanza se puede utilizar conocimiento de dominio (o simplemente comparación sintáctica entre los valores de diferentes campos de los casos).

Para que la recuperación se pueda realizar correcta y rápidamente, los casos deberán estar indexados. Lo más sencillo es considerar una organización de casos plana. Esto corresponde a comparar el caso actual con todos los casos de la base pero, evidentemente, cuando hay muchos casos esto provoca problemas por la cantidad de tiempo que necesitamos. Una alternativa consiste en utilizar índices jerárquicos con los subsiguientes problemas que supone la estructuración de los casos (en especial cuando se añaden nuevos) y encontrar el caso más adecuado para un problema dado (si la estructuración no es buena, encontraremos el caso más parecido). En este caso, los índices han de tener en cuenta cuáles son las características más relevantes en términos de encontrar la solución.

2) Reutilización: si la solución que tenemos ya se puede aplicar a la situación actual se aplica. Si esto no ocurre, habremos de adaptarla. Uno de los métodos para realizar la adaptación se basa en disponer de un conjunto de reglas definidas de manera que sus antecedentes sean las posibles diferencias entre un caso recuperado y un caso planteado, y los consecuentes sean las modificaciones que se deben efectuar en la solución del caso recuperado para adaptarla al caso planteado. Con esta información, cuando debamos adaptar una solución seguiremos los dos pasos siguientes: (1) determinar las diferencias entre el caso actual y el que hemos recuperado; (2) aplicar las reglas para decidir las transformaciones que se deben realizar en la solución del caso recuperado. Otra alternativa para la adaptación es que para cada caso tengamos indicaciones de cómo se ha obtenido la solución. Estas indicaciones se utilizan para encontrar la solución del nuevo caso.

3) Revisión: la aplicación de la solución al problema que se debe resolver realimenta el sistema ya que nos da una evaluación de la solución (podemos saber si es correcta, si no lo es y, en algunos casos, disponer de un grado de corrección). A veces, esta evaluación permite modificar la solución dada (adaptarla o encontrar una mejor que no presente problemas).

4) Retención: el caso se almacena con la solución calculada. Además, también se almacenarán los resultados de la revisión. Cuando los casos se encuentren organizados mediante un índice, el proceso de aprendizaje deberá adaptar estas estructuras para que el nuevo caso también pueda ser recuperado.

Lectura complementaria

Para más información sobre razonamiento basado en casos podéis consultar las obras siguientes:

A. Aamodt; E. Plaza (1994). "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches". *AI Communications* (núm. 7, págs. 39-59), en soporte digital en:

http://www.iiia.csic.es/es/staff/enric-plaza

Los pasos que se han descrito aquí aparecen en la mayoría de sistemas. No obstante, no todos los sistemas presentan siempre las mismas características. Por ejemplo, la manera de almacenar los casos tampoco es homogénea. Existen sistemas que guardan experiencias concretas, mientras que otros guardan formas generalizadas creadas a partir de un conjunto de casos similares.

En general, el comportamiento de un sistema de razonamiento basado en casos dependerá fuertemente de los casos que se incluyan en la base de casos, de la manera en la que se recuperan los casos para que se seleccionen los que son relevantes, y en los mecanismos de adaptación de las soluciones.

6. Sistemas de razonamiento basado en modelos

Una alternativa para tratar problemas de diagnosis consiste en relacionar las causas de los problemas con los síntomas observables sin entrar en detalle en el funcionamiento del sistema que diagnosticamos. Por ejemplo, en el caso de diagnosis médica podemos concluir sobre una enfermedad a partir de unos datos que nos suministra el paciente y de un conjunto de pruebas que se le practican. En el caso de diagnosticar el fallo de un aparato, se puede realizar el diagnóstico a partir del análisis de unos indicadores. En este caso, el conocimiento del sistema viene dado por un conjunto de relaciones causa-efecto. Por ejemplo, reglas que nos relacionan las causas con los efectos o redes bayesianas que nos relacionan las diferentes variables del problema. Otra alternativa consiste en considerar casos parecidos como ocurre en los sistemas de razonamiento basado en casos.

Lectura complementaria

J. de Kleer; B. C. Williams (1987). "Diagnosing Multiple Faults". *Artificial Intelligence* (núm. 32, págs. 97-130) y también:
http://www.ai-cbr.org/

Sin embargo, estas soluciones no entran en los detalles de por qué el fallo provoca aquellos síntomas. Este conocimiento superficial que sólo relaciona causas y efectos sin entrar en los detalles de la relación plantea dificultades en algunas aplicaciones. Por ejemplo, en un sistema experto se pueden aplicar reglas al caso que queremos resolver cuando las reglas no son las más adecuadas. En algunos casos, un análisis más detallado del problema podría seleccionar la solución correcta.

Una alternativa es tener más en cuenta los componentes de lo que queremos diagnosticar y sus relaciones. Los sistemas de razonamiento basados en modelos siguen esta línea. Consideran un modelo detallado de la estructura del aparato y una descripción de sus componentes.

Por ejemplo, en el caso de un sistema para la diagnosis de circuitos se toman descripciones de los componentes que los definen y de cómo se comportan. Además, necesitaremos saber cómo se ha construido el circuito a partir de los componentes. El modelo permite razonar sobre el circuito y, en particular, a partir de la comparación entre el comportamiento del modelo y el del sistema real se puede efectuar una diagnosis.

La diferencia entre los sistemas basados en reglas y los basados en modelos corresponde a la diferencia entre dos tipos diferentes de conocimiento: el conocimiento superficial[22] y el conocimiento profundo[23]. Una definición de este tipo de conocimiento es la que se ofrece a continuación:

[22] En inglés, *shallow knowledge.*

[23] En inglés, *deep knowledge.*

Shallow knowledge: This is the knowledge that a human expert might acquire by experience, without regard to the underlying reasons.

Deep knowledge: It is the fundamental building block of understanding. A number of deep rules might make up the causal links understanding a shallow rule.

A. A. Hopgood. (1993). *Knowledge-Based Systems for Engineers and Scientists*. CRC Press.

6.1. Aspecto formal

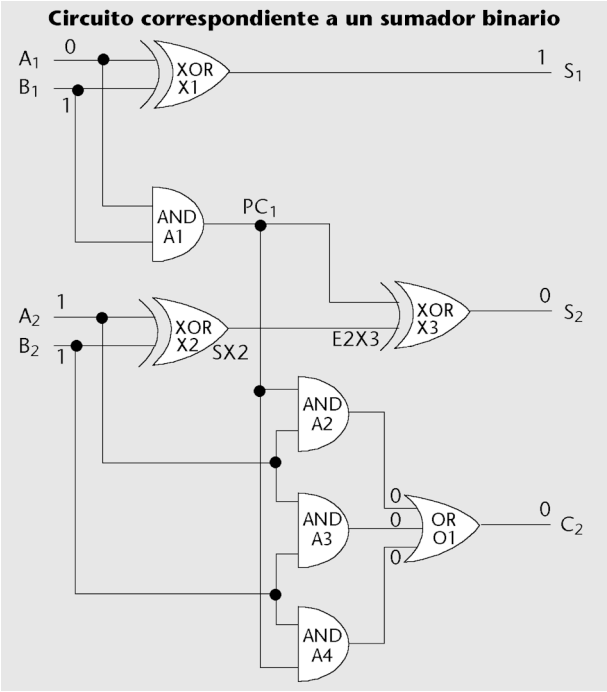
A continuación veremos con un poco más de detenimiento los elementos relacionados con los sistemas basados en modelos y cómo se utilizan para realizar diagnosis de un fallo. Empezaremos con el modelo del sistema, las posibles causas de fallo y las observaciones:

a) El modelo. En primer lugar es necesario construir el modelo del aparato. Con el fin de modelizar un aparato, se utiliza la descripción de la estructura física junto con los modelos de los constituyentes. Aquí se entiende por *constituyente* todo lo que define el sistema (componentes, procesos, etc.). Para representar el modelo, cualquier método puede ser adecuado (una representación en un lenguaje de programación o en un lenguaje de representación del conocimiento).

b) Las posibles causas. Para poder analizar el sistema es necesario disponer de información sobre qué es lo que puede fallar en los diferentes constituyentes y en la estructura que los relaciona. Por ello se considera para cada constituyente un conjunto de las posibles diferencias entre el modelo y el aparato. Esto establece la granularidad del error que detectaremos. La diagnosis sólo considerará las posibles causas que hayan sido declaradas.

c) Las observaciones. Necesitamos, además, un conjunto de medidas sobre el sistema. Estas medidas son las que nos permitirán determinar qué componentes no funcionan correctamente.

Figura 10



Para ilustrar este mecanismo de razonamiento consideramos un ejemplo sobre detección de fallos en circuitos. En particular, tomaremos el que se da en la figura 10 que corresponde a un sumador binario.

La modelización de este circuito debe definir qué puertas (AND, OR, XOR) se utilizan, cuál es su comportamiento (mediante tablas de verdad o expresiones calculables en un lenguaje de programación) y cómo están relacionadas las puertas (por ejemplo, que el resultado de la puerta A1 está conectado con el de la puerta A2). Además, la modelización puede incluir las conexiones. Por ejemplo, que hay conexiones entre la puerta A1 y el punto de contacto PC1, y entre la salida de la puerta X2 (denominada SX2) y la segunda entrada en la puerta de X3 (denotada E2X3).

También se deben definir las posibles causas de error. Por ejemplo, que fallan las puertas (o que también pueden fallar las conexiones).

Además, se han de especificar aquellos puntos en los que resulta posible consultar el valor que está transmitiendo el circuito. Por ejemplo, podemos observar los datos en cada una de las conexiones (o en puntos concretos de las conexiones si suponemos que éstas también pueden estar sujetas a fallos).

6.2. Aspecto inferencial

Cuando el sistema no funciona correctamente, habrá divergencia entre las observaciones y lo que el modelo predice (qué valor deberían tomar las observaciones). En este caso, el sistema planteará hipótesis sobre las causas de la

divergencia entre los valores. Como se dispone de los modelos, será posible evaluar las hipótesis (corroborarlas o rechazarlas) y, finalmente, seleccionar las que mejor se corresponden con las observaciones.

Por ejemplo, en la figura 8 se representa un caso de funcionamiento erróneo. Cuando las entradas al circuito tienen los valores A1 = 0, B1 = 1, A2 = 1 y B2 = 1 la salida es S1 = 1, S2 = 0 y C2 = 0. En este caso, el sistema detectaría que la salida C2 es incorrecta porque al aplicar su modelo se prevé un valor de 1. El sistema formulará la hipótesis de que el error se encuentra en la puerta O1, pero esta hipótesis es rechazada porque las entradas de esta puerta son las 0 y el modelo para una puerta OR dice que en este caso el resultado debe ser cero. Por lo tanto, O1 cumple las expectativas del modelo. Aplicando los modelos a las puertas A2, A3 y A4, el sistema detecta que el resultado de A3 debería ser 1. Este funcionamiento incorrecto explica el resultado incorrecto del sumador. Por lo tanto, formula la hipótesis de que falla la puerta lógica A3.

Un sistema de diagnosis basado en modelos ha de encontrar el conjunto de fallos más probable para explicar la conducta que se observa en el sistema. Para llevarlo a cabo plantea diferentes alternativas y rechaza las que no concuerdan con el modelo y mantiene las que son corroboradas por el modelo.

A causa de que el número de posibles errores es grande, el espacio de estados resulta considerable. Por ejemplo, en el caso de la figura 8 se tiene que si el error sólo se puede encontrar en alguna de las 8 puertas existentes, el número de estados posibles es 2^8. Esto es así porque cualquier subconjunto de las 8 puertas es una posible diagnosis.

Actividades

1. Suponiendo que los enunciados de la tabla que tenéis más abajo corresponden a reglas de un sistema basado en reglas, y suponiendo que la base de hechos iniciales es {E, F}, es decir, que E y F se cumplen, considerad la demostración de A.

a) Demostrad A cuando el intérprete del sistema basado en reglas funciona con razonamiento hacia adelante. Supongamos que la estrategia de selección de reglas se basa en la recencia y cuando dos reglas tienen la misma recencia utilizamos el orden en el que aparecen en la tabla. Además, utilizaremos la obstinancia con el fin de no repetir la aplicación de las reglas ya consideradas.

Tabla 2

Conjunto de sentencias en lógica de enunciados
$B \wedge C \rightarrow A$
$D \wedge C \rightarrow A$
$E \rightarrow C$
$E \rightarrow J$
$G \wedge H \rightarrow C$
$I \rightarrow H$
$I \rightarrow G$
$F \rightarrow D$
$J \wedge K \rightarrow B$
$L \rightarrow F$
$F \rightarrow K$

2. Suponiendo que los enunciados de la tabla que tenéis más abajo corresponden a reglas de un sistema basado en reglas, y suponiendo que la base de hechos iniciales es {A, D}, es decir, que sabemos que A y D se cumplen, considerad la demostración de Q aplicando encadenamiento hacia atrás.

Tabla 3

Conjunto de sentencias en lógica de enunciados
R1. $P \rightarrow Q$
R2. $E \rightarrow B$
R3. $R \rightarrow Q$
R4. $M \wedge N \rightarrow Q$
R5. $A \wedge B \rightarrow P$
R6. $A \rightarrow M$
R7. $C \rightarrow M$
R8. $D \rightarrow N$

3. Tenemos un lenguaje de representación del conocimiento basado en marcos donde hay herencia múltiple. Suponed que queremos saber el valor de un campo de un marco. ¿Qué relación tiene este problema con la búsqueda en un espacio de estados (el problema del

control)? ¿Qué sería aquí un estado, los operadores, la función objetivo (hay un único estado final en este problema)?

4. Considerad el sistema de marcos del ejemplo del subapartado 4.1 de este módulo. Sobre este sistema responded a las cuestiones siguientes:

a) ¿Cómo encontramos al abuelo de Heracles?

b) Dad el grafo con las relaciones de herencia que el sistema de marcos utilizará cuando lo interroguemos.

5. Según el sistema de marcos del juego Pokémon Go, representado en el ejemplo 2, ¿qué Nivel de ataque y cuánta Fuerza tiene el Grimer del jugador?

6. Según el sistema de marcos del juego *Pokémon Go*, representado en el ejemplo 2, ¿qué *Nivel de ataque* y cuánta *Fuerza* tiene el Grimer del jugador?

7. Explica por qué se genera un conflicto irresoluble cuando se aplica el algoritmo de ordenación topológica para determinar si la instancia `Bellsprout es Evolucionado`.

Solucionario

1. Para construir la solución hemos de proceder de acuerdo con lo que haría el intérprete. Debemos tener en cuenta que el conjunto de reglas está definido con las de la tabla del enunciado y que la memoria de trabajo sólo tiene al inicio de la ejecución a E y a F.

Para proceder como el intérprete hemos de empezar viendo qué reglas se pueden aplicar de acuerdo con lo que hay a la memoria de trabajo. Encontramos que las reglas seleccionables son $E \rightarrow C$, $E \rightarrow J$, $F \rightarrow D$ y $F \rightarrow K$. Como todos los hechos son del mismo instante (les asociamos una marca de tiempo igual a cero) seleccionamos la regla de acuerdo con el orden. Por lo tanto, elegimos $E \rightarrow C$. Al aplicarlo obtenemos C y lo añadimos a la memoria de trabajo con una marca de tiempo igual a 1.

A continuación volvemos a mirar las reglas que se pueden aplicar. Son las mismas. Ahora, sin embargo, como la regla $E \rightarrow C$ ya no se puede aplicar (porque aplicamos la estrategia de obstinancia) seleccionaremos y aplicaremos $E \rightarrow J$. Esto nos obliga a añadir J a la memoria de trabajo y le asignaremos una marca de tiempo igual a 2.

En el tercer paso las reglas seleccionadas son las mismas pero ahora sólo podemos aplicar $F \rightarrow D$. Así, añadimos D a la memoria de trabajo con la marca 3.

En el cuarto paso de iteración las reglas que podemos aplicar son $D \wedge C \rightarrow A$, $E \rightarrow C$, $E \rightarrow J$, $F \rightarrow D$ y $F \rightarrow K$. De acuerdo con la estrategia de recencia, seleccionamos $D \wedge C \rightarrow A$ ya que las marcas de tiempo de D y C nos dicen que son más recientes que las de E y F. Al aplicar la regla seleccionada obtenemos A y por lo tanto acaba la ejecución.

En la tabla siguiente se ofrece un resumen de la ejecución. Para cada objeto de la memoria de trabajo se da la marca de tiempo.

Tabla 4

Resumen de la ejecución		
Reglas	**Regla seleccionada**	**Memoria de trabajo**
		(E, 0) (F, 0)
$E \rightarrow C$, $E \rightarrow J$, $F \rightarrow D$, $F \rightarrow K$	$E \rightarrow C$	(E, 0) (F, 0) (C, 1)
$E \rightarrow C$, $E \rightarrow J$, $F \rightarrow D$, $F \rightarrow K$	$E \rightarrow J$	(E, 0) (F, 0) (C, 1) (J, 2)
$E \rightarrow C$, $E \rightarrow J$, $F \rightarrow D$, $F \rightarrow K$	$F \rightarrow D$	(E, 0) (F, 0) (C, 1) (J, 2) (D, 3)
$E \rightarrow C$, $E \rightarrow J$, $F \rightarrow D$, $F \rightarrow K$ $D \wedge C \rightarrow A$	$D \wedge C \rightarrow A$	(E, 0) (F, 0) (C, 1) (J, 2) (D, 3) (A, 4)

2. Nuestro objetivo es Q. Aplicando encadenamiento hacia atrás tenemos:

- Ciclo 1: Conjunto conflicto {R1, R3, R4}, seleccionamos R1 y cambiamos el subobjetivo a P.
- Ciclo 1.1: Conjunto conflicto para P: {R5}. Para demostrar P tenemos que demostrar A y B. A es cierto, por tanto, tenemos que demostrar B.
- Ciclo 1.1.1: Seleccionamos R2. Para demostrar B, entonces E tendría que ser cierto, y no lo podemos demostrar.
- Ciclo 2. Volvemos al objetivo Q. Conjunto conflicto {R3 y R4}, seleccionamos R3 y cambiamos el subobjetivo a R. No podemos demostrar R.
- Ciclo 3. Volvemos al objetivo Q. Seleccionamos R4 y cambiamos a los subobjetivos M y N. Para demostrar M, tenemos que demostrar A. A es cierto, y por tanto M es cierto (aplicando la regla 6). Por otro lado, para demostrar N tenemos que demostrar D. D es cierto puesto que está en la base de hechos iniciales. Finalmente, sabiendo que M y N son ciertos, demostramos que Q es cierto.

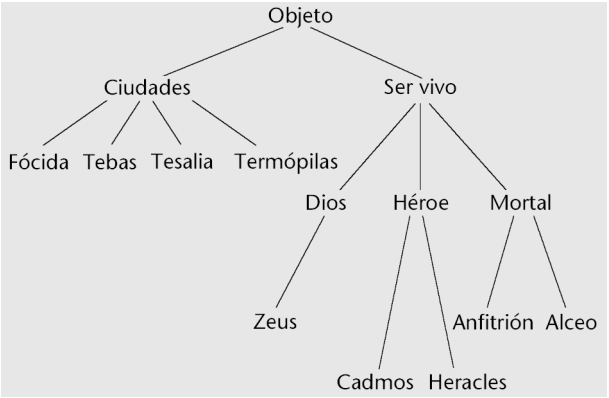
3. El problema que tenemos que resolver es el de buscar un marco que contiene un determinado campo. Como lo que haremos es consultar nodos hasta que encontremos el que nos interesa, podemos definir el estado como la visita a un nodo. Con esta interpretación un operador corresponderá a pasar de un nodo a otro. Consideraremos que se puede pasar de un

nodo A a otro nodo B si B es una superclase de A o si A es una instancia de B. La función objetivo será mirar si el marco en el cual nos encontramos contiene ahora el campo que se pide.

4. Las respuestas a las cuestiones del ejercicio son éstas:

a) Primero iremos al marco correspondiente a Heracles y miraremos si está el campo abuelo. Como no está, iremos después a buscarlo a la clase de la cual es instancia (la clase héroe) pero tampoco está. Seguimos mirando la superclase de héroe que es ser-vivo. Allí encontramos una función y la aplicamos. Como la función nos dice que el abuelo es el padre del padre del marco, lo hemos de aplicar al marco que consultábamos: Heracles. Al realizarlo, la función nos indica que debemos encontrar al padre de Heracles. Esto se encuentra en el mismo marco: es Anfitrión. Pasamos, pues, a buscar al padre de este último personaje. Lo encontramos en su marco. Es Alceo.

b) El grafo de herencias es el de la figura siguiente:



5. La herencia permite tratar fácilmente el razonamiento por defecto. Cuando tenemos un marco, hereda de la superclase (o de la clase de la cual es instancia) todos los valores por defecto. Cuando se descubre que un determinado valor por defecto no es bueno, definiremos un campo con el mismo nombre en el marco. Esto provoca razonamiento no monótono. Lo que antes se deducía, ahora no.

6. El *Grimer* tendría un *Nivel de ataque* bajo, valor que heredaría de la clase *Pokemon* y tendría 6 N de *Fuerza*, valor que estaría definido en la propia instancia.

7. Las listas de procedencia obtenidas cuando recorrimos el graf por la instancia *Bellsprout* eran, cuando el graf se recorría de izquierda a derecha: *Bellsprout, Pok_hierba_veneno, Pok_veneno, Pok_hierba*, y cuando el graf se recorría de derecha a izquierda: *Bellsprout, Pok_hierba_veneno, Pok_hierba, Pok_veneno*. Entonces si miramos qué clases tienen definido el campo *Evolucionado*, vemos que en el primer caso *Pok_veneno* tiene definido el campo *Evolucionado* y el valor es *Sí*, mientras que, en el segundo caso, *Pok_hierba* tiene definido el campo *Evolucionado* y el valor es *No* i ambos están en el mismo nivel. Por tanto, el conflicto es irresoluble.

Glosario

aspecto formal *m* Aspecto que corresponde a cómo se almacena la información cuando la tenemos representada de manera explícita en un formalismo de representación del conocimiento.

aspecto inferencial *m* Aspecto que corresponde a cómo se obtiene la información que se encuentra en el sistema pero sólo de manera implícita en un formalismo de representación del conocimiento.

base de conocimientos *f* Almacén de conocimiento de los sistemas basados en el conocimiento.

compartir el conocimiento Englobar todos los aspectos relativos al hecho de que diferentes sistemas puedan compartir el conocimiento y no sea necesario empezar un modelo de nuevo cuando se construye un nuevo sistema. También es necesario compartir el conocimiento cuando se tienen diferentes agentes autónomos y se quieren comunicar.

completitud *f* Método que es completo cuando se pueden deducir todas las consecuencias de un sistema formal.

conjunto de conflicto *m* Conjunto de reglas seleccionadas por un sistema basado en reglas en la fase de recuperación.

ingeniería del conocimiento *f* Área de la inteligencia artificial que estudia el proceso de construcción de sistemas basados en el conocimiento.

inconsistencia *f* Una base de conocimientos es inconsistente si podemos deducir una propiedad a así como también su negación $\neg a$.

obstinancia *f* Estrategia de selección de reglas en sistemas basados reglas (podéis ver II.3.1).

procedimiento demonio *m* Procedimiento que se llama como efecto secundario de alguna actuación relevante en la base de conocimientos.

recencia *f* Estrategia de selección de reglas en sistemas basados reglas.

solidez *f* Método que es sólido cuando sólo se generan oraciones implicadas.

Bibliografía

Bibliografía básica

Stefik, M. (1995). *Introduction to Knowledge Systems*. Morgan Kauffman.

Bibliografía complementaria

Reighgelt, H. (1991). *Knowledge Representation: An AI perspective*. Ablex Publishing Corporation.

Luger, G.F. (11998). *Artificial Intelligence*. Addison-Wesley.