Capítulo 4

Redes semánticas y marcos

María del Carmen Suárez de Figueroa Baonza y **Asunción Gómez Pérez**
Universidad Politécnica de Madrid

4.1 Introducción

En representación de conocimientos se puede hablar de tres modelos distintos:

- El *modelo conceptual* que es una representación de los conocimientos de un dominio, independientemente de cómo se implementen, utilizando estructuras no computables que modelizan el problema y la solución en un dominio concreto.
- El *modelo formal* que es una representación "semiinterna" o "semicomputable" de los conocimientos de un dominio. Este modelo formal se ha de obtener a partir del modelo conceptual.
- El *modelo computable* que hace que el modelo formal sea totalmente operativo, y que está formado por una base de conocimientos, un motor de inferencias y una serie de estrategias de control.

El presente capítulo está relacionado con la formalización de conocimientos. A grandes rasgos, se puede decir que **formalizar** consiste en *representar* simbólicamente los conocimientos de un dominio utilizando alguno de los formalismos de representación de conocimientos existentes, *organizarlos* de acuerdo a algún modelo de diseño y *determinar los métodos de inferencia adecuados* para manejar eficiente y efectivamente los conocimientos representados.

Para representar los conocimientos de un determinado dominio, el ingeniero del conocimiento (IC) debe utilizar uno o varios formalismos o técnicas de representación de conocimiento. Para razonar con los conocimientos representados, el IC debe analizar las técnicas de inferencia disponibles para cada formalismo utilizado y seleccionar las más adecuadas para resolver el problema que se plantea.

Los investigadores en IA han descrito varios formalismos útiles para la representación de conocimientos, cada uno de ellos más adaptado a un tipo de conocimientos. Los formalismos más populares incluyen colecciones de reglas condicionales, del tipo

"Si ..., Entonces ..." (véase el capítulo 3). Otros sistemas de representación recurren a *marcos conceptuales*, estructuras parecidas a formularios que es preciso rellenar. Existen otros métodos de representación que se valen de redes "en telaraña" o de listas, como las *redes semánticas* y los *guiones*, respectivamente.

Cuando un IC aborda un nuevo problema, una de las decisiones que tiene que tomar es determinar el formalismo de representación más adecuado para la tarea que le corresponde realizar al sistema. En sistemas pequeños y sencillos, lo normal es que solamente se utilice un formalismo para representar el conocimiento; pero a menudo se combinan dos o tres formalismos de modo que se complementen unos a otros para lograr una representación que se ajuste de la forma más natural posible a los conocimientos del dominio.

Cada uno de los formalismos existentes tiene asociadas unas técnicas básicas de inferencia. Estas técnicas son independientes del dominio representado por el formalismo, es decir, serán capaces de razonar con cualquier conjunto de conocimientos representados mediante su formalismo propietario. Básicamente, los distintos formalismos existentes permiten representar cualquier conocimiento; sin embargo, unos formalismos son más adecuados que otros para representar unos tipos de conocimientos. En función de si el formalismo es más adecuado para representar conceptos, relaciones o acciones, se puede realizar la siguiente clasificación:

- **Formalismos basados en conceptos**, los cuales representan las principales clases o entidades del dominio, sus propiedades, y los posibles valores que puede tomar cada propiedad. En este caso, se pueden distinguir entre las ternas *Objeto-Atributo-Valor* y los *Marcos*

- **Formalismos basados en relaciones**, que centran su atención en las relaciones que aparecen entre los conceptos o entidades del dominio. Los más importantes son: la *Lógica* (véase el capítulo 2), las *Redes Semánticas* y la *Teoría de la Dependencia Conceptual.*

- **Formalismos basados en acciones**, que describen los conocimientos del dominio como un conjunto de acciones básicas. Los principales formalismos de este tipo son: los *Sistemas de Producción* (véase el capítulo 3) y los *Guiones.*

Este capítulo se centra en un formalismo basado en relaciones, las *redes semánticas*, y en un formalismo basado en conceptos, los *marcos.*

El formalismo de las **Redes Semánticas** fue definido por Quillian [Quillian, 1968] como un grafo orientado formado por nodos y arcos unidireccionales, ambos etiquetados. Los nodos representan conceptos y los arcos relaciones entre conceptos. Las redes semánticas se están utilizando en IA como una técnica de representación de conocimientos para expresar relaciones entre los conceptos de un dominio. El término *red semántica* se ha utilizado para describir una gran variedad de representaciones y técnicas que tienen en común el uso de grafos orientados como representación gráfica. Dado que no existe un formalismo universalmente aceptado que permita explicar con claridad qué es una red semántica, en este capítulo se van a mostrar los trabajos más representativos realizados en este área tanto en el plano de representación como en el plano del razonamiento.

El formalismo de **Marcos** (en inglés *Frames*), fue definido por Minsky 1985] como una estructura de datos para representar estereotipos. Cada m formado por un nombre y un conjunto de propiedades. Los valores de cad dad se describen en el marco utilizando un conjunto de facetas. Los marco unos con otros mediante relaciones. La relación *subclase-de* permite crear j de conceptos por especialización y proporciona un camino para la herenci piedades. Las principales características que han hecho de los marcos el fo de representación más utilizado cuando los conocimientos del domino está en conceptos son: la organización de los conceptos que intervienen en el pro jerarquías o taxonomías de conceptos, y la posibilidad de representar cono procedimentales y declarativos en los marcos de una forma integrada.

4.2 Redes semánticas

Las redes semánticas (o redes conceptuales) son un formalismo o paradig presentación de conocimiento basado en relaciones; es decir, este formalismo atención en las relaciones que aparecen entre los conceptos o entidades de un Básicamente, una red semántica [Quillian, 1968] es un grafo orientado forn nodos etiquetados, que representan conceptos e instancias, y arcos unidire etiquetados, que representan relaciones entre conceptos o instancias. La Fi muestra los conceptos básicos de representación en redes semánticas.

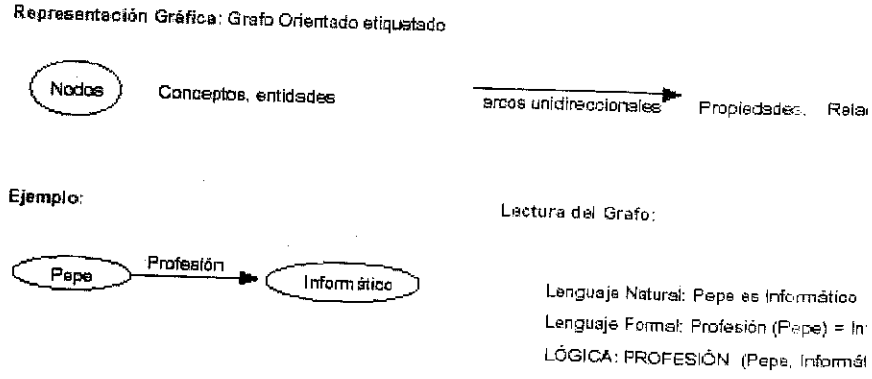


Figura 4.1: Conceptos básicos en redes semánticas.

4.2.1 Representación de conocimiento

En una red semántica, la información se representa en un grafo orient está formado por un conjunto de **nodos** y **arcos** unidireccionales, ambos etiq Los nodos representan conceptos e instancias de dichos conceptos, y los ar conectan nodos, representan relaciones binarias entre ellos. Por tanto, el signifi un concepto de la red dependerá de la forma en la que dicho concepto se relaci

otros conceptos. Se puede utilizar cualquier etiqueta para dar nombre a cualquier nodo o arco. Los principales problemas de este formalismo de representación son la falta de una terminología adecuada y universalmente aceptada, y de una semántica uniforme y precisa. Sin embargo, su gran atractivo gráfico y su intuitiva interpretación las hace rápidamente comprensibles y utilizadas en la formalización de bases de conocimiento.

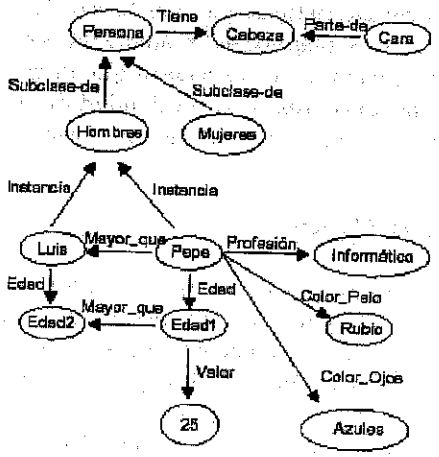


Figura 4.2: Ejemplo de red semántica.

La Figura 4.2 es un ejemplo sencillo de una red conceptual que representa a *Pepe*, un profesional de la informática que tiene los ojos azules, el pelo rubio, 25 años y que es mayor que Luis. La base de la representación de conocimientos en las redes semánticas consiste en modelar los conocimientos relativos a un objeto o concepto mediante pares atributo-valor. Los pares se representan en el grafo orientado de la siguiente manera: el nodo origen (*Pepe*) es el objeto o concepto para el cual se definen los pares atributo-valor, los arcos que parten de dicho nodo (*profesión*, *color-pelo*, etc.) son los atributos del par, y los nodos destinos (*informático*, *rubio*, etc.) representan los valores de los atributos. Por tanto, el significado de un nodo en la red de conceptos dependerá no sólo de cómo el nodo se relaciona con otros nodos, sino de las etiquetas que dan nombre a los arcos y nodos que representan los elementos del dominio. El ingeniero del conocimiento podrá utilizar tantos arcos como propiedades estime oportuno representar y, a cada uno de ellos, le asignará una etiqueta que expresará de manera representativa su semántica.

Básicamente, los arcos en las redes conceptuales se agrupan en dos categorías: **arcos descriptivos** y **arcos estructurales**.

- Los **arcos descriptivos** describen entidades y conceptos. Ejemplos de arcos descriptivos en la red de la Figura 4.2 son *profesión* y *color-pelo*. En la red conceptual de la misma figura podemos ver dos tipos de arcos descriptivos: arcos que relacionen dos entidades independientes ya existentes y arcos utilizados para

definir una nueva entidad. Por ejemplo, el arco *profesión* une dos nod
informático) con existencia propia. Sin embargo, los nodos *Edad1* y *E*
nuevos conceptos que representan la edad de *Pepe* y de *Luis* respecti
y que se definen por sus relaciones con dichos nodos.

- Los **arcos estructurales** enlazan las entidades o conceptos formando
tectura o estructura de la red. A diferencia de los arcos descriptivos, la
ca de los arcos estructurales es independiente de los conocimientos del
que se está representando. Ejemplos de arcos estructurales en la Fi
son los etiquetados como *subclase-de*, *instancia* y *parte-de*, que se corre
respectivamente con los procesos básicos de **generalización**, **instan**
y **agregación**. No obstante, el IC puede definir, a medida, tantas e
estructurales como crea oportuno. Estos procesos básicos asociados a a
tructurales se pueden definir de la siguiente forma:

  - La **generalización** pone en relación una clase con otra más gene
mando una red de nodos por especialización de conceptos. Las prop
definidas en los nodos generales se heredadan por deducción en los
específicos, siguiendo los arcos *subclase-de*. Por ejemplo en la Figur
puede deducir que *Pepe* por ser *Hombre* y por ser *Persona* tiene *C*
  - El arco **instancia** liga un objeto concreto con su tipo genérico. Por ej
la aserción *"Pepe es un Hombre"* se representa en la Figura 4.2 m
el arco instancia desde el nodo *Pepe* hacia el nodo *Hombre*.
  - La **agregación** liga un objeto con sus componentes. Siguiendo con e
plo de la figura 4.2, la aserción *"la Cara forma parte de la Cabe*
representa utilizando el arco *parte-de* desde el nodo *Cara* hacia e
Cabeza.

Alternativemanete a esta representación gráfica, los conocimientos expresa
una red semántica también pueden expresarse utilizando lógica proposicional y c
de predicados de primer orden (véase el capítulo 2). Por ejemplo, teniendo en cue
conocimiento representado en la Figura 4.2 se pueden definir los siguientes predi

$$\text{PROFESIÓN}(Pepe, informático)$$
$$INSTANCIA(Pepe, Hombre)$$
$$SUBCLASE-DE(Hombre, Persona) \text{ o } \forall\, x\ Hombre(x) \Rightarrow Persona(x)$$
$$TIENE(PersonaCabeza)$$

Sin embargo, predicados n-arios (es decir de aridad superior a dos), como
ejemplo *COMPRA-VENTA(Pepe, Luis, reloj1, 45, euros)*, que representa la info
ción de que *"Pepe compra a Luis un reloj por 45 euros"*, no pueden ser represent
en estas redes conceptuales. Además, si en la red de la Figura 2 4.2 el IC qui
representar el predicado *VIO(Pepe, museo)*, que representa que *"Pepe vio un mu*
se estarían mezclando en la red arcos que representan atributos (*profesión*, *color-*

color-ojos y *edad*) con el arco que representa la acción *vio*. Esto, que inicialmente parece no crear ningún problema en la representación de conocimiento, desde el punto de vista de la inferencia sí que ocasiona problemas, ya que el razonador deduciría que *vio* es un atributo de *Pepe* mientras que *vio* es una acción realizada por *Pepe* y, por ello, tendría que tener un tratamiento diferente.

4.2.2 Representación de predicados no binarios

Los predicados de aridad distinta a dos se pueden representar también en redes semánticas. Por ejemplo, el predicado de aridad uno *HOMBRE(Pepe)* equivaldría a *INSTANCIA(Pepe, Hombre)*. El problema aparece cuando se utilizan predicados de aridad tres o superior. En este caso, un nuevo objeto representa al predicado de aridad mayor que dos, y nuevos predicados binarios describen las relaciones entre este nuevo objeto y sus argumentos. Por ejemplo, al representar el predicado *COMPRA-VENTA(Pepe, Luis, Reloj1, 45, euros)* en una red semántica, se crearía un nodo que representa la compra-venta (*compra-venta1*) y cinco predicados (*vendedor*, *comprador*, *objeto*, *precio*, *unidad*) que representan las relaciones con las cinco piezas de información, como se muestra en la Figura 4.3.

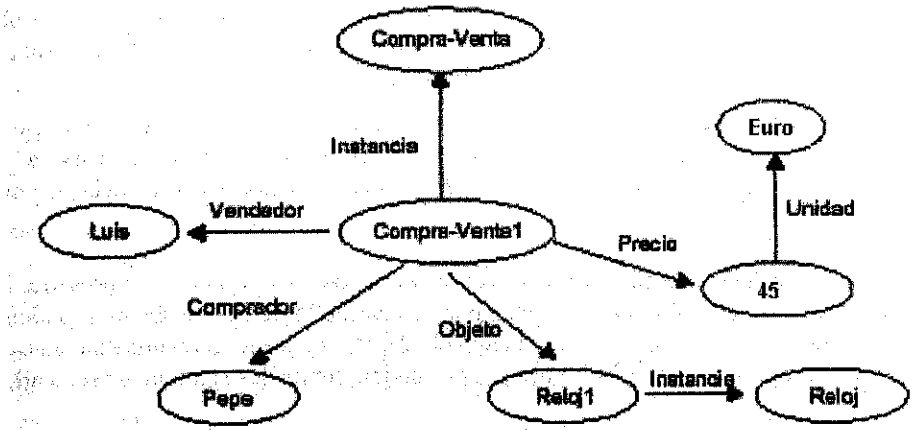


Figura 4.3: Representación de predicados no binarios.

4.2.3 Representación de acciones

La adaptación de conceptos de la gramática de casos de Fillmore [Fillmore, 1968] a las redes conceptuales permite al IC representar acciones. La gramática de casos se basa en que toda proposición contenida en una sentencia tiene una estructura

profunda formada por un verbo, que es el elemento principal, y una o r nominales. Cada frase nominal se relaciona con el verbo mediante un co casos. Los **casos** hacen referencia al:

- *Agente*: persona que realiza el evento.
- *Contra-agente*: fuerza o resistencia contra la que se ejecuta la acción.
- *Objeto*: la entidad que es movida, cambiada, o cuya posición o exi considera.
- *Resultado*: la entidad que aparece como consecuencia de la acción.
- *Instrumento*: el estímulo o causa física inmediata de un evento.
- *Origen*: el lugar del que procede el evento.
- *Propósito*: el motivo por el que se ejecuta la acción.
- *Lugar*: sitio en el que se desarrolla la acción.
- *Tiempo*: fecha o momento en el que tiene lugar la acción.
- *Sujeto*: entidad que recibe, acepta, experimenta o sufre el efecto de la

La **modalidad**, por su parte, hace referencia a características que p verbo, tales como:

- El *tiempo* en el que se ha desarrollado la acción (presente, pasado, fu
- La *voz* del verbo: activa o pasiva.

Un análisis detallado de cada verbo determina los casos que normalme asociados, permitiendo construir un patrón con sus casos obligatorios y con opcionales. El patrón se instanciará o se rellenará con los valores que toma o situación concreta.

Utilizando la información proporcionada por la gramática de casos, para tar afirmaciones que no se refieren a atributos de los objetos, sino a acciones se puede representar la afirmación *"Pepe vio el Prado en Madrid"* en una red ca utilizando nodos *situación* o *suceso*, tal y como se muestra en la Figura nodo situación tiene como atributos el conjunto de casos (*agente*, *objeto*, *lu* modalidades (*tiempo*, *voz*) que describen el evento, siendo el valor de cada a valor de cada caso, *(Pepe, El Prado, Madrid)*, o de cada modalidad *(Pasado* Con esta representación es relativamente sencillo representar acciones o se Además, las sentencias se pueden representar en lógica utilizando un pred aridad mayor que dos como *VIO(Pepe, El Prado, Madrid)*. Finalmente, n que el esquema de nodos situación permite representar sentencias compues *"Luis sabe que Pepe vio El Prado en Madrid"*, tal y como se muestra en la F

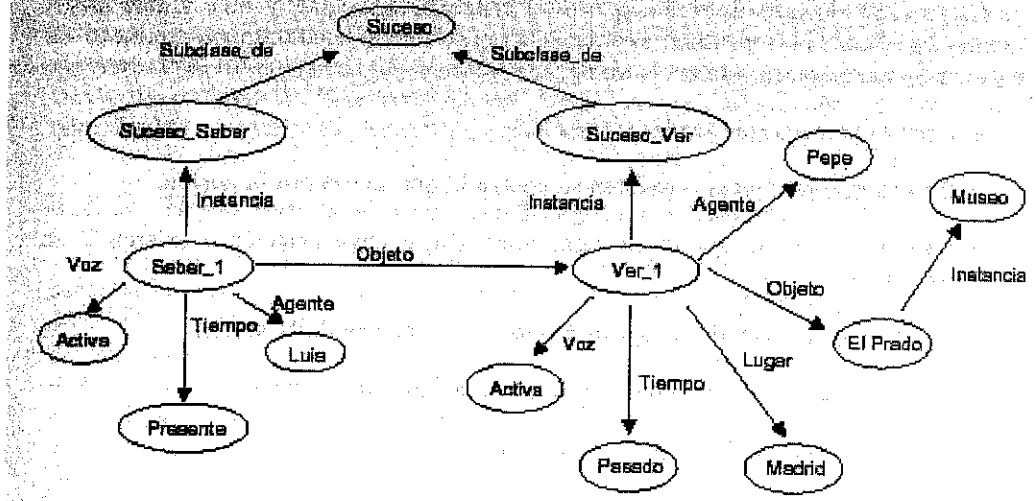


Figura 4.4: Representación de acciones en redes semánticas.

4.2.4 Representación de conocimiento disjunto

Hasta este momento se han analizado las redes semánticas para representar objetos y conceptos no excluyentes. Sin embargo, en ciertas ocasiones, el IC conoce qué entidades del dominio son disjuntas entre sí, y este conocimiento también debe expresarse en la red semántica. Conceptos o situaciones disjuntos pueden representarse en la red semántica utilizando la etiqueta disjunto (véase la Figura 4.5).

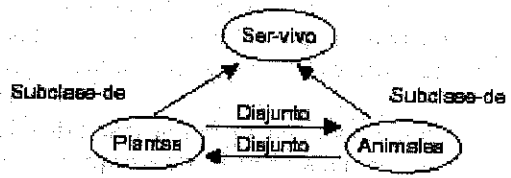


Figura 4.5: Representación de conocimiento disjunto en redes semánticas.

Sin embargo, Hendrix [Hendrix, 1975, 1979] dotó al formalismo de unos arcos especiales que permiten representar estos conocimientos de manera más sencilla, como se muestra en la Figura 4.6. Las etiquetas utilizadas en estos arcos especiales son las siguientes:

- *S*: Subconjunto.
- *SD*: Subconjunto disjunto.
- *E*: Elemento.
- *ED*: Elemento disjunto.

Figura 4.6: Representación de concimiento disjunto en redes semántic etiquetas.

4.3 Inferencia de conocimiento en redes sem

Un sistema que utilice como formalismo de representación de conoc redes semánticas, debe utilizar los conocimientos almacenados en la red los casos que se planteen. La eficacia del razonamiento en las redes depen cedimientos que trabajan con la semántica de sus arcos. Las técnicas má son la **equiparación** y la **herencia de propiedades**.

4.3.1 Equiparación

Se dice que un fragmento de red, apunte o consulta se equipara semántica, si el apunte se puede asociar con un fragmento de la red sem explicar la técnica de equiparación en redes semánticas se utilizará co conocimientos la red de la Figura 4.4, y la consulta *"¿Existe algún varón museo en Madrid?"*. Los pasos a seguir en el proceso de equiparación son

1. Se construye un apunte que responda a la pregunta en cuestión ut criterios seguidos en la construcción de la red semántica. El apunte e por un conjunto de **nodos constantes**, **nodos variables** y **arcos dos**. Los nodos constantes son los datos conocidos de la pregunta; ejemplo, *varón*, *Madrid*, *museo*, *suceso-ver*; los nodos variables son que se requieren y, por consiguiente, son desconocidos, *Ver-?* y *Va* arcos *instancia*, *agente*, *lugar*, *objeto* unen nodos constantes y nodo entre sí. El apunte descrito se muestra en la Figura 4.7.

2. Se coteja o superpone el apunte sobre la red semántica en la base mientos.

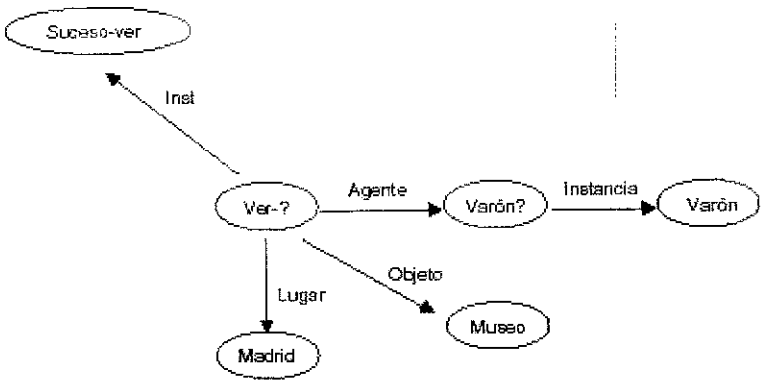


Figura 4.7: Apunte para la consulta *¿hay algún varón que viera un museo en Madrid?*

3. Los nodos variables se ligan a los nodos constantes de la red hasta encontrar una equiparación perfecta. Con este fin, en el ejemplo de la Figura 4.4, se busca en la red un nodo situación del que parta un arco *Lugar* hacia un nodo *Madrid*, y un arco *Instancia* hacia el nodo *Suceso-Ver*. Cuando esto ocurre, se liga la variable *Ver-?* del apunte con el nodo situación de la red, *Ver-1*, y el nodo *Varón?* del apunte con el nodo constante de la red *Pepe*. Dado que en el apunte se conoce que se trata de un varón, es necesario comprobar en la red que desde *Pepe* parte un arco *Instancia* hacia el nodo *Varón* (u *Hombre*).

4. La respuesta a la consulta es el fragmento de red semántica con los valores con los que se rellenan los nodos variables. En el ejemplo: *Ver-?* = *Ver-1* y *Varón?* = *Pepe*. Si no hubiera ninguna equiparación del apunte con la red semántica, la respuesta dada por la técnica de equiparación sería *"No se ha encontrado un varón que haya visto un museo en Madrid"*.

4.3.2 Herencia de propiedades

El concepto de herencia de propiedades tiene su fundamento teórico en la regla del **modus ponens**. La herencia de propiedades permite que nodos específicos de una red accedan a las propiedades definidas en otros nodos utilizando los arcos *Instancia* y *Subclase-de*, favoreciendo así la compartición de propiedades entre diferentes nodos y evitando la repetición de propiedades en la base de conocimiento. La herencia de propiedades se puede utilizar en sistemas que razonan dirigidos por la meta o por los datos.

Supóngase que se quiere determinar la veracidad de la sentencia *"Dumbo es de color gris"*, en la red semántica de la Figura 4.8. En primer lugar se debe localizar el nodo *Dumbo*, y después se debe buscar si desde dicho nodo sale un arco con la etiqueta *Color*. Al no existir el susodicho arco en *Dumbo*, el sistema buscará arcos *Instancia* que partan desde dicho nodo hacia algún otro nodo. En este caso, existen

En la red de la Figura 4.10, el problema aparece al aplicar herencia de propiedades. En este ejemplo, *Popi* hereda todas las propiedades de *Especie a Extinguir*, del mismo modo que hereda las de *Pájaro*, lo que puede ser cierto o no. La diferencia fundamental se encuentra en la naturaleza de los arcos *tienen* y *estudiado-por*, puesto que mientras que el primero se refiere a características intrínsecas al concepto *Pájaro*, el segundo no constituye un requisito para ser *Pájaro*. Por consiguiente, el problema se encuentra en el sentido del arco y en la etiqueta *estudiado-por*, y no en la técnica de inferencia. Al cambiar el sentido del arco y la etiqueta *estudiado-por* por la etiqueta *estudian*, el problema se resuelve.

4.4 Marcos

El formalismo de marcos se ha revelado como la técnica de representación de conocimientos más utilizada en IA cuando los conocimientos del dominio están organizados sobre la base de conceptos. Minsky [Minsky, 1985] definió los **marcos** como una estructura de datos que representa situaciones esteorotipadas construidas sobre situaciones similares ocurridas anteriormente, permitiendo así aplicar a situaciones nuevas los conocimientos de situaciones, eventos y conceptos previos. Los conocimientos que se expresan en los marcos son conocimientos declarativos del dominio. Dentro del marco, existen conocimientos procedimentales que se refieren a: cómo utilizar el marco, qué se espera que suceda a continuación, así como el conjunto de acciones que se deben realizar tanto si las expectativas se cumplen como si éstas fallan. Los marcos organizan los conocimientos del dominio en árboles, también llamados jerarquías, o en grafos, ambos construidos por especialización de conceptos generales en conceptos más específicos.

Las técnicas de inferencia utilizadas para razonar con los conocimientos de la base de conocimientos son: *equiparación*, para clasificar entidades en una jerarquía; *herencia simple* y *herencia múltiple*, para compartir propiedades que están distribuidas en la jerarquía de conceptos o en el grafo, respectivamente; y *valores activos* y *métodos* para representar la conducta del sistema.

4.4.1 Representación de conocimiento

Los conceptos que el IC utilizará al formalizar la base de conocimientos en marcos son: **marcos** para representar conceptos o elementos, **relaciones** para expresar dependencias entre conceptos, **propiedades** para describir cada concepto, y **facetas** para expresar de múltiples formas los valores con los que se puede rellenar cada propiedad. La Figura 4.11 muestra los conceptos básicos de representación en marcos.

4.4.1.1 Representación de conceptos e instancias

Básicamente, existen dos tipos de marcos: los **marcos clase** y los **marcos instancia**:

de partida en la inferencia. Por ejemplo, si en la red de la Figura 4.9 se pregunta por el color de *Brutus*, utilizando la herencia de propiedades se deduce que *"Brutus es de color Negro"*. Sin embargo, si se pregunta por el color de *Copito de Nieve*, el valor que se obtiene es *Blanco*, porque desde *Copito de Nieve* sale un arco etiquetado con color hacia el nodo *Blanco*.

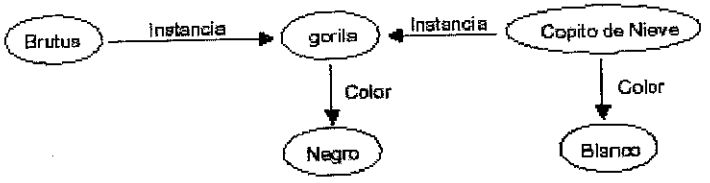


Figura 4.9: Ejemplo de tratamiento de excepciones.

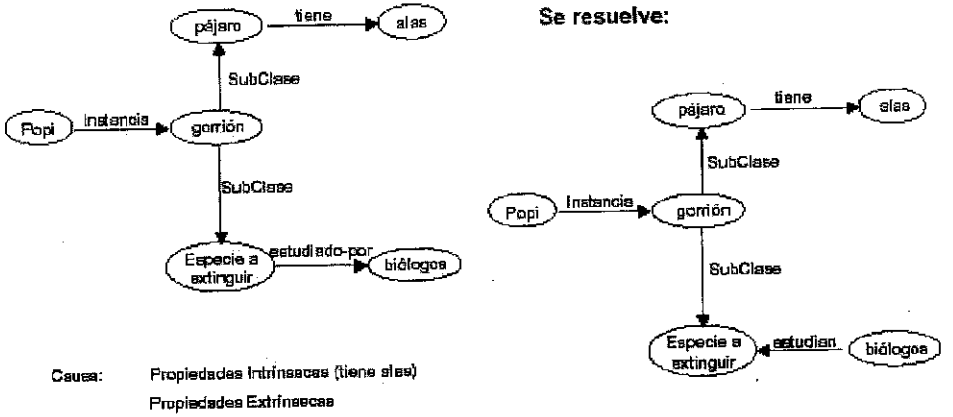


Figura 4.10: Problema en la aplicación de herencia de propiedades.

La inferencia basada en herencia de propiedades origina problemas si el IC ha formalizado mal los conocimientos. Los principales errores que se suelen cometer son:

- No distinguir los nodos que son instancias de aquellos que son conceptos.
- La etiqueta que da nombre al nodo o al arco tiene una semántica diferente al conocimiento que se quiere representar.
- El arco está en sentido contrario.
- No se han representado situaciones o acciones utilizando nodos situación.

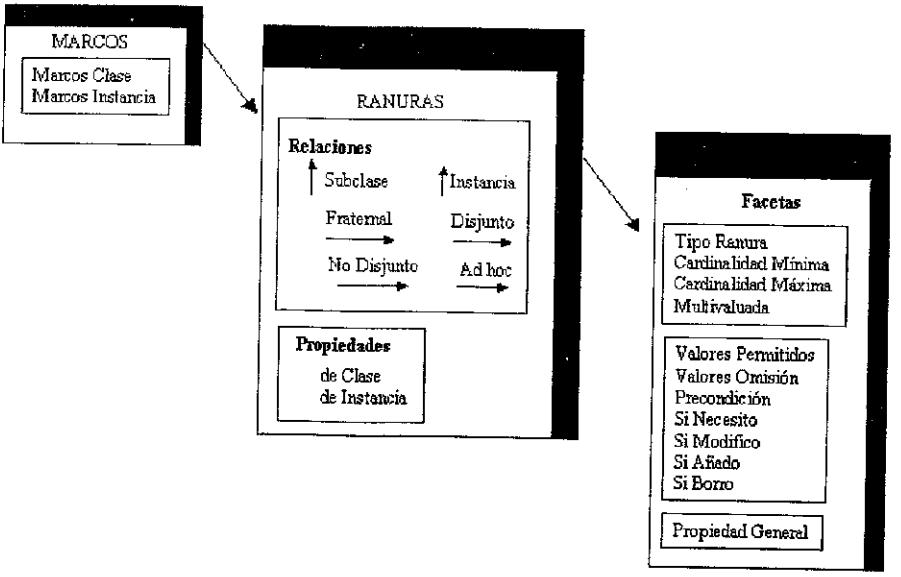


Figura 4.11: Conceptos básicos en marcos.

- Los *marcos clase* se utilizan para representar conceptos, clases o situaciones genéricas descritos por un conjunto de propiedades, unas con valores y otras sin valores asignados, que son comunes al concepto, clase o situación que el marco representa. Los marcos clase representan conceptos, es decir, entidades acerca de las cuales se desea describir cierto tipo de información. Los conceptos pueden ser de cualquier índole, ya se refieran a entidades físicas tangibles, descripción de tareas, procesos de razonamiento, entidades abstractas, etc. Los marcos *Persona*, *Hombre* y *Mujer* de la Figura 4.12 son ejemplos de marcos clase.

- En los dominios de trabajo de los expertos, existen elementos, instancias, o individuos de clases. Por ejemplo, el *martillo-1* de la clase herramienta para el experto carpintero. El formalismo de marcos permite representar estos objetos utilizando los *marcos instancia*. Los marcos instancia pueden considerarse como la representación en el dominio real de una clase determinada. Estos marcos instancia deben estar relacionados, como mínimo, con un marco clase. Además, han rellenado la mayoría de sus propiedades con valores específicos; es decir, con información concreta del elemento, instancia o individuo que representan, y que pertenece al concepto, clase o situación representado por un marco clase. El resto de propiedades que no estén almacenadas en él, las hereda de los marcos clase de los cuales son instancias. Los marcos *María*, *Ana*, *Luis*, *Pepe* y *Juan* son marcos instancia de los marcos clase *Mujer* y *Hombre* en la Figura 4.12.

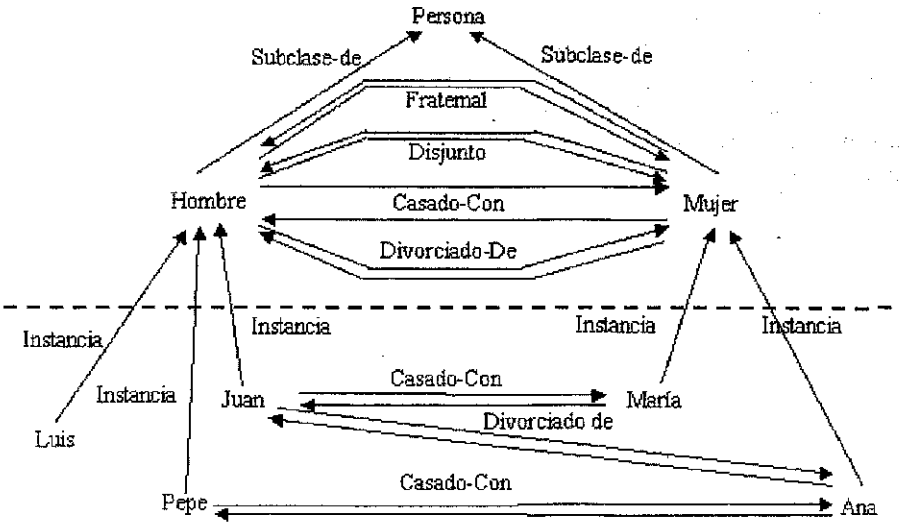


Figura 4.12: Ejemplo de una jerarquía en marcos.

4.4.1.2 Representación de relaciones entre conceptos

El formalismo de marcos representa las relaciones del dominio mediante relaciones entre marcos clase, entre marcos instancia, y entre marcos clase y marcos instancia, formando así un sistema basado en marcos (SBM). Intuitivamente, el significado del SBM de la Figura 4.12 es el siguiente:

- Los marcos clase *Hombre* y *Mujer* son subclases del marco clase *Persona*, y el marco clase *Persona* es una superclase de los marcos clase *Hombre* y *Mujer*.
- Los marcos instancia *Luis*, *Pepe* y *Juan* son instancias del marco clase *Hombre* y, por lo tanto, el conjunto de los *Hombres* está formado por *Luis*, *Pepe* y *Juan*.
- Los marcos instancia *Ana* y *María* son instancias del marco clase *Mujer* y, por lo tanto, el conjunto de las *Mujeres* está formado por *María* y por *Ana*.
- Los marcos clase *Hombre* y *Mujer* son hermanos, pues tienen el mismo padre.
- Los marcos clase *Hombre* y *Mujer* son disjuntos, pues no hay instancias que pertenezcan simultáneamente a ambos marcos clase.
- Las relaciones *casado-con* y *divorciado-de* definidas entre marcos clase representan que los *Hombres* y las *Mujeres* se casan y se divorcian. Las relaciones *casado-con* y *divorciado-de*, definidas entre marcos instancia, representan que *Juan* está casado con *María* y divorciado de *Ana*, y que *Ana* está casada con *Pepe*.

estas relaciones sea correcto. La semántica correcta es que el concepto representado por MC_i es especialización o subconjunto del concepto representado por MC_j. La relación inversa de la relación *subclase-de* es la relación *superclase-de*. Dado que un concepto puede especializarse en varios conceptos o ser clasificado desde varios puntos de vista, a un marco clase pueden llegar y/o partir un número indefinido de relaciones *subclase-de* y *superclase-de*. Las relaciones *subclase-de* en los SBM definen un camino para la herencia de propiedades. Dada la jerarquía de la Figura 4.12, la herencia de propiedades permitirá que todas las propiedades que se definen en el marco clase *Persona* sean accesibles desde *Hombre* y desde todas sus instancias, definiéndose en *Hombre* sólo aquellas propiedades que distinguen a esta clase de la clase *Persona*, y en la clase *Persona* aquellas propiedades que son comunes a los marcos clase *Hombre* y *Mujer*.

Sintácticamente, una relación *instancia* está bien definida si tiene como origen un marco instancia MI_i y como destino un marco clase MC_j. Cuando el IC utiliza la relación *instancia* debe estar seguro de que el padre es un marco clase y de que el hijo es un marco instancia. Si no es así, sintácticamente la relación no tiene sentido, y la base de conocimiento (BC) será sintácticamente incorrecta. Semánticamente, esta relación representa que el marco instancia es un elemento del conjunto o clase representado por el marco clase. Dado que un elemento puede pertenecer a varios conjuntos simultánemente, de un marco instancia pueden partir tantas relaciones instancia como conceptos lo describan consistentemente. Como se ha mencionado anteriormente, estas relaciones estándar permiten construir jerarquías o clasificaciones de conocimientos estáticos del dominio que se quiere modelar. Esta jerarquía será un árbol si existe un único camino que une cada marco instancia con el nodo raíz de la jerarquía, y un grafo en caso contrario.

Las relaciones **no estándar** representan dependencias entre conceptos de un dominio. El IC debe tener cuidado con que el significado de lo que quiere representar sea correcto. Como ejemplo de relaciones que representan la estructura de los conceptos del dominio se tienen las siguientes: *fraternal*, *disjunto* y *no-disjunto*.

- La relación *fraternal* está sintácticamente bien definida si los marcos origen y destino son marcos clase, y ambos están unidos mediante relaciones *subclase-de* con el mismo marco clase padre. Semánticamente, representa que dos conceptos son hermanos y, por consiguiente, tienen que tener el mismo padre.

- Una relación *disjunto* está sintácticamente bien definida si los marcos origen y destino son marcos clase. Semánticamente, representa que las clases son disjuntas, es decir, que tienen como intersección el conjunto vacío o, lo que es lo mismo, que los conjuntos o clases representados por ambos marcos no tienen ningún elemento en común. Además, esta relación está bien definida si no se ha definido previamente una relación *no-disjunto* entre ambos marcos clase.

- La relación *no-disjunto* es opuesta semánticamente a la relación *disjunto*. Por consiguiente, marcos clase conectados por relaciones *no-disjunto* pueden tener instancias comunes.

Al igual que ocurría en el formalismo de las redes semánticas, hay ciertas relaciones (*subclase-de* e *instancia*) que son independientes del dominio, y que reciben el nombre de **relaciones estándar**. La técnica de inferencia basada en herencia de propiedades razona sobre dichas relaciones. Sin embargo, también existen otras relaciones llamadas **relaciones no estándar**, para representar relaciones "a medida" entre conceptos de un dominio, sobre las que no se puede aplicar herencia de propiedades. La Figura 4.13 representa gráficamente la sintaxis de las relaciones más utilizadas en el paradigma de marcos, y que son las siguientes:

- *Subclase-de*, y su relación inversa *superclase-de*.
- *Instancia*, y su relación inversa *representa*.
- *Fraternal*.
- *Disjunto*.
- *No disjunto*.
- *Ad-hoc*, o relaciones *"a medida"*.

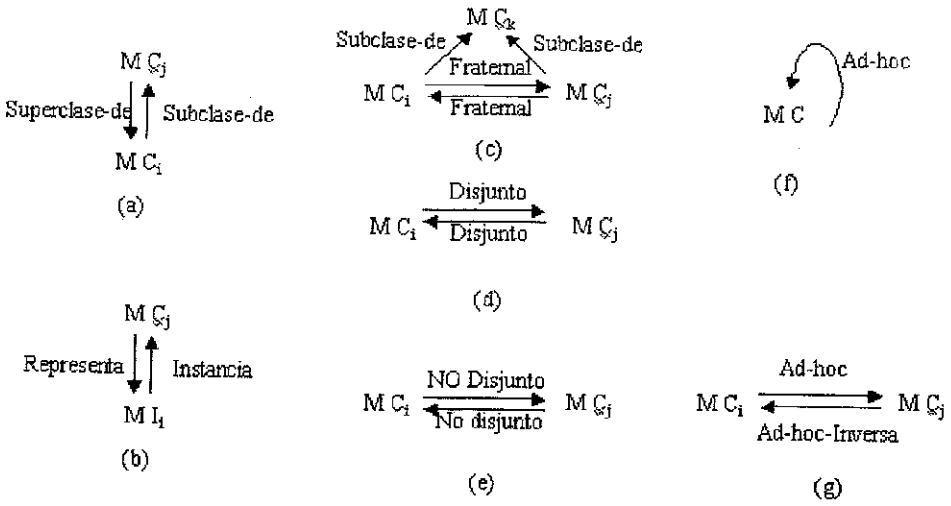


Figura 4.13: Sintaxis de relaciones en marcos.

Se consideran **relaciones estándar** las de *subclase-de* e *instancia*, y sus respectivas relaciones inversas llamadas *superclase-de* y *representa*. La relación *subclase-de* permite construir un SBM mediante la especialización de conceptos generales en conceptos más específicos. Sintácticamente, la relación *subclase-de* está bien definida si los marcos origen y destino son marcos clase. Desde el punto de vista semántico, el IC debe tener cuidado con que el significado de lo que quiere representar utilizando

Las relaciones *ad-hoc* también se utilizan para conectar marcos clase de diferentes jerarquías. Normalmente, el IC construye varias jerarquías para representar clasificaciones de los principales conceptos que forman el sistema. Por ejemplo, supongamos un sistema basado en el conocimiento (SBC) para la liga de fútbol, como mínimo se necesitan dos jerarquías: una para clasificar a los jugadores de fútbol y otra para clasificar los equipos de fútbol en función de la división en la que juegan. Se puede, por tanto, definir una relación a medida *pertenece* entre los marcos clase *jugadores-de-fútbol* y *equipos-de-fútbol* sujeta a las restricciones de que ambos, el *jugador* y el *equipo*, tiene que pertenecer a la misma división, y que un *jugador* pertenece a un único *equipo*. Cuando se crea una relación *pertenece* instanciada entre un *jugador* y un *equipo* concreto, además de comprobar que la relación está definida entre dos marcos clase y que dichas instancias lo son de dichos marcos clase, el IC, para evitar errores, comprobará que se cumplen las restricciones establecidas cuando se definió la relación. Gráficamente, la relación entre ambas jerarquías se muestra en la Figura 4.15.

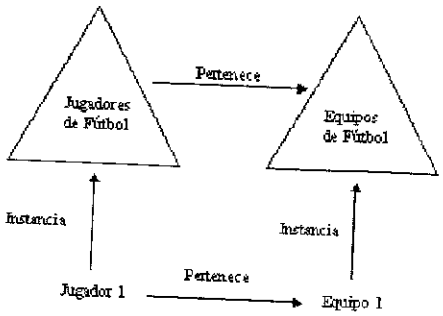


Figura 4.15: Ejemplos de relaciones *ad-hoc* entre diferentes jerarquías.

Muchas de las herramientas existentes para construir SBM no implementan este tipo de relaciones no estándar. Esto significa que el IC debe buscarse un "truco" para representarlas (por ejemplo, mediante otro marco), si realmente las necesita en su sistema. En este caso, a diferencia de las relaciones estándar, el motor de inferencias no trabajará con ellas, y el IC deberá implementar las inferencias bien con procedimientos programados, con reglas, con demonios, o con cualquier otro modo de expresar conocimientos procedimentales.

4.4.1.3 Representación de las propiedades de los conceptos

El IC utiliza dos tipos de propiedades cuando formaliza su BC en marcos: **propiedades de clase** y **propiedades de instancia**:

- Las *propiedades de clase* representan atributos o características genéricas de un concepto o clase. El IC define y rellena estas propiedades en el marco clase, y toman siempre el mismo valor en todos los elementos o instancias de la clase.

Aparte de estas relaciones no estándares, también podemos definir relaciones *a medida* o *ad-hoc*, como por ejemplo, *pertenece*, *casado-con*, *divorciado-de*, *conectado-a*, etc. Las relaciones *ad-hoc* expresan dependencias a medida entre conceptos de un dominio. Cuando se definen relaciones *ad-hoc* en un dominio, lo que ocurre frecuentemente, hay que asegurarse de definirlas entre marcos clase, y no entre marcos instancias.

Gráficamente, las figuras (f) y (g) en la Figura 4.13 muestran la sintaxis de relaciones *ad-hoc* entre marcos clase. Ejemplos de relaciones ad-hoc en la Figura 4.14 son las siguientes: la relación *casado-con* entre los marcos *Hombre* y *Mujer*; la relación *conectado-a*, que conecta, por ejemplo, computadoras en una red; y, las relaciones *compuesto-de* y *forma-parte-de*, utilizadas para describir un sistema especificando los subsistemas que lo componen y los sistemas de los que él forma parte. Cuando se definan relaciones *ad-hoc* entre dos marcos instancia, previamente hay que comprobar lo siguiente:

- Que la relación *ad-hoc* haya sido definida previamente entre dos marcos clase.
- Que los marcos instancia que se quiere conectar sean instancias de dichos marcos clase.

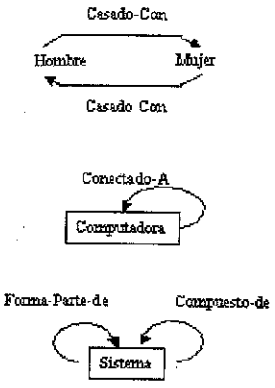


Figura 4.14: Ejemplos de relaciones *ad-hoc* entre marcos clases.

En el ejemplo de la Figura 4.12, antes de introducir la relación *casado-con* entre *Juan* y *María*, y entre *María* y *Juan*, el IC debe comprobar que la relación *casado-con* está definida en la BC y que *Juan* es instancia de *Hombre* y que *María* es instancia de *Mujer*. Solamente entonces, la relación se crea entre *Juan* y *María*, y entre *María* y *Juan*. Se procede de la misma forma para representar el hecho de que *"Ana está casada con Pepe"*. Es importante mencionar que cada relación *ad-hoc* entre dos marcos instancia es una instancia de la relación *ad-hoc* definida a nivel de clase. Por ejemplo, en la jerarquía de la Figura 4.12 existen cuatro instancias de la relación *ad-hoc casado-con*.

En la jerarquía de la Figura 4.16, la propiedad *animal-racional* en el marco clase *Persona* y la propiedad *sexo* en los marcos clase *Hombre* y *Mujer* son *propiedades de clase.*

- Las *propiedades de instancia*, aunque el IC las define en el marco clase y son comunes a todas las instancias del marco clase, se rellenan en cada instancia con valores concretos que dependen del elemento de la clase que se esté representando. Gráficamente, si la propiedad va precedida del símbolo "*" se trata de una propiedad de instancia. En el SBM de la Figura 4.16, las propiedades *nombre, edad, nacionalidad, estado-civil, tipo-de-matrimonio, religión, fecha-de-boda* y *fecha-de-nacimiento* son propiedades de instancia en el marco clase *Persona.*

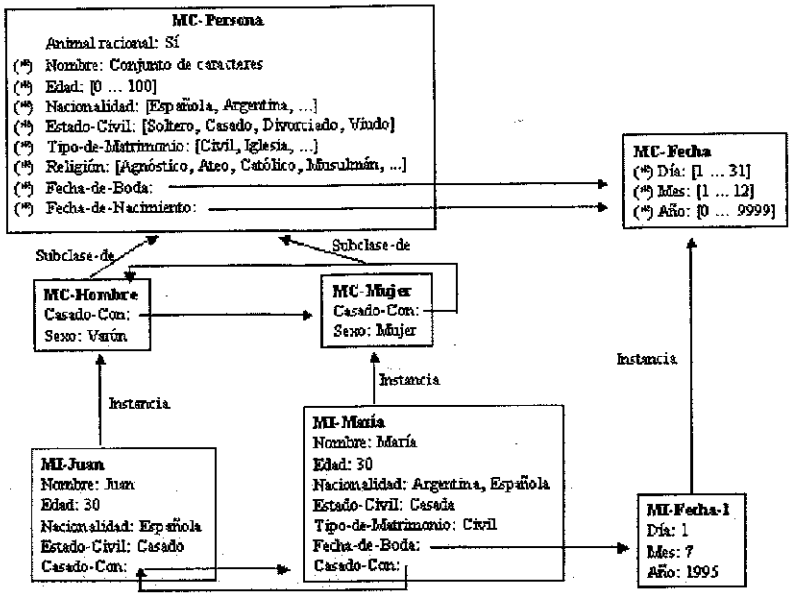


Figura 4.16: Ejemplos de propiedades en un SBM.

En la Figura 4.16, en los marcos clase, las propiedades de clase se han rellenado con el valor que toma la propiedad, y las propiedades de instancia con el tipo de valor con el que éstas se pueden rellenar en las instancias. Concretamente, con un tipo de datos (entero, conjunto de caracteres, etc.) o un puntero a un marco clase. Además, en los marcos instancia no se han rellenado todas las propiedades de instancia definidas en los marcos clase. Esto es debido bien al desconocimiento del valor de una propiedad o bien a la intención de no repetir información. Por ejemplo, si se sabe que *"Juan está casado con María"*, los dos tendrán el mismo *tipo-de-matrimonio*, *fecha-de-boda*, etc. Este tipo de inferencias van asociadas a la relación *ad-hoc casado-con.* Dado que la mayoría de las herramientas no trabajan con este tipo de relaciones, el IC deberá crear un procedimiento para que el motor de inferencias sea capaz de extraer las mismas

conclusiones que un humano en las mismas circunstancias. Al definir las propiedades de clase y de instancia en los marcos clase, el IC debe tener en cuenta lo siguiente:

- La distribución de las propiedades en el SBM debe favorecer que unos marcos compartan propiedades con otros marcos, evitando la presencia de conocimientos redundantes. Por ejemplo, la definición de la propiedad *nombre* o *animal-racional* en los marcos *Hombre* y *Mujer* haría que la jerarquía de la Figura 4.16 fuese redundante.

- Las propiedades deben proporcionar la suficiente información como para determinar la entidad que el marco clase está representando, discriminando las entidades representadas por él de las que no lo son. Como ejemplo, se tiene la propiedad *sexo* en los marcos clase *Hombre* y *Mujer*; si no estuviera definida haría indiscriminables las instancias de una y otra clase.

- El nombre de cada propiedad refleja su semántica, pero no indica la importancia que tiene la propiedad en el marco clase.

- El carácter local de cada propiedad permite tener propiedades con el mismo nombre en diferentes marcos. Por ejemplo, la propiedad *sexo* en los marcos clase *Hombre* y *Mujer*.

- Un marco clase rellena las propiedades de clase con unos valores concretos.

- En un marco clase se puede definir una propiedad de instancia utilizando otros marcos clase. Por ejemplo, las propiedades *fecha-de-nacimiento* y *fecha-de-boda* se definen a través del marco clase *Fecha*.

- Un marco instancia puede rellenar o no todas las propiedades de instancia definidas en los marcos clase con los que está conectado. Por ejemplo, en el marco instancia *Juan* no se ha rellenado las propiedades *fecha-de-boda* y *tipo-de-matrimonio* porque toman los mismos valores que los del marco instancia *María*. Otras propiedades, como la propiedad religión podrían no rellenarse, por ejemplo, por desconocimiento del valor.

4.4.1.4 Representación de facetas de propiedades

Identificadas las relaciones y distribuidas las propiedades en los marcos clase, el IC pasa a describir las facetas de cada una de las propiedades. Las **facetas** permiten modelar características de las propiedades y relaciones en los marcos clase. Declarativamente, una propiedad se define especificando un puntero o un tipo de datos (lógico, carácter, número, entero, real, o como un conjunto restringido de valores numéricos, caracteres, etc.); procedimentalmente, se define utilizando un procedimiento o una regla. Las facetas, independientemente de que se rellenen con valores, punteros, o procedimientos, se clasifican en las siguientes tres categorías: facetas que definen propiedades de clase, de instancia y relaciones; facetas que define propiedades de clase y relaciones; y facetas que definen propiedades de instancia. El motor de inferencias usa las facetas para mantener la integridad semántica de los datos, es decir, para

comprobar que los valores introducidos en las propiedades realmente pertenecen al tipo especificado, en la obtención de valores, y en la asignación de valores por defecto.

En cualquier propiedad o relación que defina el IC, obligatoriamente, debe especificar las **facetas que definen propiedades de clase, de instancia y relaciones**: *tipo ranura*, *cardinalidad mínima* y *cardinalidad máxima* de valores que puede tomar la propiedad o relación, y *multivaluada* si la propiedad o relación toma más de un valor.

- *Tipo ranura*: esta faceta establece el tipo de datos con el que se rellenará la propiedad o relación, garantizando que, una vez rellenada la propiedad o relación con unos valores concretos, dichos valores pertenecerán al tipo especificado en ella.

  - Si se trata de propiedades de clase o de instancia que se rellenan con unos valores pertenecientes a determinados tipos de datos, en esta faceta se especificará el tipo correspondiente. Por ejemplo, en el marco clase *Persona*, las propiedades *nombre*, *estado-civil*, *religión*, *nacionalidad* y *tipo-de-matrimonio* se definen como un conjunto de caracteres; la propiedad *edad* se define como un entero corto; y, la propiedad *animal-racional* como de tipo lógico.
  - Si se trata de propiedades de clase o de instancia definidas como marcos, en esta faceta se especifica, precisamente, que se trata de un marco. Por ejemplo, la definición de las propiedades *fecha-de-nacimiento* y *fecha-de-boda* como un marco evitará repetir la descripción del concepto *Fecha* en cada una de ellas.
  - Si se trata de relaciones, la relación se definirá siempre en el marco clase origen de la relación, tendrá como nombre el de la relación, y en esta faceta se deberá especificar que se trata de un marco. Como ejemplo se tiene las relaciones *subclase-de*, *casado-con* y *divorciado-de* en los marcos clase *Hombre* y *Mujer*.

- *Cardinalidad mínima*: esta faceta establece el número mínimo de valores con los que se rellena la ranura, siempre que ésta se rellene. Así, la cardinalidad mínima de todas las ranuras definidas en el marco clase *Persona* es uno.

- *Cardinalidad máxima*: esta faceta informa del número máximo de valores con los que se puede rellenar la ranura. Por ejemplo, en el marco clase *Persona*, la cardinalidad máxima de todas las propiedades es uno, salvo las cardinalidades de las propiedades *nacionalidad* y *tipo-de-matrimonio*. La propiedad *nacionalidad* podría llegar a tomar N valores, suponiendo que N sea el número máximo de nacionalidades que una persona puede tener.

- *Multivaluada*: esta faceta establece si la propiedad puede tener más de un valor o no. Si la *cardinalidad mínima* es igual a la *máxima*, y ambas son iguales a uno, entonces la propiedad no es multivaluada. En caso contrario, si la *cardinalidad*

mínima y la *máxima* son iguales y ambas diferentes de uno, o bien si la *cardinalidad mínima* es diferente de la *máxima*, entonces la ranura sí es *multivaluada*. El IC deberá realizar estas comprobaciones para evitar inconsistencias. En el ejemplo de *Persona*, las únicas propiedades multivaluadas son las de *nacionalidad* y *tipo-de-matrimonio*. El resto de las propiedades no son multivaluadas; y se pueden denominar simples.

Las propiedades de clase que se definieron en la faceta *tipo ranura* como un tipo de datos, rellenan la faceta **propiedad general**, que es una **faceta común a las propiedades de clase y relaciones**, con los valores que toma la propiedad en el marco clase. Dichos valores serán consistentes con los tipos especificados en la faceta *tipo ranura*. Las propiedades de clase definidas como marcos y relaciones rellenan esta faceta con un puntero a un marco clase. Las propiedades de instancia nunca rellenan esta faceta con valores o punteros. Conviene, no obstante, asignar a las propiedades de instancia un cierto valor que indique que no se rellenan; por ejemplo, el símbolo "––". Por ejemplo, en el marco clase *Persona* la propiedad de clase *animal-racional* rellena esta faceta con un valor de tipo lógico; la relación *superclase-de* con punteros a los marcos clase *Hombre* y *Mujer*; y, las propiedades de instancia con el símbolo "__".

Para cada una de las propiedades de instancia definidas en un marco clase, se definirán las correspondientes **facetas que definen propiedades de instancia**: *valores permitidos de la propiedad*, *valores por omisión* o por *defecto* asignados a la propiedad, *si necesito*, *si modifico*, *si añado* y *si borro*, que se utilizan al necesitar, modificar, añadir o borrar un valor de una propiedad.

- *Valores Permitidos.* Esta faceta especifica el conjunto de valores válidos que puede tomar la propiedad de instancia. Concretamente, se utilizará: un tipo de datos, un rango de valores, o un puntero a un marco clase. Independientemente de cómo se defina la propiedad, el IC debe comprobar que el conjunto de valores especificados es consistente con el tipo almacenado en la faceta *tipo ranura*. Ejemplos de definiciones para el marco clase *Persona* son los siguientes:
  - Las propiedades *nombre*, *nacionalidad* y *tipo-de-matrimonio* tienen como valores permitidos un tipo de datos.
  - Los *rangos de valores* se utilizan para restringir el número de posibles valores con los que se rellena la propiedad en la instancia. Con el fin de preservar la integridad semántica de la BC, el rango de valores impide que propiedades de instancia se rellenen con valores que sí satisfacen la definición dada en la faceta *tipo ranura*, pero que no se dan en el dominio de la aplicación. Aunque ello conlleve un mayor esfuerzo, se aconseja al IC que defina un rango de valores frente a un tipo de datos. Ejemplos del uso de un rango de valores son las propiedades *edad*, *estado-civil* y *religión*.
  - Las propiedades de instancia definidas en la faceta *tipo ranura* como marcos rellenan esta faceta con un puntero al marco clase que define la propiedad. Como ejemplo, se tienen las propiedades *fecha-de-nacimiento* y *fecha-de-boda* en el marco clase *Persona*.

- *Valores por Omisión.* Esta faceta define los valores que debe tomar la propiedad de instancia en un marco instancia si no se conoce de forma explícita otro valor. No obstante, marcos instancia y marcos clase pueden anular el valor por omisión dado a la propiedad en otro marco clase, bien al asignar un valor nuevo, como ocurre, por ejemplo, con las propiedades que presentan excepciones, bien al añadirle valores adicionales. Es aconsejable, como simpre, rellenar las propiedades de instancia que no tienen asociados valores por omisión, las propiedades de clase y las relaciones con el símbolo "––". El valor asignado en esta faceta será siempre del tipo especificado en la faceta *tipo ranura.* Por ejemplo, si la jerarquía de *Personas* se va a utilizar únicamente en *España*, por omisión, la propiedad *nacionalidad* se puede rellenar con el valor *española.*

- *Si Necesito.* Esta faceta almacena un procedimiento o regla que se ejecuta al solicitar el valor de una propiedad de instancia en un marco instancia y ser desconocido dicho valor. Estos procedimientos o reglas se utilizan para:

  - *Requerir el valor de una propiedad.* Si el valor de una propiedad de instancia en un marco instancia es desconocido, los procedimientos pueden preguntar dicho valor al usuario del sistema, o bien, pueden calcular el valor a partir de otros valores conocidos almacenados en la BC.
  - *Mantener la integridad semántica de la BC.* Cada vez que el usuario introduzca un valor en alguna propiedad de una instancia, el SBM comprobará que se cumplen las restricciones impuestas en las facetas de dichas propiedades, y sólo si los valores son correctos se aceptan. Nótese que ésta es una funcionalidad extremadamente interesante del formalismo de marcos, puesto que la propia estructura de los marcos permite mantener las restricciones de integridad semántica entre los elementos del dominio. El IC deberá aprovechar esta funcionalidad ofrecida por el formalismo para disminuir las posibilidades de valores erróneos en los contenidos de las propiedades rellenadas en los marcos instancia.
  - *Gestionar dinámicamente de valores.* Los procedimientos almacenados en la faceta *si necesito* también se utilizan para obtener dinámicamente el valor de una propiedad de un marco instancia si el conjunto de valores con los que se rellena la propiedad en la instancia se puede obtener a partir de los valores almacenados en otras propiedades y relaciones.
  - *Determinar dinámicamente el rango de valores.* Si el conjunto de valores con los que se rellena una propiedad en el marco instancia dependen de los valores que se han rellenado en otras propiedades y relaciones del mismo o de otros marcos instancia, dinámicamente se puede determinar el rango de valores permitidos definiendo un procedimiento en esta faceta.

- *Si Modifico.* Esta faceta almacena el procedimiento que se ejecuta al modificar un valor de una propiedad de un marco instancia. La ejecución de este procedimiento puede añadir, modificar y borrar valores de otras ranuras, disparando así sus procedimientos asociados. Por ejemplo, si se modifica el *estado-civil* de

Juan, pasando a estar *divorciado*, al modificar este valor, se deberían borrar del SBM las relaciones *ad-hoc casado-con* entre los marcos instancia *Juan* y *María*, y se deberían crear dos nuevas relaciones *ad-hoc divorciado-de* entre dichos marcos instancia.

- *Si Añado.* En esta faceta se almacenan procedimientos que se ejecutan al introducir un valor en una propiedad de un marco instancia que estaba vacía. Se utiliza para:(a) Mantener la integridad semántica de la BC; y (b).añadir, modificar y borrar valores de otras ranuras.

- *Si Borro.* Esta faceta almacena el procedimiento que se ejecuta al borrar un valor en una propiedad de un marco instancia. La ejecución de este procedimiento puede añadir, modificar y borrar valores en otras ranuras, disparando de este modo sus procedimientos asociados.

4.4.2 Criterios de diseño

En este apartado se presentan algunos criterios de diseño que pueden ayudar al IC a representar el conocimiento utilizando el formalismo de marcos:

- Se debe favorecer la compartición de propiedades de clase y de instancia entre marcos.

- Se debe evitar representar conocimientos redundantes.

- En cada marco clase debe haber una propiedad de clase que identifique a los elementos de dicha clase.

- Semántica de las propiedades: las propiedades deben tener o aportar significado a la representación.

- Debido al carácter local de las propiedades, se pueden tener propiedades repetidas con el mismo nombre en diferentes marcos clase.

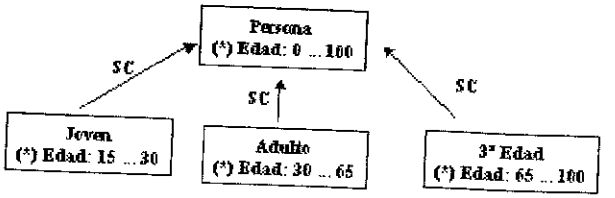


Figura 4.17: Ejemplo de redefinición de propieades.

- En caso necesario, se pueden redefinir las propiedades (de clase e instancia) en marcos clase más específicos (tal y como muestra el ejemplo de la Figura 4.17).

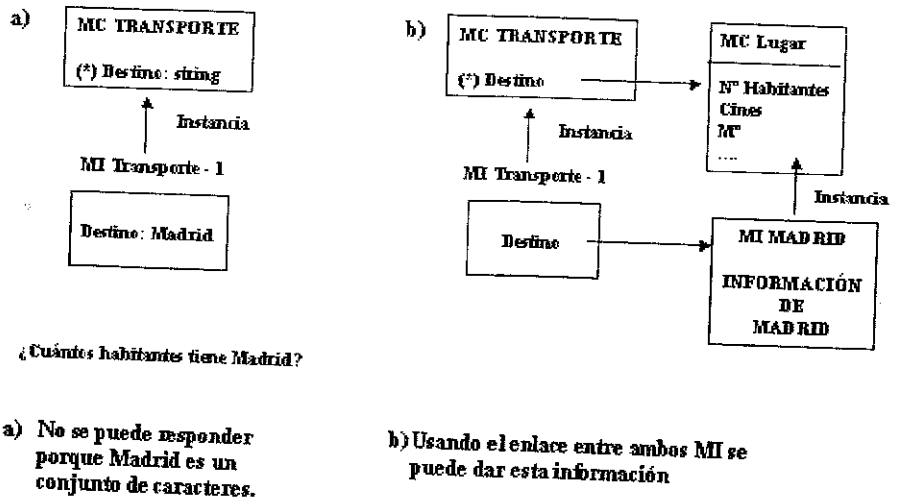


Figura 4.18: Ejemplo de propiedad definida como marco.

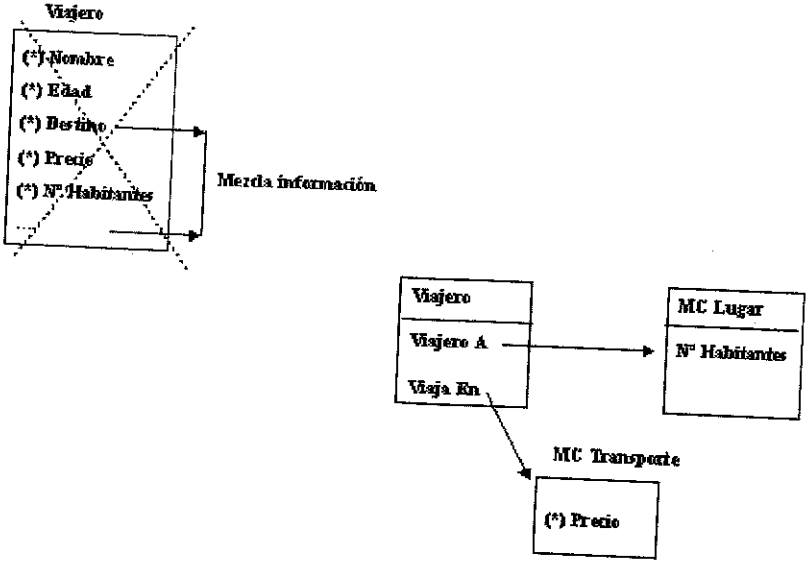


Figura 4.19: Ejemplo de representación de varios conceptos.

- Si se desea conocer información o propiedades de las propiedades de un marco, entonces dichas propiedades deben ser definidas como marcos (tal y como se muestra en el ejemplo de la Figura 4.18).

- No se deben mezclar conceptos. Por ejemplo: el caso de un viajero que va a un destino usando un transporte que tiene un precio se debería representar como se muestra en la Figura 4.19.

- Hay que tener en cuenta que en un marco instancia se pueden rellenar, o no, todas las propiedades de instancia definidas en los marcos clase con los que está conectado.

- Hay que recordar que las propiedades de instancia se rellenan con un valor concreto.

- Es importante recordar que no se pueden utilizar en las instancias propiedades que no se hayan definido en los marcos clase.

4.5 Inferencia de conocimiento en SBM

El formalismo de marcos permite realizar inferencias utilizando los conocimientos almacenados en la BC. En concreto, las tres técnicas que utilizan los marcos, de forma integrada o de manera individualizada, son: la **equiparación** para clasificar entidades en la BC; la **herencia** de propiedades para compartir propiedades entre marcos; y los **valores activos** para representar la conducta del sistema y mantener la integridad de los datos almacenados.

4.5.1 Equiparación

Equiparar significa clasificar. En este sentido, conocidos los valores de un conjunto de propiedades que describen parcialmente una nueva entidad o marco pregunta, la técnica de equiparación clasifica el marco pregunta en el árbol o en el grafo que representa los conceptos del dominio. Para ello, la técnica de equiparación tiene que encontrar los marcos clase (uno o varios) de la BC que describen más consistentemente al marco pregunta. Encontrados los marcos clase (uno o varios), el marco pregunta se convierte en un marco instancia de dichos marcos clase. El problema es el siguiente: si los valores de todas las propiedades de una entidad son conocidos y dichas propiedades se encuentran almacenadas en un único marco clase de la BC, entonces la equiparación de la nueva entidad con el marco clase es perfecta. Sin embargo, esta situación raramente ocurre debido a que:

- Las propiedades conocidas en la nueva entidad se encuentran distribuidas en la BC por todos los marcos clase del árbol o grafo de conceptos.

- No todas las propiedades de la nueva entidad son conocidas al empezar la equiparación. Los conocimientos que se poseen de la entidad son limitados, pues, habitualmente, se conoce un conjunto finito, no muy amplio, de sus propiedades.

- Aun conociendo los valores de algunas propiedades de la nueva entidad, normalmente se tienen distintos grados de seguridad en los valores que se han asignado.

- No todas las propiedades almacenadas en el marco pregunta aportan la misma información a la hora de realizar la equiparación.

- Los valores que por omisión se han asignado a las propiedades en los marcos clase no siempre representan información cierta para todas las instancias, pues existen excepciones para dichos valores. Se pueden realizar varias equiparaciones correctas de un marco pregunta con varios marcos clase.

Básicamente, las tres etapas en las que se descompone la técnica de equiparación son: *selección de marcos candidatos*, *cálculo de valores de equiparación* y, finalmente, *decisión*.

1. La *selección de marcos candidatos* se realizará siguiendo alguno de los siguientes procedimientos:

   - Si el *tipo* de una nueva entidad es conocido, se puede seleccionar el marco clase en el que se ha definido el *tipo*, y todos los marcos clase en los que éste se ha especializado. Por ejemplo, si se sabe que se trata de un perro, entonces se selecciona el marco clase *Perro* y todas sus subclases, pero no el marco clase *Mamífero* o el marco clase *Gato*.
   - Si el *tipo* de la nueva entidad es desconocido, la selección de marcos clase se realiza arbitrariamente, o se eligen aquellos marcos clase en los que, como mínimo, se encuentre definida una propiedad conocida en el marco pregunta. En este caso, los marcos clase que no tengan al menos una propiedad en común con el marco pregunta no se seleccionan. Por ejemplo, dada una entidad que tiene pelo y cuatro patas, se utilizarían las propiedades *Tiene-Pelo* y *Número-Patas* para la selección de marcos candidatos.

2. Seleccionados los marcos clase candidatos, se pasa a *calcular el valor de equiparación* (VE) de la nueva entidad en cada una de las clases seleccionadas. El VE es una medida que informa del grado de idoneidad de la equiparación que se va a realizar. El método de cálculo del VE variará de unas aplicaciones a otras. Por ejemplo, si los valores de la propiedad en la nueva entidad no coinciden o no están incluidos en los valores almacenados en la clase, y estas propiedades son esenciales, entonces el VE de la nueva entidad con la clase puede ser cero. Pero si las propiedades no son esenciales, la opción podría ser disminuir el VE. En caso de que los valores de la propiedad en la entidad coincidan con los valores de la propiedad en la clase, se aumentará proporcionalmente el VE, según sea de esencial la propiedad en la clase. El IC, según sean las características de la aplicación, elaborará un fórmula para el cálculo del VE.

3. Calculados los VE, la técnica de equiparación *decide con qué marcos clase se equiparará la nueva entidad*. Si el VE es lo suficientemente alto y el marco es lo suficientemente específico, entonces el sistema no buscará otros marcos e instanciará la nueva entidad convirtiéndola en un marco instancia. Sin embargo, si el VE no es lo suficientemente alto, o el marco no es lo suficientemente específico,

entonces la técnica tendrá que identificar marcos relevantes. Esto puede realizarse de varias maneras, entre las que cabe destacar las siguientes: ascender a lo largo de la jerarquía hasta encontrar una equiparación relevante; comenzar en el marco raíz y seleccionar el marco con el mejor VE; buscar marcos relacionados utilizando las relaciones *Fraternal*, *Disjunto*, *No-Disjunto* y, o relaciones *ad-hoc*.

Esta técnica es especialmente útil en SBC que clasifican o en sistemas que se enfrentan a situaciones parecidas a otras que ocurrieron anteriormente. Por ejemplo, la técnica de equiparación se podría utilizar en un sistema que clasifique bacterias concretas en una taxonomía de bacterias, o en un sistema médico ante el cuadro de un paciente concreto, o en sistemas industriales ante fallos particulares, etc.

4.5.2 Herencia de propiedades

Esta técnica permite compartir valores y definiciones de propiedades entre marcos de una BC usando las relaciones *instancia* y *subclase-de*. Se puede diferenciar entre **herencia simple**, que se aplica a BC en forma de árbol, **herencia múltiple**, que se aplica a BC en forma de grafo.

- *Herencia simple.* Se aplica herencia simple a una BC formalizada en marcos cuando sólo existe un único camino que une el marco instancia con el nodo raíz de la jerarquía; es decir, cuando cada marco instancia es instancia de un único marco clase y cuando cada marco clase es, a su vez, especialización de una única clase. El algoritmo que realiza la inferencia para encontrar los valores de una cierta propiedad de un marco instancia es el siguiente:

  1. Se busca la propiedad en el marco instancia. Si se encuentra, se devuelven sus valores. En caso contrario, se accede al marco clase padre utilizando la relación *instancia*.
  2. Se busca la propiedad en el marco clase. Si se encuentra la propiedad, entonces se devuelven sus valores y el algoritmo finaliza. En otro caso, se utiliza la relación *subclase-de* para acceder al marco clase padre. Este paso se repite mientras el padre no sea el marco raíz del árbol.
  3. Se busca la propiedad en el marco raíz. Si se encuentra, se devuelven sus valores. Si no, la técnica debe responder que con los conocimientos almacenados en la BC es imposible proporcionar una respuesta.

  Si en el camino que une el marco instancia con el marco raíz se encuentran propiedades que presentan excepciones, la técnica de herencia simple proporcionará el valor de la propiedad situada en el marco clase más cercano a la instancia, al ser el orden en el que se recorre el árbol un orden total.

  Si se solicita el valor de una propiedad de instancia que ya toma valor en el marco instanca, la técnica de herencia simple devuelve el valor local almacenado en él.

  Si se solicita el valor de una propiedad de instancia que no tiene valores asignados en el marco instancia, entonces la técnica, utilizando las relaciones *instancia* y

subclase-de, buscará la propiedad en los marcos clase con los que la instancia se relaciona. La ejecución de un procedimiento asociado a la faceta Si Necesito, o la herencia de un valor por omisión, permitirán, en última instancia, responder a la pregunta. Nótese que en caso de estar definidas ambas facetas, debe existir una estrategia de control que ayude al SBM a decidir si ejecuta un procedimiento o si toma el valor por omisión. Si se solicita el valor de una propiedad de clase accesible desde un marco instancia, la técnica de herencia simple buscará la propiedad en el árbol y devolverá su valor.

- Se aplica *herencia múltiple* a una BC formalizada en marcos si existen varios caminos que unen los marcos instancia con el nodo raíz de la jerarquía, bien porque un marco instancia es instancia de varios marcos clase, o bien porque cada marco clase es especialización de una o varias clases. Dado que un hijo puede heredar propiedades de varios padres, la dificultad se encuentra en determinar el orden en el cual las clases se deben explorar. Si los padres tienen propiedades con distinto nombre nunca habrá conflicto. Sin embargo, si los padres tienen propiedades con el mismo nombre, el valor de la propiedad que hereda el marco hijo dependerá de la técnica de búsqueda que se utilice para recorrer el grafo. Por consiguiente, existen diferentes algoritmos que recorren de forma diferente el grafo, y el IC deberá elegir uno de ellos:

  - *Búsqueda en profundidad.* Esta técnica explora en profundidad todos los posibles caminos que van desde el marco instancia al marco raíz del SBM. Algunos de los criterios que, de forma combinada, pueden usarse para establecer el orden en el que se recorrerán cada úno de sus marcos clase son:

    1. Recorrer el grafo de *izquierda a derecha.*
    2. Usar el criterio de *exhaustividad* para evitar la búsqueda reiterada de propiedades en marcos clase ya analizados, es decir, solamente se buscará la propiedad en cada marco una única vez.
    3. El problema que presentan las técnicas de herencia múltiple, que recorren el grafo en profundidad de forma exhaustiva, y de izquierda a derecha, es que, para una propiedad que presenta excepciones, las instancias pueden heredar los valores de clases generales en vez de heredar los valores de clases más específicas. Esto ocurre siempre que el marco que presenta la excepción está situado en la parte derecha de la jerarquía. Con el fin de paliar este error, el criterio de *especificidad* impone la restricción de que sólo se puede buscar la propiedad en una clase si previamente se ha buscado en todas sus subclases.

  Un algoritmo que cumple los tres requisitos anteriores es el *procedimiento de ordenación topológica.* Dada una instancia, este procedimiento utiliza la topología del grafo para formar su lista de criterios de preferencias. Esta lista almacena el orden en el que se recorrerán todos los marcos clase accesibles desde un marco instancia concreto.

- *Búsqueda en amplitud: Longitud del camino.* La técnica de longitud del camino recorre el grafo por niveles que están a igual distancia del marco instancia. Dado un marco instancia, se mirará si la propiedad se encuentra en alguno de los marcos clase con los que está unido. Si se encuentra, entonces se devuelve su valor. En caso contrario, utilizando las relaciones *subclase-de*, comienza a buscar la propiedad en todas las superclases de dichas clases. En otras palabras, primero se busca la propiedad en los padres, es decir, en aquellos marcos clase que están a distancia uno del marco instancia. Si no la encuentra, entonces la técnica busca en los abuelos, es decir, en aquellos marcos que están a distancia dos, y así sucesivamente. El proceso termina al encontrar la propiedad o al alcanzar el nodo raíz sin encontrarla. Este algoritmo es eficiente y es adecuado en sistemas de herencia que razonan sobre grafos sin propiedades repetidas. Si el grafo contiene propiedades con excepciones aparecerán dos problemas: razonar en presencia de ambigüedades y de conocimientos no especializados.

  Un problema adicional que presenta la técnica de la longitud del camino es que la longitud de un camino puede no corresponder con el nivel de generalidad de una clase, si hay clases generales que no se han especializado con el mismo nivel de detalle que sus clases hermanas. El IC debe evitar en lo posible construir grafos que presenten diferente granularidad en la especialización de las subclases.

- *La distancia "inferencial".* Para resolver algunos de los problemas anteriores, Touretzky [Touretzky, 1986] propuso el concepto de distancia "inferencial". La distancia "inferencial" razona correctamente en BC redundantes, pero, aunque detecta situaciones ambiguas, no las resuelve. La distancia "inferencial" se puede definir como: "la condición necesaria y suficiente para que la clase_1 esté más cercana a la clase_2 que a la clase_3 es que la clase_1 tenga un camino de inferencia a través de la clase_2 hacia la clase_3; en otras palabras, si la clase_2 está entre la clase_1 y la clase_3". Obsérvese que en el concepto de distancia "inferencial" aparecen marcos conectados por enlaces de tipo *subclase-de*. Por este motivo, al no permitir comparar marcos no conectados, el orden introducido por la distancia "inferencial" es un orden parcial y, por consiguiente, se pueden heredar valores contradictorios definidos en ramas que no están conectadas.

4.5.3 Valores activos

Además de la estructura declarativa de los marcos, también existe un aspecto dinámico o procedimental de los mismos. En muchos sistemas, estos procedimientos son el mecanismo principal para realizar razonamientos que no están cubiertos por las otras dos técnicas explicadas anteriormente. Con los nombres de *demonios*, *valores activos* o *disparadores*, se denomina a los procedimientos que recuperan, almacenan, y borran información en los SBM. Estas procedimientos, como se explicó anteriormente en este capítulo, se definen en las facetas *Si Necesito*, *Si Añado*, *Si Modifico*

y *Si Borro* de las propiedades de instancia de los marcos clase. La ejecución de estos procedimientos presenta las siguientes características:

- Los procedimientos se definen en el marco clase y permanecen latentes, sin hacer nada, a menos que, por algún motivo, se solicite su ejecución desde un marco instancia. Se puede solicitar la ejecución de un procedimiento para: recuperar, almacenar, o borrar valores de propiedades en marcos instancia; mantener la integridad semántica de la BC al impedir que se introduzcan valores en la BC que no satisfacen unas determinadas restricciones; garantizar que cambios en los valores de una propiedad se reflejan correcta y automáticamente en los valores de otras propiedades; calcular dinámicamente valores de propiedades que se requieren y que se obtienen a partir de los valores de otras propiedades almacenadas en la BC; y, finalmente, manejar errores.

- Los procedimientos se despiertan al recibir una solicitud de ejecución. Cuando un marco instancia solicita la ejecución de un valor activo, el procedimiento asociado se ejecuta con los valores almacenados en las propiedades del marco instancia.

- Los procedimientos pueden simular encadenamiento hacia adelante, y así tener demonios dirigidos por los eventos, y encadenamiento hacia atrás, y tener demonios dirigidos por las metas.

  - Los demonios dirigidos por los eventos ejecutan procedimientos antes de almacenar o borrar valores en las propiedades de un marco instancia, y están asociados a las facetas *Si Añado*, *Si Modifico* y *Si Borro*.
  - Los demonios dirigidos por las metas están asociados a la faceta *Si Necesito*, y deducen valores de propiedades a partir de valores almacenados en otras propiedades, utilizando un eficiente pero complejo mecanismo de encadenamiento hacia atrás.

- El control va pasando de unas propiedades a otras a medida que se van ejecutando los procedimientos. En cada instante, el control lo tiene la propiedad del procedimiento que se está ejecutando. Una vez ejecutado un procedimiento, el control vuelve al procedimiento que lo llamó, procedimiento que sigue ejecutándose en el mismo lugar en el que se detuvo.

4.6 Resumen

Dado que son múltiples los formalismos que permiten representar los conocimientos de un dominio, el objetivo de este libro no es realizar un estudio completo y exhaustivo de todos ellos. En la parte II del presente libro se han incluido los formalismos más relevantes explicando las nociones más importantes de cada uno de ellos. En concreto, en este capítulo 4 se han explicados los formalismos de redes semánticas y marcos. Finalmente, en este apartado se pretende resumir de una manera práctica

las ventajas e inconvenientes de los dos formalismos descritos en este capítulo: las redes semánticas y los marcos. En líneas generales, la mayoría de los SBC suelen representar los conocimientos del dominio utilizando marcos frente a redes semánticas debido a que:

- Los marcos permiten construir taxonomías de conceptos por especialización de conceptos generales en conceptos más específicos, lo que posibilita la herencia de propiedades. Con la herencia de propiedades se comparten propiedades entre marcos y se evita la presencia de conocimientos redundantes en la BC. Aunque esta característica es común a las redes semánticas, la diferencia se encuentra en que en los marcos las propiedades están recogidas dentro del marco, mientras que en las redes semánticas las propiedades no están agrupadas, es decir, se encuentran dispersas por toda la red uniendo nodos que representan conceptos con otros nodos que representan los valores que toma la propiedad.

- En las BC formalizadas en marcos las propiedades se pueden definir de forma declarativa y procedimental; sin embargo, en las redes semánticas sólo se pueden definir de forma declarativa. Además, la cantidad de información asociada a las propiedades en un marco supera con creces a la de las propiedades en una red semántica. En los marcos se puede introducir información sobre el tipo de datos al que pertenecen los valores, número de valores mínimo y máximo con los que se rellena cada propiedad, valores por omisión, excepciones, y procedimientos y reglas que permiten inferir los valores de dichas propiedades, mientras que en las redes semánticas no se puede.

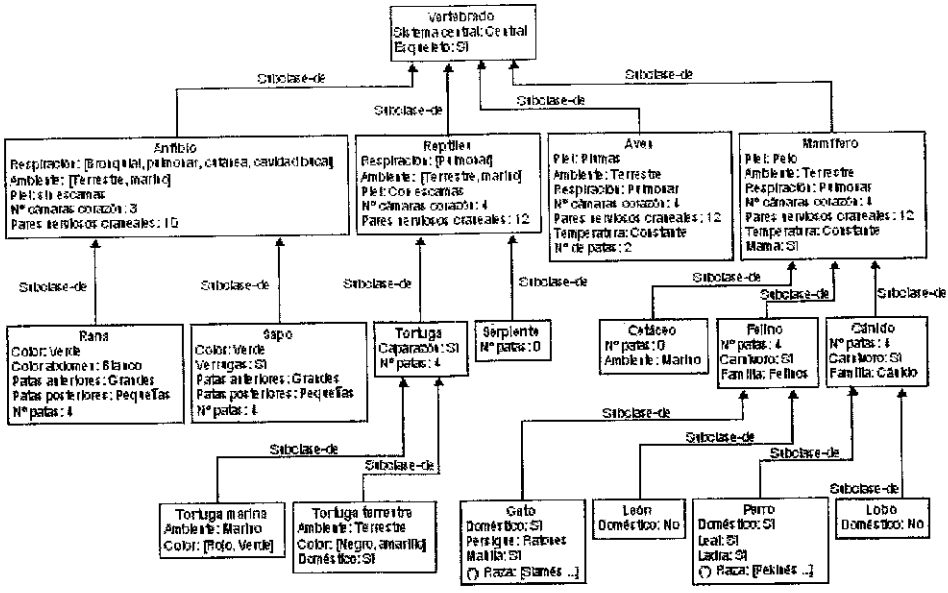
- Los SBM permiten el uso de valores por omisión y excepciones a dichos valores; sin embargo, en las redes semánticas no se pueden representar valores por omisión, aunque sí excepciones. La propia estructura interna de los marcos permite mantener internamente las restricciones de integridad semántica que existen entre los elementos de un dominio, mientras que en las redes semánticas, no.

- Uno de los principales inconvenientes que presentan las redes semánticas es que a medida que la BC del sistema aumenta, la comprensión de la red se hace cada vez más difícil. No ocurre lo mismo en los SBM, los cuales facilitan el diseño y mantenimiento de la BC.

4.7 Ejercicios resueltos

4.1. Construir una jerarquía de marcos sobre animales vertebrados. La jerarquía debe estar formada al menos por diez marcos clase y, en cada uno, debe haber al menos dos propiedades de clase.

Solución:



4.8 Ejercicios propuestos

4.1. Suponer que un usuario que desea alquilar un coche por Internet rellenando el formulario que se presenta en la Figura 4.20. Los valores que toman los campos son los siguientes:

- Clase de coche: compacto, económico y lujo.
- Tipo de coche: 2 puertas, 4 puertas y tracción a las cuatro ruedas.
- Caja de cambios: manual, automática.
- Aire acondicionado: sí, no.
- Distancia: kilómetros o millas.

Suponer que el usuario desea que el punto de recogida y de entrega sea Madrid, que la clase de coche sea un compacto, de cuatro puertas y que la caja de cambios sea manual. Las salidas que obtiene del sistema son las mostradas en la Figura 4.21.

Se pide:

- Construir la red semántica que formaliza el dominio de alquiler de coches. Debe formalizarse: el punto de entrega y de recogida, la hora de entrega y de recogida, la fecha de entrega y recogida, la clase de coche, si tiene aire acondicionado, el tipo de caja de cambio, la tarifa diaria y el precio total estimado.

- Formalizar con redes semánticas una de las 5 respuestas que da el sistema y relacionar dichas respuestas con la formalización previamente realizada en el apartado anterior.

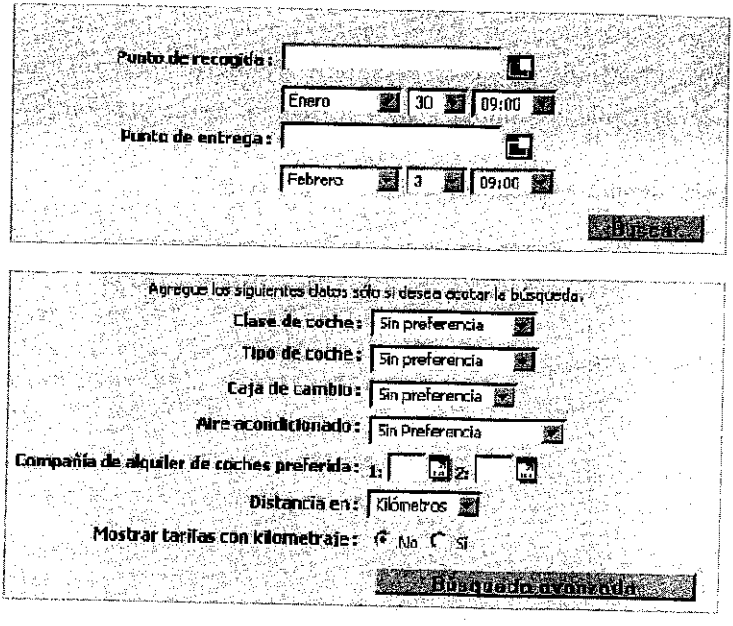


Figura 4.20: Formulario a rellenar por el cliente.

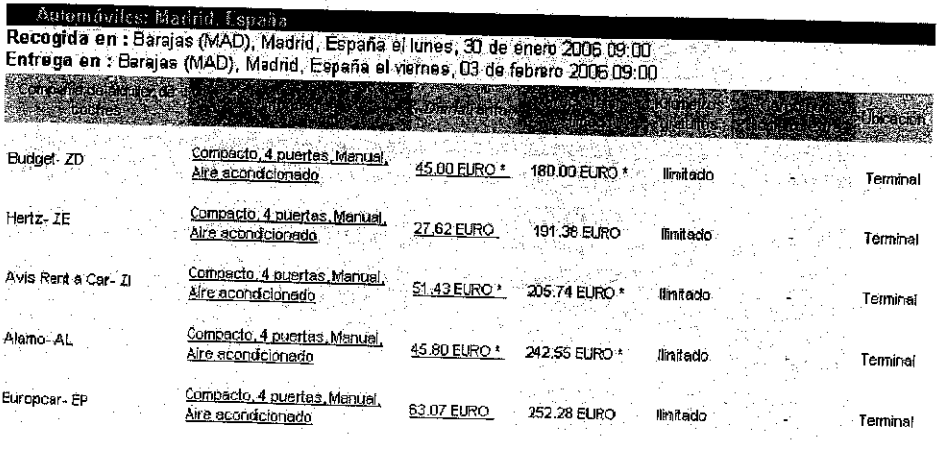


Figura 4.21: Coches disponibles.

4.2. Suponer que un usuario desea reservar un hotel por Internet rellenando el formulario de la Figura 4.22. Los valores que toman los campos son los siguientes:

- Provincia: son todas las provincias españolas.
- Población: se refiere a las poblaciones de las distintas provincias.
- Localización, que toma los valores: capital, capital y radio<25km.
- Ocupación, que toma los valores: 1 adulto, 2 adultos, 1 adulto y un niño, etc.
- Régimen: alojamiento, alojamiento y desayuno, media pensión y pensión completa.
- Categoría: de una a cinco estrellas.

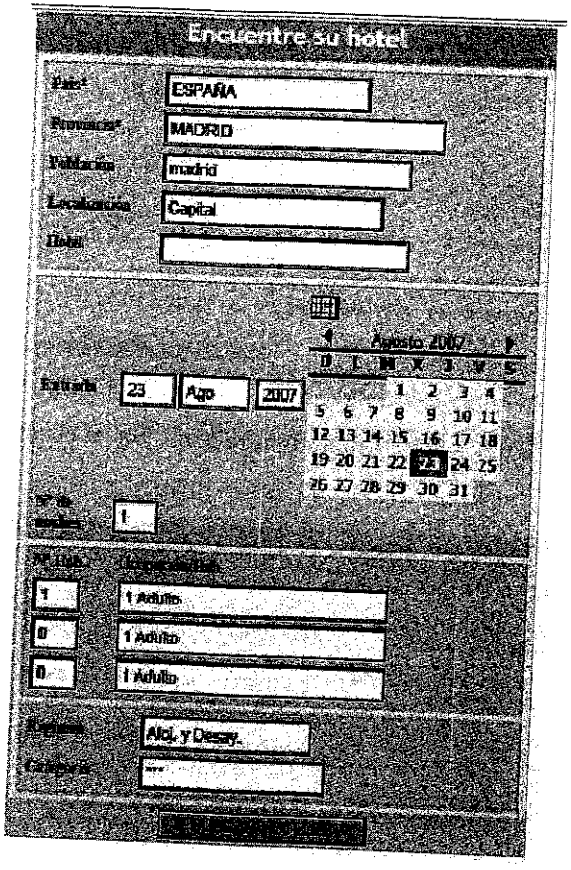


Figura 4.22: Formulario a rellenar por el cliente.

El usuario desea una habitación individual para un adulto en un hotel de al menos tres estrellas en Madrid capital para el día 23 de agosto de 2007 en régimen de alojamiento y desayuno. Las salidas que obtiene son las mostradas en la Figura 4.23.

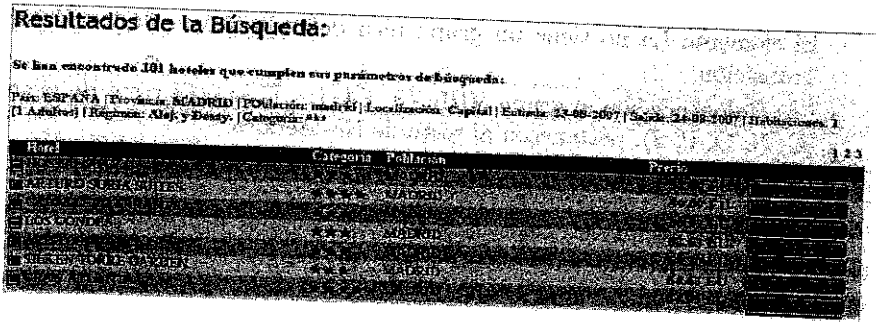


Figura 4.23: Hoteles disponibles.

Se pide:

- Construir la red semántica para este dominio.
- Formalizar con redes semánticas una de las respuestas queda el sistema y relacionar dichas respuestas con la formalización previamente realizada en el apartado anterior.

4.3. Construir una BC formalizada en marcos que represente el Sistema Periódico (SP). Tras la etapa de conceptualización se sabe que:

- Los elementos del SP se clasifican en los siguientes grupos: Metales, No-Metales, Semi-Metales y los Gases Nobles, a excepción del Hidrógeno que no tiene un grupo bien definido. Además se sabe que, el Hidrógeno, los Metales, los No-Metales y los Semi-Metales reaccionan con otros elementos, mientras que los otros no.
- Los elementos Metales se pueden clasificar en dos grupos: Metales de No Transición, que a su vez engloba a los Alcalinos y Alcalinotérreos; y los Metales de Transición, que comprenden la primera y segunda serie de transición y a los Lantánidos y Actínidos, también llamados primera y segunda serie de transición interna.
- La clasificación de los No-Metales no está tan bien definida como en la clase de los Metales. Sin embargo, existe un grupo bien diferenciado que es el de los Halógenos.
- Los elementos del SP sólo pertenecen a un único grupo. A saber:
  - Los Alcalinos son el grupo Ia (Li, Na, K, Rb, Cs, Fr).
  - Los Alcalino-Térreos son el grupo IIa (Be, Mg, Ca, Sr, Ba, Ra).
  - La primera serie de transición interna son los grupos de los Lantánidos (Ce, Pr, Nd, Pm, Sm, Eu, Gd, Tb, Dy, Ho, Er, Tm, Yb, Lu).
  - Los Actínidos (Th, Pa, U, Np, Am, Cm, Br, Cf, Es, Fm, Md, No, Lw).
  - Forman la primera serie de transición los elementos del cuarto período (Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn), y la segunda serie de transición los del quinto período (Y, Zr, Nb, Mo, Tc, Ru, Rh, Rd, Ag, Cd).

- El elemento La no tiene un grupo bien definido dentro de los Metales de Transición.
- Los Halógenos son el grupo VIIb: (F, CI, Br, I, At). El resto de elementos (C, N, O, P, S), pertenecen al resto de No-Metales.
- El grupo VIII (He, Ne, Ar, Kr, Xe, Rn) son los Gases Nobles.

4.4. Se desea construir un sistema basado en marcos que se utilizará en la enseñanza del aparato digestivo a niños de 10 años. Se sabe que el aparato digestivo lo forman los siguientes órganos: boca, esófago, estómago, intestino delgado e intestino grueso. La boca precede al esófago, y éste al estómago. A continuación, y en el siguiente orden, se encuentra el intestino delgado y el intestino grueso.

Las funciones de dichos órganos son: la boca para masticar y deglutir, el esófago para transferir alimentos; el estómago para mezclar y comenzar la digestión; el intestino delgado para absorverlos; y el intestino grueso para absorver y desecharlos.

Los órganos del aparato digestivo están unidos entre sí, pero al mismo tiempo están independizados por medios esfínteres. Así, el cardias une el esófago con el estómago e impide que el alimento pase del estómago al esófago. El píloro une el estómago y el duodeno. La válvula ileocecal une el intestino delgado con el grueso. El experto posee más conocimiento sobre estos esfínteres. Dichas propiedades se describen en otros documentos, y deberá tenerse en cuenta para incluirlas posteriormente en el sistema final.

El esófago es un tubo muscular de 30-40 cm. de longitud, cuyas contracciones empujan el bolo alimenticio desde la boca al estómago. Su mucosa no está preparada para soportar la presencia de ácido.

El estómago es un saco muscular dividido en zonas: el fundus (que está unido al esófago por el ~~píloro~~ cardias) y el cuerpo (parte intermedia) para almacenar alimentos de gran tamaño; y el antro (unido al intestino delgado por el ~~cardias~~ píloro) para mezclar y triturar los alimentos.

El intestino delgado está compuesto por el duodeno, yeyuno e ileón. El duodeno es un tubo de 25 cm. de longitud que conecta el estómago al resto del intestino delgado. El yeyuno sigue al duodeno, tiene dos metros de longitud, y comunica el duodeno con el ileón el cual mide 4 metros.

El intestino grueso comienza con el colon ascendente. Este precede al colon transverso, el cual va seguido del colon descendente, del colon sigmoideo, y del recto. El colon ascendente comunica el intestino delgado con el resto del intestino grueso.

En la digestión intervienen numerosos jugos, ácidos y enzimas. Además, se sabe que los jugos están compuestos por ácidos y enzimas. La enzima Amilasa está en la boca; el jugo gástrico, formado por ácido clorhídrico y la enzima Pepsina en el estómago; y, los jugos pancreáticos y biliar en el duodeno. El experto posee más conocimientos sobre ellos, y deberá tenerse en cuenta para incluir dicho conocimiento posteriormente en el sistema final.

Se pide:

- Construir el diagrama de la jerarquía de macros y explicar la semántica de las relaciones empleadas. Indicar en dicho diagrama cuáles son las propiedades de cada marco clase.
- Explicar detalladamente los marcos: intestino delgado, duodeno, yeyuno e ileón.
- Si a este sistema se le preguntaran las siguientes cuestiones, ¿cómo razonaría el sistema y qué respondería?:
  - ¿Cuáles son los componentes del estómago?
  - ¿Cuál es el esfinter superior del colon ascendente?
  - ¿Precede el estómago al esófago?
  - ¿Cuál es el órgano que está en la parte superior del cardias?
  - ¿Cuál es el órgano que está en la parte inferior del píloro?
  - ¿De qué órgano forma parte el colon transverso?
  - ¿Qué enzima produce la boca?
  - ¿Qué elementos forman el jugo pancreático?
  - ¿Es el ácido clorhídrico una enzima?
  - ¿Cuánto mide el intestino delgado?

Referencias

FILLMORE, C. J.: «The Case for Case». En: E. Bach. y R. Harms (Eds.), *Universals in Linguistics Theory*, pp. 1–90. Holt, Rinehart and Winston Publishing Company, 1968.

GÓMEZ, ASUNCIÓN; JURISTO, NATALIA; MONTES, CÉSAR y PAZOS, JUAN.: *Ingeniería del Conocimiento*. Editorial Centro de Estudios Ramón Areces, S.A., 1997.

HENDRIX, G. G.: «Expanding the utility of semantic networks through partitioning». En: *Proceedigns of the IJCAI-75*, pp. 115–121, 1975.

HENDRIX, G. G.: «Encoding Knowledge in Partitioned Networks». En: N. V. Findle (Ed.), *Associative Networks - The Representation and Use of Knowledge in Computers*, pp. 51–92. Academic Press, 1979.

MINSKY, M.: «A Framework for Representing Knowledge». En: R. J. Brachman y H. J. Levesque (Eds.), *Readings in Knowledge Representation*, pp. 245–262. Kaufmann, Los Altos, CA, 1985.

QUILLIAN, M. R.: «Semantic Memory». En: M. Minsky (Ed.), *Semantic Information Processing*, pp. 27–70. The MIT Press, 1968.

TOURETZKY, D.S.: *The Mathematics of Inheritance SystemsThe Mathematics of Inheritance Systems*. Morgan Kaufmann Publishers, Inc, Los Altos (California), EE.UU, 1986.