Capítulo 1
Sistemas basados en reglas

1.1. OBJETIVOS

Este capítulo introduce los conceptos básicos de un sistema basado en reglas y un entorno de programación para desarrollar tal sistema: CLIPS. Se hace énfasis en establecer claras diferencias entre la programación procedimental y la programación de los sistemas basados en reglas. Se estudia en detalle el modelo propuesto para la representación del conocimiento y la estructura que adopta el sistema. Al estudiar el modelo se hace énfasis en que está inspirado en la psicología. Al estudiar su estructura se pretende resaltar las diferencias que existen con respecto a la programación procedimental, sobre todo en la separación entre la base de hechos –representación de la información– y la base de conocimiento –representación del conocimiento en forma de reglas–, cosa que en los lenguajes procedimentales no ocurre al estar mezclado en el mismo código del programa las acciones, las estructuras de datos y los esquemas algorítmicos.

1.2. INTRODUCCIÓN

El ser humano se dice inteligente, entre otras cosas, por su capacidad de resolver problemas de una manera en que otros animales no son capaces de resolverlos. Cabe preguntarse cuáles son las características que posee el cerebro humano que le llevan a realizar estas funciones. Responder a esta pregunta ha sido, y sigue siendo, una cuestión debatida y estudiada por la humanidad a lo largo de los siglos. Más recientemente filósofos, psicólogos, neurofisiólogos, matemáticos e informáticos, entre otros especialistas, tratan de desvelar los misterios de la mente humana.

La psicología especula proponiendo que el ser humano ante un problema es capaz de crear en su mente unas representaciones mentales de la realidad que giran alrededor del problema planteado. Al mismo tiempo, tiene la capacidad de almacenar en dicha mente una serie de procesos mentales, de modo que, operando con las representaciones mentales, obtiene nuevas representaciones que suponen

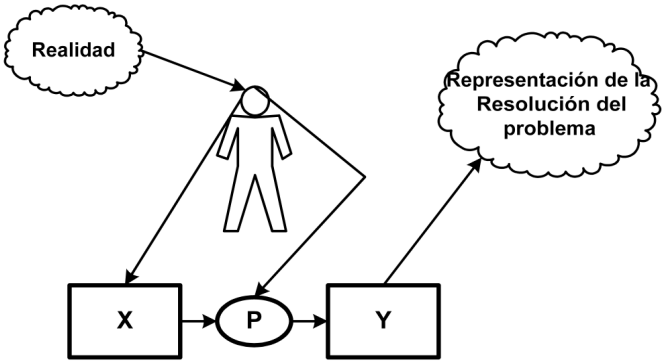

Figura 1.1: Representaciones y procesos mentales

la resolución del problema o la indicación de un conjunto de órdenes que lleva a la resolución de dicho problema [Tha96, p. 10].

Tomemos un ejemplo: si una persona se propone resolver el problema de freír un huevo obtendrá en su mente una representación de todos los materiales y utensilios necesarios para su resolución –representaciones mentales– y buscará en su mente alguna receta –procesos mentales– que le indique cómo operar con esas representaciones mentales para, finalmente, generar un conjunto de acciones que le lleva a la resolución del problema.

La figura 1.1 muestra un esquema de lo que hemos comentado. El ser humano, ante un problema obtiene una representación de la realidad que gira alrededor de ese problema (**X** de la figura). Busca un proceso mental (tarea) que pueda dar solución al problema, manipulando esas representaciones mentales (**P** del esquema), para obtener así la representación de la solución al problema (**Y** del esquema). Esa salida al proceso puede traducirse en un conjunto de acciones que den solución al problema.

En informática lo que se busca es obtener unas representaciones en forma computable. Es decir, buscamos una forma de representar la realidad –**X**– mediante estructuras de símbolos –estructuras de datos– y los procesos mentales –**P**– mediante procedimientos –algoritmos– de tal manera que la suma de estructuras de datos y algoritmos den lugar a un programa de ordenador que resuelva el problema.[1]

La dificultad es encontrar formas computables de representar tanto las representaciones mentales como los procesos mentales. Esto se logra a través de los lenguajes formales. Llegados a ese punto cabe hacerse una pregunta ¿hay diferen-

[1]Es clásico el libro de Wirth *Algoritmos + Estructuras de Datos = Programas* [Wir80].

tes modelos o paradigmas de representación? ¿Cuáles son los aspectos comunes y diferenciales entre ellos? Este libro trata de responder a esta cuestión mostrando un modelo –sistemas basados en reglas (SBR)– representado en un lenguaje formal específico –CLIPS–. Pero antes de entrar en detalle trataremos de establecer los puntos comunes y las principales diferencias con respecto a otros modelos de representación.

En cuanto a los aspectos comunes hemos de atender a las siguientes cuestiones con respecto a los SBR:

1. ¿Cuál es la estructura general del modelo? ¿Qué elementos posee?

2. ¿De qué estructuras de símbolos dispone para representar la realidad –**X**– e –**Y**–?

3. ¿De qué estructuras de símbolos dispone para representar los procesos mentales –**P**–?, es decir, ¿de qué estructuras dispone para manipular la –**X**– y obtener la –**Y**–? ¿Cómo de compleja es la sintaxis bajo la cual se disponen estas estructuras de símbolos?

4. ¿De qué mecanismos dispone para el procesamiento de los símbolos? ¿Qué mecanismos de inferencia existen en los SBR?

5. ¿De qué forma pueden controlarse la ejecución de las acciones y procedimientos que constituyen la representación de los procesos mentales –**P**– para que se ejecuten en el orden adecuado y preciso para la resolución del problema?

Los próximos capítulos trataran de responder a estas y otras cuestiones. Pero antes conviene responder a la primera pregunta, y para ello vamos a mostrar de forma resumida el modelo de representación procedimental que se observa en lenguajes de programación como C o Basic.

1.3. MODELO DE REPRESENTACIÓN PROCEDIMENTAL

La programación procedimental se basa en la representación mediante estructuras de datos (estructuras de símbolos) de una realidad y en la manipulación de esas estructuras, bien para alterarlas o bien para crear otras muevas mediante un conjunto de procedimientos. Estos procedimientos se componen de un conjunto de acciones elementales organizadas en esquemas algorítmicos básicos como son la secuencia, la iteración y la condición.

La figura 1.2 muestra los elementos básicos de la programación procedimental. Un programa consta de dos elementos básicos, las estructuras de datos y las

acciones que sobre ellos pueden realizarse. Estas acciones se disponen secuencialmente y el procesador las irá ejecutando en el orden en que han sido establecidas. En el siguiente ejemplo se ha elegido una estructura de datos elemental como es la variable. La primera línea de programa indica que se realice la acción de asignación de 20 sobre la variable `y`; la segunda línea indica que se le asigne 10 a la variable `x`; la tercera acción conlleva dos subacciones. Primero se lleva a cabo la suma de la variable `x` e `y` y a continuación el resultado de esta suma se asigna a la variable `z`; finalmente, la última acción presenta por pantalla el contenido de la variable `z` resultado de la suma.

```
main
 y=20
 x=10
 z= x + y
 print z
end
```

El procesador siempre ejecuta las acciones secuencialmente, salvo que surja un esquema condicional o iterativo. El siguiente ejemplo es ilustrativo de un esquema condicional.

```
main
 y=20
 x=10
 if x>y
    z= x + y
 else
    z = x-y
 end if
 print z
end
```

Las acciones pueden agruparse en funciones o procedimientos e invocarse desde otras partes del programa:

```
main
 y=20
 x=10
 print suma-resta(x,y)
end

function suma-resta(x as integer,y as integer) as integer
        if x>y
            z= x + y
        else
            z = x-y
        end if
return z end function
```

Este ejemplo muestra cómo se ha declarado el procedimiento (en este caso se trata de una función ya que devuelve un valor) `suma-resta` con dos argumentos enteros que devuelve otro entero. Vemos cómo este procedimiento es invocado desde el programa principal (convirtiéndose en sí mismo en una nueva acción) siguiendo el orden secuencial.

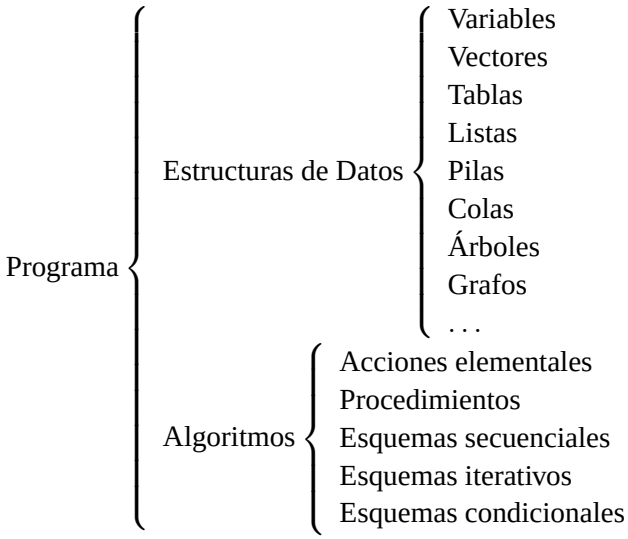

Figura 1.2: La programación procedimental

De los ejemplos anteriores podemos extraer varias conclusiones:

1. Un **programa de ordenador** está formado por un conjunto de símbolos que representan estructuras de datos (`x`, `y`), acciones primitivas (`print`, `=`), operaciones (`+`), procedimientos (`suma-resta()`) que se organizan a su vez en una estructura de acuerdo a una sintaxis.

2. El lenguaje cuenta con un conjunto de símbolos reservados que constituyen las acciones primitivas, también llamadas primitivas del lenguaje.

3. Con estas primitivas pueden formarse expresiones admitidas del lenguaje si están de acuerdo con la sintaxis establecida en el mismo.

4. El programa se representa en una estructura de símbolos.

5. El orden en que se colocan las acciones o invocación de procedimientos será el que marque el orden de procesamiento de las acciones.

6. Sobre un programa de ordenador se representa conocimiento para la resolución de un problema.[2]

¿Existen otras formas de representar conocimiento diferentes a ésta? ¿Pensamos y razonamos de esta manera procedimental? El conocimiento que permite manipular las estructuras de datos está organizado en secuencias de acciones elementales, procedimientos y funciones (que acogen en su interior otras secuencias de acciones) convertidas en acciones primitivas. El flujo de ejecución del programa está dirigido, fundamentalmente, por la secuencia. Si quisiéramos alterar el orden de ejecución de las acciones, es decir, alterar el orden de razonamiento, tendríamos que detener el programa, modificar físicamente la ubicación de las acciones primitivas o la invocación de las funciones. Si quisiéramos ampliar el programa deberíamos de añadir más líneas de código al mismo, colocándolo en el lugar apropiado y reiniciar su ejecución.

En este tipo de programación lo que va a ocurrir es totalmente previsible, una acción sigue a la otra, salvo que aparezca un esquema condicional, iterativo o la invocación de una función o un procedimiento. Además el mecanismo con el que se generan nuevas estructuras de datos a partir de las ya existentes reside en el propio programa, en el propio código que representa el conocimiento.

Para muchos tipos de problemas este tipo de representación es muy eficiente, sobre todo para la resolución de problemas de cálculo matemático, problemas de gestión de bases de datos, etc.

Pensemos en el problema de realizar diagnóstico médico de enfermedades. El médico va adquiriendo conocimiento de forma incremental, su toma de decisiones está condicionada por muchos factores. ¿Existe una forma alternativa para la representación de éste conocimiento? ¿Cual sería el modelo de representación?

1.4. MODELO DE SISTEMAS BASADOS EN REGLAS (SBR)

Frente a la representación procedimental del conocimiento para la resolución de problemas se han propuesto otras alternativas tratando de imitar el comportamiento humano.[3] Como se comentó anteriormente ha de responderse a varias

[2] Suele utilizarse el adjetivo procedimental o algorítmico para referirse a este tipo de conocimiento.

[3] El uso de las reglas de producción surgió dentro de la teoría de autómatas y lenguajes formales. También en los problemas de búsqueda que aparecen tradicionalmente en inteligencia artificial puede utilizarse un conjunto de reglas que generen los estados de un espacio de búsqueda. Dentro de este contexto, Newell y Simon [NS71] utilizaron las reglas como modelo psicológico: el comportamiento inteligente puede describirse mediante un conjunto de reglas que indiquen en cada momento cómo actuar, en función de la información disponible. Un estudio más detallado puede encontrarse en [MDBD95, pp. 231–261].

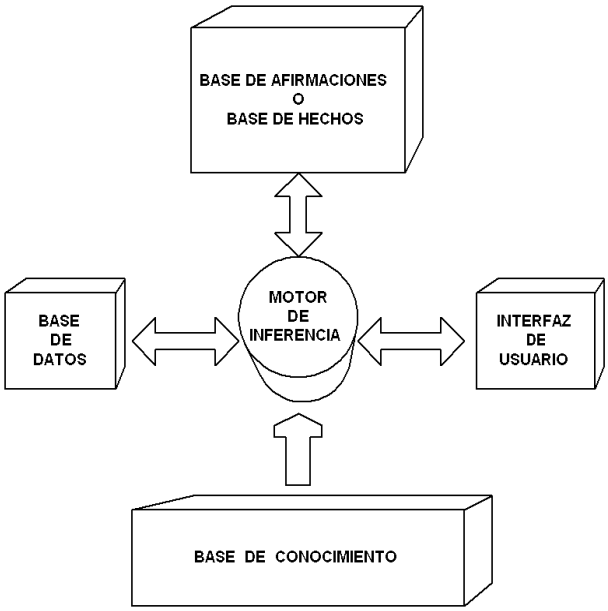

Figura 1.3: Componentes de un sistema basado en reglas

cuestiones: ¿Cuál es la estructura básica del modelo? Dentro de esta estructura ¿dónde y cómo representar la realidad –**X** e **Y**– del esquema? ¿Dónde y cómo representar las estructuras de procesamiento? ¿Qué mecanismos básicos de inferencia se utilizan en esta representación? ¿Cómo se controla el razonamiento?

1.4.1. Estructura básica

Un sistema basado en reglas consta de cinco elementos básicos, según muestra la figura 1.3. Vamos a describir a continuación cada uno de ellos y más adelante completaremos este esquema tan simple introduciendo otros elementos.

1. **Motor de inferencia**: A veces se le denomina también intérprete de reglas. Es el elemento central del sistema, se encarga de coordinar la información procedente de todos los demás elementos y de enviar los resultados de la inferencia al lugar oportuno. Este elemento viene programado en las herramientas de programación como CLIPS. Funciona en base a ciclos. Como se verá en más detalle en el capítulo 6, en cada ciclo el motor de inferencia evalúa qué reglas de la base de conocimiento están activas en un momento determinado, las ordena y dispara la de más alta prioridad. Encierra en sí mismo los mecanismos de inferencia del sistema de forma que van a ser

independientes del conocimiento depositado en la base de conocimiento en forma de reglas.

2. **Base de conocimiento:** Contiene las reglas. La base de conocimiento es específica del dominio en que el sistema puede considerarse experto: medicina, espectrografía, geología, diseño de circuitos, etc.[4] Las reglas se depositan en la base de conocimiento sin tener en cuenta el orden en que han sido escritas de forma que el control del razonamiento se realizará mediante otros mecanismos.[5] Si hacemos una comparación con la programación procedimental podríamos decir que en la base de conocimiento se deposita en forma de reglas aquel conocimiento algorítmico.[6] El capítulo 6 estudia en detalle la estructura de las reglas y su capacidad inferencial.

3. **Base de afirmaciones:** También recibe el nombre de memoria de trabajo o base de hechos. Contiene los hechos iniciales, los que se extraen de la base de datos, o los que el usuario introduce como datos y todas las que el motor de inferencia genera a partir de las anteriores aplicando las reglas oportunas. También es frecuente depositar en la base de hechos ciertas afirmaciones (hechos) que permitan el control del razonamiento marcando cuál es el objetivo que se está procesando en el momento actual.[7] Si comparamos con la programación procedimental podríamos decir que en la base de hechos se deposita toda la información que en la programación procedimental depositaríamos en las estructuras de datos. Estudiaremos en los capítulos 3, 4 y 5 qué estructuras de datos dispone CLIPS para representar esta información sobre la realidad que rodea al problema.

4. **Interfaz de usuario**: Se encarga de pedir al usuario la información necesaria y de presentarle los resultados de la inferencia; además, le ofrece explicaciones de cómo y por qué está funcionando el sistema.

5. **Base de datos**: En casi todas las aplicaciones prácticas es necesario acceder a toda la información relativa al problema que se está tratando. Para que el comportamiento del sistema llegue a ser más eficaz conviene mantener

---

[4] Los primeros sistemas expertos desarrollados bajo estos formalismos demostraron que al vaciar la base de conocimientos y rellenarla con otra el sistema funcionaba correctamente.

[5] Recordemos que en la programación procedimental el orden en que se escriben las líneas de código marca el orden de ejecución de las órdenes reflejadas en dichas líneas.

[6] Es una apreciación un poco burda y que será explicada con más detalle en los próximos capítulos.

[7] Por ejemplo si se tratara de un robot que debe de desarrollar varias tareas podríamos tener un hecho como `(objetivo abrir-puerta)` que permitiese la activación de todas las reglas relacionadas con la tarea de abrir la puerta.

un sistema de información externo y adicional a la base de hechos y cuya información será solicitada cuando sea necesario. Se dispondrá de algunas reglas que, cuando sean interpretadas por el motor de inferencia, sus acciones provocarán el paso de información de la base de datos a la base de hechos y viceversa.

1.4.2. Relaciones entre los elementos del sistema

En la figura 1.3 es importante la dirección de las flechas. Las que conectan el motor de inferencia con la base de datos, con la base de afirmaciones y con el interfaz de usuario son dobles, lo cual indica que la información fluye en ambos sentidos. En cambio, hay una flecha que va de la base de conocimiento al motor de inferencia pero no a la inversa, lo cual indica que en los sistemas expertos convencionales éste no modifica aquélla. Si describiéramos un sistema más avanzado, con capacidad de aprendizaje, esta flecha debería ser bidireccional. Ésta es la estructura más general de un sistema basado en reglas. En algunas aplicaciones puede ser innecesaria la base de datos. En aplicaciones de control y monitorización de procesos puede haber además un módulo que introduzca en el sistema la información procedente de los sensores y que aplique las conclusiones mediante un conjunto de actuadores. En el caso poco frecuente de control no supervisado podría estar ausente el interfaz de usuario. Sin embargo, en la mayor parte de los sistemas expertos aplicados a problemas reales el interfaz de usuario desempeña un papel mucho más importante de lo que se pensó al principio, pues en muchas ocasiones la aceptación del programa por parte de los usuarios finales dependerá de la calidad del interfaz (que sea fácil de aprender, cómodo y flexible) tanto como de la calidad de los resultados.

Es importante señalar que, en principio, el motor de inferencia es un programa de ordenador independiente del dominio de aplicación del sistema. Por tanto, es posible sustituir una base de conocimientos por otra diferente con la única condición de que tenga la misma sintaxis que la anterior. Si al motor de inferencia unimos las facilidades para la construcción y depuración de la base de conocimientos, obtenemos una herramienta para la construcción de sistemas expertos.

1.5. ESTRUCTURA DE LA REGLA

Estudiada en la sección anterior la estructura del modelo utilizado para representar el conocimiento conviene introducir de forma breve cuál es la estructura de cada una de las reglas que se depositan en la base de conocimiento. Esto será motivo de un estudio exhaustivo en el capítulo 6, no obstante adelantamos unas nociones básicas.

Una regla se compone, básicamente, de dos partes: un antecedente y un consecuente. El antecedente de la regla, también denominada, parte izquierda de la regla, contiene un conjunto de cláusulas (en CLIPS se les denominará elementos condicionales), las cuales han de satisfacerse todas para que el motor de inferencia active la regla en un ciclo determinado.[8] En el siguiente ejemplo mostramos una regla con dos cláusulas en su antecedente:

Base de conocimiento

```
(defrule r1
   (objetivo descubrir-mortales)
   (hombre ?x)
=>
   (assert (mortal ?x))
)
```

Base de hechos

```
 (hombre Socrates)
 (hombre Platón)
 (objetivo descubrir-mortales)
```

Observemos que aunque explícitamente no se representa el conector lógico *and* entre los dos elementos condicionales, implícitamente sí existe. De forma que para que se activen *instancias* de esta regla deberán existir hechos en la base de hechos que satisfagan estos elementos condicionales. Observemos que el primer elemento condicional patrón `(objetivo descubrir-mortales)` encaja con el hecho de la base de hechos `(objetivo descubrir-mortales)`. El otro elemento condicional `(mortal ?x)` contiene lo que llamaremos un elemento condicional patrón con variables. Como se verá en más detalle en el capítulo 6, en cada ciclo de ejecución del programa, el motor de inferencia por cada hecho de la base de afirmaciones que satisfaga este elemento condicional patrón se activará una *instancia* de la regla. Cuando aparecen variables en los elementos condicionales el motor de inferencia buscará hechos en la base de afirmaciones que casen con el elemento condicional patrón asignándole los valores de los campos correspondientes a las variables. En este ejemplo sencillo, en el primer ciclo de ejecución se activarán dos *instancias* de la regla, disparándose sólo una de ellas. En el segundo ciclo, se activará una sola regla, disparándose dicha regla.

[8] En realidad lo que se activa no es la regla sino una *instancia* de la misma de forma que en un ciclo pueden activarse más de una *instancia* de una misma regla.

El consecuente o parte derecha de la regla contiene las acciones que se llevarán a cabo cuando se dispare la *instancia* de la regla. Unas acciones modificarán la base de afirmaciones, otras modificarán la base de datos y, finalmente, otras enviarán mensajes al interfaz de usuario.

Además del antecedente y del consecuente cada regla se caracterizará por un nombre, comentarios, prioridad, etc. que se estudiarán con más detalle, junto con una amplia sintaxis para el antecedente de las reglas, en el capítulo 6 y 7.

1.6. MECANISMOS DE INFERENCIA

¿Qué entendemos por inferencia? La Real Academia de la Lengua establece que inferir es Sacar una consecuencia o deducir algo de otra cosa. Supongamos un ejemplo elemental representado de varias formas:

1. Silogismo en lenguaje natural

   Todos los hombres son mortales

   Sócrates es un hombre

   $\Rightarrow$

   Sócrates es mortal

2. Lógica de predicados

   $\forall x$ tal que $hombre(x) \Rightarrow mortal(x)$

   $hombre(Socrates)$

   La ley *modus ponens* de la lógica provocaría la inferencia $\Rightarrow$

   $mortal(Socrates)$

3. Representación en CLIPS (Lenguaje para la representación del conocimiento basado en reglas de producción)

   - base de conocimiento

   ```
   (defrule r1
      (hombre ?x)
   =>
      (mortal ?x)
   )
   ```

   - Base de hechos

   ```
   (hombre Socrates)
   ```

En el momento en que se ponga en marcha el motor de inferencia y mediante el mecanismo de inferencia de comparación de patrones, el sistema afirmará de forma automática en la base de hechos

```
(mortal Socrates)
```

4. Representación del conocimiento mediante marcos

   En esta representación tendríamos una clase, la clase persona, que tendría una propiedad[9] denominada nombrePersona y otra denominada mortal con el valor por defecto de cierto. Si se crea una *instancia* de esa clase y se le asigna a la propiedad nombre Socrates el sistema, utilizando como mecanismo de inferencia la herencia de propiedades, asignará a la propiedad de la *instancia* mortal el valor de cierto.[10]

Un ser humano inteligente al que se le presenta este silogismo infiere de manera inmediata que Sócrates es mortal. Si observamos las diferentes formas en que ha sido representado este ejemplo simbólicamente podemos preguntarnos cuál es el mecanismo, en cada uno de los formalismos de representación, mediante el cual son manipulados los símbolos para, finalmente, llegar a inferir que Sócrates es mortal (representado simbólicamente de diferentes maneras en función del modelo de representación elegido –observemos que en CLIPS este hecho viene representado por esta estructura de símbolos `(mortal Socrates)`–).

1.6.1. Comparación de patrones

Uno de los mecanismos básicos de inferencia y que se verá en detalle en el capítulo 6 es el de comparación de patrones.

La figura 1.4 nos ayudará a comprender este mecanismo de inferencia. Se observa cómo en la base de hechos se deposita el conocimiento estático de una parcela de la realidad sobre Platón, Aristóteles, etc. representada por estructuras de símbolos. En la base de conocimiento se deposita el conocimiento sobre la mortalidad de los hombres; en el antecedente de la regla se ubica el patrón `(hombre ?x)` y en el consecuente la acción `(assert (mortal ?x))`. Cuando el programa se pone en marcha el motor de inferencia comienza su primer ciclo; al evaluar la regla compara el patrón `(hombre ?x)` con los hechos de la base de hechos. Al realizar esta comparación le asigna a la variable `?x` el valor del campo con el que se empareja, en este caso, por ejemplo, con `Socrates`, esta evaluación la realiza con todos los hechos de la base de hechos, activando la regla tantas

[9] El término propiedad, para el que también se emplean otros sinónimos como campo o casilla, es la traducción de *slot*.

[10] Un estudio detallado de los Marcos puede verse en [MDBD95, pp. 305–316].

veces como hechos sean capaces de estimularla. Las *instancias* de esta regla activada las coloca en la agenda. Cuando ha terminado de evaluar todas las reglas con todos los hechos, finaliza el ciclo disparando una sola *instancia* de regla de la agenda, la que tenga más prioridad. Al dispararse la regla se ejecuta la acción, en este ejemplo se afirma en la base de hechos que `(mortal Socrates)`. Disparada la regla vuelve a comenzar el ciclo, disparándose, como se ha dicho, una *instancia* de regla por cada hecho o conjunto de hechos que sean capaces de estimularla. Ha de tenerse en cuenta que si un hecho ha sido el causante del disparo de una regla, ese mismo hecho no vuelve a disparar esa regla.

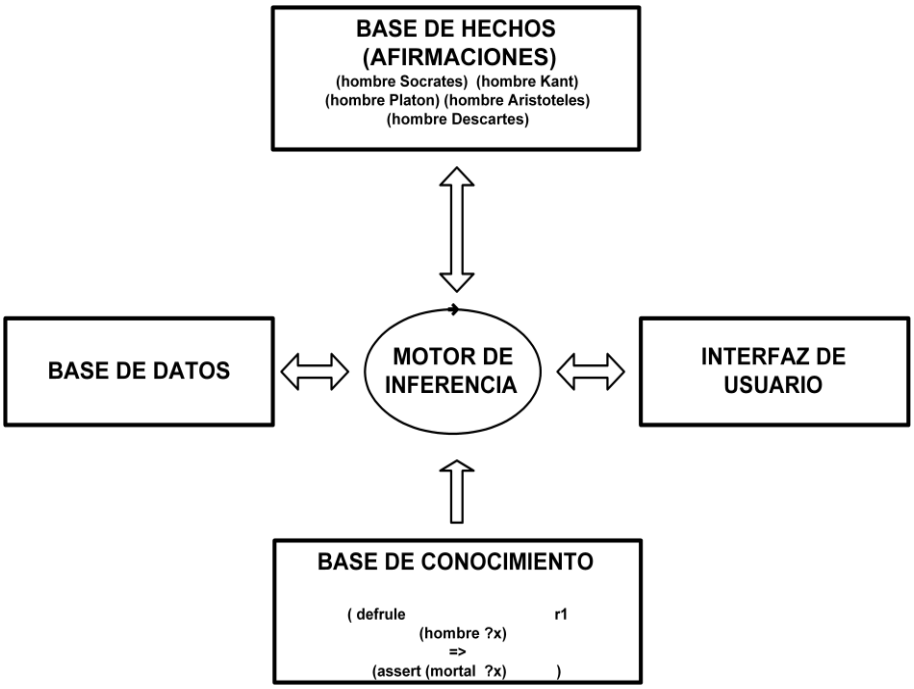

Figura 1.4: Modelo de un sistema de representación de conocimiento basado reglas

En el ejemplo de la figura 1.4 se observa que la regla se disparará cinco veces, afirmando que todas esas personas son mortales. Observamos cómo lo que en realidad existe es una manipulación de símbolos que son capaces de producir nuevos símbolos. Vemos también que el mecanismo de inferencia (la comparación de patrones) es independiente –está codificado aparte, en el motor de inferencia– del conocimiento representado a través de las reglas. Al conocimiento representado mediante las reglas en la base de conocimiento se le suele llamar conocimiento dinámico ya que es capaz de generar nueva información a partir de la informa-

ción existente en la base de hechos. Es evidente que todo este sistema no deja de ser un modelo de sistema inteligente, pero es un modelo que puede llevarse a la práctica, que podemos experimentar con él. Podemos decir que tras dos mil quinientos años se ha pasado de la especulación filosófica que inició Aristóteles a la experimentación científica.

El otro elemento básico de la estructura lo constituye el motor de inferencia. De manera muy resumida diremos que este componente actúa como un motor de una máquina dando vueltas -realizando ciclos-. En cada ciclo realiza un conjunto de operaciones con las estructuras de símbolos representadas en la base de hechos y de acuerdo con las acciones del consecuente de las reglas de la base de conocimiento:

1. Al comenzar el ciclo el motor de inferencia evalúa qué reglas de la base de conocimiento son susceptibles de activarse. Se activarán aquellas cuyos patrones encajan o emparejan con la base de hechos.

2. Las reglas activadas (*instancias* de estas reglas) se ordenan de acuerdo a unos criterios de preferencia.

3. Se dispara sólo la regla (*instancia*) con mayor prioridad.

4. Al dispararse dicha regla se añaden o se eliminan hechos de la base de hechos.

5. Se continua con el ciclo siguiente. Si en un ciclo no se activase ninguna regla el programa se detiene.

1.6.2. Encadenamiento de reglas

A veces, para llegar a deducir (inferir) un hecho no basta con una regla y se hace preciso el encadenamiento de varias reglas de forma que la inferencia de una regla del conjunto, sirve para provocar la activación de otra u otras reglas de forma que, tras un número determinado de ciclos y el disparo de varias reglas, llega a provocarse la inferencia que buscábamos. Veamos un ejemplo sencillo.

```
Todos los hombres son animales

Todos los animales respiran

Juan  es un hombre

Inferimos que Juan respira
```

Base de conocimiento

```
(defrule r1
   (hombre ?x)
=>
   (assert (animal ?x))
)


(defrule r2
   (animal ?x)
=>
   (assert (respira ?x))
)
```

Base de hechos

```
 (hombre Juan)
```

Dejamos al lector que simule la labor del motor de inferencia y observe cómo tras varios ciclos de ejecución se infiere que Juan respira –expresado simbólicamente– `(respira Juan)`.

1.7. CONTROL DEL RAZONAMIENTO

¿Qué puede entenderse por controlar el razonamiento en el transcurso de la resolución de un problema? La resolución de un problema conlleva la ejecución de una serie de acciones que pueden agruparse en lo que denominamos tareas y subtareas. Estas acciones, subtareas y tareas se realizan en un determinado orden. A veces en función del orden en que se realicen puede que el problema se resuelva de forma más eficiente o incluso no llegue a resolverse.

Los lenguajes procedimentales controlan el orden de ejecución de las acciones de varias formas. Básicamente se sigue la secuencia. Es decir, las acciones se ejecutan siguiendo el orden en que están escritas en el programa de ordenador. Este flujo u orden de ejecución puede alterarse si aparecen esquemas condicionales, iterativos o la invocación de procedimientos y funciones.

La pregunta que podemos hacernos es ¿cómo puede conseguirse que el orden de ejecución del programa –el control del razonamiento– se lleve a cabo en un determinado orden en los sistemas basados en reglas? ¿Qué mecanismos existen para lograr este control? Los capítulos 9 y 10 se ocupan de responder a estas cuestiones.[11] Introduciremos aquí las técnicas más relevantes.

[11] Un estudio teórico de este problema puede encontrarse en [MDBD95, pp. 244–249].

1.7.1. Hechos de control

Una posibilidad es incluir en la base de hechos afirmaciones (hechos de control) que marquen objetivos y al mismo tiempo escribir en el antecedente de las reglas patrones que se encajen con esos objetivos. La última regla que permite cumplir un objetivo deberá llevar una acción en su consecuente que elimine ese hecho como objetivo de la base de afirmaciones y, al mismo tiempo, afirme un nuevo objetivo.

1.7.2. Prioridades en las reglas

Como se ha indicado, en cada ciclo de ejecución de un programa el motor de inferencia evalúa todas las reglas cuyos patrones quedan satisfechos (casan o emparejan con los hechos de la base de afirmaciones) quedando activadas.[12] A continuación, ordena dichas reglas según la prioridad que se le haya asignado en tiempo de diseño o que haya sido modificada en tiempo de ejecución. El resultado de la ordenación se registra en una agenda y el motor de inferencia hace que se dispare la *instancia* de la regla que tenga mayor prioridad, ejecutándose todas las acciones de su consecuente.

1.7.3. Activación de módulos

Otro mecanismo con que se cuenta habitualmente para el control del razonamiento es el establecimiento de módulos. Es frecuente que un conjunto de acciones se agrupen en una estructura de mayor rango y a la que se le da un nombre (en la programación procedimental se definen subrutinas, procedimientos, funciones, etc.). Esto permite estructurar el programa en partes y, en cierto modo, dividir una tarea en subtareas. En los sistemas basados en reglas también se puede hacer esto dividiendo la base de conocimiento en partes o módulos de manera que en función de la tarea que se esté llevando a cabo en un momento determinado sólo las reglas relacionadas con esa tarea son evaluadas por el motor de inferencia. Esto tiene indudables ventajas, por ejemplo, facilita el mantenimiento de la aplicación,[13] mejora la eficiencia del programa ya que el motor de inferencia tendrá menos reglas que evaluar en cada ciclo y, finalmente, facilita el control del razonamiento ya que la tarea para resolver el problema se contempla como un subconjunto de subtareas pasando de unas a otras a través de unas reglas de transición que controlan

[12] En realidad lo que quedan activadas son *instancias* de las reglas, siendo posible que una regla se active varias veces en función de que existan hechos que lo permitan.

[13] Esta mejora se traduce en una disminución del tiempo para la modificación de las reglas o creación de otras nuevas.

el orden en que se deben de realizar las subtareas.

Puede observarse, por tanto, que, a diferencia de la programación procedimental, cambiar el flujo del programa en estos sistemas, es decir, controlar el razonamiento, puede llevarse a cabo mediante mecanismos que no alteran el orden en que están escritas las reglas en el código del programa. Bastará con cambiar los hechos de la base de afirmaciones, las prioridades de las reglas o la organización de los módulos para que se modifique el flujo de control del programa.

Podemos observar que todo lo dicho en estas secciones está muy relacionado con la forma de razonar del ser humano. La forma en que una tarea sea llevada a cabo estará muy relacionado con la prioridad de las reglas, cuando una persona realiza una tarea determinada es porque se ha marcado un objetivo, mientras no se haya satisfecho dicho objetivo se mantendrá en la realización de esa tarea. Cuando el ser humano está realizando una tarea, sus pensamientos se concentran en ideas relacionadas con dicha tarea (módulos) y, salvo que aparezcan acontecimientos extraordinarios que le hagan cambiar de tarea, proseguirá su realización hasta su término.