CAPÍTULO 2

La representación del conocimiento

2.1 INTRODUCCIÓN

En este capítulo se analizan algunas de las representaciones más comunes del conocimiento para los sistemas expertos. La representación del conocimiento es de gran importancia en estos sistemas por dos razones. En primer lugar, los shells de los sistemas expertos están diseñados para cierto tipo de representación del conocimiento, como las reglas o la lógica. En segundo lugar, la forma en que un sistema experto representa al conocimiento afecta su desarrollo, eficiencia, velocidad y mantenimiento. En el capítulo 3 analizaremos cómo es que se hacen las inferencias.

2.2 EL SIGNIFICADO DEL CONOCIMIENTO

Conocimiento, como *amor*, es una de esas palabras de las que todos sabemos el significado, aunque resulta difícil definirlo: ambas tienen también muchos significados. Otras palabras como *datos*, *hechos* e *información* se utilizan para sustituir *conocimiento*.

La **epistemología** es el estudio del conocimiento (Angeles 81); se ocupa de su naturaleza, su estructura y sus orígenes. En la figura 2.1 se ilustran algunas de las categorías de la epistemología. Además de los tipos de conocimiento filosófico que expresaron Aristóteles, Platón, Descartes, Hume, Kant y otros, hay dos tipos especiales llamados **a priori** y **a posteriori**. El primer término viene del latín y significa "lo que precede". Un conocimiento *a priori* precede y es independiente del conocimiento obtenido por medio de los sentidos; por ejemplo, las frases "todo tiene una causa" y "la suma de los ángulos de todos los triángulos en un plano es de 180 grados". El conocimiento *a priori* se considera verdad universal y no puede negarse sin contradicción. Las frases lógicas, las leyes matemáticas y el conocimiento de los adolescentes son otros ejemplos de conocimiento *a priori*.

Lo opuesto al conocimiento *a priori* es el que se deriva de los sentidos, o conocimiento *a posteriori*. La verdad o falsedad de este conocimiento puede verificarse mediante la experiencia de los sentidos, como en la frase "la luz es verde". Sin embargo, como la experiencia sensorial no siempre es confiable, un conocimiento *a posteriori*, puede rechazarse con base en nuevo conocimiento, sin necesidad de caer en contradicciones. Por ejemplo, si vemos a alguien de ojos café, creeríamos que de ese color son sus ojos. Sin embargo, si más tarde

observamos que esa persona se quita los lentes de contacto de color café y tiene los ojos azules, tendríamos que revisar nuestro conocimiento.

El conocimiento puede clasificarse aún más como **conocimiento por procedimientos**, **conocimiento declarativo** y **conocimiento tácito**. Los primeros dos tipos corresponden a los paradigmas de procedimientos y declarativo abordados en el capítulo 1.

A menudo el conocimiento por procedimientos se refiere a la forma en que sabemos hacer algo, un buen ejemplo sería saber cómo hervir agua en un pote. El conocimiento declarativo alude a la capacidad de saber que algo es verdadero o falso, que puede expresarse en forma de frases declarativas como "no toques con los dedos un pote con agua hirviendo".

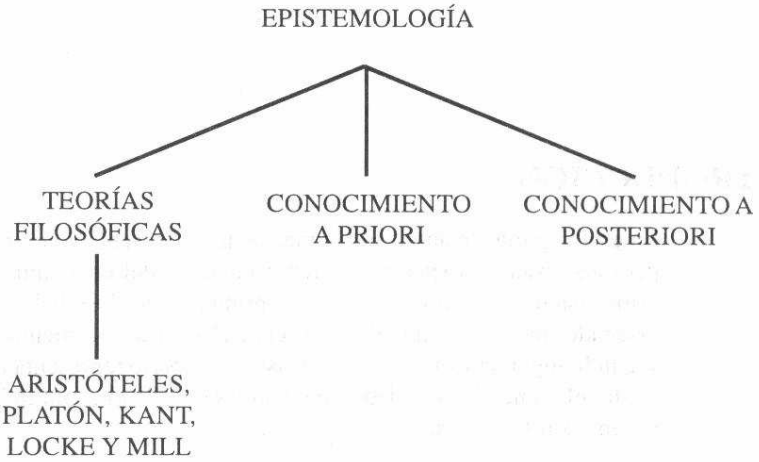


Figura 2.1 Algunas categorías de la epistemología

Al conocimiento tácito en ocasiones se le llama ***conocimiento inconsciente***, porque no puede expresarse mediante el lenguaje. Un ejemplo es saber cómo mover la mano, en gran medida, se podría decir que el movimiento proviene de tensar o relajar ciertos músculos y tendones. Pero en el siguiente nivel inferior, ¿cómo *saber* tensar o relajar los músculos y tendones? Otros ejemplos son caminar o andar en bicicleta. Los sistemas de computación ANS se relacionan con este tipo de conocimiento porque normalmente la red neuronal no puede explicar su conocimiento de manera directa, pero podría hacerlo si se tiene un programa apropiado (consulte la sección 1.14).

El conocimiento es de vital importancia en los sistemas expertos. En efecto, una analogía con la expresión clásica de Wirth

```
Algoritmos + Estructura de datos = Programas
```

para sistemas expertos es

```
Conocimiento + Inferencia = Sistemas expertos
```

En este libro se utiliza el conocimiento como parte de una jerarquía, lo que se ilustra en la figura 2.2. En la base se encuentra el ruido, formado por elementos que son de poco interés y que ocultan datos. El siguiente nivel de la jerarquía son los datos, elementos de interés poten-

cial. En el tercer nivel hay información, o datos procesados que resultan de interés. El siguiente nivel es el conocimiento, que representa información muy especializada. En el capítulo 1, el conocimiento de los sistemas expertos basados en reglas se definió como las reglas activadas por hechos para producir nuevos hechos o conclusiones. Este proceso de *inferencia* es la segunda parte esencial de un sistema experto. El término inferencia se usa generalmente para sistemas mecánicos como los sistemas expertos. Para el pensamiento humano, generalmente se utiliza *razonamiento*.

Figura 2.2 La jerarquía del conocimiento

El término *hechos* puede significar datos o información. Dependiendo de la forma en que estén escritos, los sistemas expertos pueden formular inferencias utilizando estos hechos. También pueden (1) separar los datos del ruido, (2) transformar datos en información o (3) transformar la información en conocimiento.

Como ejemplo de estos conceptos, observe la siguiente secuencia de 24 números:

```
137178766832525156430015
```

Sin conocimiento, toda esta secuencia aparecería como ruido. Sin embargo, si se sabe que esta secuencia tiene un significado, entonces se convierte en datos. Determinar lo que son datos y lo que es ruido es parecido al viejo dicho relacionado con la jardinería, "hierba es cualquier cosa que crece y que no es lo que usted quiere".

Quizá exista cierto conocimiento que permita convertir estos datos en información. Por ejemplo, el siguiente algoritmo procesa los datos para producir información.

```
Agrupe los números en pares.
Ignore cualquier número de dos dígitos menor a 32.
Sustituya los caracteres ASCII por los números de dos
dígitos.
```

La aplicación de este algoritmo a los anteriores 24 digitos produce la siguiente información

```
ORO 438+
```

Ahora el conocimiento puede aplicarse a esta información. Por ejemplo, tal vez haya una regla

```
SI oro es menor a 500
   y el precio está subiendo (+)
ENTONCES
   compra oro
```

Aunque no se muestra explícitamente en la figura 2.2, *la experiencia* es un tipo de conocimiento que poseen los especialistas. Ésta no se encuentra comúnmente en fuentes públicas de información, como libros y artículos; es el conocimiento implícito de los especialistas que debe extraerse y hacerse explícito para que pueda codificarse en un sistema experto. Por encima del conocimiento está el **metaconocimiento**, el significado del prefijo **meta** es "más allá de". El metaconocimiento es conocimiento acerca del conocimiento y la experiencia. Como es posible diseñar un sistema experto con conocimiento de varios dominios diferentes, el metaconocimiento especificaría cuál base de conocimiento es la aplicable. Por ejemplo, un sistema experto puede tener bases de conocimiento acerca de reparación de autos Chevrolet 1988, Ford 1985 y Cadillac 1989; se emplearía la base apropiada dependiendo del automóvil que se necesite reparar. Esto no es eficiente para la memoria y rapidez, porque todas la bases del conocimiento estarían trabajando al mismo tiempo. Además, pueden surgir conflictos cuando el sistema experto trate de decidir las reglas aplicables de todas las bases del conocimiento a la vez. El metaconocimiento podría utilizarse también dentro de un dominio para decidir cuál grupo de reglas es el más aplicable.

En un sentido filosófico, la **sabiduría** es la cumbre del conocimiento. La sabiduría es el metaconocimiento de determinar los mejores objetivos en la vida y cómo lograrlos. Una regla de sabiduría sería

```
SI tengo suficiente dinero para mantener feliz a mi esposa
ENTONCES me retiraré y disfrutaré de la vida
```

Sin embargo, debido a la extrema escasez de sabiduría en el mundo, debemos limitarnos a sistemas basados en el conocimiento y dejarle los basados en la sabiduría a los políticos.

2.3 PRODUCCIONES

Se han concebido varias técnicas diferentes de representación del conocimiento. Entre éstas se incluyen reglas, redes semánticas, marcos, guiones, lenguajes de representación del conocimiento como KL-1 (Woods 83) y KRYPTON (Brachman 83), gráficas conceptuales (Sowa 84) y otros (Brachman 80). Como se describió en el capítulo 1, las reglas de producción suelen utilizarse como base de conocimiento para los sistemas expertos, porque sus ventajas sobrepasan en mucho a sus desventajas.

Una notación formal para definir producciones es la forma Backus-Naur (BNF; McGettrick 80). Esta notación es un **metalenguaje** para definir la **sintaxis** de un lenguaje. La sintaxis define la forma, mientras que la **semántica** se refiere al significado. Un metalenguaje es un lenguaje para describir otros. El prefijo *meta* significa "más allá de", por lo tanto un metalenguaje va "más allá de" un lenguaje normal.

Hay muchos tipos de lenguajes: naturales, lógicos, matemáticos y de cómputo. La notación de BNF para la simple regla de una oración formada por un sujeto y un verbo seguidos por un signo de puntuación, es la siguiente regla de producción:

```
<oración> ::= <sujeto> <verbo> <signo final>
```

Los **paréntesis en ángulo, <>** y **::=** son símbolos del metalenguaje y no del lenguaje que se especifica. El símbolo "::=" significa "se define como" y es el equivalente en BNF de la flecha, →, utilizada en el capítulo 1 con las reglas de producción. Para evitar confusión con el operador de asignación de Pascal, :=, utilizaremos la flecha.

A los términos dentro de los paréntesis en ángulo se les llama *símbolos **no terminales*** o simplemente no terminales. Un símbolo no terminal es una variable en que se representa otro término. Éste puede ser no terminal o **terminal**. Un terminal no se puede reemplazar con algo más y, por tanto, es una constante.

La <frase> no terminal resulta especial porque se trata del único **símbolo de inicio** a partir del cual se definen los otros. En la definición de lenguajes de programación, al símbolo de inicio a menudo se le llama <programa>. La siguiente regla de producción

```
<oración> → <sujeto> <verbo> <signo final>
```

establece que una oración se integra con un sujeto seguido por un verbo y luego un signo final. Las siguientes reglas completan las no terminales especificando sus terminales posibles. La **barra** significa "o" en el metalenguaje.

```
<sujeto> → Yo | Tú | Nosotros
<verbo> → fui | vine | fuiste | vinistes | fuimos |
vinimos
<signo final> → . | ? | !
```

Todas las oraciones posibles en el lenguaje, las producciones, pueden producirse reemplazando sucesivamente cada uno de los no terminales con sus no terminales o terminales del lado derecho, hasta eliminar todos los no terminales. Las siguientes son algunas producciones:

```
Yo fui.
¿Yo fui?
¡Yo fui!
Tú fuistes.
¿Tú fuistes?
¡Tú fuistes!
Nosotros fuimos.
¿Nosotros fuimos?
¡Nosotros fuimos!
```

A un conjunto de terminales se le llama ***cadena*** del lenguaje. Si la cadena puede derivarse del símbolo de inicio, mediante el reemplazo de no terminales, con sus reglas de definición, entonces se le llama ***oración** válida*. Por ejemplo, "Nosotros", "NosotrosNosotros", y "fuifuistefuimos" son cadenas válidas del lenguaje, pero no son oraciones válidas.

Una **gramática** es un conjunto completo de reglas de producción que definen un lenguaje sin ambigüedades. Aunque las reglas anteriores definen una gramática, es un conjunto muy limitado porque hay pocas producciones posibles. Por ejemplo, una gramática más elaborada también podría incluir objetos directos, como en las producciones

```
<oración> → <sujeto> <verbo> <objeto> <signo final>
<objeto> → casa | trabajo | escuela
```

Aunque ésta es una gramática válida, resulta demasiado simple para el uso práctico. Una gramática más práctica es la siguiente, en la que se ha omitido el signo final para simplificar:

```
<oración> → <frase de sujeto> <verbo> <frase de objeto>
<frase de sujeto> → <determinador> <sustantivo>
<frase de objeto> → <determinador> <adjetivo> sustantivo>
<determinador> → un | uno | unos | unas | el | la | los |
las | este | esta | estos | estas | esos | esas
<sustantivo> → hombre | devorador
<verbo> → es | está | fue, estuvo
<adjetivo> → postres | gran
```

El <determinador> se utiliza para indicar un elemento específico. Utilizando esta gramática, pueden generarse oraciones como:

```
el hombre fue un devorador de postres
un devorador fue el gran hombre
```

Un **árbol de análisis gramatical** o **árbol de derivación** es una representación gráfica de una oración, descompuesta en todos los terminales y no terminales utilizados para derivarla. En la figura 2.3 se muestra el árbol de análisis gramatical para la frase "el hombre fue un gran devorador". Sin embargo la cadena "hombre fue un gran devorador" no es una oración válida porque carece del determinador en la frase de sujeto. Un compilador crea un árbol de análisis gramatical cuando trata de determinar si las oraciones de un programa se adecuan a la sintaxis válida de un lenguaje.

El árbol de la figura 2.3 muestra que la oración "el hombre fue un gran devorador" puede

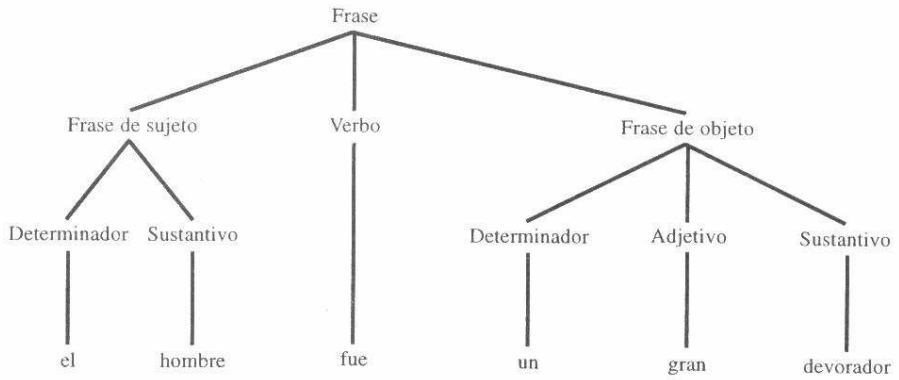


Figura 2.3 Árbol de análisis gramatical de una oración

derivarse del símbolo de inicio mediante la aplicación de las producciones apropiadas. A continuación se explican los pasos de este proceso; la **flecha doble, =>**, significa aplicar las producciones mostradas.

```
<frase> => <frase de sujeto> <verbo> <frase de objeto>
<frase de sujeto> => <determinador> <sustantivo>
<determinador> => el
<sustantivo> => hombre
<verbo> => fue
<frase de objeto> => <determinador> <adjetivo> sustantivo>
<determinador> => un
<adjetivo> => gran
<sustantivo> => devorador
```

Una forma alterna de utilizar las producciones consiste en generar frases válidas sustituyendo todas las terminales apropiadas por no terminales. Por supuesto, no todas las producciones, como "el hombre fue el postre", tendrán sentido.

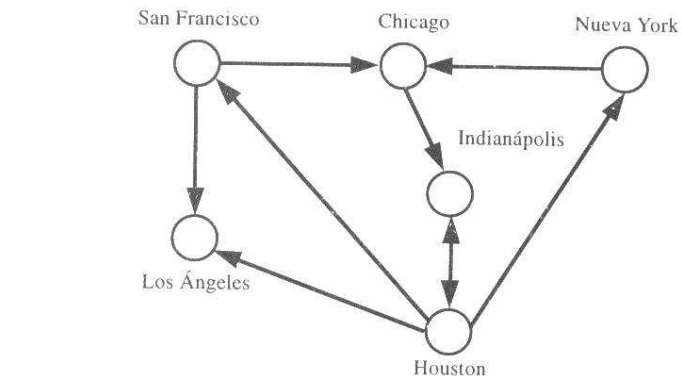
2.4 REDES SEMÁNTICAS

Una **red semántica**, o red, es una técnica clásica de representación de la AI utilizada para información relativa a las proposiciones (Stillings 87). A una red semántica también se le llama ***red de proposiciones***. Como se analizó antes, una proposición es una frase verdadera o falsa, como "todos los perros son mamíferos" y "un triángulo tiene tres lados". Son formas de conocimiento declarativo porque establecen hechos. En términos matemáticos, una red semántica es una gráfica rotulada y con dirección. Una proposición siempre es verdadera o falsa y se le llama **atómica** porque su veracidad ya no puede dividirse más, el término *atómica* se emplea en el sentido clásico de objeto indivisible. En contraste, las proposiciones que trataremos en el capítulo 5 normalmente sólo necesitan ser verdaderas o falsas.

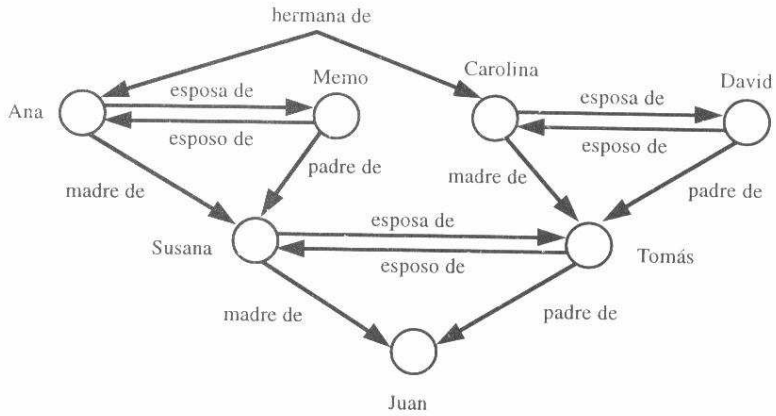
Las redes semánticas se desarrollaron para la AI como una forma de representar la memoria y la comprensión del lenguaje del ser humano (Quillian 68). Quillian las utilizó para analizar el significado de palabras en frases. Desde entonces se han aplicado a muchos problemas relacionados con la representación del conocimiento.

La estructura de una red semántica se muestra gráficamente en términos de **nodos** y los **arcos** que los conectan. A los nodos suele denominárseles como objetos y a los arcos como ***vínculos*** o ***bordes***.

Los vínculos de una red semántica se utilizan para expresar relaciones. Por lo general, los nodos se utilizan para representar objetos físicos, conceptos o situaciones. En la figura 2.4 (a) se muestra una red ordinaria, en realidad una gráfica con dirección, donde los vínculos indican las rutas de una línea aérea entre ciudades. Los nodos son los círculos, los vínculos, las líneas que los conectan y las flechas muestran la dirección en que pueden volar los aviones, de aquí el término gráfica con dirección. En la figura 2.4 (b) los vínculos muestran las relaciones entre los miembros de una familia. Éstas son de gran importancia en las redes semánticas porque proporcionan la estructura básica para la organización del conocimiento. Sin relaciones, el conocimiento es una simple colección de hechos inconexos. Con ellas, el conocimiento es una estructura cohesiva a partir de la cual puede inferirse conocimiento adicional. Por ejemplo, en la figura 2.4 (b) se puede inferir que Ann y Bill son los abuelos de John, aunque no haya un vínculo explícito rotulado "abuelo de".



(a) Una red general



(b) Una red semántica

Figura 2.4 Dos tipos de red

A las redes semánticas a veces se les denomina **redes asociativas** porque los nodos están asociados o relacionados con otros nodos. De hecho, el trabajo original de Quillian modeló la memoria humana como una red asociativa en lo cual los conceptos eran los nodos y los vínculos formaban las conexiones entre ellos. De acuerdo con este modelo, cuando un nodo de concepto se estimula al leer las palabras de una frase, sus vínculos con otros nodos se activan en un patrón de dispersión. Si otro nodo recibe activación suficiente, el concepto sería forzado dentro de la mente consciente. Por ejemplo, aunque se conozcan miles de palabras, sólo se está pensando en las palabras específicas de la frase que se lee.

Ciertos tipos de relaciones han probado ser muy útiles en una amplia variedad de representaciones del conocimiento. En lugar de definir nuevas relaciones para problemas diferentes, es habitual usar estos tipos estándar. El uso de tipos comunes facilita la comprensión de una red no muy conocida.

Dos tipos de vínculos utilizados con frecuencia son **ES-UN** y **UN TIPO DE** (Winston 84). La figura 2.5 es un ejemplo de una red semántica que utiliza estos vínculos; en esta figura, ES-UN significa "es un caso de" y se refiere a un miembro específico de una clase. Una **clase** está relacionada con el concepto matemático de conjunto, que alude a un grupo de objetos.

Aunque un conjunto puede tener elementos de cualquier tipo, los objetos de una clase tienen alguna relación entre sí. Por ejemplo, es posible definir un conjunto formado por

```
{ 3, huevos, azul, llantas, arte }
```

Sin embargo, los miembros de este conjunto no tienen relación entre sí. Por el contrario, aviones, trenes y automóviles se relacionan en una clase porque todos son tipos de transporte.

El vínculo UN-TIPO-DE se utiliza aquí para relacionar una clase con otra. No se usa para relacionar un individuo específico porque esa es la función de ES-UN. UN-TIPO-DE relaciona una clase individual con una clase padre a la que pertenece una clase hijo.

Desde otro punto de vista, el UN-TIPO-DE relaciona nodos **genéricos** con nodos genéricos, mientras que ES-UN relaciona un caso o un **individuo** con una clase genérica. Observe en la figura 2.5 que las clases más generales se encuentran en la parte superior y las más específicas en la base. La clase más general, a la que señala una flecha UN-TIPO-DE, se le denomina **superclase**; si ésta tiene un vínculo UN-TIPO-DE apuntando a otro nodo, entonces también es una clase de la superclase a la que señala el vínculo UN-TIPO-DE. Otra forma de expresarlo es que un vínculo UN-TIPO-DE señale de una **subclase** a una clase. El vínculo **SON** se usa algunas veces para UN-TIPO-DE y se lee como el verbo "son".

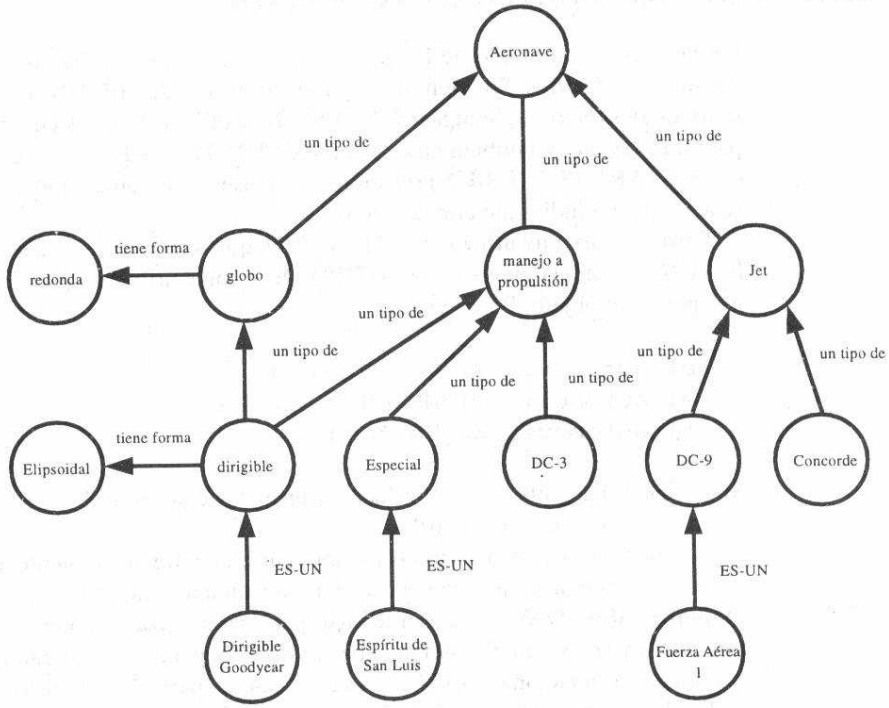


Figura 2.5 Red semántica con vínculos ES-UN y UN-TIPO-DE

Los objetos de una clase tienen uno o más **atributos** en común y cada atributo tiene un **valor**. La combinación de atributo y valor es una **propiedad**. Por ejemplo, un dirigible tiene

atributos de tamaño, peso, forma, color, etcétera. El valor del atributo de forma es elipsoidal. En otras palabras, un dirigible tiene la forma elipsoidal como una de sus propiedades. Como el Goodyear es un dirigible y los dirigibles tienen forma elipsoidal, entonces el Goodyear es elipsoidal. También se pueden encontrar otros tipos de vínculos en las redes semánticas. El vínculo **ES** define un valor. Por ejemplo, cualquier aeronave presidencial de Estados Unidos es el Fuerza Aérea 1. Si el presidente está en un DC-9, entonces el Fuerza Aérea 1 es un DC-9. El vínculo **CAUSA** expresa conocimiento causal. Por ejemplo, el aire caliente CAUSA que un globo se eleve.

A la duplicación de las características de un nodo por parte de un descendiente se le llama **herencia**. A menos que haya más evidencias específicas de lo contrario, se supone que todos los miembros de una clase heredarán todas las propiedades de sus superclases. Por ejemplo, los globos tienen una forma redonda. Sin embargo, como la clase dirigible tiene un vínculo que señala a la forma elipsoidal, ésta tiene precedencia. La herencia es una herramienta muy útil en la representación del conocimiento porque elimina la necesidad de repetir características comunes. Los vínculos y la herencia proporcionan medios muy eficientes de representación del conocimiento porque muchas relaciones complejas pueden mostrarse con pocos nodos y vínculos.

2.5 TRIPLETA OBJETO-ATRIBUTO-VALOR

Un problema con el uso de las redes semánticas es que no hay definiciones estándar para nombrar lo vínculos. Por ejemplo, algunos libros utilizan ES-UN para relaciones genéricas e individuales (Barr 81; Staugaard 87). ES-UN se utiliza entonces en el sentido ordinario de las palabras "es un" y también en el sentido de UN-TIPO-DE. En la herramienta para sistemas expertos ART, IS-A (ES-UN) relaciona dos clases de objetos e INSTANCE-OF (CASO-DE) relaciona a un individuo con una clase.

Otro vínculo muy utilizado es TIENE-UN, que relaciona una clase con una subclase. TIENE-UN señala a la inversa de UN-TIPO-DE y suele utilizarse para relacionar un objeto con una parte del objeto. Por ejemplo,

```
el automóvil TIENE-UN motor
el automóvil TIENE-UNAS llantas
el automóvil ES-UN ford
```

Más específicamente, ES-UN relaciona un valor con un atributo mientras que TIENE-UN relaciona un objeto con un atributo.

Los elementos objeto, atributo y valor ocurren tan frecuentemente que es posible construir una red semántica simplificada utilizando únicamente estos elementos. Una **tripleta objeto-atributo-valor (OAV)**, puede utilizarse para caracterizar todo el conocimiento de una red semántica y se usó en el sistema experto MYCIN para diagnosticar enfermedades infecciosas. La representación del trío OAV es conveniente para ordenar el conocimiento en forma de tabla y luego traducirla al código de la computadora mediante inducción de reglas. En la tabla 2.1 se muestra el ejemplo de una tabla con el trío OAV.

Los tríos OAV son especialmente útiles para representar los hechos y sus patrones para llevarlos al antecedente de una regla. La red semántica para este tipo de sistema consta de nodos para objetos, atributos y valores, conectados por vínculos TIENE-UN y ES-UN. Si sólo habrá de representarse un objeto y no se requiere de herencia, tal vez bastaría una simple representación llamada ***pares atributo-valor***, o simplemente ***AV***.

Objeto	Atributo	Valor
manzana	color	roja
manzana	tipo	mcintosh
manzana	cantidad	100
uvas	color	rojas
uvas	tipo	sin semilla
uvas	cantidad	500

Tabla 2.1 Tabla OAV

2.6 PROLOG Y REDES SEMÁNTICAS

Las redes semánticas son fáciles de traducir a PROLOG. Por ejemplo,

```
es un (dirigible_goodyear,dirigible).
es un (espíritu_de_san_luis,especial).
tiene forma (dirigible,elipsoidal).
tiene forma (globo,redondo).
```

son instrucciones de PROLOG que expresan algunas de las relaciones de la red semántica de la figura 2.5. Un punto marca el final de una frase.

Elementos esenciales de PROLOG

Cada una de las frases anteriores es una **expresión de un predicado** de PROLOG, o simplemente un predicado, porque está basado en lógica de predicados. Sin embargo, PROLOG no es un verdadero lenguaje de lógica de predicados porque es un lenguaje computacional con instrucciones ejecutables. En PROLOG, un predicado está formado por el nombre de predicado, como es un, seguido de cero o más argumentos encerrados entre paréntesis y separados por comas. A continuación se presentan algunos ejemplos de predicados y comentarios elaborados en PROLOG precedidos por punto y coma:

```
color(rojo).                ; rojo es un color
padre_de(tom,john).         ; tom es el padre de john
madre_de(susan,john).       ; susan es la madre de john
padres(tom,susan,john).     ; tom y susan son
                            ;  padres de john
```

Los predicados con dos argumentos son más fáciles de entender si coloca el nombre de predicado siguiendo al primer argumento. Los significados de estos predicados con más de dos argumentos deben establecerse explícitamente, como lo ilustra el predicado padres. Otra dificultad es que las redes semánticas se utilizan primordialmente para representar relaciones binarias, porque una línea sólo tiene dos extremos, por supuesto. No es posible poner la afirmación padres como un borde con una sola dirección porque hay tres argumentos. La idea de colocar a Tom y Susan juntos en un nodo de padres y a John en las otras direcciones lleva a una nueva complicación. Entonces es imposible utilizar el nodo de los padres para otra relación binaria, como madre-de, porque también se incluiría a Tom.

Los predicados pueden expresarse también con relaciones IS-A (ES-UN) y HAS-A (TIENE-UN).

```
is_a(rojo,color).
has_a(john,padre).
has_a(john,madre).
has_a(john,padres).
```

Observe que el predicado has_a no expresa el mismo significado que antes, porque el padre, la madre y los padres de John no son nombrados explícitamente. Para ello, se deben agregar algunos predicados adicionales.

```
is_a(tom,padre).
is_a(susan,madre).
is_a(tom,padre).
is_a(susan,padre).
```

Incluso estos predicados adicionales no expresan el mismo sentido de los originales. Por ejemplo, sabemos que John tiene un padre y que Tom es un padre, pero esto no informa que Tom sea el padre de John.

Todas las frases anteriores describen realmente hechos de PROLOG. Los programas de PROLOG están formados por hechos y reglas en la forma general de **metas**:

$$p :- p_1, p_2, \ldots p_N.$$

en los que p es la cabeza de la regla y p_i son las **submetas.** Normalmente esta expresión es una cláusula de Horn, que establece que la meta principal, p, se satisface si y sólo si son satisfechas todas las submetas. La excepción se presenta cuando se utiliza el predicado especial para fallas, que resulta sumamente conveniente porque no es fácil probar una negación en lógica clásica y PROLOG está basado en la lógica clásica. Una negación es vista como la falla al tratar de encontrar una prueba, esto puede ser un proceso muy largo y complicado si hay muchas coincidencias probables. El predicado de **corte** y de falla especial hace que el proceso de negación sea más eficiente reduciendo la búsqueda de una prueba.

Las comas que separan las submetas representan una disyunción Y lógica. El símbolo, :-, se interpreta como un SI. Si sólo existe la cabeza y no hay lado derecho, como en

```
p.
```

entonces la cabeza se considera verdadera. Esto se debe a que predicados como los siguientes se consideran hechos y, por tanto, deben ser verdaderos.

```
color(rojo).
has_a(john,padre).
```

Otra forma de pensar en los hechos es considerándolos una conclusión incondicional que no depende de nada más y, por lo tanto, el SI, :-, no es necesario. Por el contrario, las reglas de PROLOG necesitan el SI porque son conclusiones condicionales que en verdad dependen de una o más condiciones. Como ejemplo, las reglas de los padres quedan así:

```
padre(X,Y) :- papá (X,Y).
padre(X,Y) :- mamá (X,Y).
```

Figura 2.6

lo que significa que X es un padre de Y, si X es el papá de Y o si X es la mamá de Y. Del mismo modo, un abuelo puede definirse como:

```
abuelo(X,Y) :- padre(X,Z),padre(Z,Y).
```

y un antepasado puede definirse como:

```
(1)  ancestro(X,Y) :- padre(X,Y).
(2)  ancestro(X,Y) :- ancestro (X,Z),ancestro(Z,Y).
```

en donde (1) y (2) sólo se utilizan con propósitos de identificación.

Búsqueda en PROLOG

Un sistema para ejecución de instrucciones de PROLOG es generalmente un intérprete, aunque algunos sistemas pueden generar código compilado. En la figura 2.6 se muestra la forma general de un sistema PROLOG. El usuario interactúa con PROLOG mediante la introducción de consultas de predicados y la recepción de respuestas. La **base de datos de los predicados** contiene a los predicados de regla y de hecho que se han introducido, y con ellos forma la base del conocimiento. El intérprete trata de determinar si un predicado de consulta que el usuario introdujo se encuentra en la base de datos. La respuesta devuelta es afirmativa en caso de que se encuentre en la base de datos; de lo contrario, es negativa. Si la consulta es una regla, el interprete tratará de satisfacer los subobjetivos conduciendo primero una búsqueda a fondo, como se muestra en la figura 2.7. Por el contrario, también se muestra una búsqueda primero a lo ancho, aunque éste no es el modo normal de PROLOG. En una **búsqueda primero a fondo**, la búsqueda desciende lo más posible y luego regresa. En PROLOG también va de izquierda a derecha. Una **búsqueda primero a lo ancho** pasa de nivel en nivel antes de descender al siguiente nivel inferior.

Figura 2.6 Organización general de un sistema PROLOG

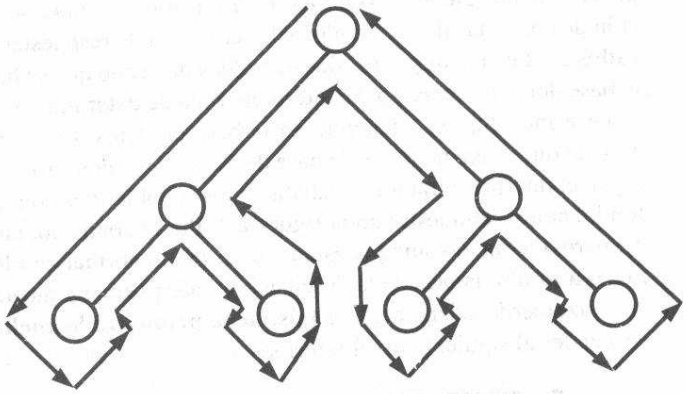
Consideremos las reglas de los antepasados, definidas como (1) y (2), como ejemplo de búsqueda de objetivos en PROLOG, Tomemos los siguientes hechos como entrada.

```
(3)  padre(ann,mary).
(4)  padre(ann,susan).
(5)  padre(mary,bob).
(6)  padre(susan,john).
```

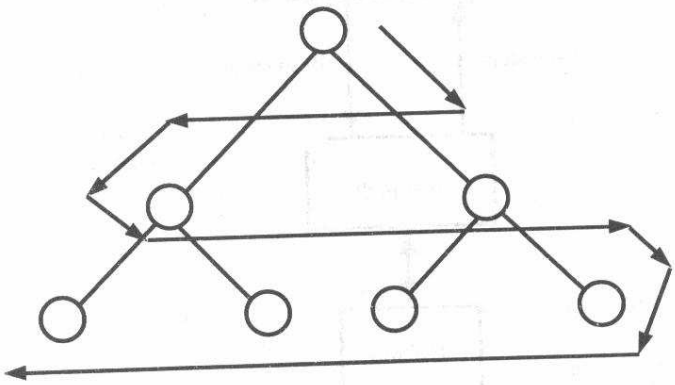
Ahora supongamos que se le consulta a PROLOG si Ann es antepasado de Susan

```
:-ancestro(ann,susan).
```

donde la ausencia de una cabeza indica que se trata de una **consulta**, que es una condición que habrá de probar PROLOG. Hechos, reglas y consultas son los tres tipos de cláusula de Horn aplicables a PROLOG. Se puede probar una condición, si es la conclusión del caso de una cláusula. Por supuesto, la propia cláusula debe poderse probar por sí misma, lo que se hace mediante la demostración de las condiciones de la cláusula.



(a) Búsqueda primero a fondo



(b) Búsqueda primero a lo ancho

Figura 2.7 Búsquedas primero a fondo y primero a lo ancho para un árbol arbitrario

Al recibir esta entrada, PROLOG empezaría a buscar una instrucción cuya cabeza coincida con el patrón de entrada ancestro(ann,susan). A esto se le llama ***coincidencia de patrones***, de manera parecida a la coincidencia de patrones de hechos en el antecedente de una regla de producción. La búsqueda empieza desde la primera instrucción que se introdujo (1), la **superior,** y sigue hasta la última instrucción (6), la **inferior**. Hay una posible coincidencia con la primera regla de ancestro (1). La variable X coincide con ann y la variable Y con susan. Como la cabeza coincide, ahora PROLOG tratará de equiparar el cuerpo de (1), produciendo la submeta padre(ann,susan). PROLOG trata de que esta submeta coincida con las cláusulas y en algún momento coincidirá con el hecho (4) padre(ann,susan). No hay más objetivos que coincidan y PROLOG responde "sí" porque la consulta original es verdadera.

Otro ejemplo, supongamos que la consulta es

```
:-ancestro(ann,john)
```

La primera regla de ancestros (1) coincide y X toma el valor de ann y Y el de john. PROLOG ahora trata de hacer que coincida el cuerpo de (1), padre(ann,john), con todas las instrucciones de padre. Ninguna coincide y, por tanto, el cuerpo de (1) no puede ser cierto. Como el cuerpo de (1) no es cierto, la cabeza no puede ser cierta.

Como (1) no puede ser verdadero, PROLOG prueba la segunda instrucción de ancestros (2). X toma el valor de ann y Y el de john. PROLOG trata de probar que la cabeza de (2) es verdadera probando que ambas submetas de (2) son verdaderas. Es decir, debe probarse que ancestro(ann,Z) y ancestro(Z,john) son verdaderos. PROLOG trata de equiparar las submetas de izquierda a derecha a partir de ancestro(ann,Z). Empezando desde la parte superior, coincide primero con la instrucción (1), por lo tanto PROLOG trata de probar el cuerpo de (1), padre(ann,Z). Empezando a buscar nuevamente desde la parte superior, este cuerpo coincide primero con la instrucción (3) y por ello PROLOG asigna a Z el valor de mary. Ahora PROLOG trata de equiparar la segunda submeta de (2) como ancestro (mary,john). Esto coincide con (1) y, por tanto, PROLOG trata de satisfacer su cuerpo, padre(mary,john). Sin embargo, no hay padre(mary,john) en la instrucción (3) a la (6) y, por tanto, esta búsqueda falla.

Entonces PROLOG examina de nuevo su suposición de que Z debe ser mary. Como esta opción no funcionó, trata de encontrar otro valor que funcione. Otra posibilidad surge de (4) al asignar a Z el valor de susan. A esta técnica de retroceso para probar una ruta de búsqueda diferente cuando otra falla, se le denomina rastreo hacia atrás o *backtrack* y se utiliza a menudo para resolver problemas.

Con la elección de Z como susan, PROLOG intenta probar que ancestro(susan,john) es verdadero. Mediante (1), debe probarse que el cuerpo padre(susan,john) es verdadero. Efectivamente, el hecho (3) coincide y entonces se prueba que la consulta

```
:-ancestro(ann,john).
```

es verdadera porque se ha probado que sus condiciones son verdaderas.

Observe que la estructura de control de PROLOG tiene el tipo del algoritmo de Markov, en el que la búsqueda de patrones coincidentes se determina normalmente por el orden en que se introducen las cláusulas de Horn. Esto difiere con los sistemas expertos basados en reglas, en que suelen seguir el paradigma de Post, donde el orden en que se introducen las reglas no afecta la búsqueda.

PROLOG tiene muchas otras funciones y capacidades que no se mencionan en esta breve introducción. Desde el punto de vista de un sistema experto, el backtrack y la equiparación de patrones son muy útiles, como también lo es la naturaleza declarativa de PROLOG, porque la especificación de un programa es un programa ejecutable.

2.7 DIFICULTADES CON REDES SEMÁNTICAS

Aunque las redes semánticas pueden ser muy útiles en la representación del conocimiento, tienen limitantes, como la falta de estándares para asignación de nombres a los vínculos, que ya se explicó. Esto hace difícil entender para qué se diseñó realmente la red y si fue diseñada de una forma congruente. Un problema complementario al asignar nombre a los vínculos es el nombre de los nodos. Si un nodo se rotula "silla", ¿Representa

una silla específica
la clase de todas las sillas
el concepto de una silla
el nombre del cerro de la Silla

o algo más? Para que una red semántica represente **conocimiento definitivo** (es decir, conocimiento que puede definirse), deben definirse con rigor los nombres de vínculos y nodos. Claro que los mismos problemas pueden ocurrir en los lenguajes de programación.

Otro problema es la explosión combinatoria en la búsqueda de nodos, sobre todo si la respuesta a una consulta es negativa. Para que una consulta produzca un resultado negativo, tendrían que buscarse muchos o todos los vínculos de la red. Como se mostró en el problema del agente viajero del capítulo 1, el número de vínculos es el factorial del número de nodos menos uno, si están todos conectados. Aunque no todas las representaciones necesitarán este grado de conectividad, existe la posibilidad de que surja una explosión combinatoria.

Las redes semánticas fueron propuestas originalmente como modelos de la memoria asociativa humana, en la que un nodo tiene vínculos con otros y la recuperación de la información ocurre debido a una dispersión de la activación de los nodos. No obstante, el cerebro humano también debe disponer de otros mecanismos, porque no le toma mucho tiempo responder la consulta: ¿hay un equipo de fútbol en Plutón? Hay alrededor de 10^{10} neuronas en el cerebro humano y alrededor de 10^{15} vínculos. Si todo el conocimiento estuviera representado por una red semántica, tomaría demasiado tiempo responder consultas negativas como en el caso de la consulta del fútbol, porque todas las búsquedas incluyen 10^{15} vínculos.

Las redes semánticas son lógicamente inadecuadas porque no pueden definir el conocimiento de la forma en que lo hace la lógica. Una representación lógica puede especificar cierta silla, algunas sillas, todas las sillas, ninguna silla, etcétera, como se analizará más adelante en este capítulo. Otro problema es que las redes semánticas también son heurísticamente inadecuadas porque no hay forma de insertar en la red información heurística relacionada con la manera de explorarla de forma eficiente. Una regla **heurística** es una regla práctica que puede ayudar a encontrar una solución, pero que no está garantizada de la misma forma que un algoritmo. Las reglas heurísticas son muy importantes en la AI porque los problemas típicos en ello son tan difíciles que no existe una solución de algoritmo o, si la hay, no resulta eficiente para uso práctico. La herencia es la única estrategia estándar de control integrada en una red que puede ayudar, pero no todos los problemas pueden tener esta estructura.

Se han probado varios métodos para corregir los problemas inherentes a las redes semánticas; se han hecho mejoras lógicas y se han probado mejoras heurísticas al adjuntar procedimientos a los nodos, que serán ejecutados cuando el nodo se active. Sin embargo, los sistemas resultantes mejoraron poco su capacidad a costa de la fuerza de expresión natural de las redes semánticas. La conclusión de toda este esfuerzo es que, como cualquier herramienta, las redes semánticas deben utilizarse para aquello que hacen mejor, mostrar relaciones binarias, y no considerarse una herramienta universal.

2.8 ESQUEMAS

Una red semántica es un ejemplo de una **estructura de conocimiento superficial.** La superficialidad se encuentra presente porque todo el conocimiento de la red semántica se encuentra en los vínculos y nodos. El término *estructura del conocimiento* es análogo a una estructura de datos en la que más que sólo datos se representa una colección ordenada de conocimiento. Una estructura de conocimiento **profundo** tiene conocimiento causal que explica por qué ocurre algo. Por ejemplo, es posible construir un sistema experto médico con conocimiento superficial como sigue:

```
SI una persona tiene fiebre
ENTONCES toma una aspirina
```

Pero estos sistemas no conocen la bioquímica fundamental del cuerpo ni la razón por la que la fiebre disminuye con la aspirina. La regla podría haberse definido:

```
SI una persona tiene un mono rosa
ENTONCES toma un refrigerador
```

En otras palabras, el conocimiento del sistema experto es superficial porque está basado en la sintaxis más que en la semántica, donde dos palabras cualquiera podrían ser sustituidas por X y Y en la siguiente regla

```
SI una persona tiene un (X)
ENTONCES toma un (Y)
```

Observe que (X) y (Y) no son variables en esta regla, sino que representan dos palabras cualquiera. Los médicos tienen conocimiento causal porque han tomado muchos cursos y han adquirido experiencia en el tratamiento de personas enfermas. Si un tratamiento no está funcionando correctamente, los médicos pueden razonar acerca de él para encontrar una alternativa. En otras palabras, un especialista sabe cuándo romper las reglas.

Muchos tipos de conocimientos del mundo real no pueden representarse con la simple estructura de una red semántica. Se necesitan estructuras más complejas para representar mejor complejas estructuras del conocimiento. En la AI, el término *esquema* se utiliza para describir una estructura de conocimiento más compleja que la red semántica. Este término viene de la psicología, donde alude a la organización continua del conocimiento o la respuesta de una criatura debida a un estímulo. Esto es, a medida que las criaturas aprenden la relación entre una causa y su efecto, tratarán de repetir la causa si el efecto es agradable, o evitarla, si no lo es.

Por ejemplo, los actos de comer y beber son esquemas **sensomotores** agradables que involucran información de coordinación de los sentidos con los movimientos motores (músculos) necesarios para realizarlos. Una persona no tiene que pensar en este conocimiento tácito para saber cómo realizar estos actos y es muy difícil explicar exactamente cómo desciende al nivel de controlar los músculos. Un esquema aún más difícil de explicar es cómo andar en bicicleta. ¡Cómo tratar de explicar el sentido del equilibrio!

Otro tipo de esquema es el **esquema de concepto.** Por ejemplo, todos tienen un concepto de *animal.* La mayoría de las personas a las que se les pida que expliquen cómo es un animal probablemente lo describirían como algo que tiene cuatro patas y pelo. Por supuesto, el concepto de *animal* será diferente si la persona creció en una granja, en una ciudad, cerca de un río, etcétera. Sin embargo, todos tenemos **estereotipos** de conceptos en nuestras mentes. Aun-

que el término *estereotipo* quizá posea un significado negativo en el lenguaje coloquial, para la AI significa un ejemplo típico. Para mucha gente un estereotipo de un animal podría ser algo como un perro.

Un esquema conceptual es una abstracción en la cual se clasifican objetos específicos, de acuerdo con sus propiedades generales. Por ejemplo, si ve un pequeño objeto rojo, redondo con tallo verde, sobre un letrero que dice "Fruta artificial", probablemente lo reconocerá como una manzana artificial. El objeto tiene las propiedades que usted asocia con el esquema conceptual de manzana, salvo que sea una manzana real, por supuesto.

El esquema conceptual de una manzana real incluirá propiedades generales de las manzanas como tamaños, colores, sabores, usos, etcétera; pero no incluirá detalles acerca de dónde se cosechó exactamente, el medio por el que se transportó hasta el supermercado, el nombre de la persona que la colocó en el anaquel, etcétera. Estos detalles no son importantes para las propiedades que conforman su concepto abstracto de una manzana. Además, tome en consideración que un ciego quizá tenga un esquema de concepto muy diferente, en el que la textura sea más importante.

Al concentrarse en las propiedades generales de un objeto, resulta más fácil razonar acerca de ellos sin distraerse con detalles poco importantes. Habitualmente los esquemas tienen una estructura interna en sus nodos, mientras que las redes semánticas no, en ellas el rótulo es todo el conocimiento acerca del nodo. Una red semántica es como un estructura de datos de la ciencia de la computación, en que la clave de búsqueda corresponde también a los datos almacenados en el nodo. Un esquema es como una estructura de datos donde los nodos contienen registros. Cada registro puede incluir datos, registros o apuntadores a otros nodos.

2.9 MARCOS

Un tipo de esquema que se ha utilizado en muchas aplicaciones de la AI es el **marco** (Minsky 75). Otro es el **guión**, que es en esencia una secuencia de marcos ordenada temporalmente (Schank 77). Propuestos como un método para comprender la visión, el lenguaje natural y otras áreas, los marcos proporcionan una estructura conveniente para representar objetos que son comunes a una situación dada, como los estereotipos. En particular, los marcos son útiles para simular conocimiento de sentido común, que es un área que le resulta muy difícil dominar a las computadoras. Las redes semánticas son básicamente representaciones bidimensionales del conocimiento; los marcos agregan una tercera dimensión para permitir que los nodos tengan estructuras, que pueden ser valores simples u otros marcos.

La característica básica de un marco, es que representa conocimiento relacionado con un tema concreto que cuenta con mucho conocimiento predeterminado. Un sistema de marcos sería una buena elección para describir un dispositivo mecánico como un automóvil; porque sus componentes, como el motor, la carrocería, los frenos, etcétera, se relacionarían entre sí para dar una visión general de sus vínculos. Sería posible obtener aún más detalles de los componentes si se examina la estructura de los marcos. Aunque las marcas individuales de automóviles variarán, muchos autos tienen características comunes como llantas, motor, carrocería, transmisión, etcétera. El marco difiere de la red semántica, que suele utilizarse para una amplia representación del conocimiento. Del mismo modo que con las redes semánticas, no hay estándares para definir los sistemas basados en marcos. Se han diseñado varios lenguajes específicos para los marcos, como FRL, SRL, KRL, KEE, HP-RL y funciones para acrecentarlos como LISP tales como LOOPS y FLAVORS (Finin 86).

Un marco es análogo a la estructura de registro en un lenguaje de alto nivel como Pascal o un átomo con su lista de propiedades en LISP. Correspondiendo a los campos y los valores de un registro son las **ranuras** y sus **rellenos** de un marco. Un marco es básicamente un grupo de ranuras y rellenos que definen un objeto estereotípico. En la figura 2.8 se muestra un ejemplo

de un marco para un automóvil. Desde el punto de vista de OAV, el automóvil es un objeto, el nombre de la ranura es el atributo y el relleno es el valor.

Ranuras	Rellenos
fabricante	General Motors
modelo	Chevrolet Caprice
año	1979
transmisión	automática
motor	a gasolina
ruedas	4
color	azul

Figura 2.8 Marco de automóvil

Casi ningún marco es tan simple como el que se muestra en la figura 2.8. La utilidad de los marcos recae en los sistemas jerárquicos de marcos y en la herencia. Al utilizar marcos en las ranuras de relleno y en la herencia, pueden construirse sistemas de representación del conocimiento muy versátiles. En particular, los sistemas expertos basados en marcos son muy útiles para representar al conocimiento causal, porque su información está organizada en causas y efectos. Por el contrario, los sistemas expertos basados en reglas dependen generalmente del conocimiento no organizado, que no es causal.

Algunas herramientas basadas en marcos como KEE permiten un amplio rango de elementos para almacenar en las ranuras. Éstas pueden contener reglas, imágenes, comentarios, depuración de información, preguntas a los usuarios, hipótesis relacionadas con una situación u otros marcos.

Los marcos se diseñan generalmente para representar conocimiento genérico o específico. En la figura 2.9 se ilustra un marco genérico para el concepto de propiedad.

Espacios	Rellenos
nombre	propiedad
especialización_de	un_tipo_de objeto
tipos	(automóvil, bote, casa)
	si-se-agrega: Procedimiento AGREGAR_PROPIEDAD
propietario	predeterminado: gobierno
	si-se-necesita: Procedimiento BUSCAR_PROPIETARIO
ubicación	(casa, trabajo, móvil)
estado	(falta, malo, bueno)
bajo_garantía	(si, no)

Figura 2.9 Marco genérico para propiedad

Los rellenos pueden ser valores, como una propiedad en la ranura del nombre, o un rango de valores, como en la ranura de tipos. Las ranuras también pueden contener procedimientos adjuntos, llamados ***anexos de procedimiento***. Por lo general son de tres tipos: los de tipo **si-es-necesario** son procedimientos que se ejecutarán cuando se necesita un valor de relleno pero no hay ninguno presente o el valor **predeterminado** no es adecuado. Los valores predeterminados son de importancia fundamental en los marcos, porque son modelos de algunos aspectos del cerebro. Corresponden a las expectativas de una situación que se construye a partir de la experiencia. Cuando encontramos una nueva situación, se modifica el marco más cercano para ayudarnos a ajustar la situación, así las personas no empiezan desde cero en cada situación nueva; solamente, se modifican las opciones predeterminadas o los otros rellenos. A menudo se utilizan las opciones predeterminadas para representar el conocimiento de

sentido común. Éste puede considerarse como el conocimiento que se conoce de manera general. Utilizamos el sentido común cuando no se dispone de más conocimiento específico para la situación.

El tipo **si-se-agrega** se ejecuta cuando se agrega un valor de un ranura. En la ranura de tipos, el procedimiento si-se-agrega ejecuta un procedimiento llamado AGREGAR-PROPIEDAD para añadir un nuevo tipo de propiedad, si es necesario. Por ejemplo, este procedimiento se ejecutaría para accesorios, TV, estéreo, etcétera, porque la ranura de tipos no contiene estos valores.

Un tipo **si-se-elimina** se ejecuta cada vez que se elimina un valor de una ranura. Este tipo de procedimiento se ejecuta cuando un valor es obsoleto.

Los rellenos de ranura también pueden contener relaciones, como en las ranuras especialización_de. Las relaciones un-tipo-de y es-un se utilizaron en las figura 2.9, 2.10 y 2.11 para mostrar la manera en que estos marcos están relacionados jerárquicamente. Los marcos de las figuras 2.9 y 2.10 son genéricos mientras que el de la figura 2.11 es un marco específico porque se trata de un caso del marco del automóvil. Supongamos aquí que las relaciones un-tipo-de son genéricas y las es-un son específicas.

Ranuras	Rellenos
nombre	automóvil
especialización_de	propiedad un-tipo-de
tipos	(sedan, deportivo, convertible)
fabricante	(GM, Ford, Chrysler)
ubicación	móvil
ruedas	4
transmisión	(manual, automática)
motor	(gasolina, diesel)

Figura 2.10 Marco de automóvil: un submarco genérico de propiedad

Los sistemas de marcos están diseñados para que los marcos más genéricos se encuentren en la parte superior de la jerarquía. Se supone que los marcos pueden personalizarse al modificar los casos predeterminados y crear marcos más específicos. Los marcos tratan de modelar objetos del mundo real utilizando el conocimiento genérico para la mayor parte de los atributos de un objeto, y el conocimiento específico para casos especiales. Por ejemplo, generalmente consideramos a las aves como criaturas que pueden volar, pero ciertos tipos de aves, como los pingüinos y los avestruces, no pueden hacerlo. Estos tipos representan clases más específicas de aves y, en la jerarquía de un marco, sus características se encontrarían en niveles inferiores a los de aves como los canarios o petirrojos. En otras palabras, la parte superior de la jerarquía del marco para aves especifica cosas que son más ciertas para todas las aves y los niveles inferiores reflejarían los límites confusos entre los objetos reales. Al objeto que tiene todas las características comunes se le llama ***prototipo***, que significa literalmente el primer tipo.

Ranuras	Rellenos
nombre	automóvil de John
especialización_de	es_un automóvil
fabricante	GM
propietario	Juan Pérez
transmisión	automática
motor	a gasolina
estado	bueno
bajo_garantía	sí

Figura 2.11 Ejemplo de un marco de automóvil

Los marcos también pueden clasificarse por sus aplicaciones. Un **marco situacional** contiene conocimiento sobre lo que se espera en una situación determinada, como una fiesta de cumpleaños. Un **marco de acción** contiene ranuras que especifican las acciones que se realizarán en una situación determinada. Es decir, los rellenos son procedimientos para realizar algunas acciones, como eliminar una parte defectuosa de un cinturón de seguridad. Un marco de acción representa conocimiento del procedimiento. La combinación de marcos de situación y de acción puede utilizarse para describir relaciones de causa y efecto, en la forma de **marcos de conocimiento causal**.

Se han construido sistemas de marcos muy sofisticados para diversas tareas. Dos de los sistemas más impresionantes, han demostrado la capacidad de los marcos para descubrir creativamente conceptos matemáticos (Lenat 77) y describir comprensión matemática en dominios como el álgebra lineal (Rissland 78). En estos sistemas, las ranuras se usaron como generadores de tareas, tales como respuestas a preguntas. El sistema AM de Lenat haría conjeturas de interesantes conceptos novedosos y los exploraría.

2.10 DIFICULTADES CON LOS MARCOS

Los marcos se concibieron originalmente como un paradigma para representar conocimiento estereotipado. La característica más importante de un estereotipo es que cuenta con funciones bien definidas, de modo que muchas de sus ranuras tienen valores predeterminados. Los conceptos matemáticos son buenos ejemplos de estereotipos bien adecuados a los marcos. El paradigma de los marcos tiene un aspecto intuitivo, porque la representación organizada de marcos de conocimiento suele ser más fácil de comprender que cualquier sistema lógico o de producción con muchas reglas (Jackson 86). Sin embargo, han aparecido problemas importantes en los sistemas de marcos que parecen una modificación o cancelación de las ranuras sin restricciones. El ejemplo clásico de este problema se ilustra en la figura 2.12 con un marco que describe elefantes.

nombre	elefante
especialización_de	un-tipo-de mamífero
color	gris
patas	4
tronco	cilíndrico

Figura 2.12 Marco de elefante

A primera vista, el marco para el elefante parece una descripción genérica razonable de los elefantes. Sin embargo, si suponemos que existe un elefante de tres patas llamado Clyde, que ha perdido una pata debido a un accidente de cacería o que simplemente ha mentido sobre el hecho de perder una pata para que su nombre apareciera en este libro. Los importante es que el marco de elefante asegura que un elefante tiene cuatro patas, no tres. Por tanto, no podemos considerar que el marco para el elefante sea una definición de elefante.

Por supuesto, el marco puede modificarse para permitir elefantes con tres patas o menos. Sin embargo, esto no proporciona una buena definición, por que pueden surgir problemas adicionales con otras ranuras. Suponiendo que Clyde adquiere ictericia y su piel se vuelve amarilla, ¿deja de ser un elefante?

Una alternativa para observar un marco como una definición, es considerarlo como descripción de un elefante típico. Sin embargo, esto conduce a otros problemas debido a la herencia. Observe que el marco indica que un elefante es un-tipo-de mamífero. Como estamos

interpretando los marcos como típicos, nuestro sistema de marco indica que un elefante típico es un mamífero típico; aunque un elefante es un mamífero, probablemente no sea un mamífero típico. Sobre la base de la cantidad, los seres humanos, las vacas, las ovejas o las ratas probablemente sean más representativos de los mamíferos típicos.

Casi ninguno de los sistemas de marcos proporciona una manera de definir ranuras inalterables. Cualquier ranura puede cambiarse, de modo que pueden modificarse o cancelarse las propiedades que un marco hereda hacia cualquier lugar de la jerarquía. Esto significa que cada marco es, en realidad, un marco primitivo y como no hay garantía de que las propiedades sean comunes, cada marco crea sus propias reglas. Nada es realmente cierto en un sistema sin restricciones como éste. Por lo tanto, es imposible hacer frases universales como las creadas en la definición de un elefante. Además, a partir de definiciones más simples como las del elefante, es imposibles construir de manera confiable objetos compuestos, como en el caso del elefante con tres patas. Los mismos tipos de problema se aplican a las redes semánticas con la herencia. Si las propiedades de cualquier nodo pueden cambiarse, entonces nada es cierto.

2.11 LÓGICA Y CONJUNTOS

Además de reglas, marcos y redes semánticas, el conocimiento puede representarse mediante los símbolos de la **lógica**, que es el estudio de las reglas del razonamiento exacto. Una parte importante del razonamiento es la inferencia de conclusiones a partir de premisas. La aplicación de computadoras para realizar razonamientos ha tenido como resultado la **programación lógica** y el desarrollo de lenguajes basados en la lógica como PROLOG. La lógica también tiene importancia primordial en los sistemas expertos, donde el mecanismo de inferencia razona a partir de los hechos para extraer conclusiones. En realidad, un término descriptivo para la programación lógica y los sistemas expertos es **sistemas de razonamiento automatizados**.

En el siglo IV antes de nuestra era, el filósofo griego Aristóteles desarrolló la lógica formal. Esta lógica aristotélica se basaba en los **silogismos**, de los que inventó 14 tipos, cinco más se inventaron en la Edad Media. Los silogismos tienen dos **premisas** y una **conclusión**, que se infiere a partir de aquéllas. El siguiente es el ejemplo clásico de un silogismo:

```
Premisa:        Todos los hombres son mortales
Premisa:        Sócrates es un hombre
Conclusión:     Sócrates es mortal
```

En un silogismo, la **premisa** proporciona la evidencia a partir de la cual surgirá la conclusión. Un silogismo es una manera de representar el conocimiento. Otra manera es un **diagrama de Venn**, como el que se muestra en la figura 2.13.

El círculo exterior representa a todas las criaturas mortales. El círculo interno representa a un hombre dibujado por completo en el interior del círculo mortal, para indicar que todos los hombres son mortales; como Sócrates es un hombre, el círculo que lo representa se dibuja por completo dentro del segundo círculo. Hablando estrictamente, el círculo que representa a Sócrates debe ser un punto, porque un círculo implica una clase, pero por razones de legibilidad utilizaremos círculos para todo. La conclusión de que Sócrates es mortal es una consecuencia del hecho de que su círculo se encuentra dentro del círculo de las criaturas mortales y, por tanto, debe ser mortal.

En términos matemáticos, un círculo del diagrama de Venn representa un conjunto, que es una colección de objetos. A los objetos de un conjunto se les llama **elementos**. Algunos ejemplos de conjuntos son

```
A = (1,3,5)
B = (1,2,3,4,5)
C = (0,2,4,...)
D = (...,-4,-2,0,2,4,...)
E = (aviones,globos,dirigibles,jets)
F = (aviones,globos)
```

donde los puntos suspensivos, ..., llamados elipses, indican que los términos continúan indefinidamente.

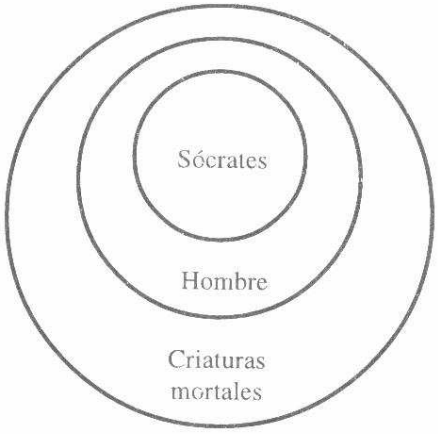


Figura 2.13 Diagrama de Venn

La letra griega épsilon, $\in$, indica que un elemento es un miembro de un conjunto. Por ejemplo, $1 \in A$ significa que el número 1 es un elemento del conjunto A previamente definido. Si un elemento no es miembro de un conjunto, se utiliza el símbolo $\notin$, como en $2 \notin A$.

Si dos conjuntos arbitrarios, como X y Y se definen para que cada elemento de X sea un elemento de Y, entonces X es un **subconjunto** de Y y se escribe en la forma matemática X Y o $Y \supset X$. A partir de la definición de un subconjunto, se desprende que cada conjunto es un subconjunto de sí mismo. A un subconjunto que no es el propio conjunto se le llama **subconjunto propio**. Por ejemplo, el conjunto X definido previamente es un subconjunto propio de Y. Al analizar los conjuntos, es útil considerarlos como subconjuntos de un **conjunto universal**. Éste cambia a medida que cambia el tema de análisis. En la figura 2.14 se ilustra un subconjunto formado como la **intersección** de dos conjuntos en el universo de todos los automóviles. Si se trata de números, el universo sería todos los números. El universo se dibuja como un rectángulo que rodea a sus subconjuntos. Los conjuntos universales no se usan sólo por conveniencia. El uso indiscriminado de condiciones para definir conjuntos puede dar como resultado paradojas lógicas.

Si dejamos que A sea el conjunto de todos los automóviles azules, B el conjunto de todos los automóviles con transmisión manual y C el conjunto de todos los automóviles azules con transmisión manual, escribiríamos:

$$C = A \cap B$$

En que el símbolo $\cap$ representa la intersección de los conjuntos. Otra manera de escribir esto es a partir de elementos x, como se muestra a continuación

$$C = \{x \in U \mid (x \in A) \wedge (x \in B)\}$$

donde

U representa el conjunto universal
| se lee como "tal que" (a veces se utilizan dos puntos, :, en lugar de |)
$\wedge$ es el **Y lógico**

La expresión para C se lee como C contiene aquellos elementos x que se encuentran en el universo tal que x es un elemento de A y x es un elemento de B. El Y lógico proviene del álgebra booleana. Una expresión integrada por dos operandos conectados por el operador lógico Y, es verdadero si y sólo si ambos operandos son verdaderos. Si A y B no tienen elementos en común, entonces $A \cap B = \varnothing$, donde la letra griega fi, $\varnothing$, representa el **conjunto vacío** o **conjunto nulo**, { }, que es el que no tiene elementos. aunque veces se utiliza la letra griega lambda, $\wedge$, para representarlo. A veces se utilizan otras notaciones para el conjunto universal: el número 1 (y el número 0 para el conjunto vacío). Aunque el conjunto vacío no tiene elementos, aún es un conjunto. Una analogía es un restaurante con un conjunto de clientes. Si nadie viene, el conjunto de clientes está vacío.

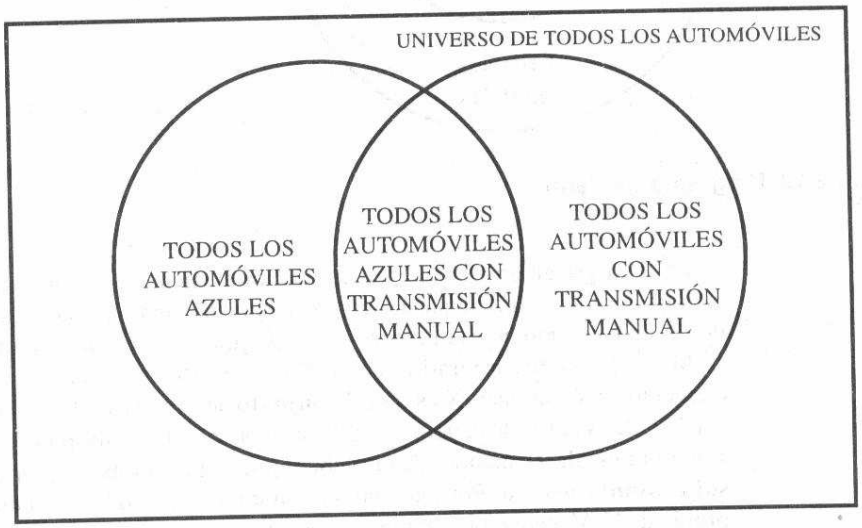


Figura 2.14 Intersección de conjuntos

Otra operación de conjunto es la **unión**, que es el conjunto de todos los elementos en A o B. Se representa de la siguiente forma

$$A \cup B = \{x \in U \mid (x \in A) \vee (x \in B)$$

donde

$\cup$ es el operador de conjunto para **unión**
$\vee$ es el operador **lógico O.**

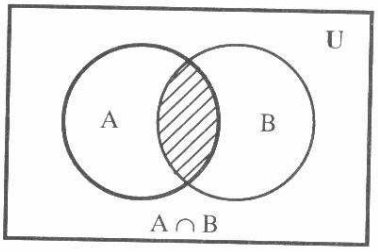
El **complemento** del conjunto A es el conjunto de todos los elementos que no están en A.

$$A' = \{x \in U \mid \sim (x \in A)\}$$

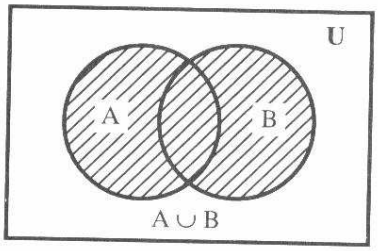
donde

el apóstrofe, ′, significa complemento de un conjunto
la tilde, ~, es el operador **NO lógico** es decir negación

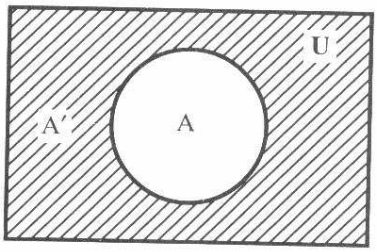
Los diagramas de Venn para estas operaciones básicas se muestran en la figura 2.15.



(a) INTERSECCIÓN DE CONJUNTOS



(b) UNIÓN DE CONJUNTOS



(c) COMPLEMENTO DE UN CONJUNTO

Figura 2.15 Operaciones básicas con conjuntos

2.12 LÓGICA DE PROPOSICIONES

La más antigua y uno de los tipos más simples de **lógica formal** es el silogismo. El término *formal* significa que la lógica se relaciona con la forma de las frases lógicas más que con su

significado. En otras palabras, tiene que ver con la sintaxis más que con la semántica de las frases. Aunque el término *lógica formal* puede sonar intimidatorio, no es más difícil que el álgebra. En realidad, el álgebra es realmente la lógica formal para números. Por ejemplo, supongamos le pidieron resolver el siguiente problema: una escuela tiene 25 computadoras con un total de 60 tarjetas de memoria. Algunas de ellas tienen dos y algunas cuatro tarjetas. ¿Cuántas computadoras hay de cada tipo? La solución puede escribirse algebraicamente como sigue:

```
25 = X + Y
60 = 2X + 4Y
```

que puede resolverse fácilmente: X = 20 y Y = 5.

Ahora considere este problema. Hay 25 animales con un total de 60 patas en un corral. Algunos tienen dos patas y otros cuatro. (Nota: Clyde, nuestro elefante de tres patas está en África, no en el corral). ¿Cuántos animales hay de cada tipo? Como puede ver, las mismas ecuaciones algebraicas se aplican si hablamos de computadoras, animales o cualquier otra cosa. De la misma forma que las ecuaciones algebraicas nos permiten concentrarnos en la manipulación matemática de símbolos sin considerar lo que representan, la lógica formal nos permite concentrarnos en el razonamiento sin confundirnos con los objetos sobre los que razonamos.

Como un ejemplo de lógica formal, tomaremos en cuenta el siguiente silogismo con las palabras sin sentido *squeeg* y *moof*.

```
Premisa:        Todos los squeegs son moofs
Premisa:        John es un squeeg
Conclusión:     John es un moof
```

Aunque las palabras *squeeg* y *moof* no tienen sentido ni significado, la *forma* de este argumento sigue siendo correcta. Es decir, el argumento es **válido** sin importar cuáles palabras se usen, porque el silogismo tiene una forma válida. En realidad, cualquier silogismo con la forma

```
Premisa:        Todas las X son Y
Premisa:        Z es una X
Conclusión:     Z es una Y
```

es válido sin importar qué se sustituye con X, Y y Z. Este ejemplo ilustra esos significados, sin importar la lógica formal. Sólo es importante la forma o apariencia. Este concepto de separar la forma del significado o semántica, es lo que convierte a la lógica en una herramienta poderosa. También hace que la validez de un argumento pueda considerarse objetivamente, sin el prejuicio causado por la semántica. Esto se parece al álgebra en que la corrección de expresiones como X + X = 2X se mantiene sin importar si X representa un entero, manzanas, aviones o cualquier cosa.

Los silogismos aristotélicos fueron el fundamento de la lógica hasta 1847, cuando el matemático inglés George Boole publicó la primer obra que describía la **lógica simbólica**. Aunque Leibnitz había desarrollado su propia versión en el siglo XVII, nunca llegó a ser de uso general. Uno de los nuevos conceptos que Boole propuso, fue una modificación de la idea aristotélica de que el sujeto tiene existencia, llamada ***importancia existencial.*** De acuerdo con la concepción clásica de Aristóteles, una proposición como "todas las sirenas nadan bien" no se podría utilizar como premisa o conclusión porque las sirenas no existen. La idea de Boole, ahora llamada *concepción moderna*, hace más relajada esta restricción; su importancia se debe a

que ahora se puede razonar acerca de clases vacías. Por ejemplo, una proposición como "todos los discos que fallan son baratos" no podría utilizarse en un silogismo aristotélico, a menos que hubiera por lo menos un disco que falló.

Otra contribución de Boole fue la definición de un conjunto de **axiomas** integrado por símbolos que representan tanto objetos como clases, y operaciones algebraicas para manipular los símbolos. Son las definiciones fundamentales a partir de las cuales se construyen sistemas lógicos como las matemáticas y la propia lógica. Utilizando únicamente axiomas se pueden elaborar teoremas. Un teorema es una afirmación que se puede probar mostrando cómo se deriva de los axiomas. Entre 1910 y 1913, Whitehead y Russell publicaron su monumental obra de tres volúmenes y 2000 páginas, *Principios matemáticos,* que mostró una lógica formal como base de las matemáticas. Fue recibido como un gran hito porque, aparentemente, pone a las matemáticas sobre bases firmes en lugar de apelar a la intuición, eliminando totalmente el significado de la aritmética y concentrándose, en cambio, en las formas y su manipulación. Debido a esto, las matemáticas se han descrito como una colección de marcas sin sentido en el papel. Sin embargo, en 1931 Gödel probó que los sistemas formales basados en axiomas no siempre resultaban internamente congruentes y, por tanto, libres de contradicciones.

La lógica de proposiciones, también llamada ***cálculo de proposiciones,*** es una lógica simbólica para la manipulación de proposiciones. En particular, se ocupa de la manipulación de las **variables lógicas** que representan proposiciones. Aunque muchas personas consideran al cálculo dentro de la perspectiva que inventaron Newton y Leibnitz, la palabra tiene un significado más general; procede de la palabra latina *calculus,* pequeña piedra utilizada para realizar cálculos. Su significado general es un sistema especial para la manipulación de símbolos. Otros términos utilizados para la lógica de proposiciones son **cálculo de afirmaciones** y **cálculo de frases.** Las afirmaciones se clasifican generalmente en cuatro tipos, como se ilustra en la tabla 2.2.

Tipo	Ejemplo
Imperativo	¡Qué te digo!
Interrogativo	¿Qué es eso?
Admirativo	¡Eso es grandioso!
Declarativo	Un cuadrado tiene cuatro lados iguales.

Tabla 2.2 Tipos de afirmaciones

La lógica de proposiciones tiene que ver con el subconjunto de oraciones declarativas que pueden clasificarse como verdaderas o falsas. Una afirmación como "Un cuadrado tiene cuatro lados iguales", tiene un valor de verdad definido como verdadero, mientras que la afirmación "George Washington fue el segundo presidente de Estados Unidos" tiene un valor de verdad falso. A una afirmación en la que es posible determinar su valor de verdad, se le llama ***frase*** o ***proposición.*** A una frase también se le llama ***afirmación cerrada,*** porque su valor de verdad no está abierto a cuestionamientos.

Si en el prefacio de este libro se afirma "todo en estas páginas es una mentira", esa frase no puede clasificarse como verdadera o falsa. Si es verdadera, entonces dije la verdad en el prefacio, cosa que no puedo hacer; si no es verdad, entonces cada palabra escrita debe ser verdadera, por lo tanto yo miento, lo que tampoco puede ser verdad. Frases de este tipo son conocidas como la Paradoja de Liar. A las afirmaciones que no se puede responder de manera absoluta se les llama **afirmaciones abiertas.** La afirmación "Las espinacas son deliciosas" es también una frase abierta, porque es cierto para unas personas y no lo es para otras. "Él es alto" es una afirmación abierta porque tiene una variable "Él". No es posible asignar valores

verdaderos para abrir afirmaciones hasta que conozcamos a la persona o el caso específico al que se refiere la variable. Otra dificultad con esta afirmación es el significado de la palabra "alto", pues lo que parece alto para algunos no lo parece para otros. Aunque la ambigüedad de "alto" no se puede manejar en el cálculo de proposiciones o de afirmaciones, puede resolverse fácilmente con la lógica de confusión, descrita en el capítulo 5.

Una **afirmación compuesta** está formada por el uso de conectores lógicos o frases individuales. Los conectores comunes de la lógica se muestran en la tabla 2.3.

Conector	Significado
$\wedge$	Y; **conjunción**
$\vee$	O; **disyunción**
~	NO; **negación**
$\rightarrow$	si...entonces; **condicional**
$\leftrightarrow$	si y sólo si; **doble condicional**

Tabla 2.3 Algunos conectores lógicos

Hablando estrictamente, la negación no es un conector porque es una operación unitaria que aplica al siguiente operando, de modo que en realidad no conecta nada. Tiene mayor prioridad que los otros operadores, así que no es necesario colocarla entre paréntesis. Una frase como $\sim p \wedge q$ significa lo mismo que $(\sim p) \wedge q$.

El condicional es análogo a la flecha de las reglas de producción en las que se expresa como una forma si...entonces. Por ejemplo

```
si está lloviendo entonces lleva una sombrilla
```

se puede poner en la forma

$$p \rightarrow q$$

donde

```
p = está lloviendo
q = lleva una sombrilla
```

Algunas veces se utiliza $\supset$ por $\rightarrow$. Otro término para el condicional es **implicación material.**

El condicional doble, $p \leftrightarrow q$, es equivalente a

$$(p \rightarrow q) \wedge (q \rightarrow p)$$

y es verdadero sólo cuando p y q tienen los mismos valores de verdad. Esto es, $p \leftrightarrow q$ sólo es verdadero cuando ambos son verdaderos o cuando ambos son falsos. El condicional doble tiene los siguientes significados:

```
p si y sólo si q
q si y sólo si p
si p entonces q, si q entonces p
```

Una **tautología** es una frase compuesta que siempre es verdadera, sin importar si sus frases individuales son verdaderas o falsas. Una **contradicción** es una frase compuesta que siempre es falsa. Una frase **condicional** no es una tautología ni una contradicción. A las tautologías y a las contradicciones se les denomina analíticamente verdaderas o analíticamente falsas, respectivamente, porque sus valores de verdad pueden determinarse a partir de sus simples formas. Por ejemplo, la tabla de verdad de $p \vee \sim p$ muestra que es una tautología, mientras que $p \wedge \sim p$ es una contradicción.

Si un condicional es también una tautología entonces se le llama ***implicación*** y tiene el símbolo $\Rightarrow$ en lugar de $\rightarrow$. A un condicional doble que también es una tautología se le llama ***equivalencia lógica*** o ***equivalencia de material*** y se simboliza con $\Leftrightarrow$ o $\equiv$. Dos frases lógicamente equivalentes siempre tienen los mismos valores de verdad. Por ejemplo, $p \equiv \sim \sim p$.

Desafortunadamente, ésta no es la única definición posible para una implicación, ya que hay 16 posibles tablas de verdad para dos variables que pueden tomar valores de verdadero o falso. En efecto, en un artículo se han revisado 11 definiciones diferentes del operador de implicación utilizado en los sistemas expertos (Whalen 85).

El condicional no significa exactamente lo mismo que SI ... ENTONCES en un lenguaje de procedimientos o un sistema experto basado en reglas. En los sistemas de procedimientos y experto, SI ... ENTONCES significa ejecutar las acciones siguiendo el ENTONCES *si* las condiciones del SI son verdaderas. En la lógica, el condicional se define por su tabla de verdad y su significado puede traducirse al lenguaje natural de varias maneras. Por ejemplo, si

$$p \rightarrow q$$

donde p y q son cualquier afirmación, esto puede traducirse como

```
p implica q
si p entonces q
p, sólo si q
p es suficiente para q
q si p
q es necesario para p
```

Como ejemplo, hagamos que p represente "usted tiene 18 años o más" y que q represente "usted puede votar". La $p \rightarrow q$ condicional puede significar

```
usted tiene 18 años o más implica que puede votar
si tiene 18 años o más entonces puede votar
usted tiene 18 años o más, sólo si puede votar
usted tiene 18 años o más, es suficiente para votar
usted puede votar si tiene 18 años o más
usted puede votar si es necesario porque tiene 18 años o más
```

En algunos casos, es necesario un cambio en el orden de las palabras para que sean frases gramaticalmente correctas. El último ejemplo dice que si no ocurre q entonces p tampoco. Se expresa de manera apropiada como "La posibilidad de votar requiere que usted tenga 18 años o más".

En la tabla 2.4 se muestran los valores para los conectores binarios lógicos. Son binarios porque requieren dos operandos. El conector de negación, ~, es un operador unitario en el operando que le sigue, como se muestra en la tabla 2.5.

Un conjunto de conectores lógicos es **adecuado** si cada función de verdad puede representarse utilizando sólo la conectividad del conjunto adecuado. Ejemplos de conjuntos adecuados son $\{\sim, \wedge, \vee\}$, $\{\sim, \wedge\}$, $\{\sim, \vee\}$ y $\{\sim, \rightarrow\}$.

A un conjunto de un solo elemento se le llama **unitario**. Hay dos conjuntos de unitarios adecuados. Son NO-O y NO-Y. El conjunto NO-O es $\{ \downarrow \}$ y el conjunto NO-Y es $\{ | \}$. Al operador " | " se le llama **negación alternativa**. Se utiliza para negar que p y q son verdaderos. Es decir p | q afirma que por lo menos una de las afirmaciones, p o q, es verdadera. El **operador de negación de unión**, "$\downarrow$" niega que p o q sean verdaderos. Es decir, p $\downarrow$ q afirma que tanto p como q son falsos.

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
V	V	V	V	V	V
V	F	F	V	F	F
F	V	F	V	V	F
F	F	F	F	V	V

Tabla 2.4 Tabla de verdad de los conectores lógicos binarios

p	**~p**
V	F
F	V

Tabla 2.5 Tabla de verdad de la negación

2.13 LA LÓGICA DE PREDICADO DE PRIMER ORDEN

Aunque la lógica de proposiciones es útil, tiene limitaciones. El principal problema es que la lógica de proposiciones sólo puede tratar con afirmaciones completas. Es decir, no puede examinar la estructura interna de una afirmación. La lógica de proposiciones ni siquiera puede probar la validez de un silogismo como

```
Todos los seres humanos son mortales
Todas las mujeres son seres humanos
Por tanto, todas las mujeres son mortales
```

La **lógica de predicado** se desarrolló con el fin de analizar casos más generales. Su forma más simple es la lógica de predicados de **primer orden**, la base de lenguajes de programación lógicos como PROLOG. En esta sección utilizaremos el término *lógica de predicado* para aludir a la lógica de predicado de primer orden. La lógica de proposiciones es un subconjunto de la lógica de predicado.

La lógica de predicado se relaciona con la estructura interna de las afirmaciones, sobre todo, se relaciona con el uso de palabras especiales llamadas ***cuantificadores***, como "todo", "algo" y "no". Estas palabras son muy importantes porque cuantifican explícitamente otras palabras y hacen más exactas las afirmaciones. Todos los cuantificadores se relacionan con "cuánto" y, por tanto, permiten un alcance más amplio de la expresión que la lógica de proposiciones.

2.14 EL CUANTIFICADOR UNIVERSAL

Una afirmación cuantificada universalmente tiene el mismo valor de verdad para todos los reemplazos en el mismo dominio. La **variable de dominio** el **cuantificador universal** se representa con el símbolo ∀ seguido por uno o más argumentos. El símbolo ∀ se interpreta como "para cada uno" o "para todos". Por ejemplo, en el dominio de los números

```
(∀ x) (x + x = 2x)
```

establece que para cada x (donde x es un número), la afirmación x + x = 2x es verdadera. Si representamos esta frase con el símbolo p, entonces puede expresarse aún más brevemente como:

```
(∀ x) (p)
```

Como otro ejemplo, dejemos que p represente la afirmación "todos los perros son animales", como en

```
(∀ x) (p) ≡ (∀ x) (si x es un perro → x es un animal)
```

La afirmación opuesta es "los perros no son animales" y se escribe como

```
(∀ x) (si x es un perro → ~x es un animal)
```

También puede leerse como

```
Cada perro no es un animal
Todos los perros no son animales
```

Otro ejemplo sería "todos los triángulos son polígonos", y se escribe como sigue:

```
(∀ x) (x es un triángulo → x es un polígono)
```

y se lee "para todo x, si x es un triángulo, entonces x es un polígono". Una manera más corta de escribir afirmaciones lógicas que incluyen predicados es utilizando las **funciones de predicado** para describir las propiedades del sujeto. La afirmación lógica anterior también puede escribirse

```
(∀ x) (triángulo(x) → polígono(x))
```

Las funciones de predicado suelen escribirse en notación más breve utilizando mayúsculas para representar los predicados. Por ejemplo, hagamos que T = triángulo y P = polígono. Entonces la afirmación de triángulo puede escribirse de manera aún más breve como

```
(∀ x) (T(x) → P(x)) o (∀ y) (T(y) → P(y))
```

Observe que cualquier variable puede utilizarse en lugar de las variables de modelo x y y. Un ejemplo más, hagamos que H sea la función de predicado de *ser humano* y M sea la función de *mortal*. Entonces la afirmación de que todos los seres humanos son mortales puede escribirse como:

```
(∀ x) (H(x) → M(x))
```

y se lee que para toda x, si x es ser humano, entonces x es mortal. Esta afirmación de lógica de predicado también puede representarse como una red semántica, tal como se muestra en la figura 2.16. También puede expresarse con reglas como

```
SI x es un ser humano
ENTONCES x es mortal
```

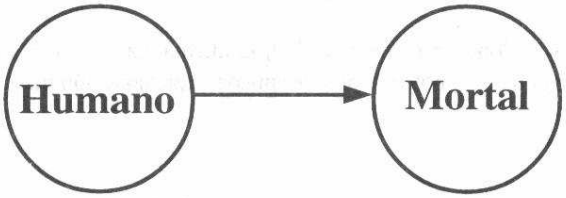


Figura 2.16 Representación de la red semántica de una afirmación de lógica de predicado

El cuantificador universal también puede interpretarse como un conjunto de predicados acerca de casos. Por ejemplo, un perro llamado Sparkler es un caso particular de la clase perros y puede escribirse como

```
Perro(Sparkler)
```

Donde Perro es la función de predicado y Sparkler es un caso.

Una frase de lógica de predicado como

$$(\forall\ x)\ P(x)$$

puede interpretarse casos de a_1 como

$$P(a_1) \wedge P(a_2) \wedge P(a_3) \wedge \ldots P(a_N)$$

donde los puntos suspensivos indican que el predicado se extiende a todos los miembros de la clase. Esta afirmación indica que el predicado se aplica a todos los casos de la clase.

Es posible utilizar cuantificadores múltiples. Por ejemplo, la ley conmutativa de la adición requiere dos cuantificadores, como en

$$(\forall\ x)\ (\forall\ y)\ (x + y = y + x)$$

que establece que "para todo x y para todo y, la suma de x y y es igual a la suma de y y x".

2.15 EL CUANTIFICADOR EXISTENCIAL

Otro tipo de cuantificador es el **cuantificador existencial**. Éste describe una afirmación como verdadera por lo menos para un miembro del dominio. Esto es una forma restringida del cuantificador universal que indica que una instrucción es verdadera para todos los miembros del dominio. El cuantificador existencial se escribe como $\exists$ seguido por uno o más archivos. Por ejemplo

```
(∃ x) (x · x = 1)
(∃ x) (elefante(x) ∧ nombre(Clyde))
```

La primera afirmación establece que hay algún x cuyo producto al multiplicarse por sí mismo es igual a 1. La segunda instrucción indica que hay algún elefante con el nombre de Clyde.

El cuantificador existencial puede leerse de varias maneras, como

```
existe
por lo menos un
para algún
hay un
algún
```

Por ejemplo:

```
(∀ x) (elefante(x) → cuatro-patas(x))
```

dice que todos los elefantes tienen cuatro patas. Sin embargo, la afirmación de que algunos elementos tienen tres patas se escribe con un Y lógico y con cuantificadores universales como sigue

```
(∃ x) (elefante(x) ∧ tres-patas(x))
```

Así como el cuantificador universal puede expresarse como conjunción, el cuantificador existencial puede expresarse como una disyunción de casos, a_1.

$$P(a_1) \vee P(a_2) \vee P(a_3) \vee \ldots P(a_N)$$

Las afirmaciones cuantificadas y sus negaciones para el ejemplo en que P representa "elefantes y mamíferos" se muestra en la tabla 2.6. Los números entre paréntesis sólo sirven para identificar los ejemplos para el análisis siguiente.

Ejemplo	Significado
(1a) (∀ x)(P)	Todos los elefantes son mamíferos
(1b) (∃ x)(-P)	Algunos elefantes no son mamíferos
(2a) (∃ x)(P)	Algunos elefantes son mamíferos
(2b) (∀ x)(-P)	Ningún elefante es mamífero

Tabla 2.6 Ejemplos de cuantificadores negados

Los ejemplos (1a) y (1b) son negaciones entre sí, como (2a) y (2b). Observe que la negación de una afirmación cuantificada universalmente de (1a) es una afirmación cuantificada existencialmente de la negación de P, como se muestra en (1b). De igual manera, la negación de una afirmación cuantificada existencialmente de (2a) es la afirmación cuantificada universalmente de la negación de P, como se muestra en (2 b).

2.16 CUANTIFICADORES Y CONJUNTOS

Los cuantificadores pueden utilizarse para definir conjuntos en un universo, U, como se muestra en la tabla 2.7.

La relación de que A es un subconjunto propio de B, escrito como $A \subset B$, significa que a pesar de que todos los elementos de A están en B, hay por lo menos un elemento de B que no se encuentra en A. Si E representa elefantes y M representa mamíferos, la relación de conjunto

$$E \subset M$$

establece que todos los elefantes son mamíferos, pero no todos los mamíferos son elefantes. Si G = gris y C = cuatro-patas, la afirmación de que todos los elefantes grises con cuatro patas son mamíferos se escribe

$$(E \cap G \cap C) \subset M$$

Expresión de conjunto	Equivalente lógico
$A = B$	$\forall x\ (x \in A \leftrightarrow x \in B)$
$A \subseteq B$	$\forall x\ (x \in A \rightarrow x \in B)$
$A \cap B$	$\forall x\ (x \in A \wedge x \in B)$
$A \cup B$	$\forall x\ (x \in A \vee x \in B)$
A'	$\forall x\ (x \in U \mid \sim (x \in A))$
U (Universo)	V (Verdadero)
$\varnothing$ (conjunto vacío	F (Falso)

Tabla 2.7 Algunas expresiones de conjuntos y sus equivalentes lógicos

Utilizando las siguientes definiciones, se muestran algunos ejemplos de afirmaciones cuantificadas

E = elefantes
R = reptiles
G = gris
C = cuatro-patas
P = perros
M = mamíferos

Ningún elefante es reptil
$E \cap R = \varnothing$
Algunos elefantes son grises
$E \cap G \neq \varnothing$
Ningún elefante es gris
$E \cap G = \varnothing$
Algunos elefantes no son grises
$E \cap G' \neq \varnothing$
Todos los elefantes son grises y de cuatro patas
$E \subset (G \cap C)$
Todos los elefantes y perros son mamíferos
$(E \cup P) \subset M$
Algunos elefantes tienen cuatro patas y son grises
$(E \cap F \cap G) \neq \varnothing$

Como otra analogía de los conjuntos y formas lógicas, las leyes de De Morgan se muestran en la tabla 2.8, donde el símbolo de equivalencia, ≡, el (bicondicional) significa que la afirmación de la izquierda tiene el mismo valor de verdad que la de la derecha. Es decir, las afirmaciones son equivalentes.

Conjunto	**Lógica**
$(A \cup B)' \equiv A' \cup B'$	$\sim(p \wedge q) \equiv \sim p \vee \sim q$
$(A \cap B)' \equiv A' \cap B'$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$

Tabla 2.8 Conjunto y formas lógicas de las leyes de De Morgan

2.17 LIMITACIONES DE LA LÓGICA DE PREDICADO

Aunque esta lógica es muy útil en muchas situaciones, hay algunos tipos de afirmaciones que no pueden expresarse en ella utilizando los cuantificadores universal y existencial (Rescher 64). Por ejemplo, la siguiente afirmación no puede expresarse con lógica de predicado:

```
La mayoría de la clase recibió dieces
```

En esta afirmación, el cuantificador *mayoría* significa más de la mitad, pero no puede expresarse con los cuantificadores universal y existencial. Para implantar *mayoría* una lógica debe proporcionar algunas predicados para conteo, como la lógica borrosa, descrita en el capítulo 5. Otra limitante de esta lógica es en la expresión de cosas que a veces son verdaderas, pero no siempre. Este problema también puede resolverse con lógica borrosa. Sin embargo, la introducción de un conteo también produce más complicaciones en el sistema lógico y la hace de manera más parecida a las matemáticas.

2.18 RESUMEN

En este capítulo revisamos los elementos de la teoría del conocimiento y las técnicas para representarlo. La representación del conocimiento es de la mayor importancia para los sistemas expertos. El conocimiento puede clasificarse de varias maneras, como *a priori*, *a posteriori*, de procedimiento, declarativo y tácito. Las reglas de producción, las redes semánticas, los esquemas, los marcos y la lógica son métodos comunes para la representación del conocimiento en los sistemas expertos. Cada uno de estos paradigmas tiene ventajas y desventajas. Antes de diseñar un sistema experto, debe decidir cuál es el mejor paradigma para el problema que habrá de resolverse. En lugar de tratar de usar una herramienta para todos los problemas, elija la mejor herramienta para el problema particular.

PROBLEMAS

2.1 Dibuje una red semántica para computadoras utilizando vínculos UN-TIPO-DE y ES-UN. Considere las clases microcomputadora, minicomputadora, supercomputadora, sistema de cómputo, dedicado, de propósito general, a nivel de tarjeta, computadora-en-un-chip, de un solo procesador y de varios procesadores. Incluya casos específicos.