es cuando toma su baño. Así, la *BC* no sería verdadera en el mundo real, sin embargo, mediante procedimientos de aprendizaje buenos no hace falta ser tan pesimistas.

7.4 Lógica proposicional: una lógica muy sencilla

LÓGICA PROPOSICIONAL

Ahora vamos a presentar una lógica muy sencilla llamada **lógica proposicional**[8]. Vamos a cubrir tanto la sintaxis como la semántica (la manera como se define el valor de verdad de las sentencias) de la lógica proposicional. Luego trataremos la **implicación** (la relación entre una sentencia y la que se sigue de ésta) y veremos cómo todo ello nos lleva a un algoritmo de inferencia lógica muy sencillo. Todo ello tratado, por supuesto, en el mundo de *wumpus*.

Sintaxis

SENTENCIAS ATÓMICAS

SÍMBOLO PROPOSICIONAL

La **sintaxis** de la lógica proposicional nos define las sentencias que se pueden construir. Las **sentencias atómicas** (es decir, los elementos sintácticos indivisibles) se componen de un único **símbolo proposicional**. Cada uno de estos símbolos representa una proposición que puede ser verdadera o falsa. Utilizaremos letras mayúsculas para estos símbolos: P, Q, R, y siguientes. Los nombres de los símbolos suelen ser arbitrarios pero a menudo se escogen de manera que tengan algún sentido mnemotécnico para el lector. Por ejemplo, podríamos utilizar $W_{1,3}$ para representar que el *wumpus* se encuentra en la casilla [1, 3]. (Recuerde que los símbolos como $W_{1,3}$ son *atómicos*, esto es, W, 1, y 3 no son partes significantes del símbolo.) Hay dos símbolos proposicionales con significado fijado: *Verdadero*, que es la proposición que siempre es verdadera; y *Falso*, que es la proposición que siempre es falsa.

SENTENCIAS COMPLEJAS

CONECTIVAS LÓGICAS

Las **sentencias complejas** se construyen a partir de sentencias más simples mediante el uso de las **conectivas lógicas**, que son las siguientes cinco:

NEGACIÓN

LITERAL

- $\neg$ (no). Una sentencia como $\neg W_{1,3}$ se denomina **negación** de $W_{1,3}$. Un **literal** puede ser una sentencia atómica (un **literal positivo**) o una sentencia atómica negada (un **literal negativo**).

CONJUNCIÓN

- $\wedge$ (y). Una sentencia que tenga como conectiva principal $\wedge$, como es $W_{1,3} \wedge H_{3,1}$, se denomina **conjunción**; sus componentes son los **conjuntores**.

DISYUNCIÓN

- $\vee$ (o). Una sentencia que utiliza la conectiva $\vee$, como es $(W_{1,3} \wedge H_{3,1}) \vee W_{2,2}$, es una **disyunción** de los **disyuntores** $(W_{1,3} \wedge H_{3,1})$ y $W_{2,2}$. (Históricamente, la conectiva $\vee$ proviene de «vel» en Latín, que significa «o». Para mucha gente, es más fácil recordarla como la conjunción al revés.)

IMPLICACIÓN

PREMISA

CONCLUSIÓN

- $\Rightarrow$ (implica). Una sentencia como $(W_{1,3} \wedge H_{3,1}) \Rightarrow \neg W_{2,2}$ se denomina **implicación** (o condicional). Su **premisa** o **antecedente** es $(W_{1,3} \wedge H_{3,1})$, y su **conclusión** o **consecuente** es $\neg W_{2,2}$. Las implicaciones también se conocen como **reglas** o afir-

[8] A la lógica proposicional también se le denomina **Lógica Booleana**, por el matemático George Boole (1815-1864).

maciones **si-entonces**. Algunas veces, en otros libros, el símbolo de la implicación se representa mediante $\supset$ o $\rightarrow$.

BICONDICIONAL

$\Leftrightarrow$ (si y sólo si). La sentencia $W_{1,3} \Leftrightarrow \neg W_{2,2}$ es una **bicondicional**.

En la Figura 7.7 se muestra una gramática formal de la lógica proposicional; mira la página 984 si no estás familiarizado con la notación BNF.

Sentencia → *Sentencia Atómica* | *Sentencia Compleja*
Sentencia Atómica → **Verdadero** | **Falso** | *Símbolo Proposicional*
Símbolo Proposicional → **P** | **Q** | **R** | ...
Sentencia Compleja → ¬ *Sentencia*
| (*Sentencia* ∧ *Sentencia*)
| (*Sentencia* ∨ *Sentencia*)
| (*Sentencia* ⇒ *Sentencia*)
| (*Sentencia* ⇔ *Sentencia*)

Figura 7.7 Una gramática BNF (Backus-Naur Form) de sentencias en lógica proposicional.

Fíjese en que la gramática es muy estricta respecto al uso de los paréntesis: cada sentencia construida a partir de conectivas binarias debe estar encerrada en paréntesis. Esto asegura que la gramática no sea ambigua. También significa que tenemos que escribir, por ejemplo, $((A \wedge B) \Rightarrow C)$ en vez de $A \wedge B \Rightarrow C$. Para mejorar la legibilidad, a menudo omitiremos paréntesis, apoyándonos en su lugar en un orden de precedencia de las conectivas. Es una precedencia similar a la utilizada en la aritmética (por ejemplo, $ab + c$ se lee $((ab) + c)$ porque la multiplicación tiene mayor precedencia que la suma). El orden de precedencia en la lógica proposicional (de mayor a menor) es: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ y $\Leftrightarrow$. Así, la sentencia

$$\neg P \vee Q \wedge R \Rightarrow S$$

es equivalente a la sentencia

$$((\neg P) \vee (Q \wedge R)) \Rightarrow S$$

La precedencia entre las conectivas no resuelve la ambigüedad en sentencias como $A \wedge B \wedge C$, que se podría leer como $((A \wedge B) \wedge C)$ o como $(A \wedge (B \wedge C))$. Como estas dos lecturas significan lo mismo según la semántica que mostraremos en la siguiente sección, se permiten este tipo de sentencias. También se permiten sentencias como $A \vee B \vee C$ o $A \Leftrightarrow B \Leftrightarrow C$. Sin embargo, las sentencias como $A \Rightarrow B \Rightarrow C$ no se permiten, ya que su lectura en una dirección y su opuesta tienen significados muy diferentes; en este caso insistimos en la utilización de los paréntesis. Por último, a veces utilizaremos corchetes, en vez de paréntesis, para conseguir una lectura de la sentencia más clara.

Semántica

Una vez especificada la sintaxis de la lógica proposicional, vamos a definir su semántica. La semántica define las reglas para determinar el valor de verdad de una sentencia respecto a un modelo en concreto. En la lógica proposicional un modelo define el va-

lor de verdad (*verdadero* o *falso*). Por ejemplo, si las sentencias de la base de conocimiento utilizan los símbolos proposicionales $H_{1,2}$, $H_{2,2}$, y $H_{3,1}$, entonces un modelo posible sería

$$m_1 = \{H_{1,2} = \textit{falso}, H_{2,2} = \text{falso}, H_{3,1} = \textit{verdadero}\}$$

Con tres símbolos proposicionales hay $2^3 = 8$ modelos posibles, exactamente los que aparecen en la Figura 7.5. Sin embargo, fíjese en que gracias a que hemos concretado la sintaxis, los modelos se convierten en objetos puramente matemáticos sin tener necesariamente una conexión al mundo de *wumpus*. $H_{1,2}$ es sólo un símbolo, podría denotar tanto «hay un hoyo en la casilla [1, 2]», como «estaré en París hoy y mañana».

La semántica en lógica proposicional debe especificar cómo obtener el valor de verdad de *cualquier* sentencia, dado un modelo. Este proceso se realiza de forma recursiva. Todas las sentencias se construyen a partir de las sentencias atómicas y las cinco conectivas lógicas; entonces necesitamos establecer cómo definir el valor de verdad de las sentencias atómicas y cómo calcular el valor de verdad de las sentencias construidas con las cinco conectivas lógicas. Para las sentencias atómicas es sencillo:

- *Verdadero* es verdadero en todos los modelos y *Falso* es falso en todos los modelos.
- El valor de verdad de cada símbolo proposicional se debe especificar directamente para cada modelo. Por ejemplo, en el modelo anterior m_1, $H_{1,2}$ es falso.

Para las sentencias complejas, tenemos reglas como la siguiente

- Para toda sentencia s y todo modelo m, la sentencia $\neg s$ es verdadera en m si y sólo si s es falsa en m.

Este tipo de reglas reducen el cálculo del valor de verdad de una sentencia compleja al valor de verdad de las sentencias más simples. Las reglas para las conectivas se pueden resumir en una **tabla de verdad** que especifica el valor de verdad de cada sentencia compleja según la posible asignación de valores de verdad realizada a sus componentes. En la Figura 7.8 se muestra la tabla de verdad de las cinco conectivas lógicas. Utilizando estas tablas de verdad, se puede obtener el valor de verdad de cualquier sentencia s según un modelo m mediante un proceso de evaluación recursiva muy sencillo. Por ejemplo, la sentencia $\neg H_{1,2} \wedge (H_{2,2} \vee H_{3,1})$ evaluada según m_1, da *verda-*

TABLA DE VERDAD

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
falso	*falso*	*verdadero*	*falso*	*falso*	*verdadero*	*verdadero*
falso	*verdadero*	*verdadero*	*falso*	*verdadero*	*verdadero*	*falso*
verdadero	*falso*	*falso*	*falso*	*verdadero*	*falso*	*falso*
verdadero	*verdadero*	*falso*	*verdadero*	*verdadero*	*verdadero*	*verdadero*

Figura 7.8 Tablas de verdad para las cinco conectivas lógicas. Para utilizar la tabla, por ejemplo, para calcular el valor de $P \vee Q$, cuando P es verdadero y Q falso, primero mire a la izquierda en donde P es *verdadera* y Q es *falsa* (la tercera fila). Entonces mire en esa fila justo en la columna de $P \vee Q$ para ver el resultado: *verdadero*. Otra forma de verlo es pensar en cada fila como en un modelo, y que sus entradas en cada fila dicen para cada columna si la sentencia es verdadera en ese modelo.

dero $\wedge$ (*falso* $\vee$ *verdadero*) = *verdadero* $\wedge$ *verdadero* = *verdadero*. El Ejercicio 7.3 pide que escriba el algoritmo ¿V-VERDAD-LP?(*s*, *m*) que debe obtener el valor de verdad de una sentencia *s* en lógica proposicional según el modelo *m*.

Ya hemos comentado que una base de conocimiento está compuesta por sentencias. Ahora podemos observar que esa base de conocimiento lógica es una conjunción de dichas sentencias. Es decir, si comenzamos con una *BC* vacía y ejecutamos DECIR(*BC*, S_1)... DECIR(*BC*, S_n) entonces tenemos $BC = S_1 \wedge \ldots \wedge S_n$. Esto significa que podemos manejar bases de conocimiento y sentencias de manera intercambiable.

Los valores de verdad de «y», «o» y «no» concuerdan con nuestra intuición, cuando los utilizamos en lenguaje natural. El principal punto de confusión puede presentarse cuando $P \vee Q$ es verdadero porque P lo es, Q lo es, o *ambos* lo son. Hay una conectiva diferente denominada «o exclusiva» («xor» para abreviar) que es falsa cuando los dos disyuntores son verdaderos[9]. No hay consenso respecto al símbolo que representa la o exclusiva, siendo las dos alternativas $\dot{\vee}$ y $\oplus$.

El valor de verdad de la conectiva $\Rightarrow$ puede parecer incomprensible al principio, ya que no encaja en nuestra comprensión intuitiva acerca de «*P* implica *Q*» o de «si *P* entonces *Q*». Para una cosa, la lógica proposicional no requiere de una relación de causalidad o relevancia entre P y Q. La sentencia «que 5 sea impar implica que Tokio es la capital de Japón» es una sentencia verdadera en lógica proposicional (bajo una interpretación normal), aunque pensándolo es, decididamente, una frase muy rara. Otro punto de confusión es que cualquier implicación es verdadera siempre que su antecedente sea falso. Por ejemplo, «que 5 sea par implica que Sam es astuto» es verdadera, independientemente de que Sam sea o no astuto. Parece algo estrafalario, pero tiene sentido si piensa acerca de «$P \Rightarrow Q$» como si dijera, «si P es verdadero, entonces estoy afirmando que Q es verdadero. De otro modo, no estoy haciendo ninguna afirmación.» La única manera de hacer esta sentencia *falsa* es haciendo que P sea cierta y Q falsa.

La tabla de verdad de la bicondicional $P \Leftrightarrow Q$ muestra que la sentencia es verdadera siempre que $P \Rightarrow Q$ y $Q \Rightarrow P$ lo son. En lenguaje natural a menudo se escribe como «*P* si y sólo si *Q*» o «*P* si *Q*». Las reglas del mundo de *wumpus* se describen mejor utilizando la conectiva $\Leftrightarrow$. Por ejemplo, una casilla tiene corriente de aire *si* alguna casilla vecina tiene un hoyo, y una casilla tiene corriente de aire *sólo si* una casilla vecina tiene un hoyo. De esta manera necesitamos bicondicionales como

$$B_{1,1} \Leftrightarrow (H_{1,2} \vee H_{2,1}),$$

en donde $B_{1,1}$ significa que hay una brisa en la casilla [1,1]. Fíjese en que la implicación

$$B_{1,1} \Rightarrow (H_{1,2} \vee H_{2,1})$$

es verdadera, aunque incompleta, en el mundo de *wumpus*. Esta implicación no descarta modelos en los que $B_{1,1}$ sea falso y $H_{1,2}$ sea verdadero, hecho que violaría las reglas del mundo de *wumpus*. Otra forma de observar esta incompletitud es que la implicación necesita la presencia de hoyos si hay una corriente de aire, mientras que la bicondicional además necesita la ausencia de hoyos si no hay ninguna corriente de aire.

[9] En latín está la palabra específica *aut* para la o exclusiva.

Una base de conocimiento sencilla

Ahora que ya hemos definido la semántica de la lógica proposicional, podemos construir una base de conocimiento para el mundo de *wumpus*. Para simplificar, sólo trataremos con hechos y reglas acerca de hoyos; dejamos el tratamiento del *wumpus* como ejercicio. Vamos a proporcionar el conocimiento suficiente para llevar a cabo la inferencia que se trató en la Sección 7.3.

Primero de todo, necesitamos escoger nuestro vocabulario de símbolos proposicionales. Para cada i, j:

- Hacemos que $H_{i,j}$ sea verdadero si hay un hoyo en la casilla $[i, j]$.
- Hacemos que $B_{i,j}$ sea verdadero si hay una corriente de aire (una brisa) en la casilla [i, j].

La base de conocimiento contiene, cada una etiquetada con un identificador, las siguientes sentencias:

- No hay ningún hoyo en la casilla [1, 1].

$$R_1: \quad \neg H_{1,1}$$

- En una casilla se siente una brisa si y sólo si hay un hoyo en una casilla vecina. Esta regla se ha de especificar para cada casilla; por ahora, tan sólo incluimos las casillas que son relevantes:

$$R_2: \quad B_{1,1} \Leftrightarrow (H_{1,2} \vee H_{2,1})$$
$$R_3: \quad B_{2,1} \Leftrightarrow (H_{1,1} \vee H_{2,2} \vee H_{3,1})$$

- Las sentencias anteriores son verdaderas en todos los mundos de *wumpus*. Ahora incluimos las percepciones de brisa para las dos primeras casillas visitadas en el mundo concreto en donde se encuentra el agente, llegando a la situación que se muestra en la Figura 7.3(b).

$$R_4: \quad \neg B_{1,1}$$
$$R_5: \quad B_{2,1}$$

Entonces, la base de conocimiento está compuesta por las sentencias R_1 hasta R_5. La *BC* también se puede representar mediante una única sentencia (la conjunción $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$) porque dicha sentencia aserta que todas las sentencias son verdaderas.

Inferencia

Recordemos que el objetivo de la inferencia lógica es decidir si $BC \models \alpha$ para alguna sentencia α. Por ejemplo, si se deduce $H_{2,2}$. Nuestro primer algoritmo para la inferencia será una implementación directa del concepto de implicación: enumerar los modelos, y averiguar si α es verdadera en cada modelo en el que la *BC* es verdadera. En la lógica proposicional los modelos son asignaciones de los valores *verdadero* y *falso* sobre cada símbolo proposicional. Volviendo a nuestro ejemplo del mundo de *wumpus*, los símbolos proposicionales relevantes son $B_{1,1}$, $B_{2,1}$, $H_{1,1}$, $H_{1,2}$, $H_{2,1}$, $H_{2,2}$ y $H_{3,1}$. Con es-

tos siete símbolos, tenemos $2^7 = 128$ modelos posibles; y en tres de estos modelos, la *BC* es verdadera (Figura 7.9). En esos tres modelos $\neg H_{1,2}$ es verdadera, por lo tanto, no hay un hoyo en la casilla [1, 2]. Por el otro lado, $H_{2,2}$ es verdadera en dos de esos tres modelos y falsa en el tercero, entonces todavía no podemos decir si hay un hoyo en la casilla [2, 2].

La Figura 7.9 reproduce más detalladamente el razonamiento que se mostraba en la Figura 7.5. En la Figura 7.10 se muestra un algoritmo general para averiguar la implicación en lógica proposicional. De forma similar al algoritmo de BÚSQUEDA-CON-BACKTRACKING de la página 86, ¿IMPLICACIÓN-EN-TV? Realiza una enumeración recursiva de un espacio finito de asignaciones a variables. El algoritmo es **sólido** porque implemen-

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	BC
falso	*falso*	*falso*	*falso*	*falso*	*falso*	*falso*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*falso*	*falso*
falso	*falso*	*falso*	*falso*	*falso*	*falso*	*verdadero*	*verdadero*	*verdadero*	*falso*	*verdadero*	*falso*	*falso*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
falso	*verdadero*	*falso*	*falso*	*falso*	*falso*	*falso*	*verdadero*	*verdadero*	*falso*	*verdadero*	*verdadero*	*falso*
falso	*verdadero*	*falso*	*falso*	*falso*	*falso*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*
falso	*verdadero*	*falso*	*falso*	*falso*	*verdadero*	*falso*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*
falso	*verdadero*	*falso*	*falso*	*falso*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*
falso	*verdadero*	*falso*	*falso*	*verdadero*	*falso*	*falso*	*verdadero*	*falso*	*falso*	*verdadero*	*verdadero*	*falso*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
verdadero	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*verdadero*	*falso*	*verdadero*	*verdadero*	*falso*	*verdadero*	*falso*

Figura 7.9 Una tabla de verdad construida para la base de conocimiento del ejemplo. La *BC* es verdadera si R_1 hasta R_5 son verdaderas, cosa que sucede en tres de las 128 filas. En estas tres filas, $H_{1,2}$ es falsa, así que no hay ningún hoyo en la casilla [1, 2]. Por otro lado, puede haber (o no) un hoyo en la casilla [2, 2].

función ¿IMPLICACIÓN-EN-TV?(*BC*, α) **devuelve** *verdadero* o *falso*
  entradas: *BC*, la base de conocimiento, una sentencia en lógica proposicional
    α, la sentencia implicada, una sentencia en lógica proposicional

  símbolos ← una lista de símbolos proposicionales de la *BC* y α
  devuelve COMPROBAR-TV(*BC*, α, *símbolos*, [ ])

función COMPROBAR-TV(*BC*, α, *símbolos*, *modelo*) **devuelve** *verdadero* o *falso*
  si ¿VACÍA?(*símbolos*) **entonces**
    si ¿VERDADERO-LP?(*BC*, *modelo*) **entonces devuelve** ¿VERDADERO-LP?(α, *modelo*)
    sino devuelve *verdadero*
  sino hacer
    P ← PRIMERO(*símbolos*); *resto* ← RESTO(*símbolos*)
    devuelve CHEQUEAR-TV(*BC*, α, *resto*, EXTENDER(*P*, *verdadero*, *modelo*)) **y**
      COMPROBAR-TV(*BC*, α, *resto*, EXTENDER(*P*, *falso*, *modelo*))

Figura 7.10 Un algoritmo de enumeración de una tabla de verdad para averiguar la implicación proposicional. TV viene de tabla de verdad. ¿VERDADERO-LP? Devuelve verdadero si una sentencia es verdadera en un modelo. La variable *modelo* representa un modelo parcial (una asignación realizada a un subconjunto de las variables). La llamada a la función EXTENDER(*P*, *verdadero*, *modelo*) devuelve un modelo parcial nuevo en el que *P* tiene el valor de verdad *verdadero*.

ta de forma directa la definición de implicación, y es **completo** porque trabaja para cualquier BC y sentencia α, y siempre finaliza (sólo hay un conjunto finito de modelos a ser examinados).

Por supuesto, que «conjunto finito» no siempre es lo mismo que «pequeño». Si la BC y α contienen en total n símbolos, entonces tenemos 2^n modelos posibles. Así, la complejidad temporal del algoritmo es $O(2^n)$. (La complejidad espacial sólo es $O(n)$ porque la enumeración es en primero en profundidad.) Más adelante, en este capítulo, veremos algoritmos que en la práctica son mucho más eficientes. Desafortunadamente, *cada algoritmo de inferencia que se conoce en lógica proposicional tiene un caso peor, cuya complejidad es exponencial respecto al tamaño de la entrada.* No esperamos mejorarlo, ya que demostrar la implicación en lógica proposicional es un problema co-NP-completo. (*Véase* Apéndice A.)

Equivalencia, validez y *satisfacibilidad*

Antes de que nos sumerjamos en los detalles de los algoritmos de inferencia lógica necesitaremos algunos conceptos adicionales relacionados con la implicación. Al igual que la implicación, estos conceptos se aplican a todos los tipos de lógica, sin embargo, se entienden más fácilmente para una en concreto, como es el caso de la lógica proposicional.

EQUIVALENCIA LÓGICA

El primer concepto es la **equivalencia lógica**: dos sentencias α y β son equivalentes lógicamente si tienen los mismos valores de verdad en el mismo conjunto de modelos. Este concepto lo representamos con $\alpha \Leftrightarrow \beta$. Por ejemplo, podemos observar fácilmente (mediante una tabla de verdad) que $P \wedge Q$ y $Q \wedge P$ son equivalentes lógicamente. En la Figura 7.11 se muestran otras equivalencias. Éstas juegan el mismo papel en la lógica que las igualdades en las matemáticas. Una definición alternativa de equivalencia es la siguiente: para dos sentencias α y β cualesquiera,

$$\alpha \equiv \beta \quad \text{si y sólo si} \quad \alpha \models \beta \text{ y } \beta \models \alpha$$

(Recuerde que $\models$ significa implicación.)

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{Conmutatividad de } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{Conmutatividad de } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{Asociatividad de } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{Asociatividad de } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{Eliminación de la doble negación} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{Contraposición} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{Eliminación de la implicación} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{Eliminación de la bicondicional} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{Ley de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{Ley de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{Distribución de } \wedge \text{ respecto a } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{Distribución de } \vee \text{ respecto a } \wedge
\end{aligned}
$$

Figura 7.11 Equivalencias lógicas. Los símbolos α, β y γ se pueden sustituir por cualquier sentencia en lógica proposicional.

VALIDEZ

TAUTOLOGÍA

El segundo concepto que necesitaremos es el de **validez**. Una sentencia es válida si es verdadera en *todos* los modelos. Por ejemplo, la sentencia $P \vee \neg P$ es una sentencia válida. Las sentencias válidas también se conocen como **tautologías**, son *necesariamente* verdaderas y por lo tanto vacías de significado. Como la sentencia *Verdadero* es verdadera en todos los modelos, toda sentencia válida es lógicamente equivalente a *Verdadero*.

TEOREMA DE LA DEDUCCIÓN

¿Qué utilidad tienen las sentencias válidas? De nuestra definición de implicación podemos derivar el **teorema de la deducción**, que ya se conocía por los Griegos antiguos:

Para cualquier sentencia α y β, $\alpha \models \beta$ si y sólo si la sentencia $(\alpha \Rightarrow \beta)$ es válida.

(En el Ejercicio 7.4 se pide demostrar una serie de aserciones.) Podemos pensar en el algoritmo de inferencia de la Figura 7.10 como en un proceso para averiguar la validez de $(BC \Rightarrow \alpha)$. A la inversa, cada sentencia que es una implicación válida representa una inferencia correcta.

SATISFACIBILIDAD

SATISFACE

El último concepto que necesitaremos es el de ***satisfacibilidad***. Una sentencia es satisfactoria si es verdadera para *algún* modelo. Por ejemplo, en la base de conocimiento ya mostrada, $(R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5)$ es *satisfacible* porque hay tres modelos en los que es verdadera, tal como se muestra en la Figura 7.9. Si una sentencia α es verdadera en un modelo m, entonces decimos que m **satisface** α, o que m **es un modelo de** α. La *satisfacibilidad* se puede averiguar enumerando los modelos posibles hasta que uno satisface la sentencia. La determinación de la *satisfacibilidad* de sentencias en lógica proposicional fue el primer problema que se demostró que era NP-completo.

Muchos problemas en las ciencias de la computación son en realidad problemas de *satisfacibilidad*. Por ejemplo, todos los problemas de satisfacción de restricciones del Capítulo 5 se preguntan esencialmente si un conjunto de restricciones se satisfacen dada una asignación. Con algunas transformaciones adecuadas, los problemas de búsqueda también se pueden resolver mediante *satisfacibilidad*. La validez y la *satisfacible* están íntimamente relacionadas: α es válida si y sólo si $\neg\alpha$ es *insatisfacible*; en contraposición, α es *satisfacible* si y sólo si $\neg\alpha$ no es válida.

$\alpha \models \beta$ si y sólo si la sentencia $(\alpha \wedge \neg\beta)$ es insatisfactoria.

REDUCTIO AD ABSURDUM

REFUTACIÓN

La demostración de β a partir de α averiguando la insatisfacibilidad de $(\alpha \wedge \neg\beta)$ se corresponde exactamente con la técnica de demostración en matemáticas de la *reductio ad absurdum* (que literalmente se traduce como «reducción al absurdo»). Esta técnica también se denomina demostración mediante **refutación** o demostración por **contradicción**. Asumimos que la sentencia β es falsa y observamos si se llega a una contradicción con las premisas en α. Dicha contradicción es justamente lo que queremos expresar cuando decimos que la sentencia $(\alpha \wedge \neg\beta)$ es *insatisfacible*.

7.5 Patrones de razonamiento en lógica proposicional

Esta sección cubre los patrones estándar de inferencia que se pueden aplicar para derivar cadenas de conclusiones que nos llevan al objetivo deseado. Estos patrones de infe-