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 $\veebar$ 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.