Por supuesto, estos resultados sólo son para los problemas generados aleatoriamente. Los problemas reales no tienen necesariamente la misma estructura (en términos de proporciones entre literales positivos y negativos, densidades de conexiones entre cláusulas, etcétera) que los problemas aleatorios. Pero todavía en la práctica, el SAT-CAMINAR y otros algoritmos de la misma familia son muy buenos para resolver problemas reales (a menudo, tan buenos como el mejor algoritmo de propósito específico para esas tareas). Problemas con miles de símbolos y millones de cláusulas se tratan de forma rutinaria con resolutores como el CHAFF. Estas observaciones nos sugieren que alguna combinación de los comportamientos de la heurística de min-conflictos y de pasada-aleatoria nos proporciona una gran capacidad de *propósito-general* para resolver muchas situaciones en las que se requiere el razonamiento combinatorio.

7.7 Agentes basados en lógica proposicional

En esta sección, vamos a juntar lo que hemos aprendido hasta ahora para construir agentes que funcionan utilizando la lógica proposicional. Veremos dos tipos de agentes: aquellos que utilizan algoritmos de inferencia y una base de conocimiento, como el agente basado en conocimiento genérico de la Figura 7.1, y aquellos que evalúan directamente expresiones lógicas en forma de circuitos. Aplicaremos ambos tipos de agentes en el mundo de *wumpus*, y encontraremos que ambos sufren de serias desventajas.

Encontrar hoyos y *wumpus* utilizando la inferencia lógica

Permítanos empezar con un agente que razona mediante la lógica acerca de las localizaciones de los hoyos, de los *wumpus* y de la seguridad de las casillas. El agente comienza con una base de conocimiento que describe la «física» del mundo de *wumpus*. El agente sabe que la casilla [1, 1] no tiene ningún hoyo ni ningún *wumpus*; es decir, $\neg H_{1,1}$ y $\neg W_{1,1}$. El agente también conoce una sentencia que indica cómo se percibe una brisa en una casilla $[x, y]$:

$$B_{x,y} \Leftrightarrow (H_{x,y+1} \vee H_{x,y-1} \vee H_{x+1,y} \vee H_{x-1,y}) \tag{7.1}$$

El agente conoce una sentencia que indica cómo se percibe el hedor en una casilla $[x, y]$:

$$M_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y}) \tag{7.2}$$

Finalmente, el agente sabe que sólo hay un *wumpus*. Esto se expresa de dos maneras. En la primera, debemos definir que hay *por lo menos* un *wumpus*:

$$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,3} \vee W_{4,4}$$

Entonces, debemos definir que *como mucho* hay un *wumpus*. Una manera de definirlo es diciendo que para dos casillas cualesquiera, una de ellas debe estar libre de un *wum-*

pus. Con n casillas, tenemos $n(n-1)/2$ sentencias del tipo $\neg W_{1,1} \vee \neg W_{1,2}$. Entonces, para un mundo de 4×4, comenzamos con un total de 155 sentencias conteniendo 64 símbolos diferentes.

El programa del agente, que se muestra en la Figura 7.19 DICE a su base de conocimiento cualquier nueva percepción acerca de una brisa o un mal hedor. (También actualiza algunas variables del programa para guardar la pista de dónde se encuentra y que casillas ha visitado. Estos últimos datos los necesitará más adelante.) Entonces el programa escoge dónde observar antes de entre las casillas que rodean al agente, es decir, las casillas adyacentes a aquellas ya visitadas. Una casilla $[i, j]$ que rodea al agente es *probable que sea segura* si la sentencia $(\neg H_{i,j} \wedge \neg W_{i,j})$ se deduce de la base de conocimiento. La siguiente alternativa mejor es una casilla *posiblemente segura*, o sea, aquella casilla para la cual el agente no puede demostrar si hay un hoyo o un *wumpus*, es decir, aquella para la que *no* se deduce $(H_{i,j} \vee W_{i,j})$.

El cálculo para averiguar la implicación mediante PREGUNTAR se puede implementar utilizando cualquiera de los métodos descritos anteriormente en este capítulo. ¿IMPLICACIÓN-TV? (Figura 7.10) es obviamente impracticable, ya que debería enumerar 2^{64} filas. El DPLL (Figura 7.16) realiza las inferencias necesarias en pocos milisegundos, principalmente gracias a la heurística de propagación unitaria. El SAT-CAMINAR también se puede utilizar, teniendo en cuenta la advertencia acerca de la incompletitud. En el mundo de *wumpus*, los obstáculos para encontrar un modelo, realizados 10.000 intercambios,

función AGENTE-WUMPUS-LP(percepción) **devuelve** una *acción*
  entradas: *percepción*, una lista, [*hedor, brisa, resplandor*]
  variables estáticas: *BC*, una base de conocimiento, inicialmente conteniendo la «física» del mundo de wumpus
    x, y, orientación, la posición del agente (inicialmente en 1, 1) y su orientación (inicialmente *derecha*)
    visitada, una matriz indicando qué casillas han sido visitadas, inicialmente *falso*
    acción, la acción más reciente del agente, inicialmente null

  plan, una secuencia de acciones, inicialmente vacía
  actualiza *x, y, orientación, visitada* basada en *acción*
  si *hedor* **entonces** DECIR(BC, $M_{x,y}$) **sino** DECIR(BC, $\neg M_{x,y}$)
  si *brisa* **entonces** DECIR(BC, $B_{x,y}$) **sino** DECIR(BC, $\neg B_{x,y}$)
  si *resplandor* **entonces** *acción* ← *agarrar*
  sino si *plan* no está vacío **entonces** *acción* ← POP(*plan*)
  sino si para alguna casilla $[i, j]$ que nos rodea es *verdadero* PREGUNTAR(BC, $(\neg H_{i,j} \wedge \neg W_{i,j})$)
      o es *falso* PREGUNTAR(BC, $(H_{i,j} \vee W_{i,j})$) **entonces hacer**
    plan ← BUSQUEDA-GRAFO-A*(PROBLEMA-RUTA([*x, y*], *orientación*, [*i, j*], *visitada*))
    acción ← POP(*plan*)
  sino *acción* ← un movimiento escogido al azar
  devolver *acción*

Figura 7.19 Un agente en el mundo de wumpus que utiliza la lógica proposicional para identificar hoyos, wumpus y casillas seguras. La subrutina PROBLEMA-RUTA construye un problema de búsqueda cuya solución es una secuencia de acciones que nos llevan de $[x, y]$ a $[i, j]$ pasando sólo a través de las casillas previamente visitadas.

se corresponden invariablemente con la *insatisfacibilidad*, así que los errores no se deben probablemente a la incompletitud.

El programa AGENTE-WUMPUS-LP se comporta bastante bien en un mundo de *wumpus* pequeño. Sin embargo, sucede algo profundamente insatisfactorio respecto a la base de conocimiento del agente. La *BC* contiene las sentencias que definen la «física» en la forma dada en las Ecuaciones (7.1) y (7.2) *para cada casilla individual*. Cuanto más grande sea el entorno, más grande necesita ser la base de conocimiento inicial. Preferiríamos en mayor medida disponer sólo de las dos sentencias para definir cómo se presenta una brisa o un hedor en *cualquier* casilla. Esto está más allá del poder de expresión de la lógica proposicional. En el próximo capítulo veremos un lenguaje lógico más expresivo mediante el cual es más fácil expresar este tipo de sentencias.

Guardar la pista acerca de la localización y la orientación del agente

El programa del agente de la Figura 7.19 «hace trampa» porque guarda la pista de su localización *fuera* de la base de conocimiento, en vez de utilizar el razonamiento lógico[13]. Para hacerlo «correctamente» necesitaremos proposiciones acerca de la localización. Una primera aproximación podría consistir en utilizar un símbolo como $L_{1,1}$ para indicar que el agente se encuentra en la casilla [1, 1]. Entonces la base de conocimiento inicial podría incluir sentencias como

$$L_{1,1} \wedge \mathit{OrientadoDerecha} \wedge \mathit{Avanzar} \Rightarrow L_{2,1}$$

Vemos inmediatamente que esto no funciona correctamente. Si el agente comienza en la casilla [1, 1] orientado a la derecha y avanza, de la base de conocimiento se deduciría $L_{1,1}$ (la localización original del agente) y $L_{2,1}$ (su nueva localización). ¡Aunque estas dos proposiciones no pueden ser verdaderas a la vez! El problema es que las proposiciones de localización deberían referirse a dos instantes de tiempo diferentes. Necesitamos $L^1_{1,1}$ para indicar que el agente se encuentra en la casilla [1, 1] en el instante de tiempo 1, $L^2_{2,1}$ para indicar que el agente se encuentra en la casilla [2, 1] en el instante de tiempo 2, etcétera. Las proposiciones acerca de la orientación y la acción también necesitan depender del tiempo. Por lo tanto, la sentencia correcta es

$$L^1_{1,1} \wedge \mathit{OrientadoDerecha}^1 \wedge \mathit{Avanzar}^1 \Rightarrow L^2_{2,1}$$
$$\mathit{OrientadoDerecha} \wedge \mathit{GirarIzquierda}^1 \Rightarrow \mathit{OrientadoArriba}^2$$

De esta manera resulta bastante difícil construir una base de conocimiento completa y correcta para guardar la pista de cada cosa que sucede en el mundo de *wumpus*; pospondremos la discusión en su detalle para el Capítulo 10. Lo que nos proponemos hacer aquí es que la base de conocimiento inicial contenga sentencias como los dos ejemplos anteriores para cada instante de tiempo t, así como para cada localización. Es

[13] El lector que sea observador se habrá dado cuenta de que esto nos posibilitó afinar la conexión que hay entre las percepciones, como *Brisa* y la proposiciones acerca de la localización específica, como $B_{1,1}$.

decir, que para cada instante de tiempo t y localización $[x, y]$, la base de conocimiento contenga una sentencia del tipo

$$L^t_{x,y} \wedge OrientadoDerecha^t \wedge Avanzar^t \Rightarrow L^{t+1}_{x+1,y} \tag{7.3}$$

Aunque pongamos un límite superior de pasos permitidos en el tiempo (por ejemplo 100) acabamos con decenas de miles de sentencias. Se presenta el mismo problema si añadimos las sentencias «a medida que las necesitemos» en cada paso en el tiempo. Esta proliferación de cláusulas hace que la base de conocimiento sea ilegible para las personas, sin embargo, los resolutores rápidos en lógica proposicional aún pueden manejar un mundo de *wumpus* de 4×4 con cierta facilidad (éstos encuentran su límite en los mundos de 100×100 casillas). Los agentes basados en circuitos de la siguiente sección ofrecen una solución parcial a este problema de proliferación de las cláusulas, pero la solución íntegra deberá esperar hasta que hayamos desarrollado la lógica de primer orden en el Capítulo 8.

Agentes basados en circuitos

AGENTE BASADO EN CIRCUITOS

CIRCUITO SECUENCIAL

PUERTAS

REGISTROS

Un **agente basado en circuitos** es un tipo particular de agente reflexivo con estado interno, tal como se definió en el Capítulo 2. Las percepciones son las entradas de un **circuito secuencial** (una red de **puertas**, cada una de ellas implementa una conectiva lógica, y de **registros**, cada uno de ellos almacena el valor lógico de una proposición atómica). Las salidas del circuito son los registros que se corresponden con las acciones, por ejemplo, la salida *Agarrar* se asigna a *verdadero* si el agente quiere coger algo. Si la entrada *Resplandor* se conecta directamente a la salida *Agarrar*, el agente cogerá el objetivo (el objeto *oro*) siempre que vea el resplandor. (*Véase* Figura 7.20.)

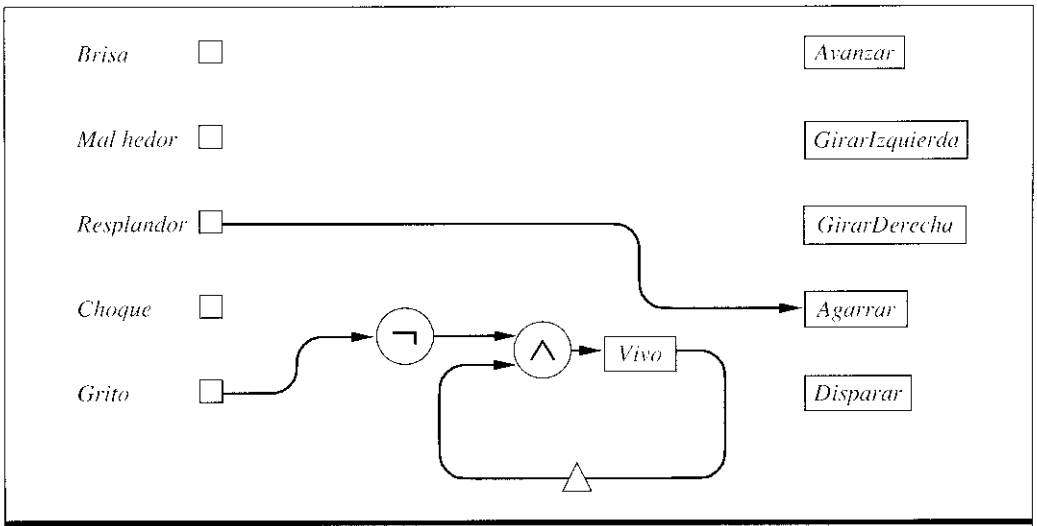

Figura 7.20 Parte de un agente basado en circuitos para el mundo de *wumpus*, mostrando las entradas, las salidas, el circuito para coger el oro, y el circuito que determina si el *wumpus* está vivo. Los registros se muestran como rectángulos, y los retardos de un paso se muestran como pequeños triángulos.

FLUJO DE DATOS

Los circuitos se evalúan de igual modo que los **flujos de datos**: en cada instante de tiempo, se asignan las entradas y se propagan las señales a través del circuito. Siempre que una puerta disponga de todas sus entradas, ésta produce una salida. Este proceso está íntimamente relacionado con el de encadenamiento hacia delante en un grafo Y-O, como el de la Figura 7.15(b).

En la sección precedente dijimos que los agentes basados en circuitos manejan el tiempo de forma más satisfactoria que los agentes basados en la inferencia proposicional. Esto se debe a que cada registro nos da el valor de verdad de su correspondiente símbolo proposicional *en el instante de tiempo actual t*, en vez de disponer de una copia diferente para cada instante de tiempo. Por ejemplo, podríamos tener un registro para *Vivo* que debería contener el valor *verdadero* cuando el *wumpus* esté vivo, y *falso* cuando esté muerto. Este registro se corresponde con el símbolo proposicional $Vivo^t$, de esta manera, en cada instante de tiempo el registro se refiere a una proposición diferente. El estado interno del agente, es decir, su memoria, mantiene conectando hacia atrás la salida de un registro con el circuito mediante una **línea de retardo**. Este mecanismo nos da el valor del registro en el instante de tiempo previo. En la Figura 7.20 se muestra un ejemplo. El valor del registro *Vivo* se obtiene de la conjunción de la negación del registro *Grito* y de su propio valor anterior. En términos de proposiciones, el circuito del registro *Vivo* implementa la siguiente conectiva bicondicional

LÍNEA DE RETARDO

$$Vivo^t \Leftrightarrow \neg Grito^t \wedge Vivo^{t-1} \tag{7.4}$$

que nos dice que el *wumpus* está vivo en el instante t *si y sólo si* no se percibe ningún grito en el instante t y estaba vivo en el instante $t-1$. Asumimos que el circuito se inicializa con el registro *Vivo* asignado a *verdadero*. Por lo tanto, *Vivo* permanecerá siendo verdadero hasta que haya un grito, con lo que se convertirá y se mantendrá en el valor falso. Esto es exactamente lo que deseamos.

La localización del agente se puede tratar, en mucho, de la misma forma que la salud del *wumpus*. Necesitamos un registro $L_{x,y}$ para cada x e y; su valor debería ser *verdadero* si el agente se encuentra en la casilla $[x, y]$. Sin embargo, el circuito que asigna el valor de $L_{x,y}$ es mucho más complicado que el circuito para el registro *Vivo*. Por ejemplo, el agente está en la casilla [1, 1] en el instante t si: (a) estaba allí en el instante $t-1$ y no se movió hacia delante o lo intentó pero tropezó con un muro; o (b) estaba en la casilla [1, 2] orientado hacia abajo y avanzó; o (c) estaba en la casilla [2, 1] orientado a la izquierda y avanzó:

$$\begin{aligned} L_{1,1}^t \Leftrightarrow \quad & (L_{1,1}^{t-1} \wedge (\neg Avanzar^{t-1} \vee Tropezar^t)) \\ & \vee (L_{1,2}^{t-1} \wedge (OrientadoAbajo^{t-1} \wedge Avanzar^{t-1})) \\ & \vee (L_{2,1}^{t-1} \wedge (OrientadoIzquierda^{t-1} \wedge Avanzar^{t-1})) \end{aligned} \tag{7.5}$$

En la Figura 7.21 se muestra el circuito para $L_{1,1}$. Cada registro de localización tiene enlazado un circuito similar a éste. En el Ejercicio 7.13(b) se pide diseñar un circuito para las proposiciones de orientación.

Los circuitos de las Figuras 7.20 y 7.21 mantienen los valores correctos de los registros *Vivo* y $L_{x,y}$ en todo momento. Sin embargo, estas proposiciones son extrañas, en el sentido de que *sus valores de verdad correctos siempre se pueden averiguar*. En lu-

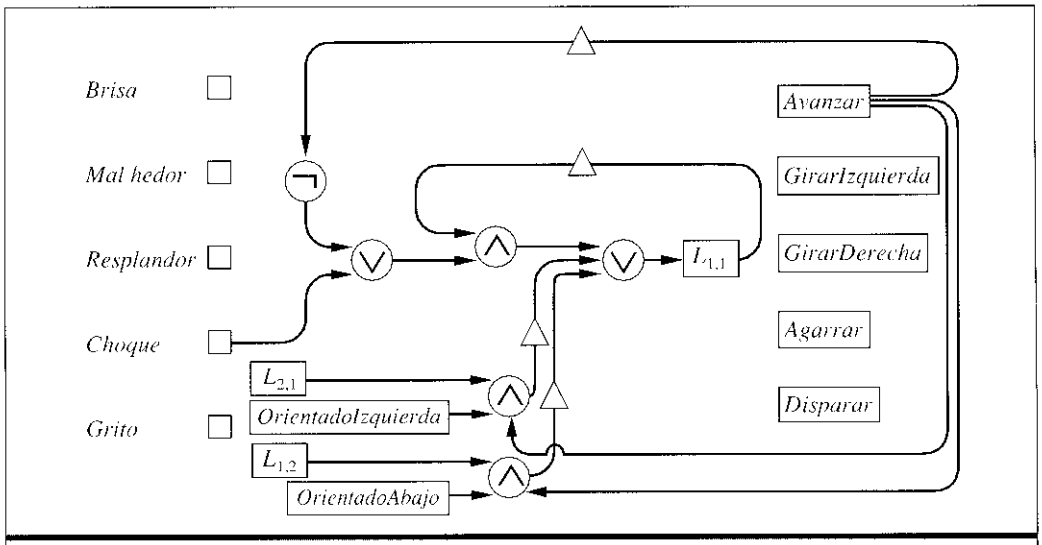

Figura 7.21 El circuito para determinar si el agente está en la casilla [1, 1]. Cada registro de localización y de orientación tiene enlazado un circuito similar.

gar de ello, consideremos la proposición $B_{4,4}$: en la casilla [4, 4] se percibe una brisa. Aunque el valor de verdad de esta proposición se mantiene fijo, el agente no puede aprender su valor hasta que haya visitado la casilla [4, 4] (o haya deducido que hay un hoyo cercano). Las lógicas proposicional y de primer orden están diseñadas para representar proposiciones verdaderas, falsas, o inciertas, de forma automática. Los circuitos no pueden hacerlo: el registro $B_{4,4}$ debe contener *algún* valor, bien *verdadero* o *falso*, aun antes de que se descubra su valor de verdad. El valor del registro bien podría ser el erróneo, y esto nos llevaría a que el agente se extraviara. En otras palabras, necesitamos representar tres posibles estados (se sabe que $B_{4,4}$ es verdadero, falso, o desconocido) y sólo tenemos un *bit* para representar estos estados.

La solución a este problema es utilizar dos bits en vez de uno. $B_{4,4}$ se puede representar mediante dos registros, a los que llamaremos $K(B_{4,4})$ y $K(\neg B_{4,4})$, en donde K significa «conocido». (¡Tenga en cuenta que esta representación consiste tan sólo en unos símbolos con nombres complicados, aunque parezcan expresiones estructuradas!) Cuando ambas, $K(B_{4,4})$ y $K(\neg B_{4,4})$, son falsas, significa que el valor de verdad de $B_{4,4}$ es desconocido. (¡Si ambas son ciertas, entonces la base de conocimiento tiene un fallo!) A partir de ahora, utilizaríamos $K(B_{4,4})$ en vez de $B_{4,4}$, siempre que ésta aparezca en alguna parte del circuito. Por lo general, representamos cada proposición que es potencialmente indeterminada mediante dos **proposiciones acerca del conocimiento** para especificar si la proposición subyacente se sabe que es verdadera y se sabe que es falsa.

PROPOSICIONES ACERCA DEL CONOCIMIENTO

En breve veremos un ejemplo de cómo utilizar las proposiciones acerca del conocimiento. Primero necesitamos resolver cómo determinar los valores de verdad de las propias proposiciones acerca del conocimiento. Fíjese en que, mientras $B_{4,4}$ tiene un valor de verdad fijo, $K(B_{4,4})$ y $K(\neg B_{4,4})$ *cambian* a medida que el agente descubre más cosas acerca del mundo. Por ejemplo, $K(B_{4,4})$ comienza siendo falsa y entonces se transforma en verdadera tan pronto como se puede determinar $B_{4,4}$ que es verdadera (es decir, cuan-

do el agente está en la casilla [4, 4] y detecta una brisa). A partir de entonces permanece siendo verdadera. Así tenemos

$$K(B_{4,4})^t \Leftrightarrow K(B_{4,4})^{t-1} \vee (L^t_{4,4} \wedge Brisa^t) \tag{7.6}$$

Se puede escribir una ecuación similar para $K(\neg B_{4,4})^t$.

Ahora que el agente sabe acerca de las casillas con brisa, puede ocuparse de los hoyos. La ausencia de un hoyo en una casilla se puede averiguar si y sólo si se sabe que en una de sus casillas vecinas no hay ninguna brisa. Por ejemplo, tenemos

$$K(\neg H_{4,4})^t \Leftrightarrow K(\neg B_{3,4})^t \vee K(\neg B_{4,3})^t \tag{7.7}$$

Determinar si *hay* un hoyo en una casilla es más difícil, debe haber una brisa en una casilla adyacente que no sea causada por otro hoyo:

$$\begin{aligned} K(H_{4,4})^t \quad \Leftrightarrow \quad & (K(B_{3,4})^t \wedge K(\neg H_{2,4})^t \wedge K(\neg H_{3,3})^t) \\ \vee \quad & (K(B_{4,3})^t \wedge K(\neg H_{4,2})^t \wedge K(\neg H_{3,3})^t) \end{aligned} \tag{7.8}$$

Mientras que utilizar los circuitos para determinar la presencia o ausencia de hoyos puede resultar algo peliagudo, *sólo se necesitan un número constante de puertas para cada casilla*. Esta propiedad es esencial si debemos construir agentes basados en circuitos que se pueden ampliar de forma razonable. En realidad ésta es una propiedad del propio mundo de *wumpus*; decimos que un entorno manifiesta **localidad** si el valor de verdad de una proposición relevante se puede obtener observando sólo un número constante de otras proposiciones. La localidad es muy sensible a la «física» precisa del entorno. Por ejemplo, el dominio de los dragaminas (Ejercicio 7.11) no es un dominio localista, porque determinar si una mina se encuentra en una casilla dada puede requerir observar las casillas que están arbitrariamente lejos. Los agentes basados en circuitos no son siempre practicables para los dominios no localistas.

ACÍCLICO

Hay un asunto por el que hemos caminado de puntillas y con mucho cuidado: el tema es la propiedad de ser **acíclico**. Un circuito es acíclico si cada uno de sus caminos en el que se conecta hacia atrás la salida de un registro con su entrada se realiza mediante un elemento de retardo. ¡Necesitamos que todos los circuitos sean acíclicos porque los que no lo son, al igual que los artefactos físicos, no funcionan! Éstos pueden entrar en oscilaciones inestables produciendo valores indefinidos. Como ejemplo de un circuito cíclico, tenga en cuenta la siguiente ampliación de la Ecuación (7.6):

$$K(B_{4,4})^t \Leftrightarrow K(B_{4,4})^{t-1} \vee (L^t_{4,4} \wedge Brisa^t) \vee K(H_{3,4})^t \vee K(H_{4,3})^t \tag{7.9}$$

Los disyuntores extras, $K(H_{3,4})^t$ y $K(H_{4,3})^t$, permiten al agente determinar si hay brisa a partir del conocimiento de la presencia de hoyos en las casillas adyacentes, algo que parece totalmente razonable. Pero ahora, desafortunadamente, tenemos que la detección de la brisa depende de los hoyos adyacentes, y la detección de los hoyos depende de las brisas adyacentes, si tenemos en cuenta la Ecuación (7.8). Por lo tanto el circuito, en su conjunto, tendría ciclos.

La dificultad no es que la Ecuación (7.9) ampliada sea *incorrecta*. Más bien, el problema consiste en que las dependencias entrelazadas que se presentan en estas ecuaciones no se pueden resolver por el simple mecanismo de la propagación de los valores de

verdad en el correspondiente circuito Booleano. La versión acíclica utilizando la Ecuación (7.6), que determina si hay brisa sólo a partir de las observaciones directas en la casilla es *incompleta*, en el sentido de que en algunos puntos, el agentes basado en circuitos podría saber menos que un agente basado en inferencia utilizando un procedimiento de inferencia completo. Por ejemplo, si hay una brisa en la casilla [1, 1], el agente basado en inferencia puede concluir que también hay una brisa en la casilla [2, 2], mientras que el agente basado en circuitos no puede hacerlo, utilizando la Ecuación (7.6). *Se puede* construir un circuito completo (después de todo, los circuitos secuenciales pueden emular cualquier computador digital) pero sería algo significativamente más complejo.

Una comparación

Los agentes basados en inferencia y los basados en circuitos representan los extremos declarativo y procesal en el diseño de agentes. Se pueden comparar según diversas dimensiones:

- *Precisión*: el agente basado en circuitos, a diferencia del agente basado en inferencia, no necesita disponer de copias diferentes de su «conocimiento» para cada instante de tiempo. En vez de ello, éste sólo se refiere al instante actual y al previo. Ambos agentes necesitan copias de la «física» de cada casilla (expresada mediante sentencias o circuitos), y por lo tanto, no se amplían adecuadamente a entornos mayores. En entornos con muchos objetos relacionados de forma compleja el número de proposiciones abrumará a cualquier agente proposicional. Este tipo de entornos requiere del poder expresivo de la lógica de primer orden. (*Véase* Capítulo 8.) Además, ambos tipos de agente proposicional están poco preparados para expresar o resolver el problema de encontrar un camino a una casilla segura que esté cercana. (Por esta razón, el AGENTE-WUMPUS-LP recurre a un algoritmo de búsqueda.)
- *Eficiencia computacional*: en el *peor de los casos*, la inferencia puede tomar un tiempo exponencial respecto al número de símbolos, mientras que evaluar un circuito toma un tiempo lineal respecto al tamaño del circuito (o lineal respecto a la *intensidad de integración* del circuito, si éste se construye como un dispositivo físico). Sin embargo, vemos que en la *práctica*, el DPLL realiza las inferencias requeridas bastante rápidamente[14].
- *Completitud*: anteriormente insinuamos que el agente basado en circuitos podría ser incompleto, debido a la restricción de que sea acíclico. Las causas de la incompletitud son en realidad más básicas. Primero, recordemos que un circuito se ejecuta en tiempo lineal respecto a su tamaño. Esto significa que, para algunos entornos, un circuito que sea completo (a saber, uno que calcula el valor de verdad de cada proposición determinable) debe ser exponencialmente más grande que la *BC* de un agente basado en inferencia. Dicho de otro modo, deberíamos poder resolver el problema de la implicación proposicional en menor tiempo que el tiempo exponencial, lo que parece improbable. Una segunda causa es la naturaleza del

[14] De hecho, ¡todas las inferencias hechas por un circuito se pueden hacer por el DPLL en tiempo lineal! Esto es debido a que la evaluación de un circuito, al igual que el encadenamiento hacia delante, se puede emular mediante el DPLL, utilizando la regla de propagación unitaria.

estado interno del agente. El agente basado en inferencia recuerda cada percepción que ha tenido, y conoce, bien implícita o explícitamente, cada sentencia que se sigue de las percepciones y la *BC* inicial. Por ejemplo, dado $B_{1,1}$, el agente conoce la disyunción $H_{1,2} \vee H_{2,1}$, de lo cual se sigue $B_{2,2}$. Por el otro lado, el agente basado en circuitos olvida todas sus percepciones anteriores y recuerda tan sólo las proposiciones individuales almacenadas en los registros. De este modo, $H_{1,2}$ y $H_{2,1}$ permanecen desconocidas *cada una de ellas* aún después de la primera percepción, así que no se sacará ninguna conclusión acerca de $B_{2,2}$.

- *Facilidad de construcción*: este es un aspecto muy importante acerca del cual es difícil ser precisos. Desde luego, los autores encontramos mucho más fácil especificar la «física» de forma declarativa, mientras que idear pequeños circuitos acíclicos, y no demasiado incompletos, para la detección directa de hoyos, nos ha parecido bastante dificultoso.

En suma, parece que hay *compensaciones* entre la eficiencia computacional, la concisión, la completitud y la facilidad de construcción. Cuando la conexión entre las percepciones y las acciones es simple (como en la conexión entre *Resplandor* y *Agarrar*) un circuito parece que es óptimo. En un dominio como el ajedrez, por ejemplo, las reglas declarativas son concisas y fácilmente codificables (como mínimo en la lógica de primer orden), y en cambio, utilizar un circuito para calcular los movimientos directamente a partir de los estados del tablero, sería un esfuerzo inimaginablemente enorme.

Podemos observar estos diferentes tipos de compensaciones en el reino animal. Los animales inferiores, con sus sistemas nerviosos muy sencillos, quizá estén basados en circuitos, mientras que los animales superiores, incluyendo a los humanos, parecen realizar inferencia con base en representaciones explícitas. Esta característica les permite ejecutar funciones mucho más complejas de un agente. Los humanos también disponen de circuitos para implementar sus reflejos, y quizá, también para **compilar** sus representaciones declarativas en circuitos, cuando ciertas inferencias pasan a ser una rutina. En este sentido, el diseño de un **agente híbrido** (*véase* Capítulo 2) podría poseer lo mejor de ambos mundos.

COMPILACIÓN

7.8 Resumen

Hemos introducido los agentes basados en conocimiento y hemos mostrado cómo definir una lógica con la que los agentes pueden razonar acerca del mundo. Los principales puntos son los siguientes:

- Los agentes inteligentes necesitan el conocimiento acerca del mundo para tomar las decisiones acertadas.
- Los agentes contienen el conocimiento en forma de **sentencias** mediante un **lenguaje de representación del conocimiento**, las cuales quedan almacenadas en una **base de conocimiento**.
- Un agente basado en conocimiento se compone de una base de conocimiento y un mecanismo de inferencia. El agente opera almacenando las sentencias acerca del mundo en su base de conocimiento, utilizando el mecanismo de inferencia para