Agentes que razonan con lógica

Tema 5

Esquema

- ♢ Introducción
- ♢ Agentes basados en el conocimiento
- ♢ El "mundo del Wumpus"
- ♢ Lenguajes de representación y lenguajes lógicos
- ♢ Lógica proposicional
- ♢ Un ejemplo de agente proposicional: un agente para el "mundo del Wumpus"

Introducción

Los agentes basados en objetivos están mejor preparados para resolver problemas que los agentes reflejos simples

Vamos a ver cómo dotando a los agentes de razonamiento lógico pueden obtener un buen desempeño realizando tareas más difíciles

Un **agente lógico** o **basado en el conocimiento** parte de conocimiento y utiliza el razonamiento lógico para decidir las acciones que le llevarán a su objetivo

Un agente basado en el conocimiento necesita conocimiento muy variado:

- sobre el mundo: estado actual, cómo inferir sus propiedades a partir de percepciones, cómo evoluciona, ...
- sobre el objetivo a conseguir
- sobre las acciones a realizar dependiendo de las circunstancias

Agentes basados en el conocimiento

Un componente fundamental del agente basado en el conocimiento es la **base de conocimiento** o KB (de *knowledge base*)

La base de conocimiento está formada por un conjunto de **sentencias**,

donde una sentencia es la representación de un hecho del mundo en un **lenguaje de representación del conocimiento**

Un agente basado en el conocimiento necesita mecanismos para introducir sentencias en la KB y para realizar consultas sobre el contenido de la KB

Otro componente fundamental es el **mecanismo de inferencia**, que debe asegurar que la respuesta a cualquier consulta sobre la KB siga/se derive de lo que ha sido introducido en la KB

El agente se construye añadiendo a la KB sentencias que representan el conocimiento que posee el diseñador del problema: **enfoque declarativo**

El esquema de un agente basado en el conocimiento genérico:

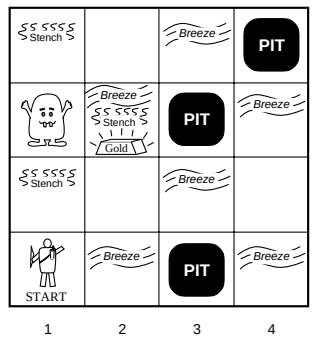
```
function KB-AGENT( percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

notar que

- el esquema puede hacer pensar en un agente con estado interno
- TELL sirve para introducir sentencias y ASK para realizar consultas
- los detalles del lenguaje de representación y del mecanismo de inferencia están ocultos

El "mundo del Wumpus"

El mundo es una cueva de 4×4
En los recuadros puede haber sorpresas: oro, un precipicio (PIT) o el Wumpus
Objetivo: encontrar el oro y volver a la posición inicial
Percepciones: $Stench$ (hedor), $Breeze$ (corriente), $Glitter$ (brillo), $Bump$ (golpe), ...
Acciones: $Leftturn$, $Rightturn$, $Forward$, $Grab$ (coger), $Release$ (soltar), ...



Entorno

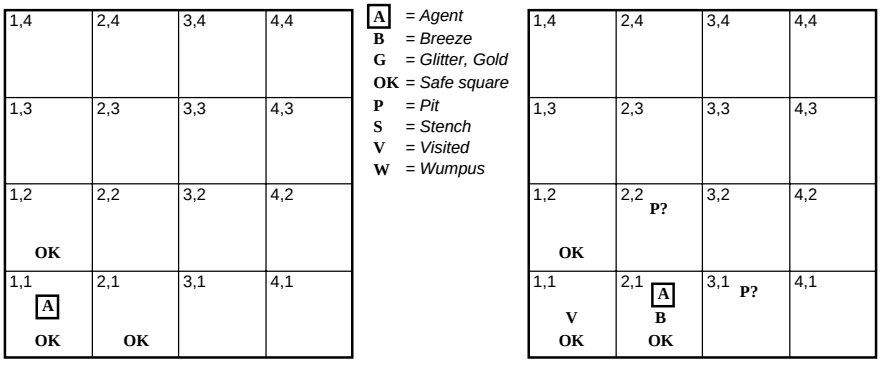
- en los recuadros adyacentes al Wumpus hay hedor,
- en los recuadros adyacentes a un precipicio hay corriente de aire,
- en los recuadros que guardan oro hay brillo, ...

No existe una secuencia de acciones que garantice llegar al objetivo desde distintas configuraciones

Caracterización del "mundo del Wumpus":

Es determinista?? Sí —los resultados están claramente especificados
Es accesible?? No —sólo percepción local
Es estático?? Sí —el Wumpus y los precipicios no se mueven
Es discreto?? Sí

cómo actuar en el "mundo del Wumpus"

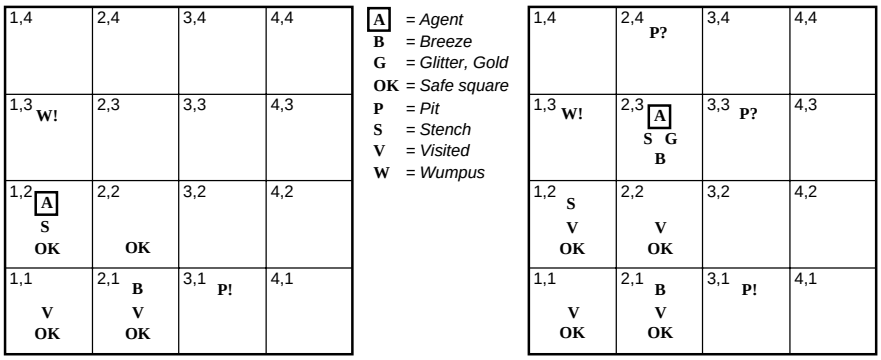


(a) (b)

Conocimiento del agente en la posición inicial después de la percepción
$[None, None, None, None, None]$

. . . y después de un movimiento y de
$[None, Breeze, None, None, None]$

cómo actuar en el "mundo del Wumpus" (cont.)



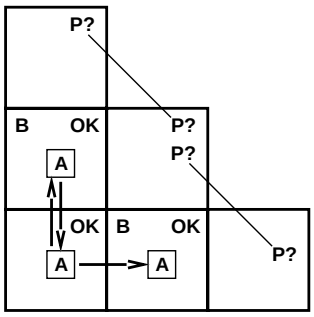
(a) (b)

Conocimiento del agente después del tercer movimiento y de la percepción $[Stench, None, None, None, None]$

... y después del quinto y de $[Stench, Breeze, Glitter, None, None]$

$\rightsquigarrow$ Necesitamos que el agente pueda representar conocimiento como "hay un precipicio en $[2,2]$ o en $[3,1]$" y realizar con él inferencias

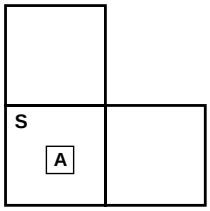
otros movimientos en el "mundo del Wumpus"



$Breeze$ en $[1,2]$ y en $[2,1]$

$\Rightarrow$ no existen posiciones seguras

Suponiendo una distribución uniforme, es más probable que haya un precipicio en $[2,2]$



$Stench$ en $[1,1]$

$\Rightarrow$ no podemos movernos

Podemos usar una estrategia de coerción:

disparar hacia adelante

si el Wumpus estaba allí $\Rightarrow$ muerto $\Rightarrow$ segura

si el Wumpus no estaba allí $\Rightarrow$ segura

Lenguajes de representación y lenguajes lógicos

Podríamos intentar utilizar lenguajes que conocemos:
Lenguajes de programación, idóneos para describir algoritmos y estructuras de datos:

- ✓ "hay un precipicio en $[2,2]$": $World[2,2] \leftarrow Pit$
- × no se puede expresar "hay un precipicio en $[2,2]$ o en $[3,1]$"

Lenguajes naturales, idóneos para la comunicación:

- ✓ muy expresivos
- × el significado de las sentencias depende de ellas mismas y del contexto en que se utilizan, p.e. *"¡Mira!"*
- × existen sentencias ambigüas, p.e. *"Perros y gatos pequeños"*

Un buen lenguaje de representación del conocimiento deberá ser expresivo y conciso, independiente del contexto y no ambigüo; también deberá tener un mecanismo de inferencia asociado

Se han diseñados muchos lenguajes lógicos con estas características

Todo lenguaje de representación está definido por dos aspectos:

- la **sintaxis** describe las configuraciones que constituyen sentencias
- la **semántica** determina los hechos del mundo a los que hacen referencia las sentencias, o **significado** de las sentencias, a través de una **interpretación**. Con el significado se puede determinar el **valor de verdad** de las sentencias

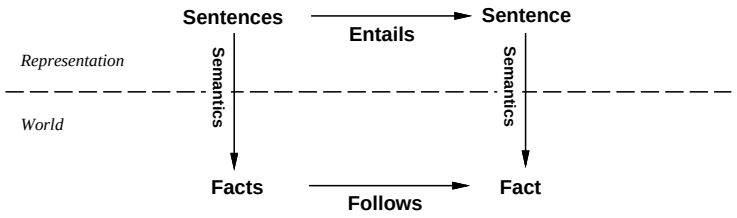
Por ejemplo, en el lenguaje de la aritmética:

- $x+2 \geq y$ es una sentencia, pero $x2+y >$ no
- $x+2 \geq y$ es cierta si y sólo si el número $x+2$ no es menor que el y
- $x+2 \geq y$ es cierta con la interpretación $x=7,\ y=1$

Si la sintaxis y la semántica están definidas de manera precisa diremos que el lenguaje es una **lógica**

A partir de la sintaxis y la semántica se puede derivar un mecanismo de inferencia para utilizarlo en combinación con el lenguaje

lenguajes lógicos



Los hechos forman parte del mundo, mientras que las sentencias son la representación que de ellos tiene el agente

Queremos generar sentencias que sean necesariamente ciertas a partir de otras que también lo son (*entails*), de igual manera que algunos hechos siguen/se derivan de otros (*follows*)
$\rightsquigarrow$ necesitamos implementar la relación de **implicación** (*entailment*) mediante un **procedimiento de inferencia**

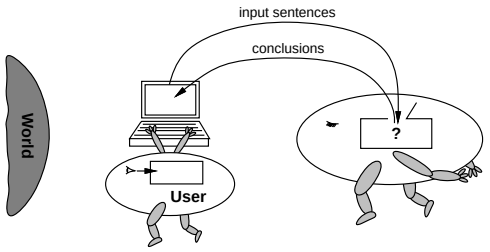
Un procedimiento de inferencia que genere sólo sentencias ciertas a partir de sentencias anteriores ciertas se dice que es **robusto** (*sound*)

Una sentencia es **válida** si y sólo si es cierta sea cual sea su significado y sea cual sea el estado del mundo, p.e.

"hay hedor en $[1,1]$ o no hay hedor en $[1,1]$"

La validez sirve para que el agente pueda razonar en el "mundo del Wumpus":
si preguntamos "¿es $[2,2]$ OK?" intentará determinar si la sentencia "si la KB es cierta entonces $[2,2]$ es OK" es válida

Normalmente el agente tendrá el conocimiento de la KB pero no nuestra interpretación, y no tendrá acceso al mundo al que se refieren las sentencias:



Lógica proposicional

La lógica proposicional es la más simple pero ilustra las ideas básicas

Supone que existen hechos (**proposiciones**) que pueden darse o no en el mundo, es decir, ser ciertos o falsos

♢ **La sintaxis** es muy simple: los símbolos que se usan son las constantes $True$ y $False$, los símbolos proposicionales como P y Q, las conectivas lógicas $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ y $\neg$, y los paréntesis

Las sentencias se construyen siguiendo las reglas:
- las constantes y los símbolos proposicionales son sentencias
- una sentencia entre paréntesis es una sentencia
- una sentencia puede construirse combinando otras:
  - con $\wedge$ (y), p.e. $P \wedge Q$
  - con $\vee$ (o), p.e. $P \vee Q$
  - con $\Rightarrow$ (implicación), p.e. $(P \wedge Q) \Rightarrow R$
  - con $\Leftrightarrow$ (equivalencia), p.e. $(P \wedge Q) \Leftrightarrow (Q \wedge P)$
  - con $\neg$ (negación), p.e. $\neg P$

Existe un orden de precedencia entre los operadores: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ y $\Leftrightarrow$

♢ **La semántica** es también simple: se define mediante la interpretación de los símbolos y la especificación del significado de las conectivas

Los símbolos pueden tener cualquier significado, p.e. la interpretación de P puede ser "Paris es la capital de Francia" o "el Wumpus está muerto"

Las sentencias compuestas derivan su significado a partir del de sus partes y del de las conectivas

El significado de las conectivas se especifica mediante sus **tablas de verdad**:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

Para definir el significado de sentencias mas complejas procederemos de manera incremental, p.e.

para definir $(P \vee Q) \wedge \neg S$, obtendremos primero el significado de $(P \vee Q)$ y el de $\neg S$

$\diamondsuit$ **Un procedimiento de inferencia**: las tablas de verdad se pueden utilizar como método para comprobar la validez de una sentencia

Por ejemplo, para comprobar la validez de $((P \vee H) \wedge \neg H) \Rightarrow P$

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

notar que en la tabla hay una fila por cada combinación de valores de verdad de los símbolos de la sentencia

$\diamondsuit$ **Otro procedimiento de inferencia**: aplicación de las reglas de inferencia

Una **regla de inferencia** es un "patrón" de inferencia que se utiliza a menudo y cuya robustez (*soundness*) se ha probado

Las reglas de inferencia más comunes:

- Modus Ponens o eliminación de la implicación: $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$
- Eliminación del "y": $\dfrac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$
- Introducción del "y": $\dfrac{\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$
- Introducción del "o": $\dfrac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$
- Eliminación de la doble negación: $\dfrac{\neg\neg\alpha}{\alpha}$

Las reglas de inferencia más comunes (cont.):

- Resolución unitaria: $\dfrac{\alpha \vee \beta,\ \neg\beta}{\alpha}$
- Resolución: $\dfrac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$

Siguiendo con ejemplo, para probar que P se deriva de $(P \vee H)$ y $\neg H$:

aplicar la regla de resolución unitaria con P en lugar de α y H en lugar de β

¿Qué procedimiento de inferencia debe utilizar un agente proposicional?
El método basado en las tablas de verdad

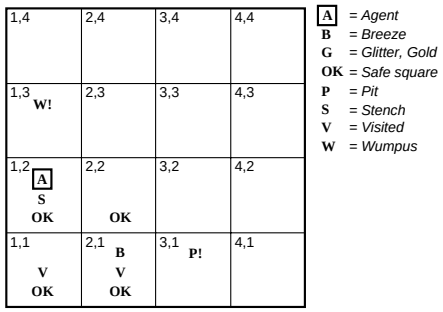
es completo, porque siempre es posible enumerar las 2^n filas de una tabla de n símbolos

pero tiene un coste exponencial con n

Afortunadamente se conocen clases de sentencias para las que existen métodos de inferencia de coste polinómico

Un ejemplo de agente proposicional

Supondremos que el agente ha llegado al punto:



La KB contendrá, entre otras, las siguientes sentencias derivadas de sus percepciones:

$\neg S_{1,1}$ $\neg B_{1,1}$
$\neg S_{2,1}$ $B_{2,1}$
$S_{1,2}$ $\neg B_{1,2}$

Además, la KB incluirá conocimiento del entorno, p.e.

"si una posición no huele, ni la posición ni sus adyacentes alojan al Wumpus", ésto para cada recuadro:

R1: $\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
R2: $\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$
R3: $\neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$
...

o "si una posición huele, el Wumpus debe estar en alguna de las posiciones adyacentes", para cada recuadro:

R4: $S_{1,2} \Rightarrow W_{1,1} \vee W_{2,2} \vee W_{1,3}$
...

Dadas las sentencias anteriores el agente puede concluir $W_{1,3}$ mediante una tabla de verdad ($\times$) o mediante la aplicación de reglas de inferencia ($\checkmark$)

Para decidir cuál es la acción correcta una vez hemos inferido información útil, la KB incluirá conocimiento sobre las acciones a realizar, p.e.

"si el agente está delante del Wumpus, no debe ir hacia adelante", ésto para cada recuadro y para cada orientación:

R5: $A_{1,1} \wedge East_A \wedge W_{2,1} \Rightarrow \neg Forward$
...

Con estas reglas podremos consultar a la KB sobre qué acción debemos llevar a cabo, p.e. "¿debería ir hacia adelante?"
notar que no se pueden hacer consultas del tipo "¿qué acción debería realizar?" porque la lógica no lo permite

El esquema para un agente basado en lógica proposicional:

```
function PROPOSITIONAL-KB-AGENT( percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
  for each action in the list of possible actions do
      if ASK(KB, MAKE-ACTION-QUERY(t, action)) then
          t ← t + 1
          return action
  end
```

La lógica proposicional tiene muchos inconvenientes:

- conocimiento sencillo requiere un gran número de reglas, p.e. R5 y versiones similares
- no es sencillo reflejar cambios a lo largo del tiempo, p.e. si queremos seguir la traza de las posiciones por las que pasa el agente necesitamos muchos símbolos ($A^0_{1,1}$ ...) y muchas más reglas