



SISTEMAS INTELIGENTES

T5: Representación del Conocimiento

{jdiez, juanjo} @ aic.uniovi.es



Índice

- Necesidad de representar el conocimiento
- Agentes basados en el conocimiento
- Representación de conocimiento:
 - Enfoques
 - Características deseables
 - Modelos de representación
- Lenguajes de representación
 - Lógica proposicional
 - Inferencia
 - Resolución
 - Lógica de predicados
 - Unificación



La necesidad del conocimiento

- No puede hablarse de **inteligencia**, sin hablar antes de **conocimiento**
- Todos los sistemas relacionados con la IA manejan conocimiento de una u otra forma
- Ejemplo: La resolución de problemas por técnicas de búsqueda en espacios de estados, aunque no lo parezca, encierra conocimiento en:
 - la representación de los estados,
 - en la función sucesores, y
 - en la función heurística h
- En el caso de las técnicas de búsqueda se trata de representaciones ad-hoc, adaptadas a cada problema
- Nos interesan las técnicas generales de representación



Objetivos

- **Objetivo 1: Estudiar técnicas generales de representación del conocimiento**
- **Objetivo 2: Estudiar nuevas estrategias de resolución de problemas**

El uso explícito de conocimiento es la diferencia fundamental entre la IA y la informática convencional

“El papel de la representación del conocimiento en IA es reducir los problemas de construcción de sistemas inteligentes a problemas de búsqueda”

(Gingsberg, 1993)



IA vs Computación tradicional (I)

<i>Computación tradicional</i>	<i>Inteligencia Artificial</i>
Idear el algoritmo	Identificar el conocimiento necesario para resolver el problema
Seleccionar un lenguaje de programación	Seleccionar un lenguaje de representación del conocimiento
Escribir el algoritmo en dicho lenguaje	Escribir el conocimiento en dicho lenguaje
Ejecutar el programa	Usar las consecuencias del conocimiento para resolver el problema

En IA los pasos más difíciles son el primero y el tercero
En el cuarto intervienen las estrategias de búsqueda



IA vs Computación tradicional (II)

Diseñar un programa que dados los trabajadores de una compañía y dos trabajadores X e Y, diga si X es un superior de Y

```
boolean superior(individuo X, individuo Y, conjunto C) {  
    if ( jefe(X,Y) ) return true;  
    else {  
        C = sacar(C,X); C = sacar(C,Y);  
        while ( !vacio(C) ) {  
            Z = buscar_en(C);  
            if ( jefe(X,Z) && superior(Z,Y) ) return true;  
            C = sacar(C,Z);  
        }  
        return false;  
    }  
}
```

```
superior(X,Y)    :- jefe(X,Y).  
superior(X,Y)    :- jefe(X,Z), superior(Z,Y).
```



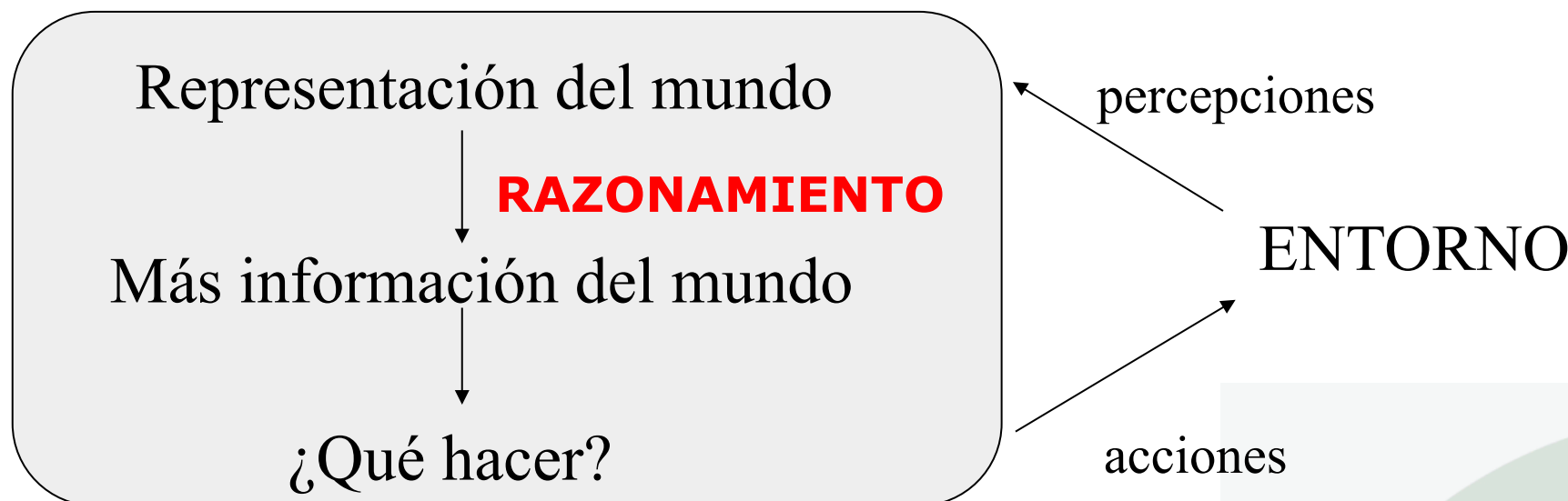
El Agente Inteligente Perfecto

- Para construir un Agente Inteligente autónomo y capaz de evolucionar con el tiempo, debería estar dotado de tres cualidades fundamentales:
 1. Representar conocimiento del mundo que le rodea
 2. Razonar para generar nuevo conocimiento a partir del conocimiento del que ya dispone
 3. Aprender nuevo conocimiento de forma automática a partir de las observaciones que obtiene del entorno



Agentes basados en conocimiento (I)

Agente que razona lógicamente





Agentes basados en conocimiento (II)

- Capacidades:
 - Aceptar nuevas tareas
 - Ser rápidamente competente (usando el conocimiento o aprendiendo)
 - Adaptarse a cambios en el entorno
- Necesidades:
 - Conocer el estado actual del mundo
 - Cómo inferir propiedades no vistas
 - La evolución del mundo con el tiempo
 - Acciones realizables en distintas circunstancias
- Componentes:
 - Base de conocimientos (BC)
 - » {"sentencias" en un *lenguaje de representación*}
 - » Tareas de interfaz: TELL, ASK
 - Mecanismo de inferencia



Enfoques para la representación de conocimiento

- Planteamiento Conexionista

Emulan el sistema neuronal humano

- Planteamiento Simbólico

Cada elemento de la representación (símbolo) se refiere a un objeto (particular o abstracto) del mundo a representar

La representación de conocimiento es un proceso de transformación

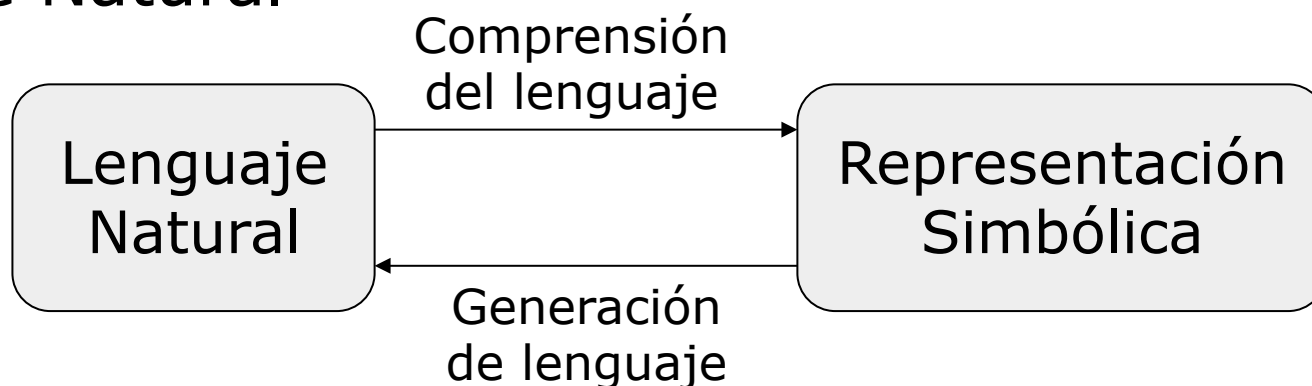
Los algoritmos utilizan la representación simbólica y generan nuevos símbolos



Representación e interpretación no siempre son biunívocas

Planteamiento Simbólico

- Lenguaje Natural



- Lógica de predicados:

$\forall x, \text{perro}(x) \Rightarrow \text{tiene-cola}(x)$

puede interpretarse como:

- Todos los perros tienen cola
- Cada perro tiene cola
- Si algo es perro, entonces tiene cola
- ... pero no queda claro si todos los perros tienen la misma cola, cada uno una distinta o una solución intermedia



Propiedades deseables de los sistemas de representación (I)

- Un buen lenguaje de representación debe combinar las ventajas de los lenguajes naturales y las de los lenguajes formales
- Expresivo y conciso
- No ambiguo
- Independiente del contexto
- Efectivo



Propiedades deseables de los sistemas de representación (II) [Rich, Knight 91]

- **Adecuación de la representación:** todo conocimiento pertinente debe ser representable
- **Adecuación de la inferencia:** posibilidad de manejar las estructuras de conocimiento de tal forma que se puedan derivar de ellas nuevas estructuras que correspondan a nuevos conocimientos inferidos de los antiguos
- **Eficiencia de la inferencia:** posibilidad de incorporar en la representación heurísticos o guías para hacer más eficiente la inferencia
- **Eficiencia de la adquisición:** posibilidad para incorporar fácilmente nuevo conocimiento, manual o automáticamente



Modelos de representación (I)

1) Declaraciones estáticas

- Relaciones tipo “bases de datos”
- No reflejan explícitamente relaciones
- No es conocimiento desde la perspectiva de la Inteligencia Artificial, pero es útil

¿quiénes están conectados a una red inalámbrica y tienen un procesador de 2GHz?

	GHz.	Wi-Fi	RAM
Pepe	1.66	SI	512 Mb
Juan	2.00	SI	512 Mb
Luis	2.00	NO	512 Mb



Modelos de representación (II)

2) Conocimiento estructurado

- Tipos:
 - **Representaciones basadas en lógica**
 - Representaciones con herencia de propiedades
 - » Redes semánticas
 - » Marcos (frames)
 - Reglas de producción (tema siguiente)
- Facilitan los procesos de razonamiento



Representaciones basadas en lógica (I)

- Modelo (funcional) para los razonamientos humanos
- Alto grado de formalización
 - Sintáctico:
 - Cómo construir sentencias legales
 - Qué símbolos se pueden usar
 - Semántico:
 - Cómo se interpretan las sentencias lógicas, es decir, qué significan
- Ejemplo: “todos los estudiantes aprueban”
 - $\text{estudiante}(x) \Rightarrow \text{aprueba}(x)$
 - Sentencia válida
 - Entendemos el significado
 - La sentencia es falsa, hay contraejemplos



Representaciones basadas en lógica (II)

- Separación conocimiento/razonamiento
- Mecanismos de inferencia potentes y fiables
- Otros lenguajes se pueden expresar en él
- Suficientemente expresivo para muchos dominios
- Inconvenientes:
 - A veces “demasiado expresivo”
 - A veces “se queda corto”
 - Eficiencia



Representaciones basadas en lógica (III)

- **Lógica de proposiciones (orden 0)**

- Utiliza **proposiciones, conectivas** (\neg \wedge \vee \Leftrightarrow \Leftarrow \Rightarrow), **paréntesis, *cierto* y *falso***

- Ej: “Siempre me mojo y me irrito cuando llueve”

- $$\text{llueve} \Rightarrow \text{me-mojo} \wedge \text{me-irrito}$$

- **Lógica de predicados (orden 1)**

- Más expresiva que la lógica de orden 0

- Se usan también **constantes, variables, predicados, funciones y cuantificadores** (\forall, \exists)

- Ej: “Cada lunes y jueves voy a casa de Juanjo a cenar”

- $$\forall X \ (\text{día}(X, \text{lunes}) \vee \text{día}(X, \text{miércoles})) \Rightarrow$$
$$(\text{ir}(\text{yo}, \text{casa-de}(\text{juanjo})) \wedge \text{tomar}(\text{yo}, \text{cena}))$$

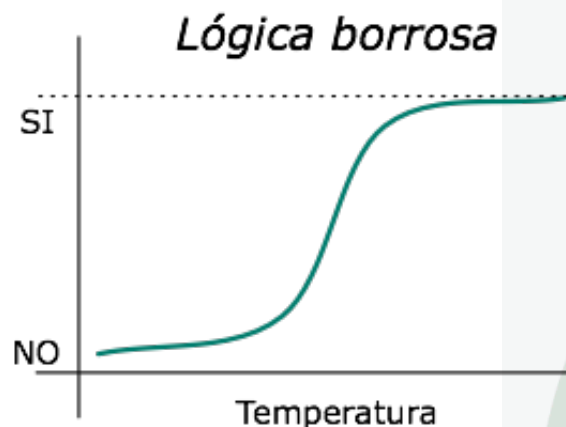
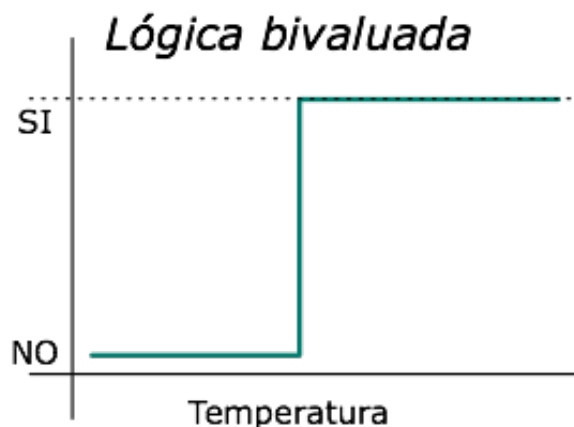
Representaciones basadas en lógica (IV)

- **Extensiones**

- no monótona: razonamiento por defecto, mantenimiento de la verdad
- multivaloradas y borrosa: imprecisión e incertidumbre
- modales: creencias, deseos, intenciones. . .

- **Lógica Borrosa (fuzzy logic)**

A veces es difícil asignar un valor de verdad a una sentencia
"El paciente tiene fiebre", pero ¿tiene mucha, poca...?





Razonamiento Automático (I)

- Representar estructuras con conocimiento adquiere su importancia cuando podemos obtener nuevo conocimiento. Esto es lo que se llama **Inferencia**
- Hay tres formas de Inferencia: Deducción, Inducción y Abducción

1. Deducción

- Son las reglas de inferencia de la lógica. Una forma de deducción es la regla de la Resolución o la del Modus Ponens
- **El nuevo conocimiento es cierto si se parte de otro conocimiento cierto:** esta es la fuerza de la inferencia lógica y, por lo tanto, de la lógica
- Ejemplo

$$\left. \begin{array}{l} \forall X (p(X) \rightarrow q(X)) \\ p(a) \end{array} \right\} \Rightarrow q(a)$$



Razonamiento Automático (II)

2. Inducción

- Son las reglas que permiten generalizar observaciones para así sintetizar conocimiento de más alto nivel. Es el mecanismo del Aprendizaje Automático

$\text{llovió}(\text{lunes}) \wedge \text{calle_mojada}(\text{lunes})$

$\text{llovió}(\text{martes}) \wedge \text{calle_mojada}(\text{martes})$

..... INDUCCIÓN

$\forall x (\text{llovió}(x) \Rightarrow \text{calle_mojada}(x))$

$\forall x (\text{calle_mojada}(x) \Rightarrow \text{llovió}(x))$

3. Abducción

- Es un método de razonamiento por explicación posible. De un conjunto de conocimiento (reglas y hechos observados) produce un conjunto de explicaciones posibles o hipótesis que harían, usando la deducción, coherente el conocimiento de partida

$B \Rightarrow A$

A

..... ABDUCCIÓN

B




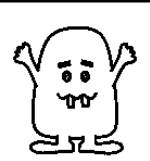

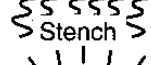
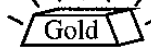





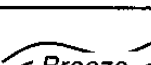




Características de los métodos de inferencia

- Sólido
 - Se dice que un método de inferencia es sólido o mantiene la verdad si sólo deriva conocimiento implicado
- Completo
 - Si puede derivar cualquier conocimiento que está implicado



El entorno "WUMPUS" (I)

4	 Stench		 Breeze	 PIT
3		 Breeze  Stench  Gold	 PIT	 Breeze
2	 Stench		 Breeze	
1	 START	 Breeze	 PIT	 Breeze
	1	2	3	4

Percepciones: (5 sensores)

{Hedor, Brisa, Resplandor, Nada, Nada}

(no golpe, no grito)

El agente no percibe su situación

Acciones:

Ir hacia adelante, girar izda/dcha 90°

Agarrar (estando en la casilla)

Disparar (sólo una flecha)

Objetivo: salir con el oro lo antes posible

1000 ptos: salir con el oro

1 pto penaliza cada acción. Penaliz. Máxima: 10000 ptos



Lógica

- La Lógica es un lenguaje formal y como tal se define mediante:
 - Alfabeto
 - Reglas sintácticas
- Sobre él se desarrollan dos formas de abordar el problema de la deducción/demostración:
- Teoría de modelos (método semántico)
 - Interpretación y reglas de evaluación
 - Fórmula válida, Deducción...
 - Métodos para el estudio de la validez:
 - Tablas de verdad (L0)
 - Árboles semánticos (L0)
 - Refutación...
- Teoría de la demostración (método axiomático)
 - Conjunto de axiomas (finito o numerable)
 - Reglas de inferencia
 - Demostración de teoremas
 - Corrección
 - Completud



Lógica de proposiciones (orden 0) (I)

- Sintaxis:

- Proposiciones primitivas: p, q, r, \dots

- Conectivas lógicas:

- \neg (no) \wedge (y) \vee (o)

- \Rightarrow (implicación) \Leftrightarrow (bicondicional)

- V (Verdadero) F (Falso)

- Sentencias (siendo α y β proposiciones primitivas o sentencias):

- $\neg \alpha, \alpha \wedge \beta, \alpha \vee \beta, \alpha \Rightarrow \beta, \alpha \Leftrightarrow \beta$

- Ejemplos:

p

$\neg p \wedge r$

$q \vee r \rightarrow p \wedge \neg s$



Lógica de proposiciones (II)

- Semántica

- Define las reglas para asociar un valor de verdad (V ó F) a cada sentencia
- Interpretación
 - Correspondencia entre elementos del lenguaje y el mundo a representar
 - Consistirá en asignar el valor V ó F cada proposición
- Reglas de evaluación
 - Significados de las conectivas

- Modelo para una sentencia

- Interpretación para la cual esa sentencia es cierta



Lógica de proposiciones (III)

- Reglas de las conectivas: tabla de verdad

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V



El entorno “WUMPUS” (II)

- Podemos crear una Base de Conocimiento para representar el problema con lógica proposicional
- Primero, escogemos los símbolos proposicionales:
 - H_{ij} es verdadero si hay un hoyo en i,j
 - B_{ij} es verdadero si hay brisa en i,j
 - W_{ij} es verdadero si el wumpus está en i,j
 - M_{ij} es verdadero si huele mal en i,j
- Podemos establecer nuestras primeras sentencias, en virtud de la definición del problema y de las percepciones en las dos primeras casillas (11 y 21)
 - $R_1: \neg H_{11}$
 - $R_2: B_{11} \Leftrightarrow (H_{12} \vee H_{21})$
 - $R_3: B_{21} \Leftrightarrow (H_{11} \vee H_{22} \vee H_{31})$
 - $R_4: \neg B_{11}$
 - $R_5: B_{21}$



Lógica de proposiciones (IV)

- Conceptos fundamentales
 - Una sentencia es:
 - Satisfacible si tiene algún modelo
 - Insatisfacible si no tiene modelos
 - Válida si es V bajo cualquier interpretación ($\models \alpha$)
 - Dos sentencias α y β son:
 - Equivalentes si tiene el mismo valor de verdad bajo cualquier interpretación ($\alpha \equiv \beta$)
 - Equisatisfacibles si α es satisfacible sii β lo es
 - β es consecuencia lógica de α si todo modelo de α lo es de β
 - α es **consecuencia lógica** de un conjunto ϕ si todo modelo de ϕ lo es de α ($\phi \Rightarrow \alpha$ ó $\phi \models \alpha$)
 - No existe ninguna interpretación que haga V a ϕ y F a α
 - $\phi \models \alpha$ si, y sólo si $\phi \cup \{\neg \alpha\}$ es insatisfacible



Lógica de proposiciones (IV)

- Leyes lógicas

- La \equiv es una relación de equivalencia que verifica la propiedad de sustitución: el resultado de reemplazar en α una subfórmula por otra equivalente proporciona una fórmula equivalente a α
- Equivalencias o leyes lógicas:
 - Asociativa ($\vee \wedge$): $\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \vee \gamma$
 - Conmutativa ($\vee \wedge$): $\alpha \vee \beta \equiv \beta \vee \alpha$
 - Distributiva ($\vee \wedge$): $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
 - De Morgan ($\vee \wedge$): $\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$



Lógica de proposiciones (V)

- Leyes lógicas

- Idempotencia: $\alpha \wedge \alpha \equiv \alpha \equiv \alpha \vee \alpha$
- Doble negación: $\neg \neg \alpha \equiv \alpha$
- Absorción: $\alpha \vee (\alpha \wedge \beta) \equiv \alpha \equiv \alpha \wedge (\alpha \vee \beta)$
- Cero y Uno: $\alpha \vee V \equiv V$; $\alpha \vee F \equiv \alpha$; $\alpha \wedge V \equiv \alpha$; $\alpha \wedge F \equiv F$
- Negación: $\alpha \vee \neg F \equiv V$; $\alpha \wedge \neg F \equiv \alpha$
- Eliminación de la implicación: $\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$
- Contraposición: $\alpha \Rightarrow \beta \equiv \neg \beta \Rightarrow \neg \alpha$
- Eliminación bicondicional: $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
- $\neg F \equiv V$
- $\neg V \equiv F$



Reglas de Inferencia

- **Todas las leyes lógicas antes vistas**

- **Modus Ponens**

$\alpha \Rightarrow \beta \wedge \alpha$ se puede inferir β

- **Eliminación de \wedge**

$\alpha \wedge \beta$ se puede inferir α

- Se trata de reglas sólidas, generan inferencias sólidas



El entorno “WUMPUS” (III)

- Tenemos $BC = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$ y vamos a demostrar $\neg H_{12}$

$$BC = \{ \begin{array}{lll} R_1: \neg H_{11} & R_4: \neg B_{11} & R_5: B_{21} \\ R_2: B_{11} \Leftrightarrow (H_{12} \vee H_{21}) & R_3: B_{21} \Leftrightarrow (H_{11} \vee H_{22} \vee H_{31}) & \end{array} \}$$

bicondicionalidad R_2 :

$$R_6: (B_{11} \Rightarrow (H_{12} \vee H_{21})) \wedge ((H_{12} \vee H_{21}) \Rightarrow B_{11})$$

eliminamos \wedge en R_6 :

$$R_7: (H_{12} \vee H_{21}) \Rightarrow B_{11}$$

por equivalencia lógica de contraposición

$$R_8: \neg B_{11} \Rightarrow \neg (H_{12} \vee H_{21})$$

modus ponens con R_4 y

$$R_9: \neg (H_{12} \vee H_{21})$$

ley de Morgan

$$R_{10}: \neg H_{12} \wedge \neg H_{21} \text{ es decir en las casillas 12 y 21 no hay hoyo}$$



La Resolución (I)

- Es un método constructivo que, realizando manipulaciones sintácticas, permite probar la consecuencia lógica

- **Resolución unitaria:**

$\alpha_1 \vee \dots \vee \alpha_k$ **y** β **se infiere** $\alpha_1 \vee \dots \vee \alpha_{i-1} \vee \alpha_{i+1} \vee \dots \vee \alpha_k$
siempre que α_i **y** β **sean complementarios**

Ej1: Sabemos que $p \vee q$ y $\neg p \vee r$ son ciertas, entonces:

- si p es cierto, entonces deducimos r
- si p es falso, entonces deducimos q

Ej2: Si $H_{11} \vee H_{31}$ y $\neg H_{11}$ son ciertas, se infiere H_{31}
(Si hay un hoyo en [11] ó en [31] y no lo hay en [11] entonces está en [31])

- **Resolución generalizada:**

$\alpha_1 \vee \dots \vee \alpha_k$ **y** $\beta_1 \vee \dots \vee \beta_n$ **se infiere**
 $\alpha_1 \vee \dots \vee \alpha_{i-1} \vee \alpha_{i+1} \vee \dots \vee \alpha_k \vee \beta_1 \vee \dots \vee \beta_{j-1} \vee \beta_{j+1} \vee \dots \vee \beta_n$
siempre que α_i **y** β_j **sean complementarios**

Ej1: Si $p \vee q$ y $\neg p \vee r$ son ciertas, entonces deducimos $q \vee r$



El entorno “WUMPUS” (IV)

- El agente se ha movido de 11 a 21, ha vuelto a 11 y se ha movido a 12 donde percibe un hedor y no percibe brisa ($\neg B_{12} M_{12}$). Añadimos a BC

$$R_{11}: \neg B_{12}$$

$$R_{12}: B_{12} \Leftrightarrow (H_{11} \vee H_{22} \vee H_{13})$$

Mediante el mismo proceso que antes podemos deducir:

$$R_{13}: \neg H_{22}$$

$$R_{14}: \neg H_{13}$$

$$R_{15}: H_{11} \vee H_{22} \vee H_{31}$$

Ahora podemos aplicar Resolución combinando las reglas $H_{11} \vee H_{22} \vee H_{31}$ y $\neg H_{22}$ obtenemos

$$R_{16}: H_{11} \vee H_{31}$$

De igual forma si resolvemos $H_{11} \vee H_{31}$ y $\neg H_{11}$ obtenemos

$$R_{17}: H_{31}$$



La Resolución (II)

- Necesita las sentencias en forma causal ($\alpha_1 \vee \dots \vee \alpha_k$): una disyunción de literales
- Factorización: la cláusula resultante contiene sólo una copia de cada literal
- Prueba por refutación

Para demostrar que $BC \models \alpha$, prueba que $BC \cup \{\neg \alpha\} \models F$ (es insatisfacible)
- Correcto y completo

$BC \models \alpha$ si y solo si $BC \cup \{\neg \alpha\} \models F$



Forma Normal Conjuntiva

- La resolución sólo se puede aplicar a disyunciones de literales
- Una conjunción de disyunciones de literales se dice que está en **Forma Normal Conjuntiva** (FNC)
- Toda sentencia de lógica proposicional se puede poner en forma FNC

- Ejemplo: Pasar $B_{11} \Leftrightarrow (H_{12} \vee H_{21})$ a FNC
eliminamos \Leftrightarrow :

$$(B_{11} \Rightarrow (H_{12} \vee H_{21})) \wedge ((H_{12} \vee H_{21}) \Rightarrow B_{11})$$

eliminamos \Rightarrow

$$(\neg B_{11} \vee H_{12} \vee H_{21}) \wedge (\neg(H_{12} \vee H_{21}) \vee B_{11})$$

hacemos que \neg sólo se aplique a literales

$$(\neg B_{11} \vee H_{12} \vee H_{21}) \wedge ((\neg H_{12} \wedge \neg H_{21}) \vee B_{11})$$

aplicando distributividad

$$(\neg B_{11} \vee H_{12} \vee H_{21}) \wedge (\neg H_{12} \vee B_{11}) \wedge (\neg H_{21} \vee B_{11})$$



Algoritmo de resolución

- El objetivo es demostrar que $BC \models \alpha$ para ello demostramos que $(BC \wedge \neg \alpha)$ es insatisfacible. Demostramos una contradicción
- Primero se convierte $(BC \wedge \neg \alpha)$ a FNC
- Se aplica la regla de resolución a las cláusulas obtenidas:
 - Cada par de cláusulas con literales complementarios se resuelven para generar una nueva cláusula. Si no existía se añade
 - El proceso continua hasta que ocurre una de estas cosas:
 - no hay nuevas cláusulas que se puedan añadir (no se implica α)
 - se deriva la cláusula vacía (se implica α)

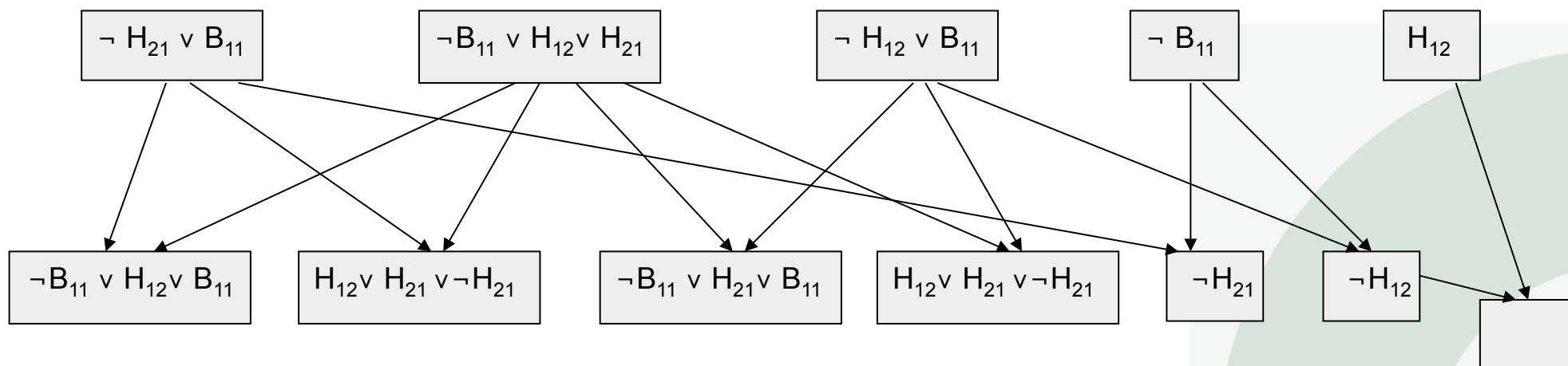
El entorno “WUMPUS” (V)

- Cuando el agente está en 11 no percibe brisa (por lo tanto no hay hoyos en las casillas vecinas). Tenemos:

$$BC = \{R_2: B_{11} \Leftrightarrow (H_{12} \vee H_{21}) \quad R_4: \neg B_{11}\}$$

Deseamos demostrar $\neg H_{12}$

Convertimos $BC \wedge H_{12}$ a FNC





Completitud de la Resolución

- La resolución se puede utilizar para confirmar o refutar una sentencia, no para enumerar sentencias verdaderas

Ej: Dado A , no podemos usar la resolución para generar de forma automática $A \vee B$, pero sí para responder a la pregunta de si $A \vee B$ es verdadero

- Cualquier algoritmo de búsqueda completo aplicado sobre la regla de resolución, puede derivar cualquier conclusión implicada por una BC en lógica proposicional
- **Teorema fundamental de la resolución:** Si un conjunto de cláusulas S es insatisfacible su cierre (cjto de todas las cláusulas derivables por resolución a partir de S) contiene la cláusula vacía
Ej: $BC = \{ p, p \Rightarrow q \}$ ¿Se cumple q ?
Sí, ya que de $BC \wedge \neg q$ se obtiene la cláusula vacía



Cláusula de Horn (I)

- Una cláusula es de Horn si es una disyunción de literales de los cuales como mucho *uno es positivo*

$$\begin{array}{ll}\neg L_{11} \vee \neg B_{21} \vee B_{11} & \text{SI es de Horn} \\ \neg B_{11} \vee H_{12} \vee H_{21} & \text{NO es de Horn}\end{array}$$

- Ventajas:

1. Representa una implicación que tiene como:

- premisa (cuerpo): una conjunción de literales
- conclusión (cabeza): un literal positivo

$$\neg L_{11} \vee \neg B_{21} \vee B_{11} \text{ representa } L_{11} \wedge B_{21} \Rightarrow B_{11}$$

- **Hecho:** es una cláusula sin literales negativos (sin cuerpo)

$$\Rightarrow A, \text{ en forma de Horn es: } A$$

- **Restricción de integridad:** sin literal positivo (sin cabeza)

$$B \wedge C \Rightarrow , \text{ en forma de Horn es: } \neg B \vee \neg C$$

- BC en forma de Horn contiene hechos y no tiene restricciones de integridad



Cláusula de Horn (II)

- Ventajas:

2. La inferencia con cláusulas de Horn se puede realizar mediante los algoritmos de:

- » encadenamiento hacia delante
- » encadenamiento hacia atrás

Se trata de algoritmos naturales y fáciles de seguir y entender para las personas

3. Averiguar si hay o no una implicación con las cláusulas de Horn se puede realizar en un tiempo que es lineal respecto al tamaño de la base de conocimientos

- » es un proceso barato para BC reales



Encadenamiento hacia delante (I)

- Puede servir para determinar si un símbolo proposicional q se deduce de una BC o para deducir los hechos que se derivan de una BC
- Pasos:
 1. El algoritmo comienza con los hechos conocidos de la BC
 2. Para todas aquellas implicaciones cuyas premisas son conocidas, se añaden sus respectivas conclusiones al conjunto de hechos conocidos
 3. El paso 2 se repite hasta que se deduce el hecho q que se busca o hasta que no se pueden realizar más inferencias



Encadenamiento hacia delante (II)

H1: A

H2: B

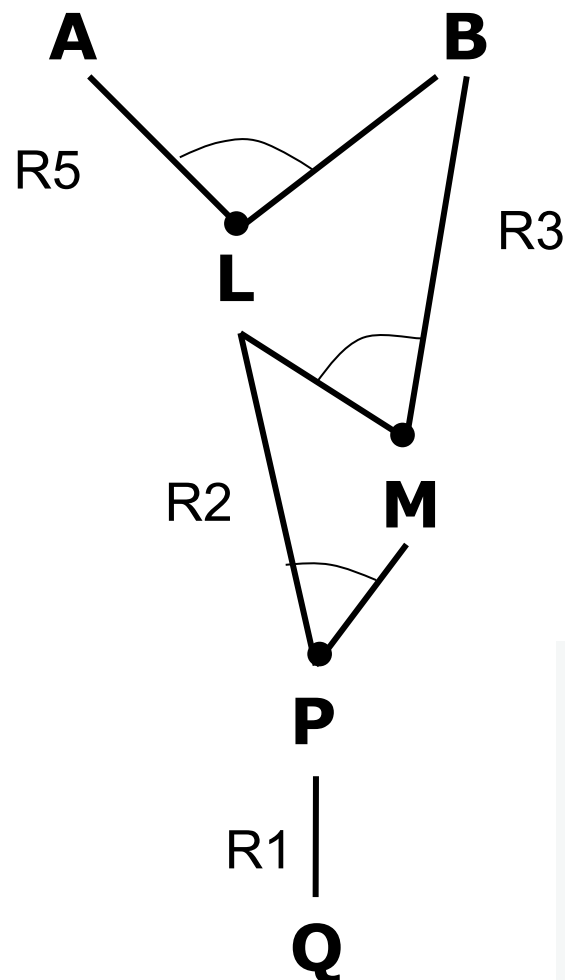
R1: $P \Rightarrow Q$

R2: $L \wedge M \Rightarrow P$

R3: $B \wedge L \Rightarrow M$

R4: $A \wedge P \Rightarrow L$

R5: $A \wedge B \Rightarrow L$





Encadenamiento hacia delante (III)

- Es un algoritmo que se ejecuta en **tiempo lineal**
- Es un proceso **sólido**, cada inferencia es una aplicación del Modus Ponens
- Es **completo**, cada sentencia atómica implicada será derivada

No puede ocurrir que la BC contenga una cláusula $\alpha_1 \wedge \dots \wedge \alpha_k \Rightarrow \beta$ falsa en el modelo, es decir, que $\alpha_1 \wedge \dots \wedge \alpha_k$ sea cierto y β falsa, ya que en ese caso se habría inferido β
- El encadenamiento hacia delante está **dirigido por los datos** (parte de los datos conocidos)



Encadenamiento hacia atrás (I)

- **Sirve sólo** para determinar si un símbolo proposicional q se deduce de una BC
- Trabaja a partir de una petición concreta
- Pasos:
 1. Si q es un hecho, entonces no hace falta realizar ningún trabajo: q es verdadero
 2. Se buscan todas aquellas implicaciones que concluyen q
 3. Si se pueden probar todas las premisas de una de esas implicaciones (mediante encadenamiento hacia atrás) entonces q es verdadero



Encadenamiento hacia atrás (II)

H1: A

H2: B

R1: $P \Rightarrow Q$

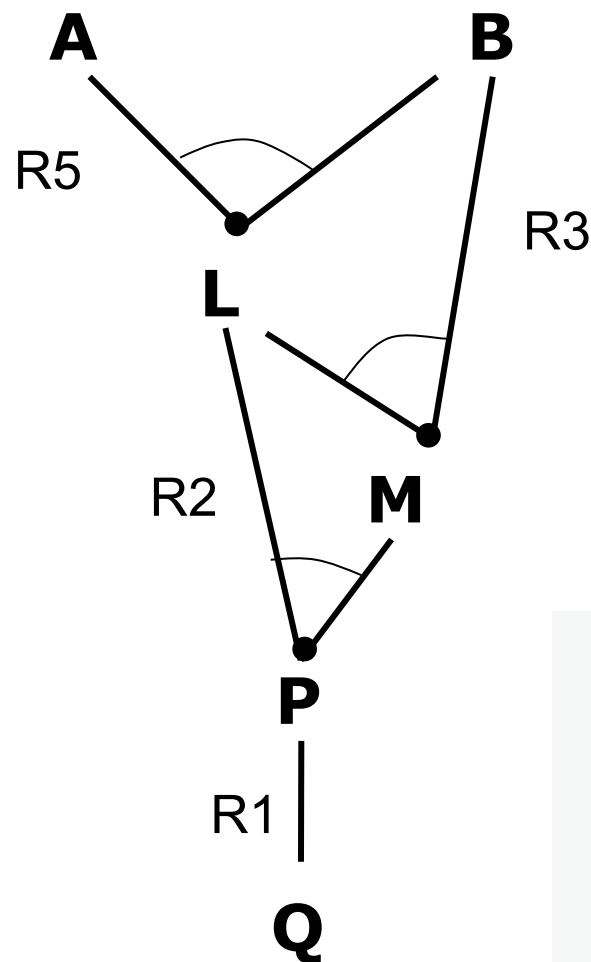
R2: $L \wedge M \Rightarrow P$

R3: $B \wedge L \Rightarrow M$

R4: $A \wedge P \Rightarrow L$

R5: $A \wedge B \Rightarrow L$

¿Q?





Encadenamiento hacia atrás (III)

- Una implementación eficiente se ejecuta en **tiempo lineal**
- A menudo su coste es mucho menor que el orden lineal respecto al tamaño de la BC ya que sólo trabaja con los hechos relevantes
- El encadenamiento hacia atrás está **dirigido por los objetivos**. Es útil para responder a preguntas
- Consideraciones:
 - Se deben evitar bucles infinitos: $a \Rightarrow b$ $b \Rightarrow a$
 - Complicaciones con el uso de variables



Lógica de predicados (orden 1) (I)

- **Lógica de predicados (orden 1)**

- Más expresiva que la lógica de orden 0
- Su importancia se debe a la existencia de la **resolución**
- Se usan **constantes, variables, predicados, funciones y cuantificadores**

$$\forall X, Y \exists Z p(a, Y) \wedge \overbrace{\neg q(Y, Z)}^{\text{literal}} \Rightarrow r(f(X), Y, Z)$$

átomo

donde:

- \forall, \exists : cuantificadores
- $\Rightarrow \wedge \vee \neg$: conectivas
- X, Y, Z : variables
- p, q, r : predicados
- a : constante
- $f()$: función formal



Semántica

- Depende de la interpretación que se le dé a los símbolos.
- Interpretación: Correspondencia entre predicados, funciones y constantes y los elementos del mundo a representar

Ejemplo: $\forall X \text{ real}(X) \rightarrow \text{positivo}(f(X))$

- Si $f(X) = X^2 + 1$ es verdadera
- Si $f(X) = -X^2 - 1$ es falsa



Definiciones

- **Modelo:** Interpretación que hace verdaderas a todas las sentencias de un conjunto
- **Sentencia satisfacible:** La que tiene algún modelo
- **Sentencia válida:** Cualquier interpretación es modelo para ella ($\models \alpha$)
- **Sentencias equisatisfacibles:** Una es satisfacible sólo cuando la otra lo es
- **Sentencias equivalentes:** Toman el mismo valor de verdad en cualquier interpretación para ambas ($\alpha \equiv \beta$)
- **Consecuencia Lógica:** Sentencia que es verdadera en todos los modelos de un conjunto de sentencias ($\alpha \models \beta$)



Resolución

- Método que permite probar la relación de consecuencia lógica mediante manipulaciones sintácticas de las fórmulas
- Las fórmulas deben estar en Forma Clausal (Kowalski)

Fórmulas \rightarrow Forma Normal de Skolem \rightarrow Forma Clausal

- Una sentencia está en FNS sii:
 - Todos los cuantificadores están al principio
 - No tiene cuantificadores existenciales
 - El núcleo está en Forma Normal Conjuntiva

FNC: Conjunción de disyunciones o literales donde cada elemento de la conjunción se denomina cláusula



Teorema de Skolem

- Eliminar \Rightarrow utilizando la equivalencia $a \Rightarrow b \equiv \neg a \vee b$
- Reducir el ámbito de cada \neg a un término simple, utilizando las leyes de doble negación, De Morgan, y Gran De Morgan
- Renombrar las variables teniendo en cuenta que $\forall X P(X) \vee \forall X Q(X)$ es equivalente a $\forall X P(X) \vee \forall Y Q(Y)$ (variantes)
- Poner todos los cuantificadores al principio de la expresión conservando su orden (Gran Distributividad Restringida, interpretaciones de Horn)
- Eliminar los cuantificadores existenciales introduciendo constantes y funciones de Skolem si es preciso. Ejemplo: $\forall X \exists Y \text{ padre}(X,Y)$ se transforma en $\forall X \text{ padre}(X, S1(X))$, siendo $S1$ una función de Skolem. O bien $\exists Y \forall X p(X,Y)$ se transforma en $\forall X p(X,c)$, siendo c una constante de Skolem
- Convertir la expresión en una conjunción de disyunciones utilizando las propiedades asociativa y distributivas de \vee y \wedge



De FNS a Forma Clausal

- Se prescinde de los cuantificadores (todos son universales)
- Se escribe cada cláusula como una expresión independiente
- Se expresa cada cláusula en notación de Kowalski

Ejemplo: $p(X) \vee q(Y) \vee \neg r(X,Z) \vee \neg m(Z)$
 $p(X), q(Y) \leftarrow r(X,Z), m(Z)$



Consecuencias del Th. de Skolem

Comprobar si una sentencia es consecuencia lógica de un conjunto de sentencias se reduce a probar, o no, la insatisfacibilidad de un conjunto de cláusulas

Se simplifica el proceso de comprobación



Derivabilidad, corrección, completud

- De un conjunto de cláusulas se deriva, mediante la regla de inferencia Inf, otra cláusula G

$$\{C_1, \dots, C_n\} \text{ Inf } G$$

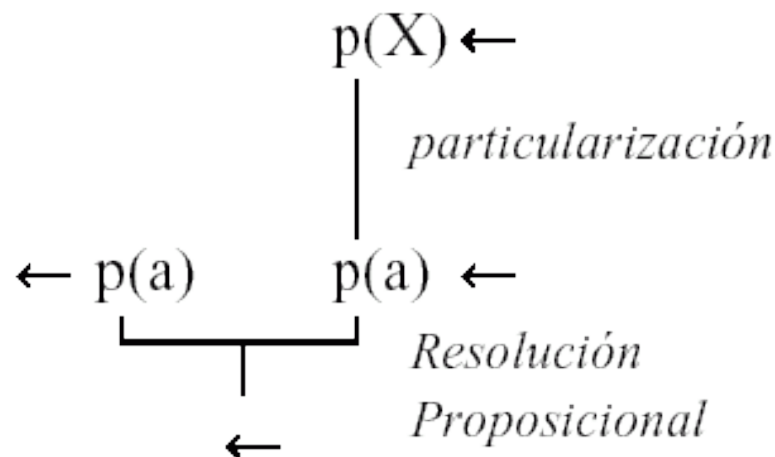
sii existe una cadena finita F_0, \dots, F_m donde

- $F_m = G$
- F_i se infiere utilizando inferencia a partir de un par de fórmulas del conjunto $\{C_1, \dots, C_n\} \cup \{F_0, \dots, F_{i-1}\}$
- Th. de Corrección: Si S es un conjunto de cláusulas tal que $S \vdash_{RP} C$ entonces $S \models C$
- Th. de Completud: Si S es un conjunto de cláusulas proposicionalmente insatisfacible entonces S es refutable por Resolución Proposicional $S \vdash_{RP} \leftarrow$



Resolución General

- Para probar la insatisfacibilidad del conjunto $\{\forall X p(X), \neg p(a)\}$ necesitamos la *unificación*
- El u.m.g. (unificador más general) es el conjunto mínimo de particularizaciones para que un conjunto de fórmulas se reduzcan a una misma fórmula



$$\text{u.m.g.}\{p(X), p(a)\} = \langle X, a \rangle$$



Unificación y Resolución General

- Algoritmo de Unificación: Dado un conjunto de expresiones lógicas, siempre termina proporcionando el u.m.g., si existe, o indicando que el conjunto no es unificable (Robinson, 1965)
- El algoritmo de unificación permite extender la resolución al caso de la L1
- Afortunadamente, la resolución general también es correcta y completa, como la proposicional
- Tenemos así la Resolución General, en este caso la regla de resolución se aplica a dos cláusulas después de un proceso de unificación

$$\begin{array}{c} C_1 \equiv \Gamma_1 \leftarrow \Delta_1 + E_1 \quad C_2 \equiv \Gamma_2 + E_2 \leftarrow \Delta_2 \\ \left[\sigma = \text{u.m.g}\{E_1, E_2\} \right] \\ \downarrow \\ \textit{Resolvente:} \boxed{\Gamma_1 \sigma + \Gamma_2 \sigma \leftarrow \Delta_1 \sigma + \Delta_2 \sigma} \end{array}$$



Decibilidad

- L0 es decidable
- L1 es semidecidible: sólo es posible diseñar algoritmos que siempre finalizan cuando el conjunto de cláusulas de partida es insatisfacible, pero no se puede garantizar que un algoritmo termine cuando el conjunto de partida no es insatisfacible

Ej: trata de refutar $\{p(X) \leftarrow p(f(X)), \leftarrow p(a)\}$



Programación Lógica (I)

- Modelo de programación basado en la posibilidad de conseguir demostraciones constructivas de fórmulas lógicas de la forma
$$\phi \models \exists X p(X)$$
- Hay que construir ϕ y diseñar un mecanismo deductivo constructivo
- PROLOG: Lenguaje de programación lógica por excelencia
 - Se usan únicamente cláusulas Horn
 - Las pruebas se hacen por refutación usando resolución



Programación Lógica (II)

Ejemplo:

$\phi = \{\text{mortal}(X) \leftarrow \text{persona}(X), \text{persona}(\text{sócrates}) \leftarrow\}$

$\psi \equiv \exists Y \text{ mortal}(Y) \Rightarrow \neg\psi \equiv \leftarrow \text{mortal}(Y)$

