



UNIVERSIDAD  
DE GRANADA



# Inteligencia Artificial

Tema 5: Comportamiento inteligente: Representación del  
Conocimiento e inferencia basados en lógica

E.T.S. de Ingenierías Informática y de Telecomunicación  
Universidad de Granada

© Antonio González

gonzalez@ugr.es

Departamento de Ciencias de la  
Computación e Inteligencia Artificial  
<http://decsai.ugr.es>

*Este documento está protegido por la Ley  
de Propiedad Intelectual ([Real Decreto Ley  
1/1996 de 12 de abril](#)).*

*Queda expresamente prohibido su uso o  
distribución sin autorización del autor.*

# Objetivos

- Entender que la resolución de problemas en IA implica definir una representación del problema y un proceso de búsqueda de la solución.
- Comprender la necesidad de representar el conocimiento y realizar inferencia para que un sistema pueda exhibir comportamiento inteligente.
- Conocer los fundamentos de la representación del conocimiento en lógica proposicional y de predicados y sus mecanismos de inferencia asociados.
- Aplicar los aspectos de representación basada en la lógica y mecanismos de inferencia, mediante técnicas y herramientas de programación lógica.

# Estudia este tema en ...

- Nils J. Nilsson, “*Inteligencia Artificial: Una nueva síntesis*”, Ed. Mc Graw Hill, 2000. pp. 215-284

# Contenido

- Representación del conocimiento en IA
- El cálculo proposicional
- Cálculo de predicados
- Introducción a los Sistemas Basados en el Conocimiento

# Representación del conocimiento en IA

- Hemos estudiado varias formas de modelar el mundo de un agente, entre ellas:
  - **Representaciones icónicas:** Simulaciones del mundo que el agente podía percibir.
  - **Representaciones descriptivas:** Valores binarios que describían aspectos ciertos o falsos sobre el mundo.
- Las representaciones descriptivas tienen ciertas ventajas sobre las icónicas:
  - Son más sencillas.
  - Son fáciles de comunicar a otros agentes.
  - Se pueden descomponer en piezas más simples.

# Representación del conocimiento en IA

- Además, hay información del entorno del agente que no se puede representar mediante modelos icónicos, tales como:
    - **Leyes generales.** “Todas las cajas azules pueden ser cogidas”.
    - **Información negativa.** “El bloque A no está en el suelo”, sin decir dónde está el bloque A.
    - **Información incierta.** “O bien el bloque A está sobre el bloque C, o bien el bloque A está sobre el bloque B”.
  - Sin embargo, este tipo de información es fácil de formular como conjunto de restricciones sobre los valores de las características binarias del agente.
  - Estas restricciones representan **conocimiento sobre el mundo.**
-

# Representación del conocimiento en IA

- A menudo, este **conocimiento sobre el mundo** puede utilizarse para razonar sobre él y hallar nuevas características del mismo.

## Ejemplo:

- El conocimiento que se tiene es “Todos los pájaros vuelan”; y “Piolín es un pájaro”.
- Se puede *razonar*, por tanto, que “Piolín vuela”.
- **Otro Ejemplo:** Un robot sólo puede levantar un bloque si tiene suficiente batería y el bloque es elevable. Entonces, el conocimiento sobre el mundo es: “Si el bloque es elevable y hay suficiente batería, entonces es posible levantar el bloque”.
- El robot “sabrá” si es capaz de levantar el bloque a partir de este **conocimiento** sobre su entorno.

# Representación del conocimiento en IA

- Estudiaremos 2 tipos básicos para representar el conocimiento y razonar sobre él:
  - Cálculo proposicional.
  - Cálculo de predicados.



# Cálculo Proposicional: el lenguaje

- Elementos de representación: proposiciones y conectivas

$\wedge$  (y),  $\vee$  (o),  $\rightarrow$  (implica),  $\neg$  (no)

- Inferencia: deducciones con reglas, hechos y Modus-Ponens
- Ejemplos: llueve,  $(\neg \text{Nieva} \wedge \text{llueve}) \vee \text{Hay-hielo}$
- Ventaja: representación de tipo general, y decidible (en tiempo finito es capaz de decidir si una proposición es deducible de la información disponible o no)
- Problema: si se quiere razonar sobre conjuntos de cosas. Por ejemplo, grafos, o jerarquías de conceptos.

# Reglas de inferencia

- Las **reglas de inferencia** nos permiten producir nuevas FBFs a partir de las que ya existen, Algunas de las más comunes son:
  - $Q$  puede inferirse a partir de  $P$  y  $P \supset Q$  (*modus ponens*)
  - $P \wedge Q$  se puede inferir a través de la conjunción de  $P$  y  $Q$
  - $Q \wedge P$  se puede inferir desde  $P \wedge Q$  (*conmutatividad*)
  - $P$  (también  $Q$ ) se puede inferir desde  $Q \wedge P$
  - $P \vee Q$  se puede inferir bien desde  $P$ , bien desde  $Q$
  - $P$  se puede inferir desde  $\neg(\neg P)$

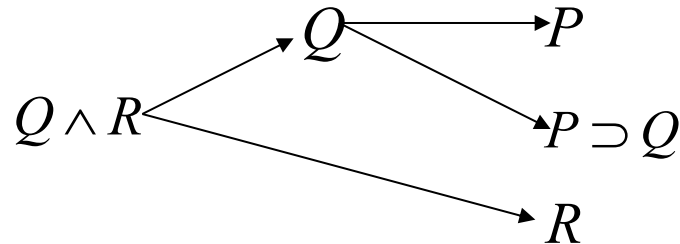
# Definición de demostración

- Supongamos  $\Delta$  un conjunto de FBFs, y una secuencia de  $n$  FBFs  $\{w_1, w_2, w_3, \dots, w_n\}$ .
- Esta secuencia de FBFs se llama **demostración o deducción** de  $w_n$  a partir de  $\Delta$  si, y sólo si, cada  $w_i$  de la secuencia pertenece a  $\Delta$  o puede inferirse a partir de FBFs en  $\Delta$ .
- Si existe tal demostración, entonces decimos que  $w_n$  es un **teorema de  $\Delta$** , y decimos que  $w_n$  puede demostrarse desde  $\Delta$  con la siguiente notación:  $\Delta \vdash w_n$ ,
- o como  $\Delta \vdash_R w_n$  para indicar que  $w_n$  se demuestra desde  $\Delta$  mediante las reglas de inferencia **R**.

# Demostración

- **Ejemplo:**

- Sea el conjunto de FBFs  $\Delta$ ,  $\Delta = \{P, R, P \supset Q\}$
- Entonces, la siguiente secuencia es una demostración de la Fórmula Bien Formada  $R \wedge Q$  :  
$$\{P, P \supset Q, Q, R, Q \wedge R\}$$
- La demostración se puede llevar a cabo fácilmente a través del siguiente **árbol de demostración**, utilizando  $\Delta$  y las reglas de inferencia:



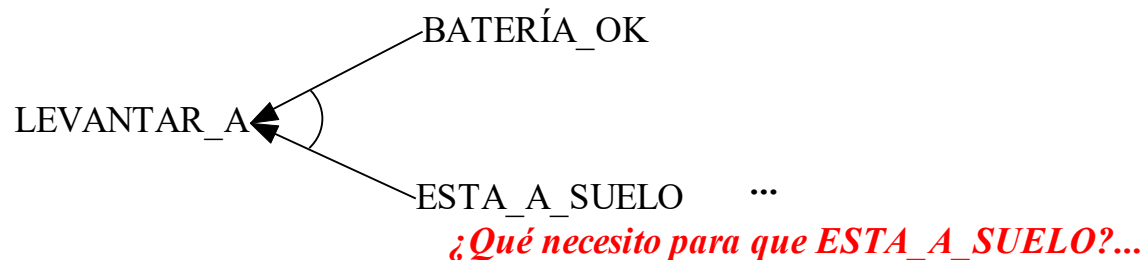
# Interpretación

- A la hora de resolver problemas con IA, el papel de la semántica es esencial: Hay que hacer una correcta **interpretación** del sistema lógico subyacente.
- Conlleva asociar conceptos del lenguaje lógico con su significado (semántica) en el mundo real o en el mundo del entorno del agente.
- **Ejemplo:** Se desea implantar el conocimiento “Si la batería funciona y el bloque A está en el suelo, entonces se puede levantar” dentro de un agente.
  - Definimos los átomos *BATERIA\_OK*, *ESTA\_A\_SUELO*, *LEVANTAR\_A*.
  - Definimos la FBF:

$$BATERIA\_OK \wedge ESTA\_A\_SUELO \supset LEVANTAR\_A$$

# Interpretación

- En un agente cuyo objetivo sea “*levantar el bloque A*”, con este conocimiento puede especificar las acciones que debe llevar a cabo para realizar su acción.
- Esta planificación se puede hacer mediante árboles de demostración.
- La representación de **grafos Y/O** es muy útil en este tipo de problemas.
- **Ejemplo:** “*Debo levantar el bloque A, ¿qué necesito para poder levantarlo?*”



# Tablas de la verdad

- Las **tablas de verdad** establecen la semántica de las conectivas proposicionales.
- Para una representación interna de un agente con **n** características, el número de combinaciones (formas de ver el mundo) es **2<sup>n</sup>**.
- Para dos características **A** y **B**:

A	B	A y B	A o B	No A	A implica B
V	V	V	V	F	V
V	F	F	V	F	F
F	V	F	V	V	V
F	F	F	F	V	V

# Satisfacibilidad y Modelos

- Una **interpretación satisface una FBF** cuando a la FBF se le asocia el valor **V** bajo esa interpretación.
- A la interpretación que satisface una FBF se le denomina **modelo**.
- Bajo una interpretación una FBF debe indicar una restricción que nos informe sobre algún aspecto del mundo. **Ejemplo:**  $A \wedge B \supset C$ 
  - Para la interpretación  $A=\text{BATERIA\_OK}$ ,  $B=\text{ESTA\_A\_SUELO}$ ,  $C=\text{LEVANTAR\_A}$ , la semántica se corresponde con el entorno y el mundo a modelar.
  - Para la interpretación  $A=\text{TOCA\_LOTERIA}$ ,  $B=\text{TENGO\_SALUD}$ ,  $C=\text{TIRAR\_POR\_VENTANA}$ , la semántica es inconsistente con lo que se modela. **Esta interpretación no es válida porque no satisface la FBF.**



# Consecuencia lógica

- Si una fbf  $w$  tiene el valor verdadero bajo todas aquellas interpretaciones para las cuales cada una de las fbfs del conjunto  $\Delta$  tiene el valor verdadero, entonces decimos que  $\Delta$  lleva lógicamente a  $w$ , que  $w$  se sigue lógicamente de  $\Delta$ , o que  $w$  es una consecuencia lógica de  $\Delta$ .
- $\{P\} \models P$
- $\{P, P \supset Q\} \models Q$
- $P \wedge Q \models P$
- La noción de consecuencia lógica es importante ya que nos proporciona un mecanismo para demostrar que si ciertas proposiciones son ciertas entonces otras deben serlo también.

# Solidez y completitud

- Si, para el conjunto de fbfs  $\Delta$  y para la fbf  $w$ ,  $\Delta \vdash_R w$  implica que  $\Delta \models w$  decimos que el conjunto de reglas de inferencia  $R$  es sólido.
- Si, para para el conjunto de fbfs  $\Delta$  y para la fbf  $w$ , tenemos que siempre que  $\Delta \models w$ , existe una demostración de  $w$  a partir de  $\Delta$  utilizando el conjunto de reglas de inferencia  $R$ , decimos que  $R$  es completo.

# Resolución en el cálculo proposicional

- **La refutación** es útil para demostrar que la negación de una cláusula es inconsistente en el sistema, quedando así demostrada, por tanto, la veracidad de dicha cláusula.
- **Ejemplo:**
  - 1. Convertir las FBFs de  $\Delta$  como conjunciones de cláusulas
    - a)  $BATERIA\_OK$
    - b)  $\neg ROBOT\_SE\_MUEVE$
    - c)  $\neg BATERIA\_OK \vee \neg OBJETO\_ELEVABLE \vee ROBOT\_SE\_MUEVE$
  - 2. Convertir  $\neg w$  como conjunción de cláusulas
    - $OBJETO\_ELEVABLE$

# Resolución en el cálculo proposicional

- **Ejemplo:**

- 3. Unir el resultado de los pasos 1 y 2 en un único conjunto  $\Gamma$

$\Gamma = \{$

a) *BATERIA\_OK*

b)  $\neg$ *ROBOT\_SE\_MUEVE*

c)  $\neg$ *BATERIA\_OK*  $\vee$   $\neg$ *OBJETO\_ELEVABLE*  $\vee$  *ROBOT\_SE\_MUEVE*

d) *OBJETO\_ELEVABLE*

$\}$

# Resolución en el cálculo proposicional

- **Ejemplo:**

- 4. Aplicar la resolución a las cláusulas de  $\Gamma$  de forma iterativa, hasta que no haya nada más que resolver o se llegue a **Nil**

- Resolviendo c) y d), tenemos que

$$e) \neg BATERIA\_OK \vee ROBOT\_SE\_MUEVE$$

- Resolviendo e) y b), tenemos:

$$f) \neg BATERIA\_OK$$

- Resolviendo f) y a), tenemos **Nil**.
- Queda demostrado que  $\neg OBJETO\_ELEVABLE$

# Resolución en el cálculo proposicional

- Como hemos visto, el procedimiento de refutación mediante resolución consiste en “*aplicar resoluciones hasta que se genere la cláusula vacía o no se puedan hacer más resoluciones*”.
- La selección de cláusulas para su resolución, de forma manual, es sencilla.
- Existen distintas estrategias que permiten determinar la selección de las cláusulas a resolver para conseguir una mayor eficiencia.

# El Cálculo de Predicados

- **El cálculo proposicional es limitado.** Supongamos nuestro mundo de bloques. Para decir que el bloque **A** está sobre el bloque **B**, deberíamos establecer una interpretación **SOBRE\_A\_B**.
- Para representar esta situación con todos los bloques usando cálculo proposicional, necesitaríamos tantos literales como posibilidades.
- Además, supongamos dos literales **P** y **Q**, con la semántica asociada **P $\equiv$ SOBRE\_A\_B**, **Q $\equiv$ SOBRE\_B\_C**.
- En lenguaje natural y mediante el conocimiento que tenemos del problema, nosotros (diseñadores, personas, etc.) *sabemos* que **A** está sobre **B**, y que **B** está sobre **C**. Por tanto, **C** está por debajo de **A**. Sin embargo, necesitaríamos más proposiciones y más complejas para implementar este conocimiento utilizando únicamente cálculo proposicional.

# El Cálculo de Predicados

- Sería de gran utilidad un lenguaje que permitiese definir objetos y relaciones entre ellos.
- El **cálculo de predicados** nos permite esta opción y, además solventa los problemas planteados en la diapositiva anterior.
- Ejemplo: Para decir que  $SOBRE\_B\_C \supset \neg LIBRE\_C$ , para cualquier bloque, el cálculo de predicados nos evita tener que reescribir todas las proposiciones del cálculo proposicional de las situaciones que pueden darse. Podemos abstraer los objetos a variables y escribir:

$$SOBRE(x, y) \supset \neg LIBRE(y)$$

- El significado sería “*cuando un objeto  $x$  esté sobre otro  $y$ , entonces  $y$  no estará libre*”.



# Reglas de inferencia

- Inferencia: Todas las de lógica proposicional + instanciación universal
- Instanciación universal: si tenemos  $\forall X p(X)$  entonces se puede deducir  $p(a)$ ,  $p(Y)$  . . .
- Ejemplo: Todos los hombres son mortales, Sócrates es un hombre, por tanto, Sócrates es mortal:
  - a. R1:  $\forall X \text{ hombre}(X) \rightarrow \text{mortal}(X)$
  - b.  $\text{hombre}(\text{sócrates})$
  - c. R1 y  $X=\text{sócrates}$ :  $\text{hombre}(\text{sócrates}) \rightarrow \text{mortal}(\text{sócrates})$
  - d. (b y c)  $\text{mortal}(\text{sócrates})$

# Características del cálculo de predicados

- Ventaja: representación de tipo general más rica que la proposicional
- Características de un sistema de razonamiento lógico:
  - **solidez**: para estar seguro de que una conclusión inferida es cierta.
  - **completitud**: para estar seguros de que una inferencia tarde o temprano producirá una conclusión verdadera.
  - **decidibilidad**: para estar seguros de que la inferencia es factible.
- La refutación mediante resolución es sólida y completa.
- Problema: el cálculo de predicados es **semidecidible** y además en los casos en que la refutación mediante resolución termina, el procedimiento es **NP-duro**.

# PROLOG

- Solución: subconjuntos decidibles de lógica de predicados (cláusulas de Horn)
- Existe un lenguaje de programación que permite crear y ejecutar programas en lógica de predicados: PROLOG

conectados(X,Y) :- conectados(Y,X).

alcanzable(X,Y) :- conectados(X,Y).

alcanzable(X,Y) :- conectados(X,Z), alcanzable(Z,Y).

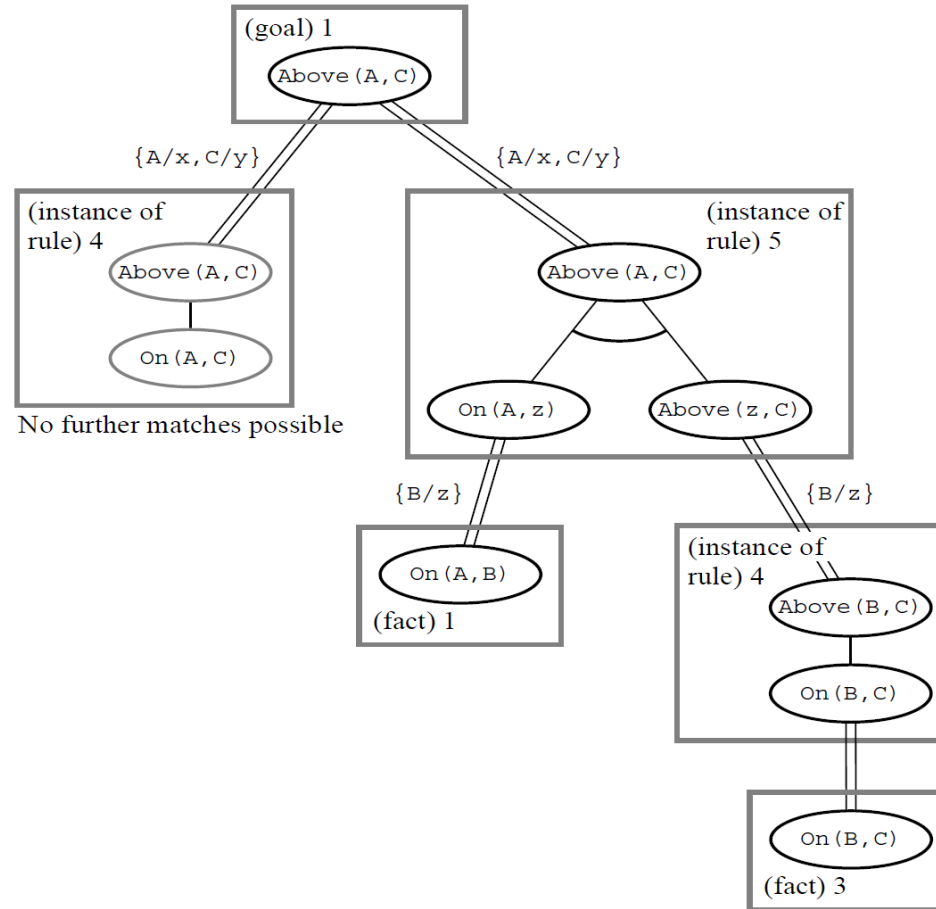
# Ejemplo de PROLOG

$$(\forall x, y, z)[Sobre(x, y) \supset Encima(x, y)]$$

$$(\forall x, y)\{(\exists z)[Sobre(x, z) \wedge Encima(z, y)] \supset Encima(x, y)\}$$

1. :- Encima(A,C)
2. Sobre(A,B) :-
3. Sobre(B.C) :-
4. Encima(x,y) :- Sobre(x,y)
5. Encima(x,y) :- Sobre(x,z), Encima(z,y)

# Ejemplo de Prolog



# Organización jerárquica del conocimiento

- Organización jerárquica del conocimiento
  - Snoopy es una impresora láser
  - Todas las impresoras láser son impresoras
  - Todas las impresoras son máquinas

*Impresora.laser(Snoopy)*

$(\forall x)[Impresora.laser(x) \supset Impresora(x)]$

$(\forall x)[Impresora(x) \supset Maquina.de.oficina(x)]$

- Herencia de Propiedades

$(\forall x)[Maquina.de.oficina(x) \supset$

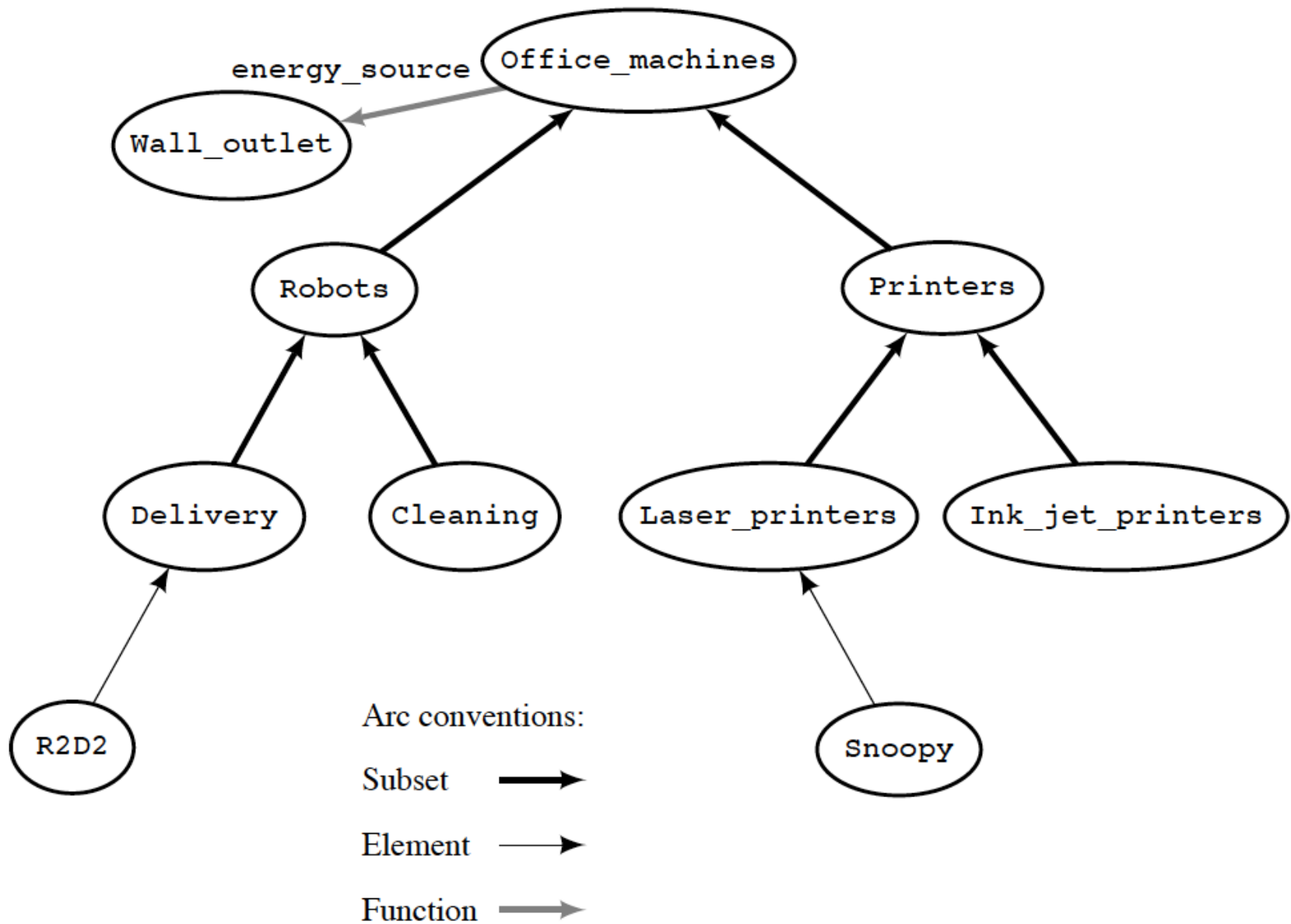
$[Fuente.de.alimentacion = Toma.de.la.pared]]$

$(\forall x)[Impresora.laser(x) \supset$

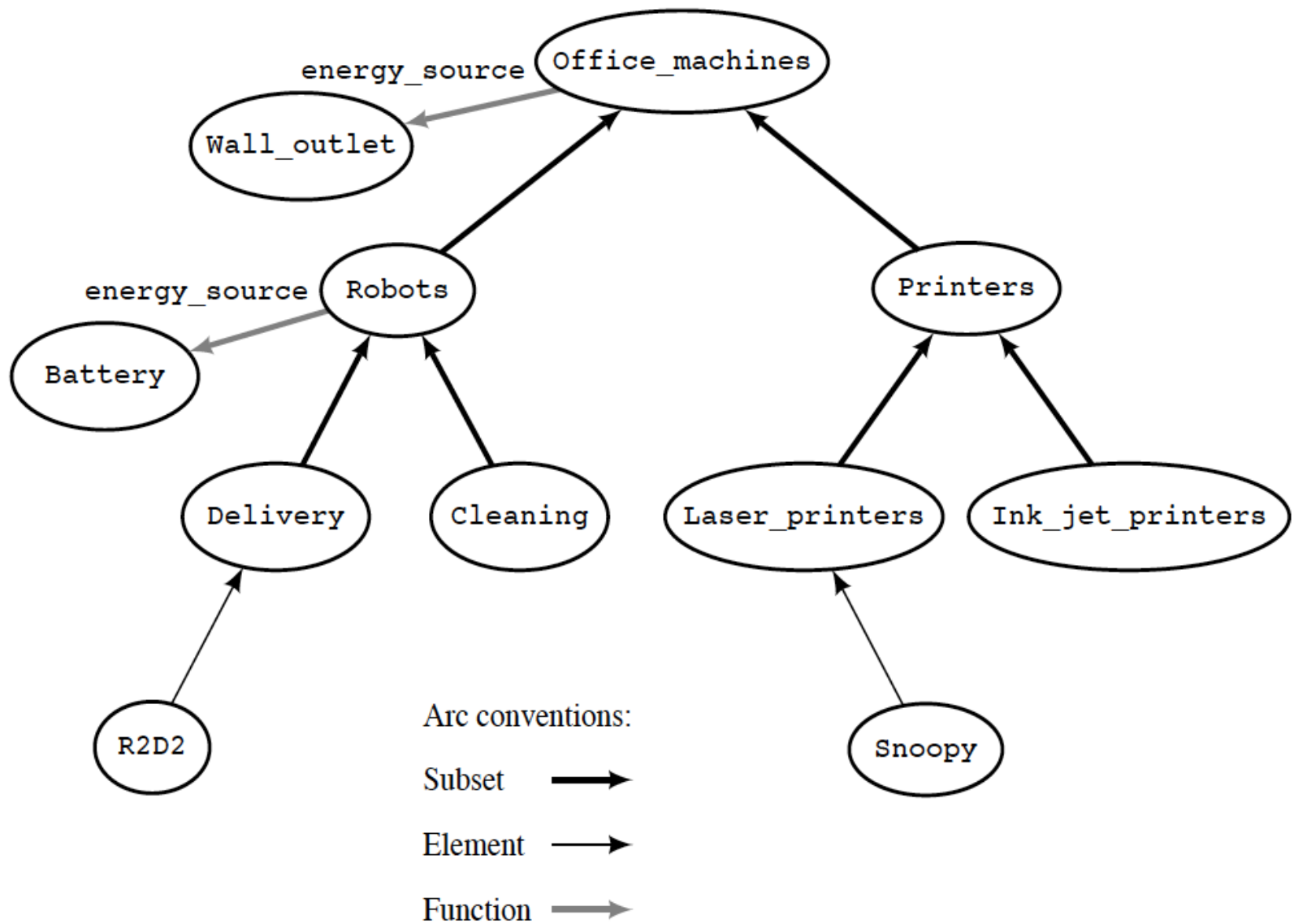
$[Fuente.de.alimentacion = Toma.de.la.pared]]$

# Redes semánticas

- Las redes semánticas son estructuras gráficas que codifican el conocimiento taxonómico sobre objetos y propiedades de estos
- PROPIEDADES: nodos etiquetados con constantes de relación
- OBJETOS: nodos etiquetados con constantes de objetos
  - Arcos de jerárquica
  - Arcos de pertenencia
  - Arcos de función







# Otras lógicas

- Lógicas de segundo orden (o de orden superior)
  - tienen dos (o tres) tipos definidos: los objetos y los conjuntos o funciones sobre los mismos (o ambos)
  - es equivalente a decir que los predicados pueden tomar otros predicados como argumentos
- Lógicas modales y temporales
  - necesario y posible
- Lógica difusa
  - grados de pertenencia
- Otras: multi-valuadas, no-monótonas, cuánticas, .  
..

# Sistemas Basados en el Conocimiento

- Una gran cantidad de aplicaciones reales de la IA se basan en la existencia de una gran masa de conocimiento:
  - Diagnóstico médico.
  - Diseño de equipos.
  - Sistemas de Recomendación.
  - Etc.
- Este tipo de sistemas se denominan **Sistemas Basados en el Conocimiento**, ya que este ocupa la parte central de la solución al problema a resolver.

# Sistemas Basados en el Conocimiento

- Un **Sistema Basado en el Conocimiento (SBC)** necesita 3 componentes básicas:
    - Una **Base de Conocimiento (BC)**, que contenga el conocimiento experto necesario sobre el problema a resolver. Puede ser:
      - **Estática**, si la BC no varía a lo largo del tiempo.
      - **Dinámica**, cuando se añaden nuevos hechos o reglas, o se modifican las existentes a lo largo del tiempo.
    - Un **Motor de Inferencia**, que permite razonar sobre el conocimiento de la BC y los datos proporcionados por un usuario.
    - Una **interfaz de usuario** para entrada/salida de datos.
-

# Sistemas Expertos basados en Reglas (SEBR)

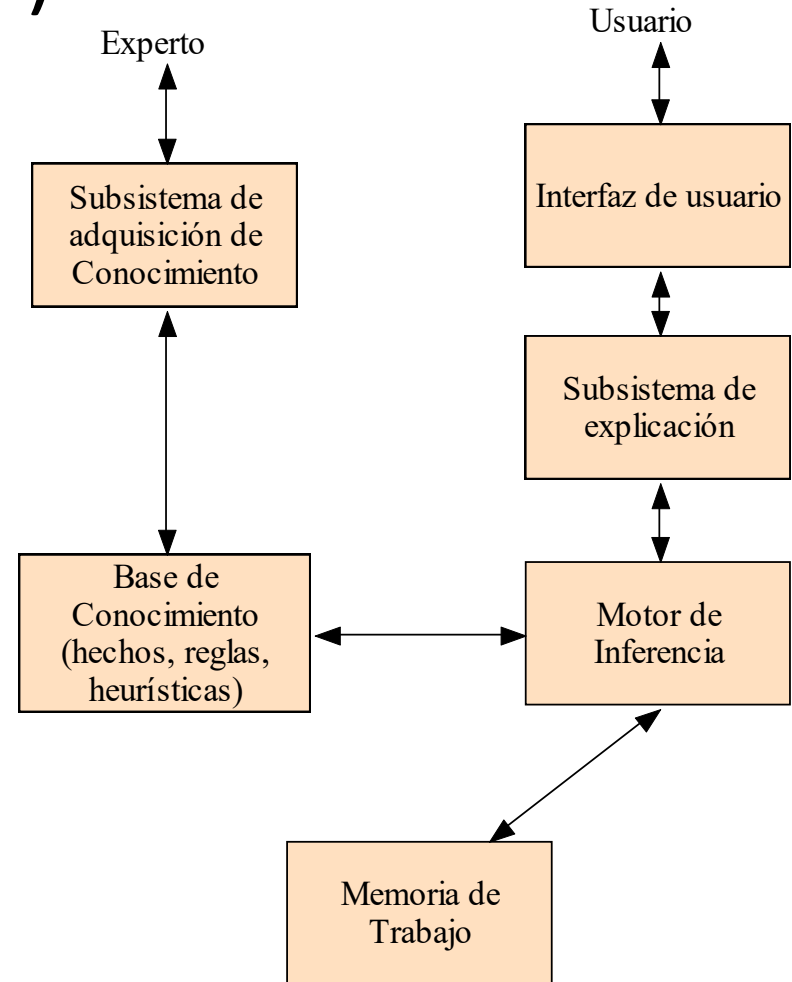
- Un **SEBR** es un **SBC** donde el conocimiento se incluye en forma de reglas y hechos.
- Estas reglas y hechos pueden implementarse, por ejemplo, mediante el **cálculo de predicados**.
- El proceso de construcción de un SEBR es el siguiente:
  - Se **extrae el conocimiento** experto (bibliografía, entrevistas a expertos reales, etc.).
  - Se **modela y se adquiere el conocimiento**, utilizando un lenguaje adecuado (cálculo de predicados, otras lógicas más avanzadas, etc.)
  - Se **crea la Base de Conocimiento** con el conocimiento adquirido.

# Sistemas Expertos basados en Reglas (SEBR)

- Por otra parte, también se necesita:
  - Una **interfaz de usuario**, para poder utilizar el sistema y adquirir/enviar datos.
  - Un **subsistema de explicación**, para los casos en los que sea necesario indicar al usuario porqué se llega a las conclusiones que se llegan.
  - Un **Motor de Inferencia**, para razonar sobre la Base de Conocimiento y los datos proporcionados por el usuario.

# Sistemas Expertos basados en Reglas (SEBR)

- El esquema general de diseño de un SEBR es el siguiente:
- La **memoria de trabajo** contiene la información relevante que el Motor de Inferencia está usando para razonar las respuestas para el usuario.



# Ideas finales

- Representación del conocimiento de sentido común
- Razonamiento temporal



# Representación del conocimiento de sentido común

- Lógica difusa: Extensión de la lógica clásica diseñada para permitir el razonamiento sobre conceptos imprecisos
  - “la velocidad del motor es muy alta”
  - “el paciente tiene fiebre moderada”
  - “si el paciente tiene fiebre muy alta y es muy joven, entonces la dosis debe de ser moderada”
- Ejemplo: control del movimiento de un robot

# Razonamiento temporal

- Allen (1983,1984): El tiempo es algo dinámico, sobre el cual los procesos y los eventos transcurren
  - E evento o suceso
  - I intervalo de tiempo

Ocurre(E,I)

- Intervalos temporales: instantes de inicio y final

$$(\forall x)[inicio(x) \leq fin(x)]$$

# Razonamiento temporal

$$(\forall x, y)[Se.encuentra.con(x, y) \equiv (fin(x) = inicio(y))]$$

$$(\forall x, y)\{Antes.de(x, y) \equiv \\ \exists(z)[Se.encuentra.con(x, z) \wedge Se.encuentra.con(z, y)]\}$$

$$(\forall x, y)\{Antes.de(x, y) \equiv [(fin(x) < inicio(y))]\}$$

# Razonamiento temporal

- Ejemplo: representación de hechos de sentido común el evento salir agua de un grifo está precedido por el de abrir una válvula, y seguido por el de cerrarla

$$(\forall y)\{Ocorre(Saleagua, y) \supset$$
$$(\exists x, z)[Ocorre(Abrir.V, x) \wedge Ocorre(Cerrar.V, z) \wedge$$
$$Se.solapa.con(x, y) \wedge Se.solapa.con(y, z)]\}$$