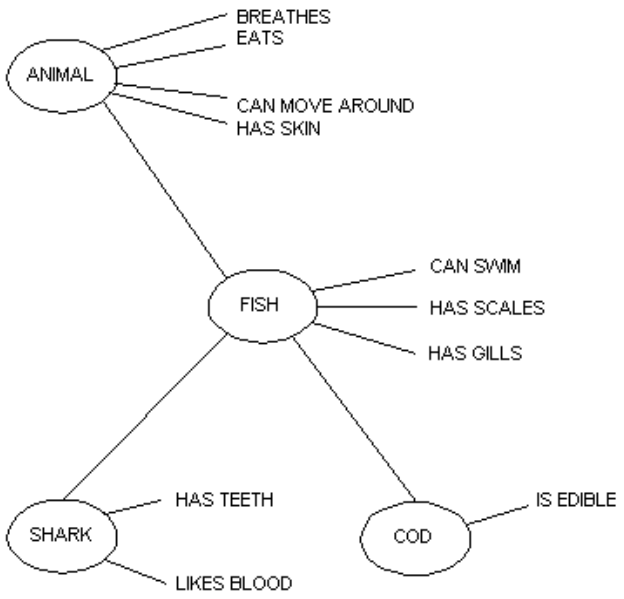
Tema 5. Comportamiento inteligente: Representación del Conocimiento e inferencia basados en lógica



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

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

- Inferencia: deducciones con reglas, hechos y Modus-Ponens
- Ejemplos: llueve, (¬Nieva∧llueve)vHay-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 (fórmulas bien formadas) a partir de las que ya existen, Algunas de las más comunes son:
  - **Q** puede inferirse a partir de **P** y $\mathbf{P} \supset \mathbf{Q}$ (*modus ponens*)
  - $\mathbf{P} \wedge \mathbf{Q}$ se puede inferir a través de la conjunción de **P** y **Q**
  - $\mathbf{Q} \wedge \mathbf{P}$ se puede inferir desde $\mathbf{P} \wedge \mathbf{Q}$ (*conmutatividad*)
  - **P** (también **Q**) se puede inferir desde $\mathbf{Q} \wedge \mathbf{P}$
  - $\mathbf{P} \vee \mathbf{Q}$ se puede inferir bien desde **P**, bien desde **Q**
  - **P** se puede inferir desde $\neg(\neg\mathbf{P})$

Definición de demostración

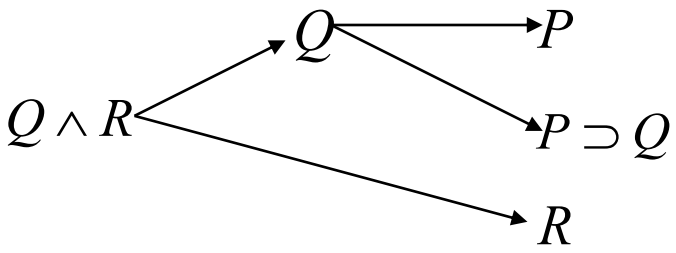
- Supongamos Δ un conjunto de FBFs, y una secuencia de **n** FBFs $\{\mathbf{w_1}, \mathbf{w_2}, \mathbf{w_3}, \ldots, \mathbf{w_n}\}$.
- Esta secuencia de FBFs se llama **demostración** o **deducción** de $\mathbf{w_n}$ a partir de Δ si, y sólo si, cada $\mathbf{w_i}$ de la secuencia pertenece a Δ o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

Demostración

- **Ejemplo:**
  - Sea el conjunto de FBFs Δ, $\Delta = \{P, R, P \supset Q\}$
  - 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 Δ y las reglas de inferencia:



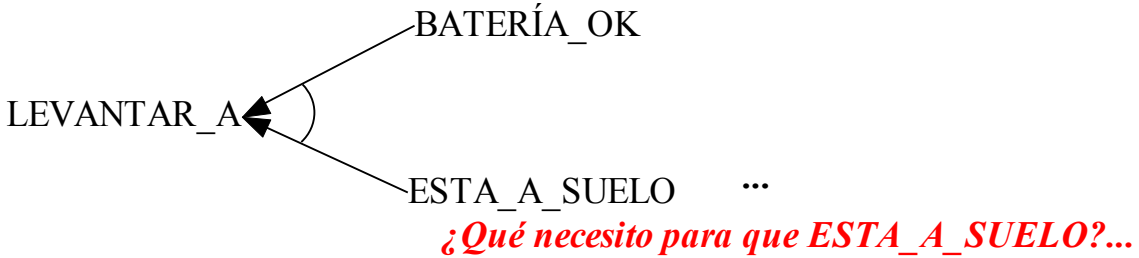
Interpretación

- 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 $\mathbf{2^n}$.
- 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=BATERIA_OK, B=ESTA_A_SUELO, C=LEVANTAR_A, la semántica se corresponde con el entorno y el mundo a modelar.
  - Para la interpretación A=TOCA_LOTERIA, B=TENGO_SALUD, C=TIRAR_POR_VENTANA, la semántica es inconsistente con lo que se modela. **Esta interpretación no es válida porque no satisface la FBF.**

Validez y equivalencia

- Se dice que una FBF es **válida** si se cumple independientemente de la interpretación que se le asocie. Ejemplo: $P \supset P, \neg(P \wedge \neg P)$
- Las interpretaciones válidas no modelan aspectos del mundo y deben ser evitadas en el diseño del comportamiento del agente.

- Dos FBFs son **equivalentes** si sus tablas de verdad son idénticas. Ejemplo:$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
- En el diseño de agentes, debemos evitar FBFs con interpretaciones equivalentes para hacer más eficiente el proceso de razonamiento.

Consecuencia lógica

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

Solidez y completitud

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

Ejemplo: proposiciones

- Una fabrica tiene cuatro sensores que detectan fuego y dos sensores que detectan fugas en el circuito del agua. Existen tres alarmas que se producen en diferentes ocasión

$$s_1, s_2, s_3, s_4, f_1, f_2, a_1, a_2, a_3$$

- Si el detector 3 de fuego o el detector 2 de fugas saltan, se debe producir la alarma 1

$$R1: s_3 \vee f_2 \rightarrow a_1$$

- Si saltan los detectores de fuego 1 y 4, se debe producir la alarma 2

$$R2: s_1 \wedge s_4 \rightarrow a_2$$

- Si salta la alarma 1, y el detector de fuego 2 o el de fugas 1, se debe producir la alarma 3

$$R3: a_1 \wedge (s_2 \vee f_1) \rightarrow a_3$$

Ejemplo de Inferencia: deducción

- Han saltado el detector de fuego 2 y el de fugas 2.
- ¿Qué alarmas saltaran?
  a. s_2
  b. f_2
  c. (R1 y a) a_1
  d. (R3, a y c) a_3

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 Δ 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 ¬**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 **Γ**

$$\Gamma = \{$$
$$a) BATERIA\_OK$$
$$b) \neg ROBOT\_SE\_MUEVE$$
$$c) \neg BATERIA\_OK \vee \neg OBJETO\_ELEVABLE \vee ROBOT\_SE\_MUEVE$$
$$d) OBJETO\_ELEVABLE$$
$$\}$$

Resolución en el cálculo proposicional

- **Ejemplo:**
  - 4. Aplicar la resolución a las cláusulas de **Γ** 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 ¬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≡SOBRE_A_B**, **Q≡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*".

Cálculo de Predicados

- Elementos de representación:
  - Términos: Constantes (UGR), Variables (X), Funciones (siguiente(X))
  - Fórmulas atómicas: Predicados definidos sobre términos
    - trabaja-como(empleado1,director)
    - tiene-hijos(empleado1,1)
  - Fórmulas bien formadas (fbf): Fórmulas atómicas unidas por conectivas ($\wedge$,$\vee$,$\neg$,$\rightarrow$) y cuantificadas ($\forall$,$\exists$)
    - $\forall$ X,Y trabaja-como(X,director), tiene-hijos(X,Y), Y<=2 $\rightarrow$ gana(X,60000)
    - $\forall$ X,Y trabaja-como(X,director), tiene-hijos(,;Y), Y>2 $\rightarrow$ gana(X,70000)

Reglas de inferencia

- Inferencia: Todas las de lógica proposicional + instanciación universal
- Instanciación universal: si tenemos ∀ 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: ∀ X hombre(X) → mortal(X)
  b. hombre(sócrates)
  c. R1 y X=sócrates: hombre(sócrates) → mortal(sócrates)
  d. (b y c) mortal(sócrates)

Representación

- Una universidad imparte un conjunto de titulaciones en un conjunto de centros y campus.
  - imparte(Universidad,Titulación,Centro,Campus)
  - imparte(UGR,Informática,ETSIIT,Aynadamar)
  - Imparte(UGR,Matemáticas,FC,Ciencias)
- La representación no es única, alternativa:
  - imparte-titulación(Universidad,Titulación)
  - imparte-titulación(UGR,Informática)
  - imparte-titulación(UGR,Matemáticas)
  - imparte-centro(Informática,ETSIIT)
  - imparte-centro(Matémáticas,FC)
  - centro-en-campus(ETSIIT,Aynadamar)
  - centro-en-campus(FC,Ciencias)

Sigue el ejemplo

- Cada titulación tiene un plan de estudios formado por un conjunto de asignaturas troncales, obligatorias, optativas y de libre elección.
  - asignatura-en-plan(Asignatura,Titulación)
  - asignatura-en-plan(IA,Informática)
  - tipo-asignatura(Asignatura,Tipo)
  - tipo-asignatura(IA,obligatoria) o
  - tipo-asignatura(IA,troncal)
- Cada asignatura se imparte en un curso y cuatrimestre determinados y tiene un determinado numero de créditos.
  - curso-asignatura(Asignatura,Curso)
  - cuatrimestre-asignatura(Asignatura,Cuatrimestre)
  - créditos-asignatura(Asignatura,Créditos)

o

  - asignatura(Asignatura,Curso,Cuatrimestre,Creditos)

Inferencia

- Cuando un alumno se matricula por primera vez en primero, debe matricularse de todas las asignaturas del primer curso.

R1: ∀ X,U,Y primera-matrícula(X,U), curso-asignatura(Y,1) → matriculado-en(X,Y)

Inferencia: Deducción con Modus-Ponens

- En primero del grado en Informática de la UGR se imparte las asignaturas de FundamentosdeProgramación, . . .
  1. curso-asignatura(FundamentosdeProgramación,1)
  2. curso-asignatura(FundamentosdelSoftware,1)

- Ana Morales Pérez acaba de matricularse en primero de la titulación.
  a) primera-matricula(anaMorales,UGR)

- Si X=anaMorales, U=UGR, e Y =FundamentosdeProgramación, (unificación) por el Modus-Ponens, a partir de la regla R1, de 1 y de a), se puede deducir que

  matriculado-en(anaMorales, FundamentosdeProgramación)

- Si X=anaMorales, U=UGR, y Y =FundamentosdelSoftware, por el Modus-Ponens, a partir de la regla R1, de 2 y de a), se puede deducir que

  matriculado-en(anaMorales, FundamentosdelSoftware)

Deducción hacia atrás

- Y si se desea conocer ¿en qué asignaturas se debe matricular Ana Morales?
- Pregunta: matriculado-en(anaMorales,Y)
- Se busca una implicación lógica en la que aparezca matriculado-en(V, V1) en la parte derecha (puede haber más de una).
- Si se pueden unificar, se intentan deducir los literales que aparezcan en la parte izquierda de la implicación.
- Si existe alguna asignación de valor a las variables de la parte izquierda que permita deducir como ciertas las condiciones, se podrá deducir la pregunta de diferentes formas (diferentes valores de Y: FundamentosdeProgramación, FudamentosdelSoftware, . . .)

Características del cálculo de predicados

- Ventaja: representación de tipo general mas rica que la proposicional
- Características de un sistema de razonamiento lógico:
  - solidez: para estar seguro 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.

Características del cálculo de predicados

- Solución: subconjuntos decidibles de lógica de predicados (clausulas 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).
```

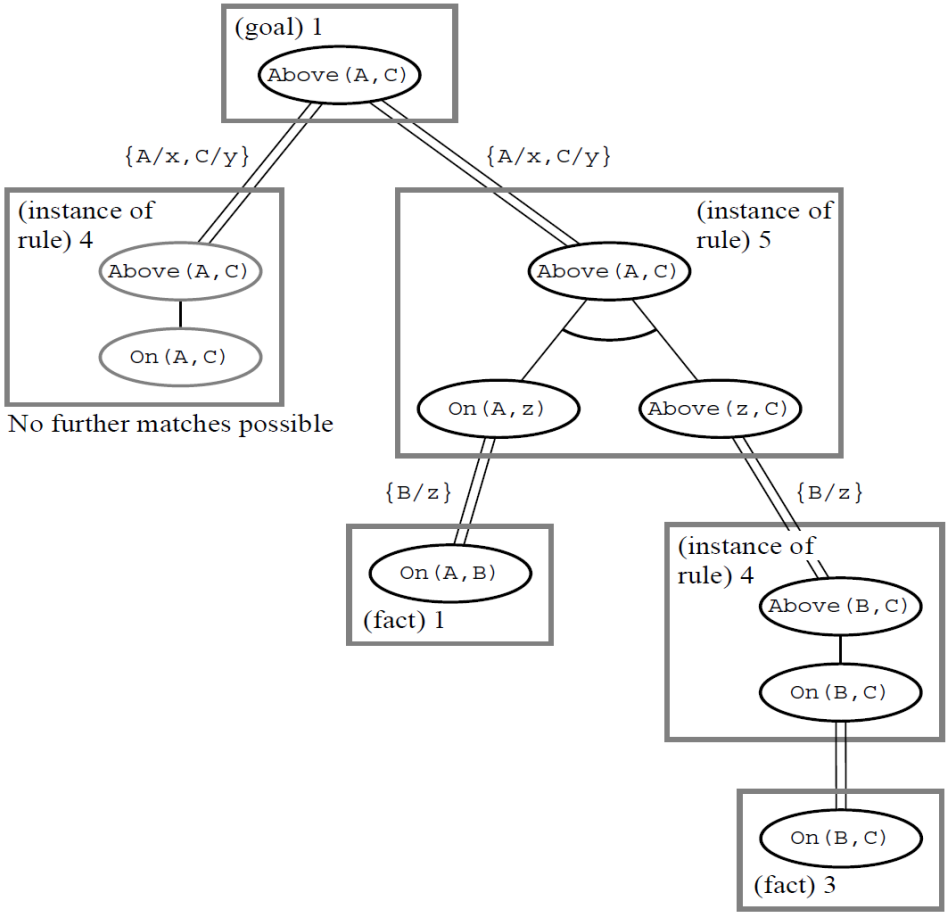
Ejemplo de PROLOG

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

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

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



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, . . .

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

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)

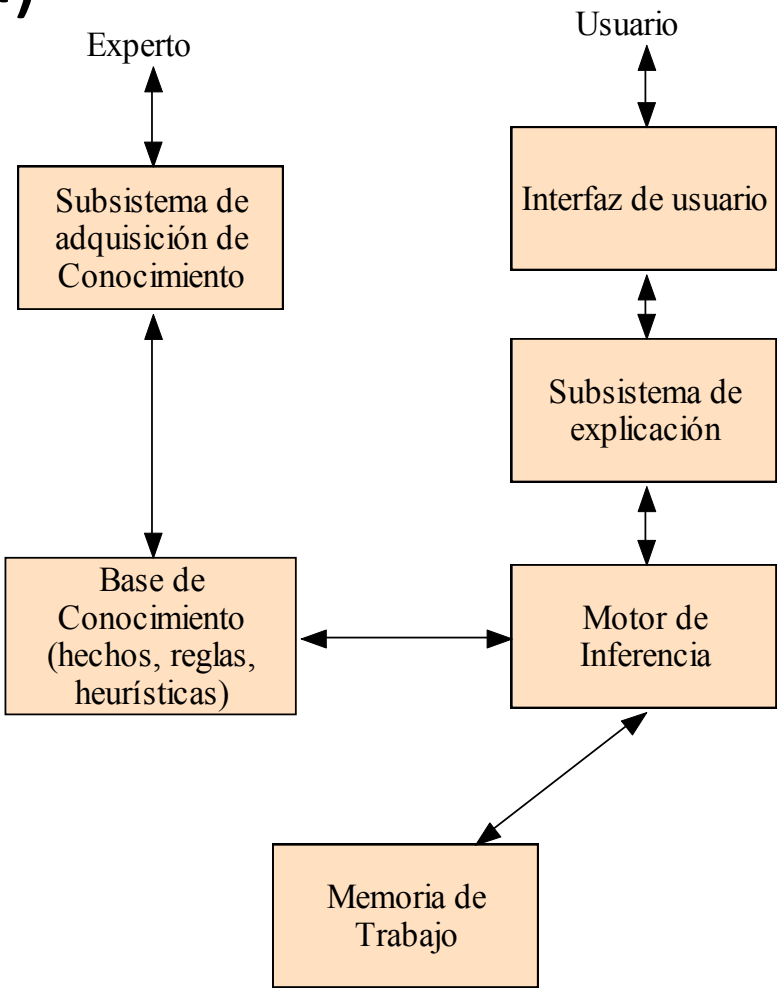
- Un **SEBR** es un **SBC** donde el conocimiento se incluye en forma de reglas y hechos.
- Estas reglas y hechos pueden implementarse, por ejemplo, mediante el **cálculo de predicados**.
- El proceso de construcción de un SEBR es el siguiente:
  - Se **extrae el conocimiento** experto (bibliografía, 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.



Otros modelos/problemas de representación del conocimiento

- Representación del conocimiento de sentido común
- Organización jerárquica del conocimiento
- Razonamiento temporal
- …

Organización jerárquica del conocimiento

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

$$Impresora.laser(Snoopy)$$

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

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

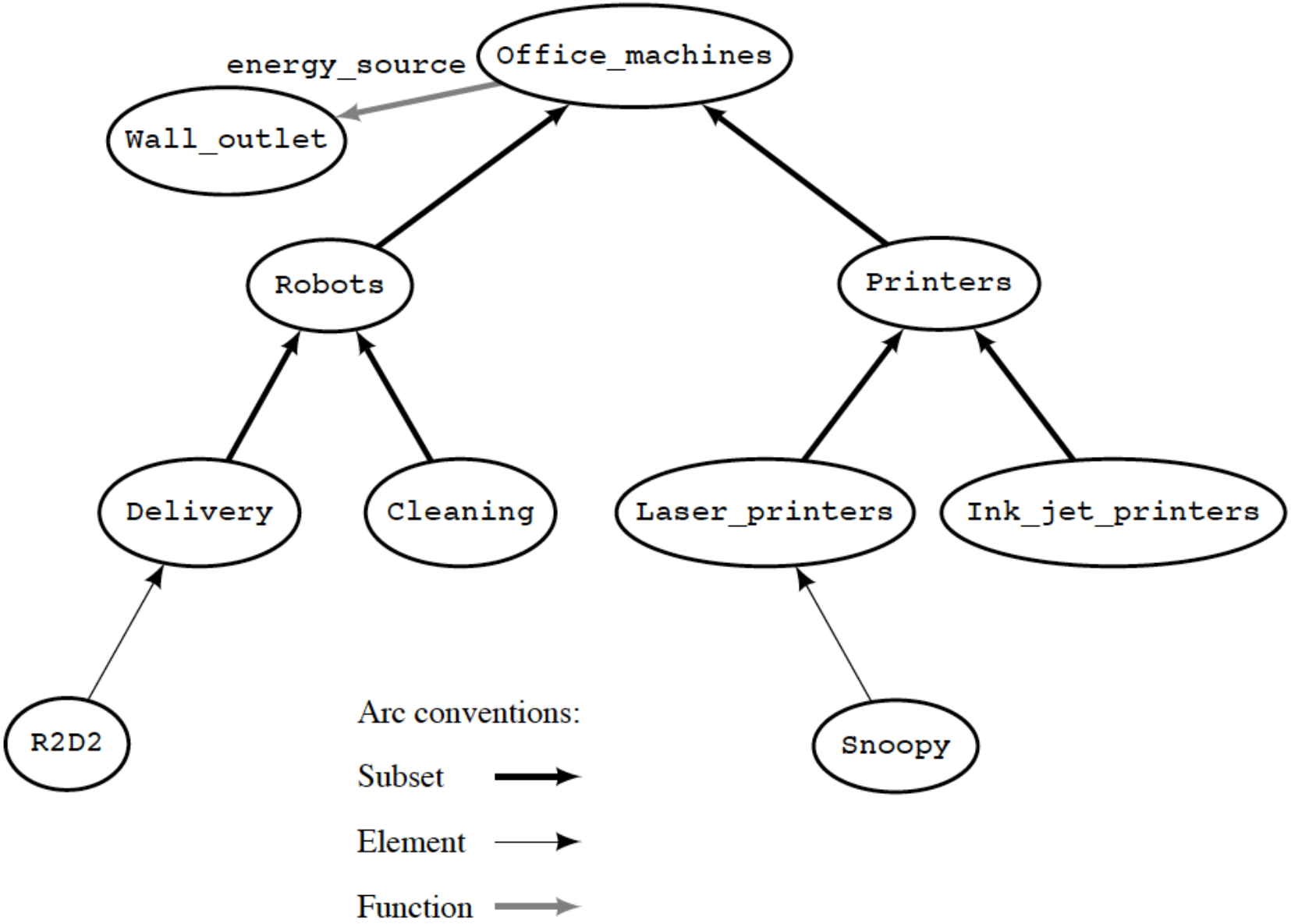
- Herencia de Propiedades

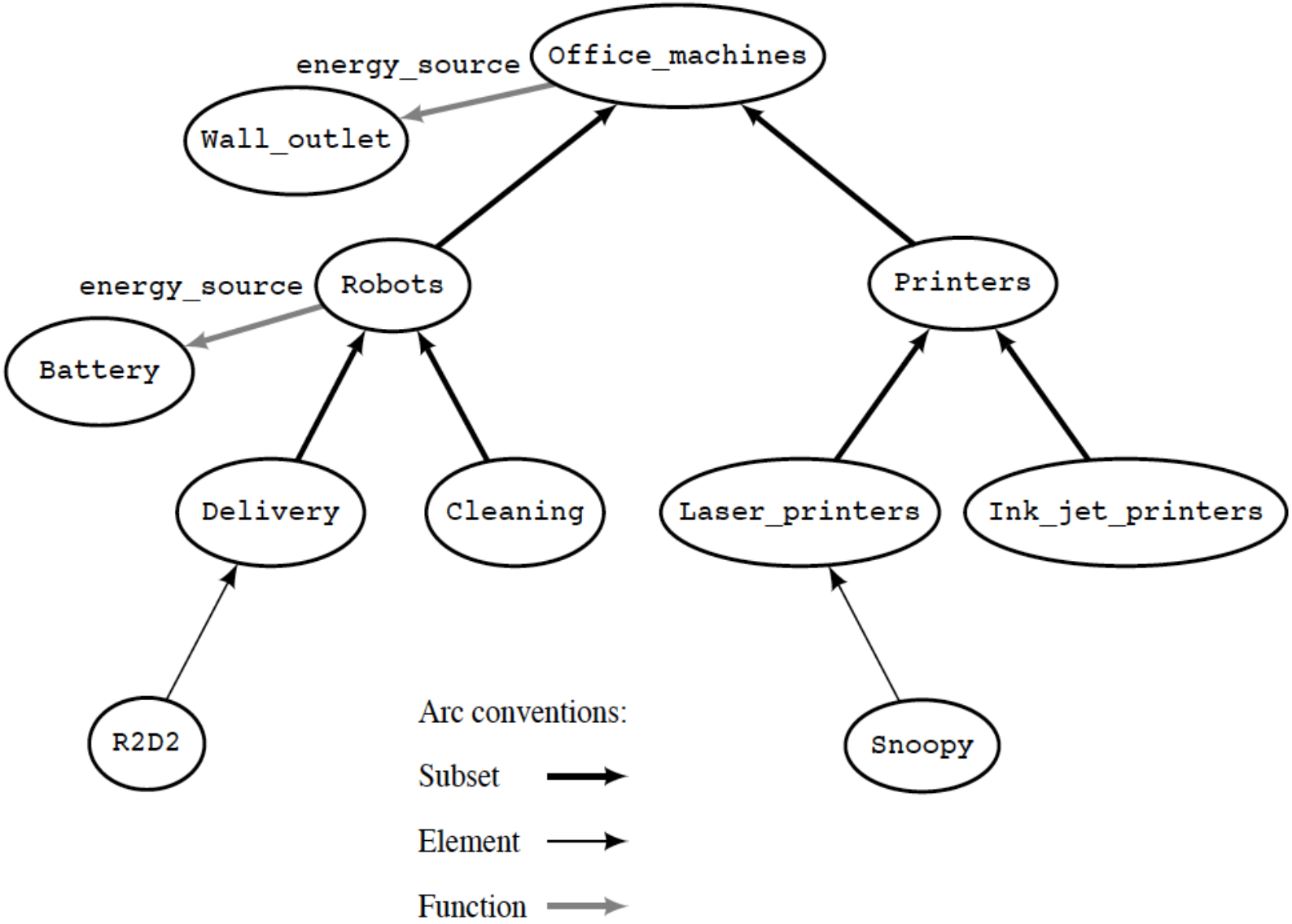
$$(\forall x)[Maquina.de.oficina(x) \supset [Fuente.de.alimentacion = Toma.de.la.pared]]$$

$$(\forall x)[Impresora.laser(x) \supset [Fuente.de.alimentacion = Toma.de.la.pared]]$$

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


energy_source
Office_machines
Wall_outlet
Robots
Printers
Delivery
Cleaning
Laser_printers
Ink_jet_printers
R2D2
Snoopy
Arc conventions:
Subset
Element
Function


Office_machines
energy_source
Wall_outlet
Robots
energy_source
Battery
Printers
Delivery
Cleaning
Laser_printers
Ink_jet_printers
R2D2
Snoopy
Arc conventions:
Subset
Element
Function

Razonamiento temporal

- Allen (1983,1984): El tiempo es algo dinámico, sobre el cual los procesos y los evento 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

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