Tema 5. Representación del Conocimiento

- Representación del conocimiento en IA.
- El cálculo proposicional.
- Resolución en el cálculo proposicional.
- Cálculo de predicados.
- Resolución en el cálculo de predicados.
- Modelos de conocimiento heredable
- Sistemas Basados en el Conocimiento.

Objetivos

- Adquirir las habilidades básicas para construir sistemas capaces de resolver problemas mediante técnicas de IA.
- 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
- S. Russell, P. Norvig, "*Inteligencia Artificial: Un enfoque moderno",* Ed. Prentice Hall, 2ª edición, 2004.

MÉTODOS EN INTELIGENCIA ARTIFICIAL

- **Inteligencia:** ***capacidad de encontrar rápidamente una solución adecuada en lo que en principio es un inmenso espacio de alternativas.***
- ***La búsqueda*** **aparece como algo que debe realizar todo (o casi todo) sistema que pretendamos que manifiesta una conducta inteligente**

MÉTODOS EN INTELIGENCIA ARTIFICIAL

- ***Principio del Conocimiento*:**

Un sistema exhibe un comportamiento inteligente, debido principalmente al conocimiento que puede manejar: conceptos, hechos, representaciones, métodos, modelos, metáforas y heurísticas en su dominio de actuación

Representación del conocimiento en IA

Conocimiento: Representación simbólica de “aspectos” de un cierto dominio o “universo” de discurso.

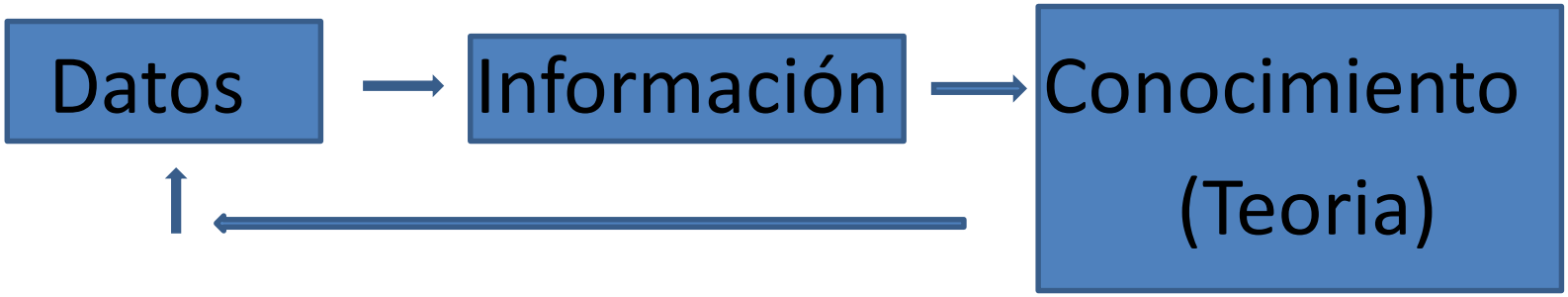
Los “aspectos” pueden ser hechos, relaciones o procedimientos.

Información: Cualquier estímulo capaz de alterar el estado o respuesta de un sistema.

Debe ser un conocimiento “sorprendente”.
La medida de la “sorpresa” será la Entropía.

Representación del conocimiento en IA

- La adquisición de conocimientos se produce a partir de la información extraíble de los datos y su "encapsulamiento" en un modelo teórico siguiendo un proceso cíclico



Representación del conocimiento en IA

Hemos estudiado el modelado del mundo de un agente reactivo empleando:

- **Representaciones icónicas:** "dibujos/mapas" del mundo

Y solo hemos mencionado para agentes deliberativos:

- **Representaciones descriptivas:** "aspectos/situaciones/hechos ciertos o falsos sobre el mundo.

Representación del conocimiento en IA

- Las representaciones descriptivas tienen ciertas ventajas sobre las icónicas. Genéricamente:
  - Son más sencillas.
  - Son fáciles de comunicar a otros agentes.
  - Se pueden descomponer en piezas más simples.

Representación del conocimiento en IA

Por otra parte 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 afirmaciones o negaciones sobre ciertos atributos/características del agente o su entorno.

Representación del conocimiento en IA

El **conocimiento sobre el mundo** puede utilizarse para razonar sobre él y hallar nuevas características del mismo.

Ejemplo 1: "Si Los hombres aman a las mujeres"; y "Juan es un hombre"

Entonces "Juan ama a las mujeres".

Ejemplo 2 : Un robot sólo puede levantar un bloque si tiene suficiente batería y el bloque es elevable.

"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 él y su entorno.

Representación del conocimiento en IA

Normalmente cualquier representación del conocimiento pertinente a un dominio real conlleva representar conceptos (conjuntos de objetos) y propiedades de los objetos del conjunto

No es lo mismo representar conjuntos que propiedades de los objetos del conjunto.

Ejemplo:
Un perro tiene patas, pero el concepto perro no las tiene.

Representación del conocimiento en IA

Todo concepto puede ser definido por:

- Intensión: Explicitar las características que debe poseer cada elemento de un conjunto.
- Extensión: Se indican los elementos que pertenecen al conjunto. Hipótesis del mundo cerrado.

Cualquier definición intensional debe ser computable.

Niveles de conocimiento

- Conocimiento de los elementos del dominio o **conocimiento factual**. Hechos relevantes. (Ej. un coche falla si se estropea el sistema de inyección)
- **Heurísticas** sobre el conocimiento. Jerarquizan el conocimiento factual. (Ej. las averías mecánicas son mas frecuentes que averías eléctricas)
- **Metaconocimiento**, conocimiento acerca del conocimiento pero independiente del mismo. (Ej: la aparición de una sola avería es mas frecuente que la aparición simultanea de dos).

Niveles de representación del conocimiento

- **Esquema o modelo de representación:** Formalismo simbólico empleado para expresar el conocimiento. Modelo mental.

- **Lenguaje de representación:** Lenguaje formal (cadenas de caracteres) empleado para representar el modelo o esquema elegido.

- **Lenguaje de programación:** Lenguaje formal empleado para implementación en un computador.

Representación del Conocimiento

- **Cuestión general:**

  Semántica del modelo de representación.

- **Cuestiones particulares:**
  - Qué cosas pueden representarse.
  - Granularidad de la representación (nivel de detalle).
  - Descripciones intensionales versus descripciones extensionales.
  - Representación de metaconocimiento.
  - Jerarquización de objetos y herencia.

Representación del Conocimiento

- Adicionalmente:
  - Problema del marco: Saber que cosas son fijas en el sistema y no van a cambiar.
  - Problema de cualificación: Hasta que “granularidad” se contemplan las excepciones.

Representación del Conocimiento

- Átomo básico: Triple <objeto, atributo, valor>

<antonio, edad, 21>

- Hecho: lista de triples <objeto, atributo, valor>

(<antonio, edad, 21>, <antonio, estado-civil, soltero>)

Modelos de representación inspirados por la Psicología

1.- Modelos de conducta: Reglas de Producción o reglas IF__THEN

Si A entonces B (A: condición; B: Acción)

2.- Modelos de razonamiento: Lógica (s).

Lenguaje formal para representar conocimiento y formular inferencia.

3.- Modelos de memoria: Almacenaje y representación.

3.1 Modelos simples: Ficheros y Bases de datos.

3.2 Modelos de conocimiento heredable. Almacenamiento jerarquizado:

Redes Semánticas,

Frames, guiones,

Representaciones orientadas a objetos…

Modelos de Razonamiento: Lógicas

Una lógica es un modelo de representación de conocimiento que consta de:

1) Un lenguaje formal con:

   1.1) Símbolos,

   1.2) Sintaxis: Reglas para construir “cadenas” válidas (f.b.f o w.f.f),

   1.3) Semántica: Relación entre las f.b.f de la lógica y el conocimiento que se quiere representar.

2) Unas reglas de inferencia (Teoría de la demostración) que permitan obtener nuevo conocimiento a partir de conocimiento previo.

Lógicas

Las lógicas más sencillas son:

- Lógica de proposiciones: Cálculo proposicional
- Lógica de primer orden: Cálculo de predicados

Cálculo Proposicional

Símbolos:

- Proposiciones {p, q, r…….}
- Valores lógicos : {V, F} , {0,1}

(Proposiciones y valores lógicos: ATOMOS)

- Conectivos lógicos: $\wedge, \vee, \Leftarrow, \neg, \Leftrightarrow$
- Separadores. Paréntesis, llaves, coma, punto y coma.

Cálculo Proposicional

Sintaxis (Reglas para construir fbf):

- Un atomo es una fbf
- Si P y Q son fbf entonces ¬P, ¬Q , P∧Q, P∨Q,
  P⇒ Q, P⇔Q son fbf.
- Una f.b.f entre separadores es una f.b.f

Dos fbf's con los mismos símbolos pueden ser diferentes dependiendo de la posición de los separadores:

(P∨Q)∧R, P∨(Q∧R)

Cálculo Proposicional

Ejemplos de fbfs:

$$(P \wedge Q) \Rightarrow \neg P$$

$$P \Rightarrow \neg P$$

$$P \vee P \Rightarrow P$$

$$(P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)$$

Cálculo Proposicional

Semántica:

- Se asocia a cada proposición un hecho de un universo de discurso: U $\longleftrightarrow$ P

- V: P $\longrightarrow$ {0,1} (lo mismo que) {F,V}

Función de veritación: valor de verdad.

Interpretación

Significado.

Tablas de la verdad

La lógica de proposiciones es funcional:

El valor de verdad de una fbf compleja se obtiene a partir de los valores de verdad de sus componentes empleando las **Tablas de verdad.**

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

Interpretación

A la hora de resolver problemas de IA con CP es esencial hacer una correcta **interpretación del CP,** dicho de otro modo dar una semántica correcta.

Ejemplo: Se desea representar el conocimiento

"Si la batería del robot funciona y el bloque A está en el suelo, entonces se puede levantar".

Definimos los átomos:

- $BATERIA\_OK$, $ESTA\_A\_SUELO$, $LEVANTAR\_A$.
- Definimos la fbf:

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

Satisficibilidad 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: $A \wedge B \Rightarrow C$

Si $A=BATERIA\_OK,\ B=ESTA\_A\_SUELO,\ C=LEVANTAR\_A,$ la semántica se corresponde con el entorno y el mundo a modelar.

Si $A=TOCA\_LOTERIA,\ B=TENGO\_SALUD,\ C=TIRAR\_POR\_VENTANA,$ la semántica es inconsistente con lo que se modela.

Validez y equivalencia

- Se dice que una fbf es **válida** si se cumple independientemente de la interpretación que se le asocie. Ejemplo: $P \Rightarrow P, \neg(P \wedge \neg P)$

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.

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

En el diseño de agentes, debemos evitar fbfs con interpretaciones equivalentes para hacer más eficiente el proceso de razonamiento.

Deducción y Reglas de inferencia

Se dice que Q se sigue (se infiere a partir) de un conjunto de axiomas $\{P^1 \ldots\ldots\ldots\ldots\ldots P^n\}$

$$\{P^1 \ldots\ldots\ldots\ldots\ldots P^n\} \vdash Q,$$

si Q es verdadero siempre que $P^1 \wedge \ldots\ldots \wedge P^n$ sea verdadera.

Para probar que $\{P^1 \ldots\ldots\ldots\ldots\ldots P^n\} \vdash Q$, hay que encontrar una DEMOSTRACIÓN.

Para la demostración se emplean las denominadas REGLAS DE INFERENCIA

Monotonia

La Lógica de proposiciones es **monótona**:

Si $\{P^1 \ldots\ldots\ldots\ldots P^n\} \vdash Q$, entonces también $\{P^1 \ldots\ldots\ldots\ldots P^n, W) \vdash Q$

cualquiera que sea W, (incluso la negación de alguno de los P^i o cualquier contradicción con varios de ellos).

Esta propiedad es indeseable en I.A. porque impide la revisión de las demostraciones al actualizar las Bases de Conocimiento

Reglas de Inferencia

Las reglas de inferencia son mecanismos para obtener fórmulas verdaderas a partir de otras que lo sean.

Modus Ponens $[(P \Rightarrow Q), P] \vdash Q$

Modus Tolens $[(P \Rightarrow Q), \neg Q] \vdash \neg P$

Reglas de inferencia

- Además de Modus ponenes y modus Tolens existen reglas de inferencia comunes que utilizamos de modo “automático”

  - $\mathbf{Q} \wedge \mathbf{P}$ 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 $\neg(\neg \mathbf{P})$

Demostración

Sea **Δ** un conjunto de fbfs.

- Una secuencia de **n** fbfs **{w_1, w_2, w_3, …, w_n}**, se denomina **demostración** o **deducción** de **w_n** a partir de **Δ** si, y sólo si, cada **w_i** de la secuencia pertenece a **Δ** o puede inferirse a partir de fbfs en **Δ**.
- Si existe tal demostración, entonces decimos que **w_n** es un **teorema de Δ**, y que **w_n** puede demostrarse desde **Δ:**

$$\Delta \vdash w_n, \text{ o como } \Delta \vdash_R w_n$$

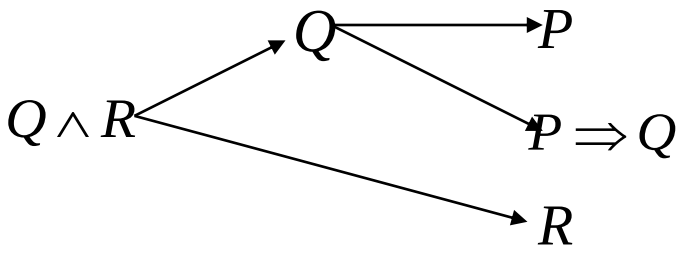
para indicar que **w_n** se demuestra desde **Δ** mediante las reglas de inferencia **R**.

Demostración

- **Ejemplo:**
  - Sea el conjunto de fbfs Δ, $\Delta = \{P, R, P \Rightarrow Q\}$
  - Entonces, la siguiente secuencia es una demostración de la fbf: $R \wedge Q$

$$\{P, P \Rightarrow Q, Q, R, Q \wedge R\}$$

  - La demostración se puede llevar a cabo fácilmente a través del siguiente **árbol de demostración**, utilizando **Δ** y las reglas de inferencia:



Resolución en C.P.

- **Claúsulas**
  - **Un Literal** es un átomo o su negación. Ejemplo: **P**, **¬P**
  - **Una cláusula Σ** es una disyunción de literales. Ejemplo:

$$\Sigma = P \vee Q \vee \neg R$$

  - Por contener solamente el operador OR una cláusula puede verse como un conjunto de átomos. Por tanto, **Σ={P,Q,¬R}**
  - Denominaremos **Nil** a la cláusula vacía **Σ={}**
  - Empleando clausulas se ha desarrollado un potente mecanismo de inferencia: La Resolución.

Resolución en C.P.

- **Idea general de la Resolución:**
  - Siendo **λ** un literal y **Σ_1, Σ_2** dos cláusulas, a partir de **$\{\lambda\} \cup \Sigma_1$** y **$\{\neg\lambda\} \cup \Sigma_2$** se puede inferir **$\Sigma_1 \cup \Sigma_2$**.

- Ejemplo: Resolviendo {R,P} y {Q , ¬P}, es decir

$$R \vee P, \neg P \vee Q$$

se obtiene {R,Q}, es decir $R \vee Q$

Modus Ponens y Modus Tolens son casos particulares del mecanismo de Resolución

Resolución en C.P.

- Si una fbf está escrita como conjunción de cláusulas, se dice que está en **FNC** (Forma Normal Conjuntiva).
- Si una fbf está escrita como disyunción de cláusulas, se dice que está en FND (Forma Normal Disyuntiva).

Resolución en el cálculo proposicional

- **Cualquier fbf en el cálculo proposicional puede escribirse como conjunción de cláusulas.**
- Para transformar una fbf a FNC se siguen 3 pasos:
  - 1. Eliminar implicaciones mediante su equivalente

$$A \Rightarrow B \equiv \neg A \vee B$$

  - 2. Reducir el ámbito de aplicación de la negación ¬ mediante las leyes de DeMorgan
  - $$\neg(A \vee B) \equiv \neg A \wedge \neg B; \neg(A \wedge B) \equiv \neg A \vee \neg B$$
  - 3. Usar asociatividad y distributividad para pasar a FNC.

Resolución en el cálculo proposicional

- **Ejemplo:** $\neg(P \Rightarrow Q) \vee (R \Rightarrow P)$
  - 1. Eliminar implicaciones.

    $$\neg(\neg P \vee Q) \vee (\neg R \vee P)$$

  - 2. Reducir el ámbito de aplicación de la negación ¬.

    $$(P \wedge \neg Q) \vee (\neg R \vee P)$$

  - 3. Usar las leyes asociativas y distributivas para pasar a FNC.
    - Distributiva: $(P \vee \neg R \vee P) \wedge (\neg Q \vee \neg R \vee P)$
    - Simplificamos: $(P \vee \neg R) \wedge (\neg Q \vee \neg R \vee P)$
  - **Solución:** El conjunto de cláusulas $\{(P \vee \neg R), (\neg Q \vee \neg R \vee P)\}$

Demostración por Refutación

- **La refutación** es un método para demostrar que una clausula **w** se sigue de un conjunto Δ de ellas.
- La idea es intentar demostrar que la negación de **w** es inconsistente con Δ, quedando así demostrada, por tanto, la veracidad de dicha cláusula.
- La idea es que, si “algo” es cierto y se puede demostrar, entonces se puede demostrar que lo contrario de ese “algo” no puede ser cierto.

Demostración por Refutación

- **Se siguen 4 pasos p**ara refutar una fbf **w:**
  - 1. Convertir las fbfs de Δ como conjunciones de cláusulas
  - 2. Convertir **¬w** como conjunción de cláusulas
  - 3. Unir el resultado de los pasos 1 y 2 en un único conjunto **Γ**
  - 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**

Resolución en el cálculo proposicional

- **Se podrán producir dos resultados:**
  - **Se llega a generar la cláusula vacía Nil**. El proceso de refutación termina con éxito y queda demostrada **w**.
  - **El proceso termina sin generar la cláusula vacía**. Entonces no se puede demostrar **w**.
- **Ejemplo:** Supongamos que en el mundo de los bloques Δ es:
  Hechos: $BATERIA\_OK \ ; \neg ROBOT\_SE\_MUEVE$.

  Conocimiento : $BATERIA\_OK \wedge OBJETO\_ELEVABLE \Rightarrow ROBOT\_SE\_MUEVE$
  - Queremos demostrar **w**= $\neg OBJETO\_ELEVABLE$

Demostración por Refutación

- **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

$$OBJETO\_ELEVABLE$$

Demostración por Refutación

- **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 $\neg OBJETO\_ELEVABLE$

Demostración por Refutación

- 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.
- Sin embargo, si queremos elaborar un procedimiento automático que implemente este método, ¿cómo debemos seleccionar las cláusulas a resolver?

Demostración por Refutación

- ¿En qué orden se deben realizar las resoluciones para obtener **Nil** con la mayor brevedad?
  - Esta cuestión es análoga a la de qué nodo/estado hay que expandir en la búsqueda en espacios de estados (temas 2 y 3):
    - Estrategias de primero en anchura.
    - Estrategias de primero en profundidad.etc.
- Llamamos **resolvente de nivel 0** a las cláusulas iniciales, incluída la negación de la cláusula a demostrar.
- Una **resolvente de nivel i+1** es aquella que se obtiene tras resolver una resolvente de nivel **i** con una resolvente de nivel **j**, con **j<=i**.

Demostración por Refutación

- **Estrategias de primero en profundidad.etc.**
  - En primer lugar se generaría una resolvente de nivel 1.
  - Luego se usaría la resolvente del nivel 1 con alguna del nivel 0 para generar una nueva resolvente de nivel 2.
  - Seguidamente, se usaría la resolvente de nivel 2 y alguna entre las resolventes de los niveles 1, 0 para generar la resolvente del nivel 3
  - Etc.
- Si fijamos un nivel de profundidad máximo, se puede aplicar una técnica de vuelta atrás para hacer la refutación.

Demostración por Refutación

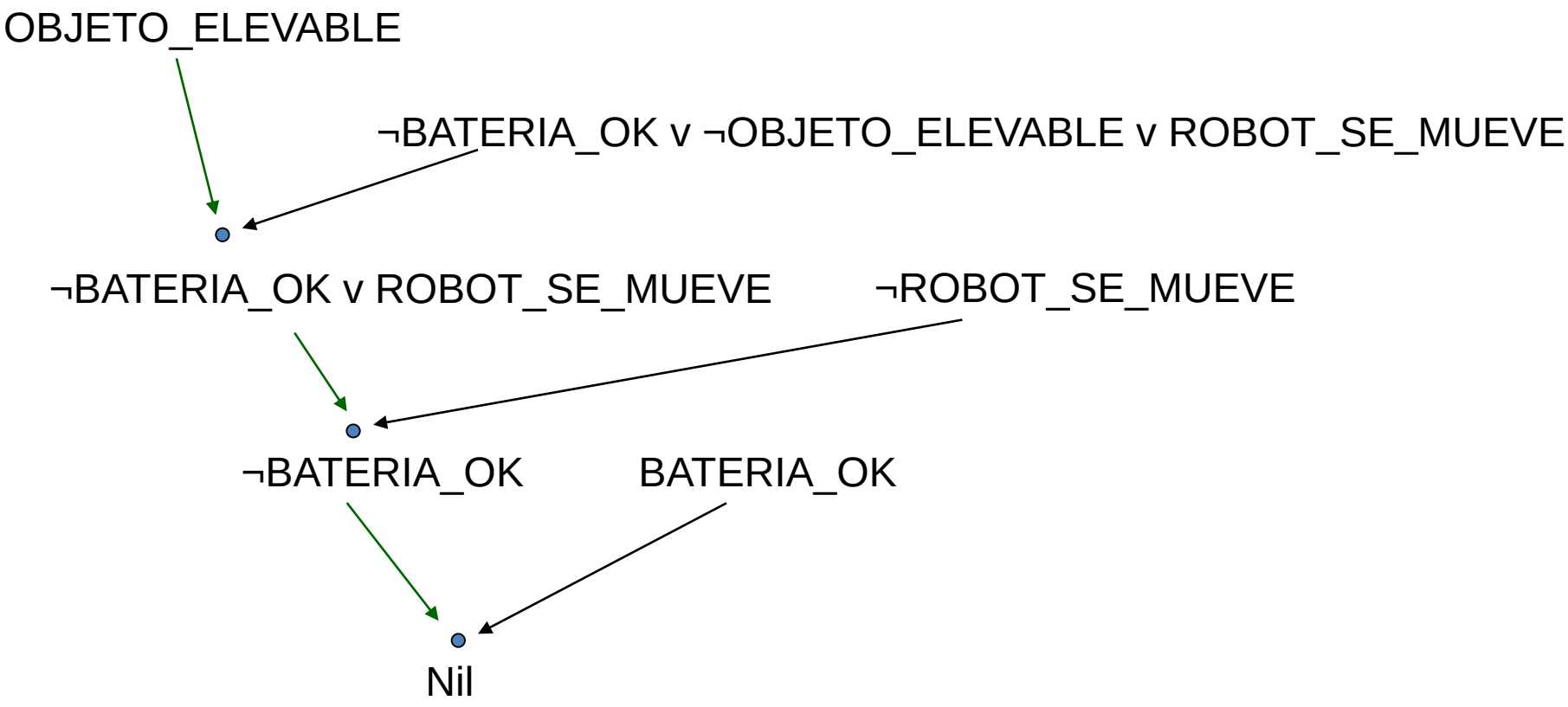
- **Estrategia de preferencia unitaria:**
  - Se da preferencia a resolver las cláusulas en las que, al menos una de las cláusulas a resolver, tienen un único literal.
  - Esta estrategia se puede generalizar en “resolver primero las más simples”, intentando que al menos una de las dos cláusulas a resolver sea unitaria. En caso de que no pueda ser, se escogerá que al menos una de las dos cláusulas tenga sólo dos literales, etc.

Demostración por Refutación

- **Estrategia de refinamiento de conjunto soporte:**
  - Se dice que una cláusula C_2 es descendiente de otra cláusula C_1 si: a) $\mathbf{C_2}$ es resultado de resolver $\mathbf{C_1}$ y alguna otra cláusula.
    - b) $\mathbf{C_2}$ es resultado de resolver alguna cláusula sucesora de $\mathbf{C_1}$ y alguna otra cláusula.
- **Conjunto soporte**: Conjunto de cláusulas que provienen de la negación de la cláusula a demostrar, o bien descendientes de estas cláusulas.
- **La estrategia del conjunto soporte** realiza resoluciones en las que una de las cláusulas que se resuelven pertenece siempre al conjunto soporte.

Resolución en el cálculo proposicional

- **Ejemplo de estrategia del conjunto soporte:** El proceso de refutación del ejemplo anterior sigue esta estrategia:



Notas finales

- **El cálculo proposicional es limitado en capacidad semántica.**
- Supongamos nuestro mundo de bloques. Para decir que el bloque **A** está sobre el bloque **B**, deberíamos establecer una proposición P≡$SOBRE\_A\_B$.
- Para representar la situación de todos los bloques, necesitaríamos tantos literales como combinaciones de bloques y posiciones relativas.
- 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 sin relación formal entre ellas para implementar este conocimiento utilizando únicamente cálculo proposicional**.**

El Cálculo de Predicados

- Sería de gran utilidad un lenguaje que permitiese manejar 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

"si un bloque está sobre otro el de debajo no está libre"

para cualquier par de bloques , el cálculo de predicados nos permite abstraer los objetos mediante variables y escribir:

$$\forall x, y SOBRE(x, y) \Rightarrow \neg LIBRE(y)$$

- "*cuando un objeto x esté sobre otro y, entonces y no estará libre*".

Lógica de Predicados

- La lógica de predicados tiene su origen en los trabajos de G. Frege (1879).
- Su forma actual y sus propiedades fueron dadas por Tarski (1935)
- Se basa en la lógica aristotélica, por la que una cosa es verdadera o falsa.
- La lógica de Predicados es la base del lenguaje declarativo Prolog.

El Cálculo de Predicados

- SINTAXIS
- Símbolos:
- Alfabeto latino junto con los caracteres numéricos.
- Valores de verdad (V y F ó 0 y 1…)
- Conectivos lógicos: $\wedge, \vee, \Rightarrow, \neg, \Leftrightarrow$
- Separadores. Paréntesis, llaves, coma, punto y coma, etc.
- Cuantificadores $\forall(\text{para todo}), \exists(\text{existe})$

Cálculo de Predicados

- Componentes. Sintaxis
- **Constantes** y **variables**: cadenas de caracteres.
- **Función**: cadena_caracteres(t_1……..t_n) con ti término.
- **Predicado**: cadena_caracteres(t_1……..t_n) con ti término.
- **Término**: constante, variable o función.
- Los predicados tienen igual sintaxis que las funciones, pero no pueden aparecer como términos de funciones o predicados (Lógica de Primer Orden) y hay que distinguirlos.
- Llamaremos **aridad** de una función o un predicado al número de terminos que tiene.

El Cálculo de Predicados

Reglas para construir f b f

- Todo predicado (o un valor de verdad) es una fórmula bien formulada (fórmula atómica), las funciones no lo son.

- Si P y Q son f.b.f entonces $\neg P$, $\neg Q$, $P \wedge Q$, $P \vee Q$, $P \Rightarrow Q$, $P \Leftrightarrow Q$, (P), (Q), son fbf.

- Si P(x) es una f.b.f. que contiene a la variable x entonces $\forall$ x P(x) y $\exists$ x P(x) son fbf.

El Cálculo de Predicados

- **Literal:** fórmula atómica o su negación.
- **Cláusula:** disyunción de literales.
- **Variable libre:** No ligada a un cuantificador. No vamos a poder dar semántica (valor de verdad) a las variables libres. Ej: Hombre (x).
- **Sentencia (Expresión):** f.b.f que no contiene variables libres.
- Nos centraremos en sentencias.

Semántica en Cálculo de Predicados

- Como en la lógica de proposiciones la idea es asociar a cada sentencia un hecho, propiedad o relación de un universo **(interpretación).**
- $S \leftrightarrow U$
- V: S ----> {0,1} [o en {V,F}]

- OJO: Las formulas con variable libres no pueden tener semántica

Semántica en Cálculo de Predicados

- Más concretamente:
- Objetos del dominio: Términos.
  - Objetos concretos: constantes.
  - Objetos indeterminados: variables.
  - Objetos vinculados: funciones.
- Propiedades: predicados : unarios.
- Relaciones:
  - predicados n-arios
  - funciones
  - predicados sin argumentos
- Hechos complejos: Fórmulas, Expresiones o sentencias.

Modelos e interpretaciones

- En cálculo de predicados, una **interpretación** es una correspondencia biunivoca entre

  1.- Los objetos del mundo y los nombres de objetos,

  2.- Las funciones n-arias y los nombres de funciones n-arias, y

  3.- Las relaciones n-arias y los nombres de relaciones n-arias.

- Dada una interpretación, un **átomo tendrá valor V** sólo si está asociado a un hecho verdadero . En caso contrario, tendrá el **valor F**.

- Las veracidad o falsedad del resto de sentencias se determinan mediante las tablas de verdad y las reglas de la cuantificación.

Cuantificación

- Equivalencias útiles entre cuantificadores:

$$\neg(\forall x)P(x) \equiv (\exists x)\neg P(x)$$
$$\neg(\exists x)P(x) \equiv (\forall x)\neg P(x)$$

- Semántica de cuantificadores:
  - $(\forall x)P(x)$ es cierta si **P(x)** es cierta para cualquier valor de **x**.
  - $(\exists x)P(x)$ es cierta si, hay al menos un valor de **x** para el que **P(x)** sea cierta.
  - Ejemplos:

$$(\forall x)[Sobre(x,C) \Rightarrow \neg Libre(C)]$$
$$(\exists x)[Sobre(x,Suelo)]$$

Cuantificación

- **¡Cuidado con la cuantificación!:**

$$(\forall x)(\forall y) \equiv (\forall y)(\forall x)$$
$$(\exists x)(\exists y) \equiv (\exists y)(\exists x)$$
$$(\forall x)(\exists y) \neq (\exists y)(\forall x)$$

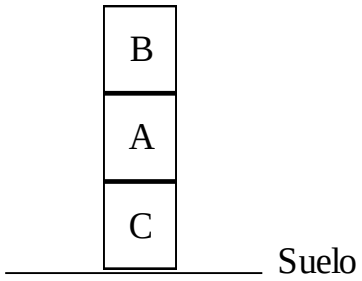
- La primera parte de la última sentencia quiere decir que "Para cualquier objeto **x**, existe al menos un objeto **y** tal que...".La segunda parte de la última sentencia quiere decir que "Existe al menos un objeto **y** tal que todos los objetos **x**...".
- **Ejemplo:** a) Parte 1. "**Para cualquier modelo de pasarela existe al menos un hombre fan**". b) Parte 2. "**Existe al menos un hombre que es fan de todas las modelos de pasarela**".

$$(\forall x)(\exists y) Modelo(x) \wedge Hombre(y) \wedge Fan(x, y)$$
$$(\exists y)(\forall x) Modelo(x) \wedge Hombre(y) \wedge Fan(x, y)$$

Modelos e interpretaciones

Ejemplo. El mundo de los bloques,

- Introducimos las constantes **A, B, C, Suelo**, y les asignamos su interpretación con los bloques reales y el suelo.
- Introducimos los predicados **Sobre(x,y)** y **Libre(x)**, y los interpretamos indicando que **x** está sobre **y**, y que **x** está libre (no tiene nada encima) respectivamente.
- Si el estado del mundo es el de la figura adjunta el siguiente estado del mundo entonces se tiene que : **Sobre(B, A), Sobre(A, C), Sobre(C, Suelo)** y **Libre(B)** son atomos con valor de verdad **V**.



Modelos e interpretaciones

- **Ejemplo:** Esta asignación se representa en la siguiente tabla (donde se hace uso de la Hipótesis del Mundo Cerrado y del Conjunto de Objetos Cerrado):

Cálculo de Predicados	Mundo
A	A
B	B
C	C
Suelo	Suelo
Sobre	<B,A>, <A,C>, <C,Suelo>
Libre	<B>

Modelos e interpretaciones

- **Ejemplo:** La información de la tabla anterior se puede utilizar para conocer la veracidad de otras fbfs del cálculo de predicados. Ejemplo:
  - **Sobre(A,B)** es falsa porque **<A,B>** no está en la lista de relaciones **Sobre**.
  - **Libre(B)** es verdadera porque **<B>** está en la relación **Libre**
  - **Sobre(C, Suelo)^¬Sobre(A,B)** es verdadera porque ambas relaciones son verdaderas.

Modelos e interpretaciones

- Una interpretación **satisface** una fbf, si la fbf es verdadera bajo esa interpretación.
- Una interpretación que satisface una fbf es un **modelo** de esta.
- Las fbf con valor **V** en cualquier interpretación son **fbf válidas**.
- Las fbf que no tiene ningún modelo se llaman **inconsistentes** o **insatisfacibles**.
- Si una fbf **w** es **V** para todas las interpretaciones posibles en las que el conjunto **Δ** es verdadero, entonces **w** se demuestra desde **Δ**.
- Dos fbf son **equivalentes** si sus valores **V** son idénticos en cualquier interpretación.

Modelos e interpretaciones

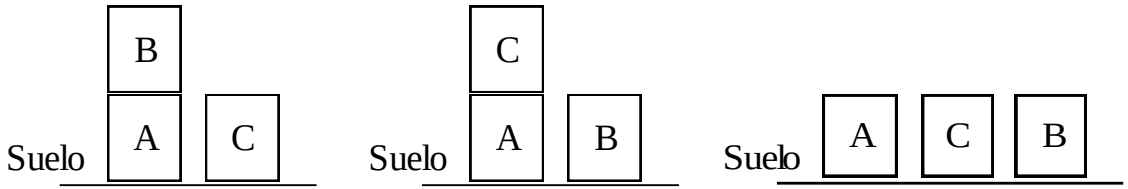
- El cálculo de predicados se utiliza para representar el conocimiento que tiene un agente sobre el mundo.
- Al conjunto Δ formado por las sentencias asociadas a los hechos del mundo se le llama **Base de Conocimiento.**
- Al construir una Base de Conocimiento, debemos pensar en generar el suficiente número de fórmulas que permitan identificar el modelo del mundo que se representa.
- Pero también, a su vez, debemos evitar ambigüedades e intentar evitar excluir otros modelos posibles adicionales.

Modelos e interpretaciones

- **Ejemplo:** Supongamos la siguiente Base de Conocimiento (B.C.):

$$1.\ Sobre(A, Suelo) \Rightarrow Libre(B)$$
$$2.\ Libre(B) \wedge Libre(C) \Rightarrow Sobre(A, Suelo)$$
$$3.\ Libre(B) \vee Libre(A)$$
$$4.\ Libre(B)$$
$$5.\ Libre(C)$$

- Entonces, cualquiera de los siguientes modelos podría ser válido:



- Añadiendo **Libre(A)** a la B.C., se elimina toda ambigüedad, definiendo claramente el modelo del mundo.

Conceptualización

- El gran problema en IA no es cómo representar, sino **qué** representar. Requiere de una gran habilidad del diseñador. Ejemplos:
  - *"Todos los paquetes que están en la habitación 27 son más pequeños que los de la 28"*

$$(\forall x, y)[[\text{Paquete}(x) \wedge \text{Paquete}(y) \wedge \text{EnHabitacion}(x, \text{H27}) \wedge \text{EnHabitacion}(y, \text{H28})] \Rightarrow \text{MasPequeño}(x, y)]$$

  - *"Cada paquete de la habitación 27 es más pequeño que uno de los paquetes de la habitación 29"*
    - Esta frase es ambigua (cosa frecuente en el lenguaje natural), ya que puede representar dos situaciones.

Conceptualización

- Ejemplos:
  - *"Cada paquete de la habitación 27 es más pequeño que uno de los paquetes de la habitación 29"*
    - Situación 1:

$$(\exists y)(\forall x)[\text{Paquete}(x) \wedge \text{Paquete}(y) \wedge \text{EnHabitacion}(x, H27) \wedge \text{EnHabitacion}(y, H29)] \Rightarrow \text{MasPequeño}(x, y)]$$

    - Situación 2:

$$(\forall x)(\exists y)[\text{Paquete}(x) \wedge \text{Paquete}(y) \wedge \text{EnHabitacion}(x, H27) \wedge \text{EnHabitacion}(y, H29)] \Rightarrow \text{MasPequeño}(x, y)]$$

Conceptualización

El cálculo de predicados nos permite también conceptualizar la noción del tiempo.

Ejemplo, si se desea decir:

"El paquete A ha llegado antes que el paquete B",

introducimos una nueva variable, **z**, que modele el tiempo:

$$(\exists z1, z2)\mathbf{[HaLlegado(A, z1)} \wedge \mathbf{H}aLlegado\mathbf{(B, z2)} \wedge \mathbf{A}ntes\mathbf{(z1, z2)]}$$

Deducción y Reglas de inferencia

Se dice que Q se sigue de un conjunto de axiomas $\{P^1 \ldots\ldots\ldots\ldots\ldots P^n\}$

$$\{P^1 \ldots\ldots\ldots\ldots\ldots P^n\} \vdash Q,$$

si Q es verdadero siempre que $P^1 \wedge \ldots\ldots \wedge P^n$ sea verdadera.

Para probar que $\{P^1 \ldots\ldots\ldots\ldots\ldots P^n\} \vdash Q$, hay que encontrar una DEMOSTRACIÓN.

Para la demostración se emplean las denominadas REGLAS DE INFERENCIA

Monotonia

La Lógica de Predicados, igual que la de Proposiciones es **monótona** ya que

$$\text{Si } \{P^1 \ldots\ldots\ldots\ldots P^n\} \vdash Q, \text{ entonces también}$$
$$\{P^1 \ldots\ldots\ldots\ldots P^n, W) \vdash Q$$

Cualquiera que sea W, (incluso la negación de alguno de los P^i o una contradicción con varios de ellos).

Esta propiedad es indeseable en I.A. porque impide la revisión de las demostraciones al actualizar las Bases de Conocimiento.

Demostración

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

Reglas de Inferencia

Las reglas de inferencia son mecanismos para obtener fórmulas verdaderas a partir de otras que lo sean:

- Modus Ponens: $[(P \Rightarrow Q), P] \vdash Q$
- Modus Tolens: $[(P \Rightarrow Q), \neg Q] \vdash \neg P$
- Resolución: $[\{\lambda\}UP$ y $\{\neg\lambda\}UQ] \vdash PUQ$

La resolución engloba al Modus Ponens y al Modus Tolens. Es "util" cuando se emplea sobre clausulas, en particular clausulas de Horn.

La aplicación de las reglas de inferencia precisa acoplamiento perfecto, en particular unificación de variables.

Inferencia y Cuantificación

Además, para tratar con los cuantificadores, se introducen dos nuevas reglas:

- **Particularización**. Este se elimina instanciando la/s variable/s cuantificadas en los objetos que hacen que la fbf sea cierta, dentro de la interpretación actual.

$$(\forall x)P(x, f(x), B), \text{instanciando } x \text{ en } A : P(A, f(A), B)$$

- **Introducción del cuantificador existencial.** Se introduce abstrayendo los objetos de una fbf.

$$(\forall x)Q(A, f(A), x), \text{abstrayendo } A : (\exists y)(\forall x)Q(y, f(y), x)$$

Resolución en el cálculo de predicados

- Podremos hacer demostración por resolución siempre que se pueda convertir cualquier conjunto de sentencias en un conjunto de clausulas semánticamente equivalentes.

- Como ocurre en el caso del Cálculo de Proposiciones existe un procedimiento sistemático para esta conversión, solo que ahora tiene nueve pasos

Conversión en clausulas

1.- Eliminar implicaciones (como en cálculo proposicional).

2.- Reducir ámbito de negaciones a los literales (como en c. proposicional).

3.- Reetiquetar variables para que cada cuantificador tenga su símbolo de variable.

4.-Eliminar cuantificadores existenciales, empleando objetos o funciones de Skolem:

$$\exists x P(x) \rightarrow P(S)$$

$$(\forall x)[(\exists y)P(x,y)] \rightarrow (\forall x)[P(x, h(x))]$$

Conversión en clausulas

5.-**Colocar los cuantificadores universales a la izquierda.**

Se tiene la denominada Forma Prenexa. La parte que contiene los cuantificadores se denomina “prefijo” y el cuerpo de la fórmula “matriz” (formada por uniones, intersecciones, negaciones…)

6.- **Colocar las conjunciones fuera de los literales.** Empleamos la propiedad distributiva.

$$(P \cap Q) \cup R \Leftrightarrow (P \cup R) \cap (Q \cup R)$$

Conversión en Clausulas

7. **Eliminar cuantificadores universales.**
8. **Eliminar los símbolos** ∩. Las expresiones del tipo $w_1 \cap w_2$ se pueden reemplazar por conjuntos de fbfs **{**w_1, w_2**}**.
9. **Renombrar variables**, para que el mismo símbolo no aparezca en las diferentes cláusulas

Demostración por Refutación

Se hace de forma análoga como lo hacíamos en el cálculo proposicional:

- Se niega el Teorema,
- Se añade al conjunto de axiomas,
- Se convierte todo en clausulas,
- Se resuelve hasta obtener NIL o concluir que el teorema no puede probarse.

Estrategias de resolución: Igual que en Calculo de Proposiciones

Refutación para demostrar teoremas

- **Ejemplo:** La Base de Conocimiento de un agente contiene los siguientes elementos:

$$(\forall x, y)[\{Paquete(x) \wedge Paquete(y) \wedge$$
$$\wedge EnHabitacion(x, H27) \wedge EnHabitacion(y, H28)\} \Rightarrow MasPequeño(x, y)]$$
$$Paquete(A)$$
$$Paquete(B)$$
$$EnHabitacion(B, H27)$$
$$MasPequeño(B, A)$$

- **Pregunta: ¿Dónde está el paquete A?**

Resolución para demostrar teoremas

- **Ejemplo (continuación):** El robot puede inferir fácilmente que EnHabitacion(A,H27) v EnHabitacion(A,H28).
- Mediante refutación, se comienza a probar si EnHabitacion(A,H27), introduciendo su negación ¬EnHabitacion(A, H27) en la Base de Conocimiento y comprobando si se llega a **Nil**.

- **Ejercicio:** Elaborar el árbol de refutación mediante resolución, partiendo de

EnHabitacion(A,H27) v EnHabitacion(A,H28),¬EnHabitacion(A, H27).

PROBLEMAS DEL USO DE L.P. EN I.A.

1.- Problemas semánticos

1.1 Es difícil expresar todo en fórmulas lógicas: heurísticas, metaconocimiento, jerarquía y herencia, igualdad, sentido común, etc.
2.- Razonamiento temporal.
3.- Razonamiento acerca de predicados
4.- Información incompleta y/o imprecisa (vaguedad, probabilidad).
5.- Excepciones.
6.- Monotonia

2.- Problemas computacionales

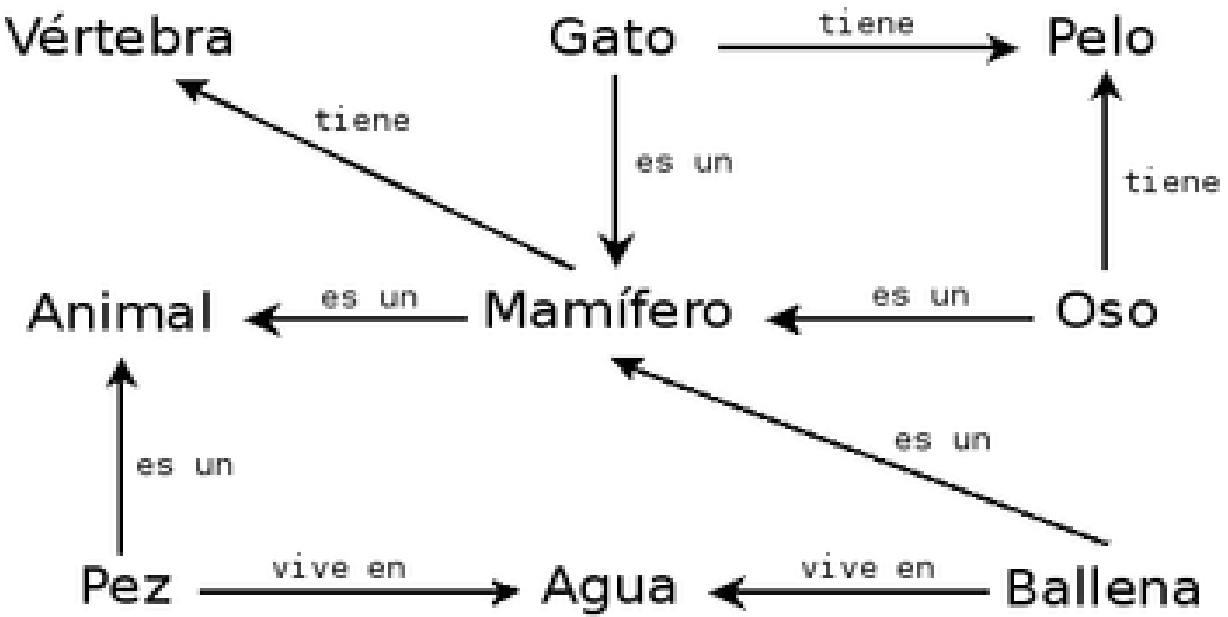
2.1.- Consistencia (solidez, soundness): lo que se demuestra como verdadero lo es realmente. Refutación por resolución.
2.2.- Completitud: si una cosa es verdadera puede demostrarse. Refutación por resolución. Semidecible.
2.3.- Complejidad Computacional (tratabilidad)

REDES ASOCIATIVAS

En las **redes asociativas** cada nodo representa un concepto (o una proposición) y los enlaces corresponden a relaciones (inclusión, pertenencia, causalidad) o a categorías gramaticales (verbo principal, sujeto, objeto, complementos, etc.). entre ellas, se conocen:

- **redes semánticas**: son las destinadas a representar o a comprender el lenguaje natural.
- **redes de clasificación**: es exactamente lo que su nombre indica, una clasificación de objetos o conceptos con sus características propias (herencia y demás).
- **redes causales**: son las que llevan asociadas, junto a sus nodos que representan variables, una relación de influencia, representada por los enlaces.

Ejemplo Red Semántica



Marcos

Un marco es una estructura de datos para representar una situación estereotipada, como encontrarse en un cierto tipo de sala de estar o asistir a un cumpleaños infantil.

Cada marco se caracteriza por un conjunto de campos o slots que se asocian en general a atributos, y que en conjunto sirven para identificar los marcos.

Los marcos están especialmente concebidos para tareas de reconocimiento: la información recibida hace que se activen unos marcos y esta a su vez provoca la activación de otros marcos conectados con los primeros, dando lugar así a una *red de activación*, cuyo objetivo es predecir y explicar la información que se va a encontrar en esa situación. Este reconocimiento suele denominarse herencia o más generalmente reconocimiento descendente.

Ejemplo Marcos

- FRAME: pájaro
  - is_a: animal
  - forma_moverse: volar
  - activo_durante: día
- FRAME: pingüino
  - is_a: pájaro
  - color: blanco_y_negro
  - forma_moverse: andar
  - activo_durante: noche
  - tamaño: mediano
- FRAME: pepe
  - is_a: pingüino

Sistemas basados en el conocimiento, Sistemas Expertos

Un Sistema Basado en el Conocimiento (SBC) es aquel Programa (en general sistema hardware-software) que emplea masivamente conocimiento para resolver un problema dentro de un dominio determinado.

Los SBC se aplican a una gran diversidad de campos y/o áreas. A continuación se listan algunas de las principales:

	Militar		Informática		Telecomunicaciones	
	Química		Derecho		Aeronáutica	
	Geología		Arqueología		Agricultura	
	Electrónica		Transporte		Educación	
	Medicina		Industria		Finanzas y Gestión	

Sistemas Basados en el Conocimiento

Dentro de esta categoría destacan los Sistemas Expertos (SE) que pueden definirse como aquellos SBC capaces de comportarse como un experto (humano) en un determinado dominio de actividad:

- Resuelven un problema
- Pueden ser consultados y justifican su Razonamiento

Durante bastante tiempo y para un amplio sector de la comunidad científica no especialista, los SE fueron sinónimos de IA.
http://aaai.org/AITopics/AIApplicationsAreas

Ejemplos de SBC

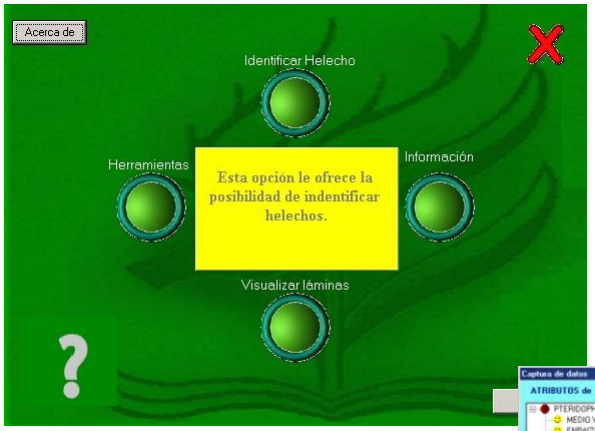
Anna (Agente conversacional de IKEA). Su Base de Conocimiento está descrita en lenguaje AIML (*Artificial Intelligence Markup Language*).

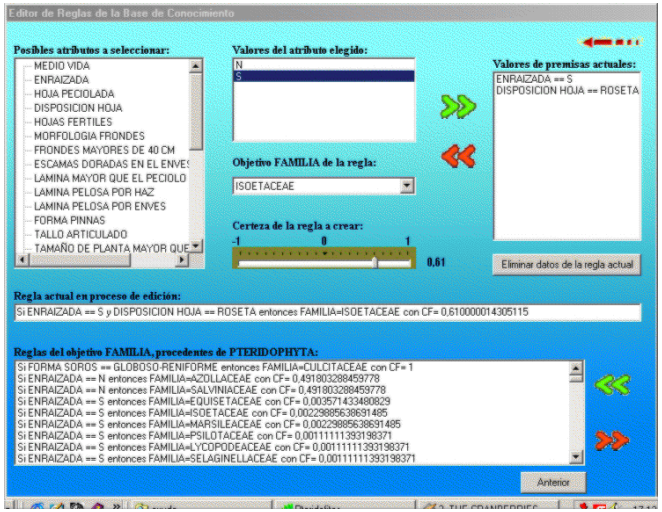
Ejemplos de SBC

El **Paciente Simulado Virtual (PSV)** es un agente que simula enfermedades. Se utiliza para la formación de especialistas en medicina: El médico pregunta y/o visualiza síntomas que el paciente presenta y debe decidir un diagnóstico válido.

Ejemplos de SBC

Pteridophita es un experto en helechos que, de forma interactiva con un usuario, permite identificar tipos de helechos y proporcionar información sobre los mismos.





Sistemas Basados en el Conocimiento

Un **Sistema Basado en el Conocimiento (SBC)** necesita 3 componentes básicas:

1.- Una **Base de Conocimiento (BC)**, que contenga el conocimiento necesario sobre el dominio/Universo del problema a resolver.

La BC contendrá una descripción del mundo: sus objetos y las relaciones entre ellos.

2.- Un **Motor de Inferencia**, que permite razonar sobre el conocimiento de la BC y los datos proporcionados por un usuario.

3.- Una **interfaz de usuario** para entrada/salida de datos de cada problema. A veces los datos de un problema se conocen como Base de Hechos.

4.- Los SE incorporan además un **módulo de explicaciones /justificación.**

Tareas en la construcción de un SBC

- La adquisición del conocimiento y como representarlo de una forma abstracta efectiva.
- La representación del conocimiento en términos de un formalismo que una máquina pueda procesar.
- La realización de de inferencias o como hacer uso de esas estructuras abstractas para generar información útil en cada caso específico.

Principales formalismos de representación

Pieza básica: triple objeto-atributo-valor

- lógica
- reglas de producción
- redes asociativas (semánticas)
- Marcos

Cada formalismo de representación usa un método de inferencia específico:

- Resolucion en Lógica
- Razonamiento hacia a delante y hacia atrás en sistemas de reglas
- Herencia en sistemas de frames

Lenguajes de SBC

Por otra parte, dependiendo del formalismo de representación hay que (se puede) emplear un Lenguaje de Programación diferente:

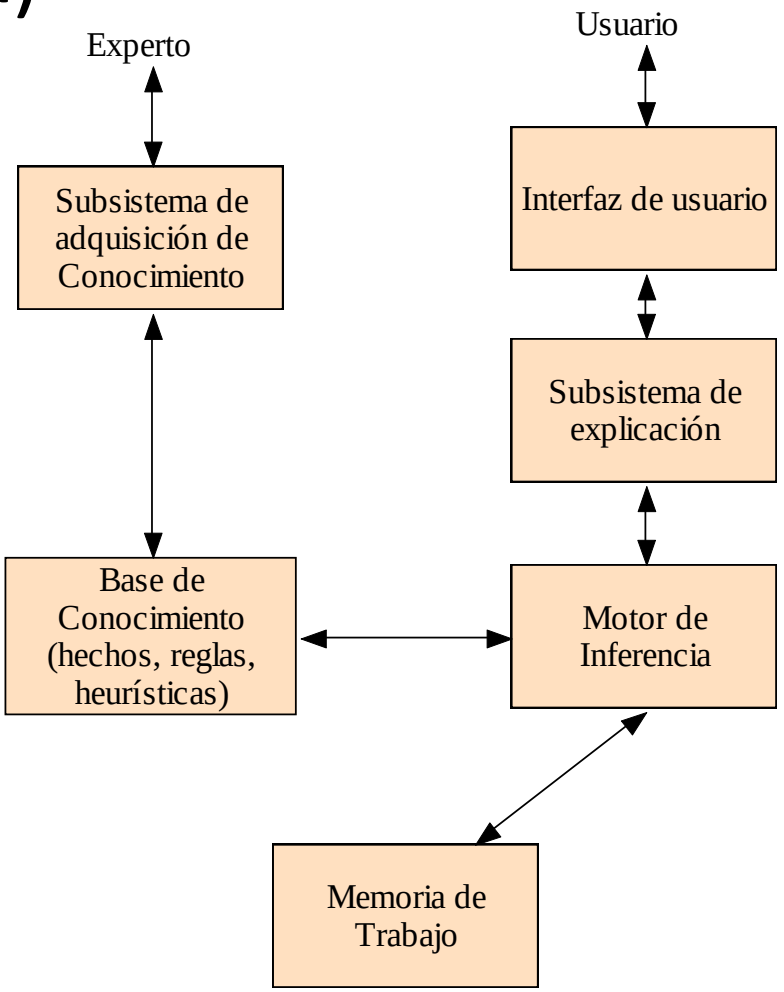
- LISP, PROLOG
- EMYCIN
- OPS5, ART, CLIPS
- KEE

Sistemas Expertos basados en Reglas (SEBR)

- Un **SEBR** es un **SBC** donde el conocimiento se incluye en forma de reglas y hechos. Son los Sistemas Expertos más clásicos y comunes.
- Aunque estas reglas y hechos pueden implementarse, por ejemplo, mediante el **cálculo de predicados**, no obstante es más "adecuado" emplear **reglas de producción** para las reglas.
- El proceso de construcción de un SEBR (estas ideas se extienden a SBC's con diferente estructura) 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,
- Se **crea la Base de Conocimiento** con el conocimiento adquirido.

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.

Extracción del conocimiento en SEBR

- Los métodos para extraer el conocimiento se estudian en el área de **aprendizaje automático** dentro de la IA (**extracción de reglas** o **aprendizaje de reglas).**
- Además, normalmente, la percepción del mundo por parte de un agente no es perfecta.
- Del mismo modo, es posible que una regla no sea aplicable siempre (aunque sí en un gran número de casos). Este hecho no permite que la regla sea admitida en un sistema de cálculo proposicional o de predicados, dado que daría lugar a sistemas inconsistentes.
- Se hace necesario establecer mecanismos para **tratar con incertidumbre**.

Un ejemplo

Objeto	Conjunto de posibles valores
Tarjeta	{verificada, no verificada}
Fecha	{expirada, no expirada}
NIP	{correcto, incorrecto}
Intentos	{excedidos, no excedidos}
Balance	{suficiente, insuficiente}
Límite	{excedido, no excedido}
Pago	{autorizado, no autorizado}

Table 1: Objetos y posibles valores para el ejemplo del cajero automático.

Un ejemplo

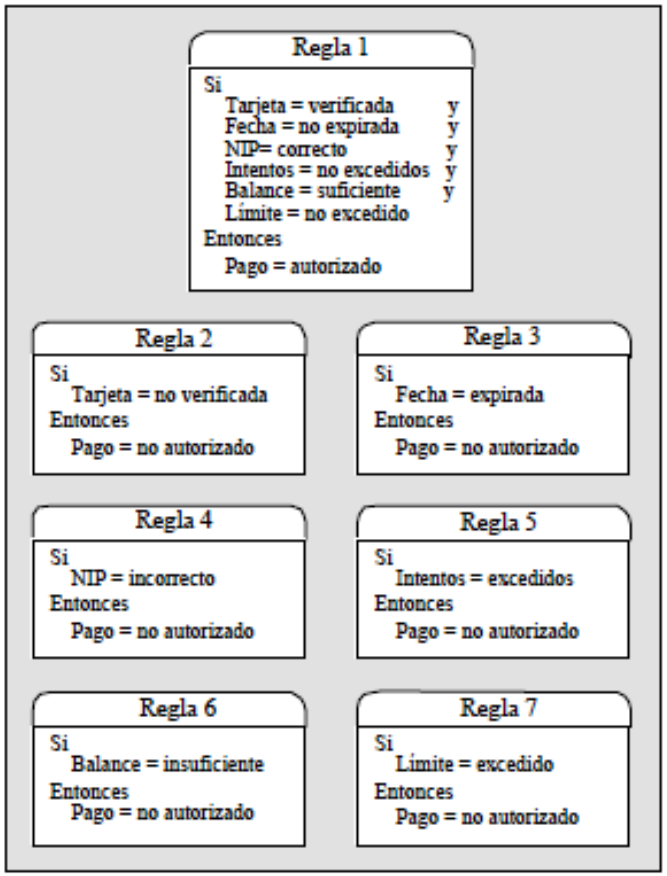


Figure 1: Ejemplos de reglas para sacar dinero de un cajero automático.

Otros modelos/problemas de representación del conocimiento

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