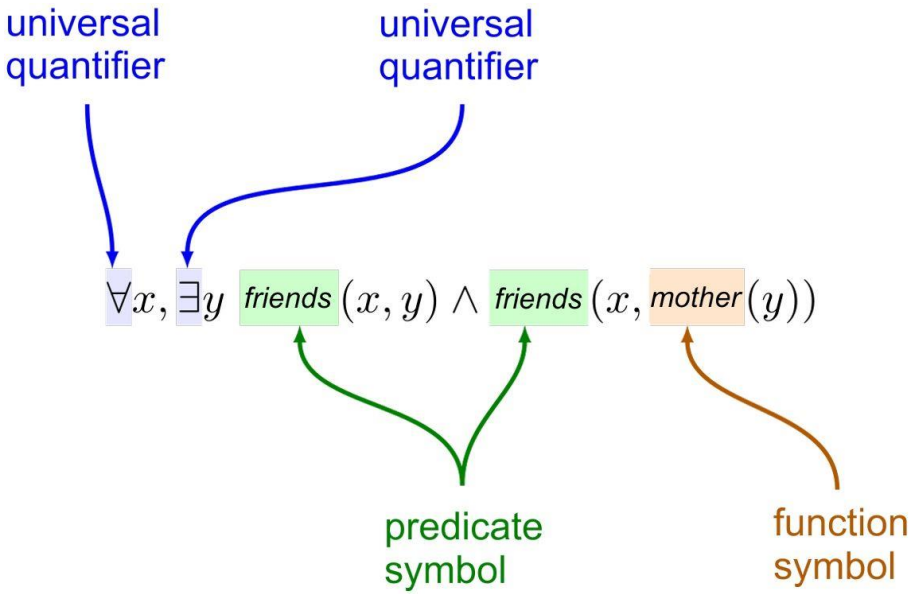
Grado en ingeniería informática
Inteligencia artificial 2020/2021

3. Lógica de predicados



UAM
Universidad Autónoma de Madrid

Esquema

3. Lógica de predicados

3.1 Elementos de la lógica de predicados

3.1.1. Elementos comunes con la lógica proposicional

3.1.2. Variables y cuantificadores

3.1.3. Predicados

3.1.4. Funciones

3.1.5. Átomos, términos, literales y cláusulas

3.1.6. Formas normales

3.2 Sustitución y unificación

3.3 Inferencia en lógica de predicados

3.3.1 Reglas de inferencia generalizadas

3.3.1.1 Modus ponens

3.3.1.2 Resolución

3.3.2 Extracción de respuestas mediante el truco de Green.

3.4 El predicado de igualdad

Lecturas :

- CAPÍTULOS 8,9 de Russell & Norvig
- CAPÍTULOS 15,16 de Nilsson

BIBLIOGRAFÍA:

- E. Paniagua Arís, J. L. Sánchez González, F. Martín Rubio, “Lógica computacional”, Thomson

- Melvin Fitting “First-order logic and automated theorem proving”, Springer-Verlag (New-York 1990)

Representación de conocimiento e inferencia

- Representación de conocimiento:
  algo más amplio y flexible que una heurística
  - Hemos visto heurísticas para guiar los algoritmos de búsqueda
  - Las heurísticas son un tipo de representación de conocimiento demasiado simple: Insuficiente para representar la complejidad de los sistemas reales
  - La resolución de problemas en dominios complejos requieren representaciones más generales y flexibles

- El conocimiento puede ser representado de múltiples maneras
- Elegir un formalismo que nos permita representar de forma adecuada ciertos hechos
  - Formalismo de representación de conocimiento
    - ¿De qué información dispongo?
  - Mecanismos de razonamiento o inferencia asociados
    - ¿Qué puedo hacer (deducir) con la información de la que dispongo?

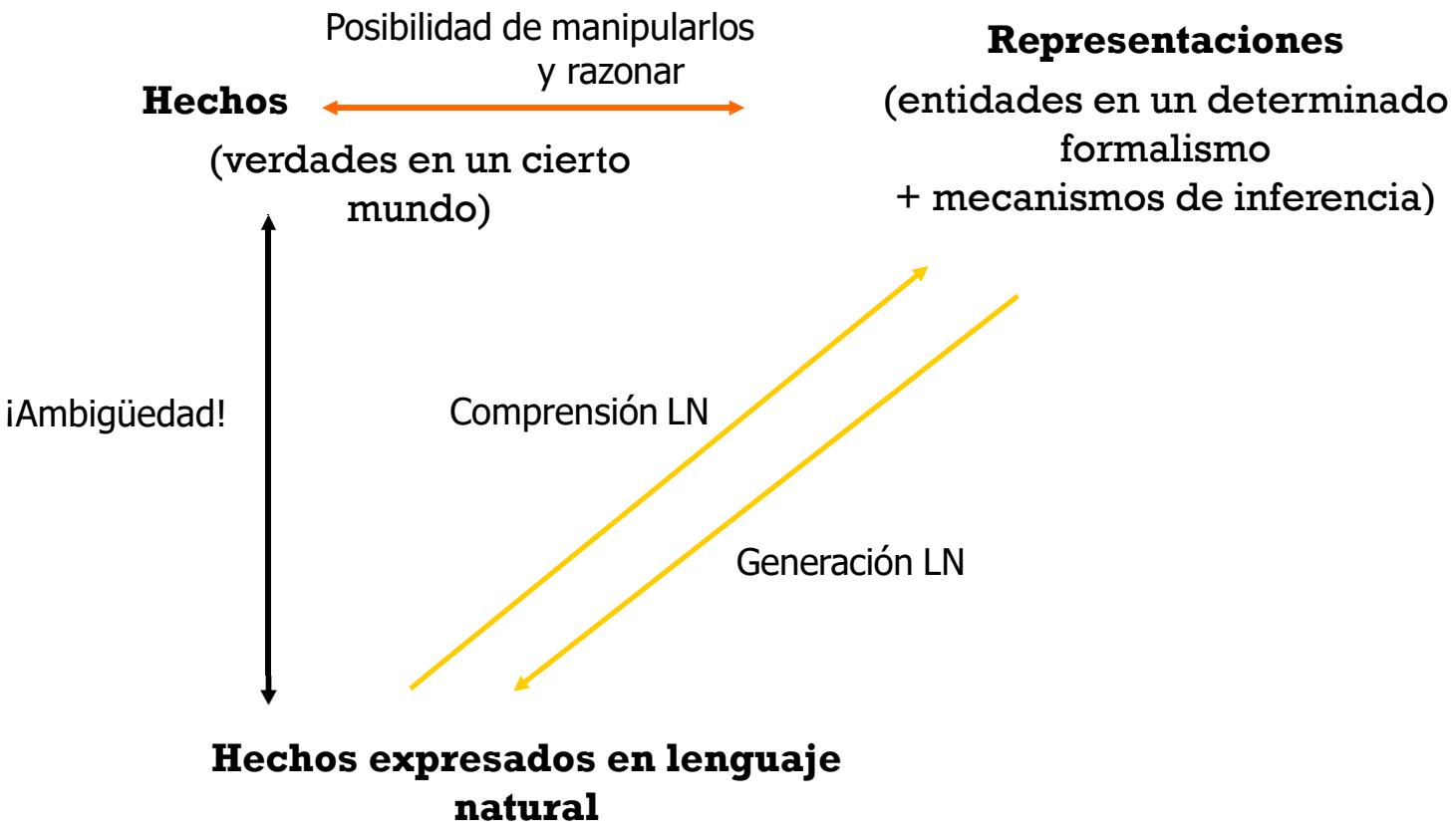
Tipos de Técnicas de representación

- Diversos formalismos para construir bases de conocimiento
  - Representaciones basadas en relaciones
    - Lógica
    - Redes semánticas
  - Representaciones basadas en objetos
    - Marcos
    - Objeto-Atributo-Valor
  - Representaciones basadas en acciones
    - Sistemas de producción (o de Reglas)
    - Guiones
  - Combinaciones y modificaciones de los anteriores

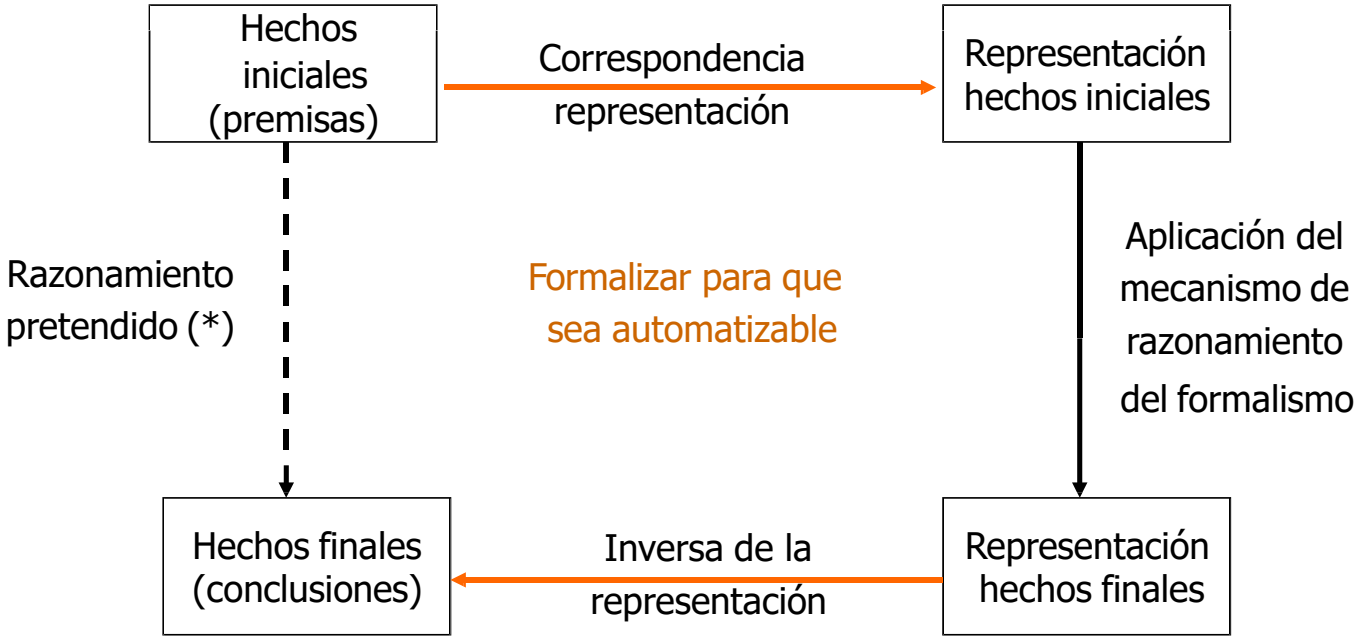
Técnicas de representación y razonamiento

- Mecanismos de inferencia
  - Obtención de nuevo conocimiento a partir del conocimiento actual
  - Deducción
    - A partir de leyes generales obtenemos conocimiento particular
      - Si las premisas son ciertas, la conclusión es verdadera.
  - Inducción
    - Es la generalización de la información extraida de casos particulares.
    - No se puede garantizar su validez. Es la base del aprendizaje
  - Abducción
    - Es la capacidad de generar explicaciones plausibles para un cierto hecho que ha ocurrido

Rol de la Representación: Ejemplo



Elementos que participan



Propiedades de una representación

- Para un determinado dominio hay que valorar (+ ó -)
  - Idoneidad representativa: capacidad de representación de todos los tipos de conocimiento necesarios en ese dominio
  - Idoneidad inferencial: capacidad de manipular los símbolos del formalismo de representación e inferir nuevo conocimiento (deseado)
  - Eficiencia inferencial: capacidad de incorporar meta-conocimiento que permita mejorar los procesos de razonamiento
  - Eficiencia adquisitiva: capacidad de adquirir fácilmente nuevo conocimiento del exterior, idealmente bajo control del propio sistema (o, simplemente, añadiéndolo una persona) manteniendo la consistencia con el conocimiento existente
- Ninguna técnica de representación optimiza todas estas propiedades para todos los dominios y tipos de conocimiento
  - Multitud de técnicas. Muchos sistemas basados en más de una

Tipos de conocimiento

Bases de conocimiento (BC)

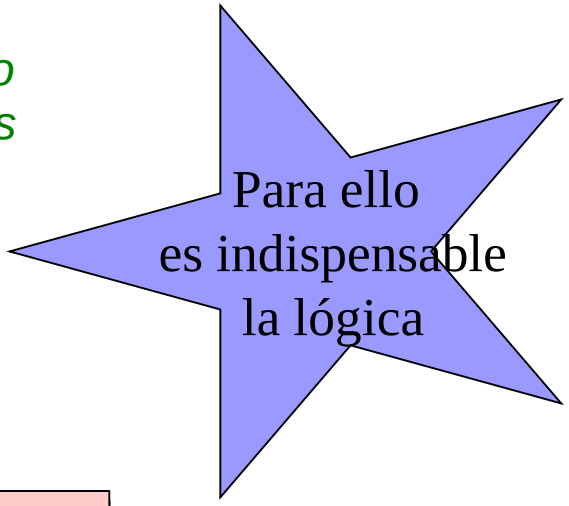
Todo el cuerpo de conocimiento utilizable por el sistema, representado en algún formalismo dado, junto con los mecanismos de gestión de ese conocimiento *(incorporación, supresión, modificación, consulta exacta, consulta aproximada, inferencia, control de consistencia, etc.)*

- Tipos de conocimiento
  1. **Factual o declarativo (representación de hechos)**
     - **Explícito: se introduce directamente**
     - **Implícito: se infiere a partir del conocimiento explícito**
  2. Procedimental
     - Indica cómo actuar mediante pasos en diversas situaciones
  3. Meta-conocimiento o conocimiento de control
     - Conocimiento a un nivel superior: conocimiento sobre el propio conocimiento, que permite gestionarlo con más eficiencia

Meta: **el conocimiento aparezca explícitamente y se logren conclusiones del conocimiento declarado**

- Aprender a diseñar agentes que:
  - Construyan interpretaciones del mundo
  - Deriven nuevas representaciones del mundo por inferencia
  - Usen esas nuevas representaciones para saber que hacer
- Representación de conocimiento:

El papel pretendido de la representación del conocimiento en IA es reducir problemas de acción inteligente en meros problemas de BÚSQUEDA
Grinsberg

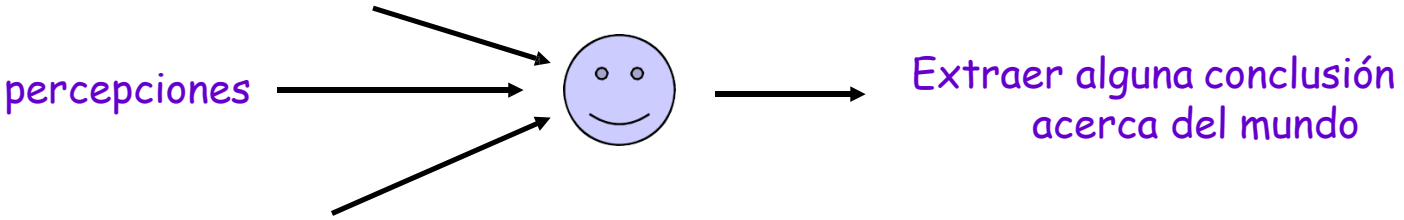


Analogía entre Programación y Problemas de IA

Programación	***Inteligencia Artificial***
1 .Crear un algoritmo para resolver el problema	1. Identificar el conocimiento necesario para resolver el problema
2. Seleccionar un lenguaje de programación para codificar la tarea	2. Seleccionar el lenguaje con el cual dicho conocimiento pueda ser representado
3. Capturar el algoritmo como programa	3. Escribir el conocimiento dentro de ese lenguaje
4. Ejecutar el programa	4. Usar las consecuencias del conocimiento para resolver el problema

¿Qué es una lógica?

- Una lógica es un lenguaje formal
  - Tiene una sintaxis que determina qué expresiones son *(la forma)*
  - También cuenta con una semántica que determina qué representan las expresiones legales *(el contenido)*
  - Y suele disponer de un sistema de inferencia que permite manipular expresiones sintácticas para obtener otras expresiones sintácticas
    - ¿Con qué propósito?
      - Obtener expresiones con un significado "interesante"
      - Que nos digan algo "nuevo" del mundo



Lógica

- compromiso ontológico
  - para el agente, qué existe en el mundo
    - en el caso de la lógica proposicional, para el agente existen hechos que serán verdaderos o falsos
- compromiso epistemológico
  - para el agente, cuál es la actitud con respecto a los hechos
    - en el caso de la lógica proposicional, el agente cree que una oración es verdadera o falsa, o no ha llegado a conclusión alguna

Tipos de lógicas y sus preocupaciones

Lenguaje	Ontología (lo que existe)	Epistemología (qué cree de los hechos)
Lógica Proposicional	hechos	verdadero/falso/no sabe
Lógica de primer orden	hechos, objetos, enlaces	verdadero/falso/no sabe
Lógica temporal	hechos, objetos, enlaces, tiempos	verdadero/falso/no sabe
Lógica de la probabilidad	hechos	grado de certidumbre
Lógica difusa	grado de verdad	grado de certidumbre

Lógicas de orden superior

- La lógica proposicional carece de capacidad para expresar ciertos tipos de conocimiento. Por ejemplo:
  - Descripción concisa de entornos con muchos objetos.
  - Enunciados generales relativos a todos o algunos de los objetos en un dominio específico, las relaciones entre ellos, la existencia de objetos con ciertas propiedades, etc.
- Tipos de lógica
  - **Lógica de orden cero** o **"Lógica proposicional"**

    Objetos, conectores lógicos.
  - **Lógica de primer orden** o **"Lógica de predicados"**

    ++ funciones, predicados, cuantificadores cuyos argumentos son variables cuyo dominio incluye un conjunto objetos

    ej. $(\forall x)\ \text{Humano}(x) \Rightarrow \text{Mortal}(x)$
  - **Lógica de segundo orden**:

    ++ funciones, predicados, cuantificadores cuyos argumentos pueden ser predicados

    ejs. $(\exists P)\ [P(A) \wedge P(B)]$

    $(\forall x)(\forall y)\ [(x=y) \Leftrightarrow (\forall P)(P(x) \Leftrightarrow P(y))]$ [***Leibniz***, 1666]
  - **Lógicas de orden superior**:

    ++ funciones, predicados, cuantificadores cuyos argumentos pueden ser predicados de predicados ...
  - **Teoría de tipos**

Lógica de primer orden

Ejemplo: Familia + amigos

- **Ontología**
  - **Conectores, Variables, Cuantificadores**
  - **Constantes**
    - **Objetos**: Pedro, Pablo, María
    - **Funciones**: madreDe^{1}, mejorAmigoDe^{1}
    - **Predicados**: Hombre^{1}, Mujer^{1} , Niñ@^{2} , Hijo^{2} , Hija^{2}, Casad@^{1}, Feliz^{1},…

- **Definición**: "La madre de uno es su progenitor mujer"

  $(\forall x,y)\ (\text{MadreDe}(x,y) \Leftrightarrow \text{ProgenitorDe}(x,y) \wedge \text{Mujer}(x))$

- **Aserciones generales**:

  "Todos los descendientes de María están casados, pero no son felices"

  $(\forall x)\ (\text{DescendienteDe}(x,\text{María}) \Rightarrow \text{Casado}(x) \wedge \neg\text{Feliz}(x))$

- **Aserciones existenciales**: "Algún hijo del mejor amigo de Pedro tiene algún descendiente"

  $(\exists x)(\exists y)(\text{HijoDe}(x,\text{mejorAmigoDe}(\text{Pedro})) \wedge \text{DescendienteDe}(y,x))$

Lenguaje, I

- **Constantes**:
  - **Objetos simbólicos** (normalmente en mayúscula)
    `Ej.  P, Q, Juan, PuertaDeAlcalá`
  - **Funciones** (en general, en minúscula)
    Argumentos de entrada: lista de términos entre paréntesis
    Evalúa a: un término.

    `Ej. padreDe`^{1} `, distanciaEntre`^{2}
  - **Relaciones o predicados** (en general, en mayúscula)
    Argumentos de entrada: lista de términos entre paréntesis
    Evalúa a: un valor de verdad ("Verdadero" o "Falso").

    `Ejs. EsPadreDe`^{2}`, EsBlanco`^{1}
    Las relaciones unarias se denominan "propiedades"

Notas:
- El superíndice denota la aridad de la función o predicado (es decir, el número de argumentos)
- Los objetos simbólicos pueden ser considerados como funciones de aridad cero

Lenguaje, II

- **Signos de puntuación**
  - **Coma:** ,
  - **Paréntesis:** ( ) [ ] { }

- **Conectores de lógica proposicional**:
  $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ (en orden de prioridad)

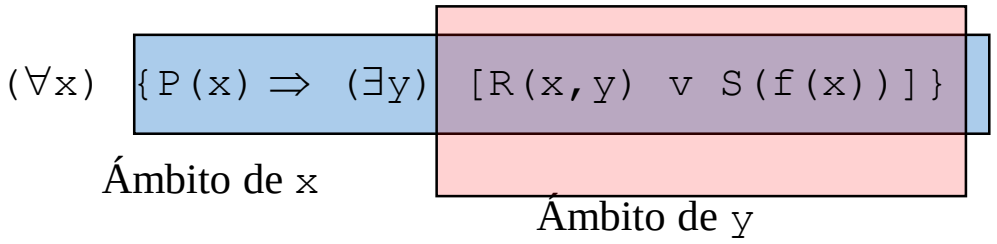
- **Variables** (en general, en minúscula)

  Las variables pueden tomar valores en un dominio.
  Ejs. x (reales), n (enteros), p (personas), etc.

  Ejs. x,y,p,q,…, p1,p2,…

- **Cuantificadores**
  - **Cuantificador universal ($\forall$)**
  - **Cuantificador existencial ($\exists$)**

Se usan con variables. Dichas variables se dicen que están ligadas al cuantificador.



Término, átomo, literal

- **Un término** puede ser
  - Un **objeto**

    Ej. Pedro, P, P1,…
  - Una **variable**

    Ej. x,y
  - Una **función** de aridad n **seguida de n términos** separados por comas y entre paréntesis.

    Ej. padreDe(x), distanciaEntre(A,B)
  - Un caso especial es un **término sin variables** (“término base”, “ground term”).

- Una **fórmula atómica (átomo)** es una **relación** de aridad n **seguida de n términos** separados por comas y entre paréntesis.

  Ej. PadreDe(x,Pedro)

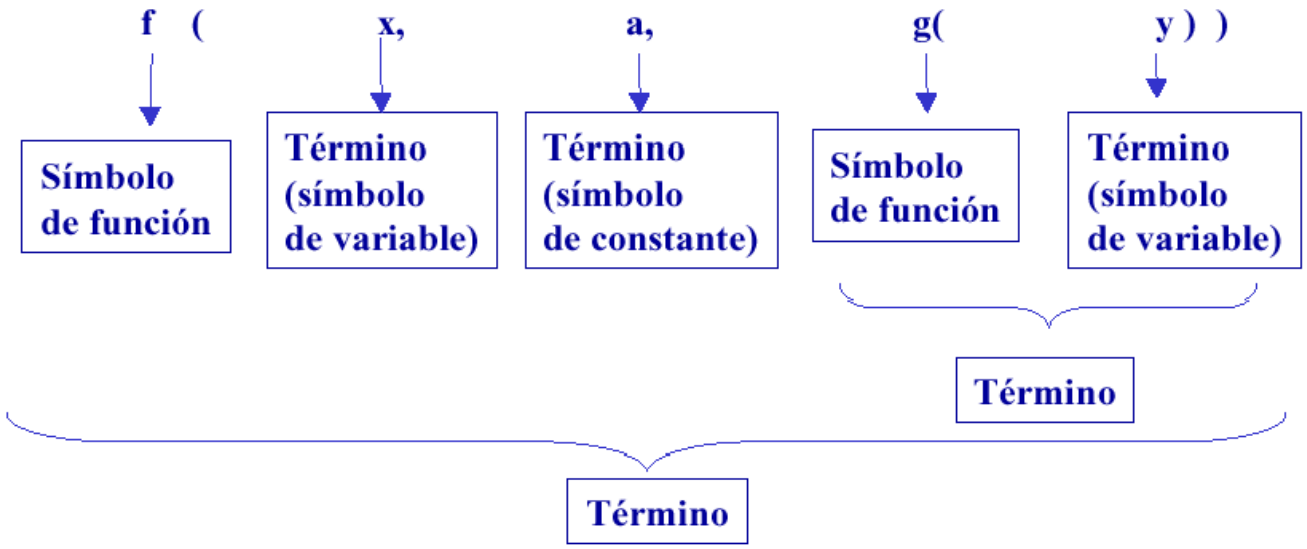
- **Un literal** puede ser:
  - Un **literal positivo**: Una fórmula atómica

    Ej. HermanoDe(x,Pedro)
  - Un **literal negativo**: La negación de una fórmula atómica

    Ej. ¬ PadreDe(Juan,padreDe(Pedro))

Lógica de predicados: sintaxis

- Términos
  - Un símbolo de constante es un *(a, b, c...)*
  - Un símbolo de variable es un término *(x, y, z...)*
  - Si *f* es un símbolo de función (o functor) de aridad *n*, y $t_1, t_2, \ldots, t_n$ son términos, entonces $f(t_1, t_2, \ldots, t_n)$ es un término compuesto
- Ejemplo: *f (x, a, g(y))*

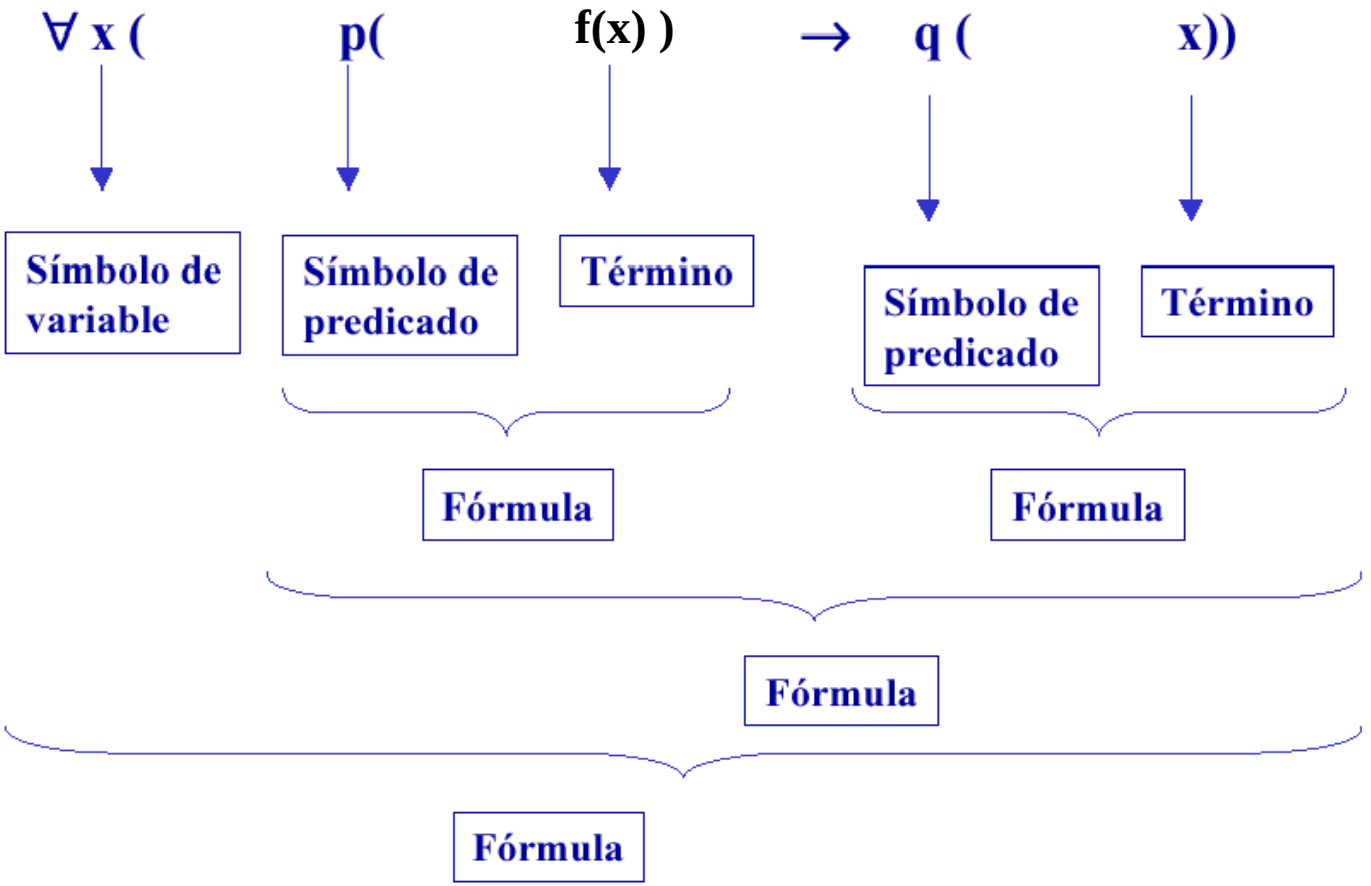


Lógica de predicados: sintaxis

- Fórmulas
  - ATÓMICAS
    - Los símbolos de verdad $\top$ y el de falsedad o contradicción $\bot$ son fórmulas
    - Si p es un símbolo de predicado de aridad n, y $t1, t2, ..., tn$ son términos, entonces $p(t1, t2, ..., tn)$ es una fórmula
  - COMPUESTAS
    - Si F es una fórmula, entonces $\neg F$ es una fórmula
    - Si F y G son fórmulas, entonces:
      - $(F \wedge G)$ es una fórmula
      - $(F \vee G)$ es una fórmula
      - $(F \rightarrow G)$ es una fórmula
      - $(F \leftrightarrow G)$ es una fórmula
    - Si x es un símbolo de variable, y F es una fórmula, entonces:
      - $\forall x\, F$ es una fórmula
      - $\exists x\, F$ es una fórmula

Lógica de predicados: sintaxis

- Ejemplo de fórmula: $\forall x\,(p(f(x)) \rightarrow q(x))$



Interpretación y semántica

- **Constantes**
  - Las **constantes son objetos simbólicos** que corresponden a **objetos del mundo real**
  - Los **predicados de aridad 1** corresponden a **propiedades del objeto**
  - Los **predicados de aridad n > 1** expresan **relaciones entre objetos**

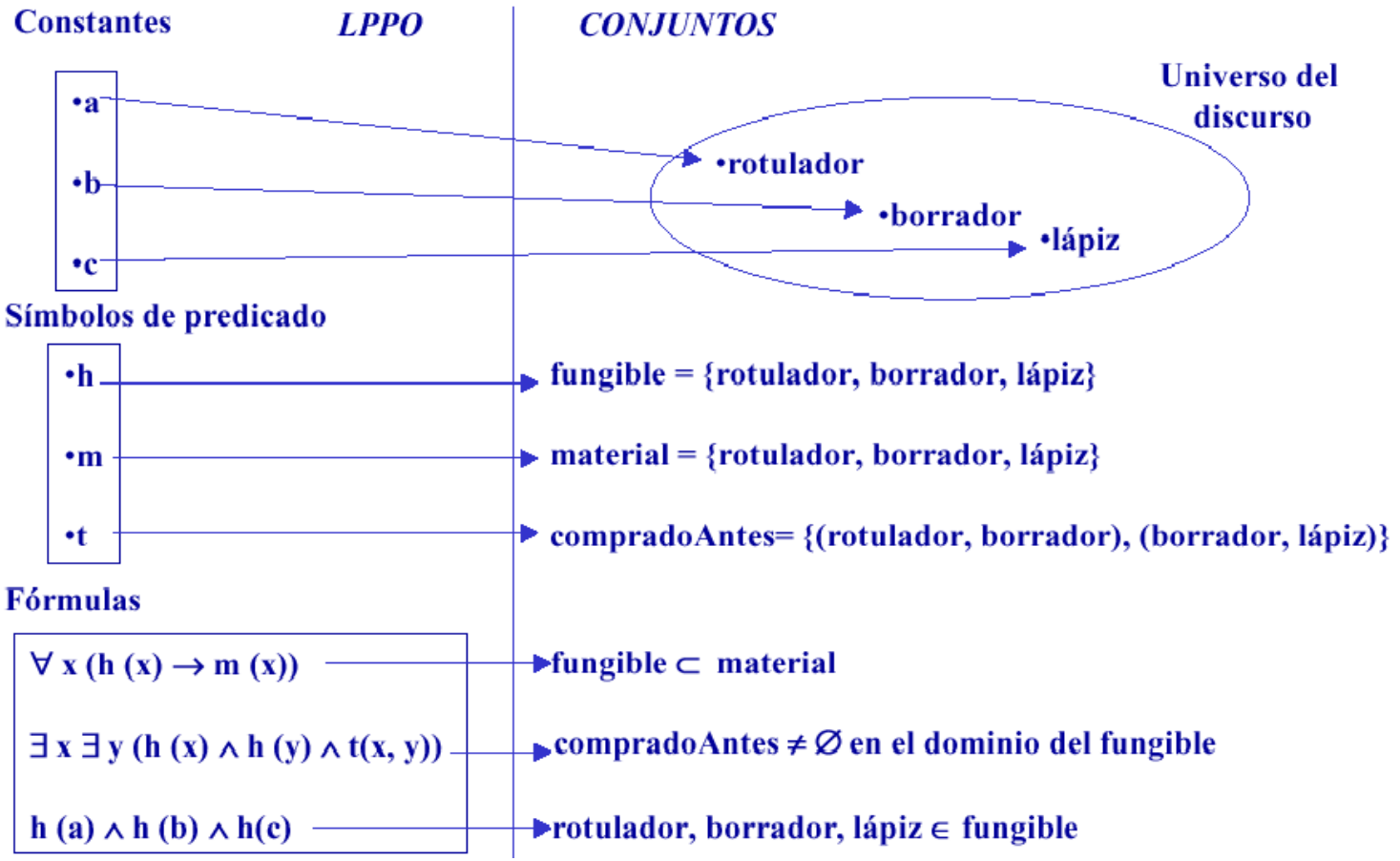
- **Variables**
  - El **dominio** de una variable es el conjunto de objetos simbólicos entre los que puede tomar valor dicha variable
  - **Asignación**: Operación mediante la cual una variable que aparece en una fbf es reemplazada por un valor particular dentro de su dominio.

- **Cuantificadores**
  - **Cuantificador universal**: `(∀x) w(x)` tiene valor `Verdadero` si `w(x)` tiene valor `Verdadero` para todas las posibles asignaciones de la variable `x`
  - **Cuantificador existencial**: `(∃x) w(x) tiene` valor `Verdadero` si `w(x)` tiene valor `Verdadero` para alguna de las posibles asignaciones de `x`

Lógica de predicados: semántica

- Los significados de términos y fórmulas se obtienen fijando una interpretación *I* que consta de
  - Un conjunto llamado *U* universo (o dominio) de discurso
  - Una asociación de elementos de *U* a los símbolos de constante
  - Una asociación de relaciones en *U* a los símbolos de predicado
  - Una asociación de funciones en *U* a los símbolos de función

Lógica de predicados: semántica



Variables libres y ligadas

- La aparición de una variable X dentro de una formula φ se dice ligada si está afectada por un cuantificador, o con más precisión: X aparece ligada en φ si está dentro de una subfórmula de φ de la forma $\forall\psi$ o $\exists\psi$.
- La aparición de X en φ es libre si no está ligada, i.e., si no está afectada por ningún cuantificador.

$$\exists X\ p(\underbrace{X}_{\text{ligada}}, f(\underbrace{Y}_{\text{libre}})) \vee \forall Y\ q(\underbrace{Y}_{\text{ligada}}, g(\underbrace{X}_{\text{libre}}))$$

- Una variable X está libre en φ si todas las apariciones de X en φ son libres.
- Una variable X está ligada en φ si alguna de las apariciones de X en φ es ligada.

Variables libres y ligadas

- ¿Puede una misma variable aparecer ligada y libre en la misma fórmula?

$$\varphi \equiv \forall X\ r(\underbrace{X}_{\text{ligada}}, \underbrace{Y}_{\text{libre}}) \wedge \forall Y\ p(\underbrace{X}_{\text{libre}}, \underbrace{Y}_{\text{ligada}}, Z)$$

- Para evitar confusiones, renombramiento:

$$\varphi' \equiv \forall X'\ r(X', Y) \wedge \forall Y'\ p(X, Y', Z)$$

- ¿Son iguales φ y φ'?

- "Iguales" no son… parece son equivalentes: tienen el mismo significado, pero para verlo tendríamos que dar semántica o significado a las fórmulas.
- Nos interesará especialmente la semántica de las sentencias: fórmulas cerradas, i.e., sin variables libres.

Lógica de predicados: propiedades

- Representación declarativa
  - No está fijada la forma en la que debe ser usado el conocimiento
    - Sencillez para incorporar nuevo conocimiento, o eliminarlo
    - Sencillez de integración de dos o más BCs
- Corrección y completitud
  - Cuenta con mecanismos deductivos correctos y completos
    - Permiten deducir nuevo conocimiento (conclusiones) a partir del conocimiento de partida (premisas)
    - Pueden usarse para responder a preguntas o resolver problemas
      - Demostración automática de teoremas: una de las áreas iniciales de la IA
- La deducción es semi-decidible en LPO
  - Si lo que pretendemos demostrar no se deduce del contenido de la base de conocimiento, no está garantizado que termine el proceso de demostración
    - No existe un procedimiento de decisión, ni siquiera de coste exponencial

Dificultades en la representación

- Lenguaje natural → lógica de predicados
  - Ambigüedad, sentido común, implicaciones y disyunciones…
- Incompletitud del conocimiento representado
  - Es imposible representar TODO el conocimiento
  - Representamos sólo lo más importante
  - No incluimos conocimiento de "sentido común"
    → habrá razonamientos que no podrán hacerse
- Infinidad de alternativas de representación dentro de la lógica
  - Influyen en el proceso de razonamiento. Algunas representaciones facilitan un tipo de razonamiento y dificultan otro
  - Lo importante es usar una representación consistente dentro de la misma BC
- Dificultades en la representación del conocimiento por defecto
  - Por ejemplo, las excepciones a la herencia

Alternativas de representación

- Ejemplo
  - Representación de conocimiento factual terminológico
    - Relaciones de ejemplar ($\in$) y subclase ($\subseteq$)
    - Propiedades esenciales
- Ventajas e inconvenientes de distintas alternativas
  - "Flipper es un delfín" — *ejemplar de una clase*
  - "Todos los delfines son vertebrados" — *relación de subclase*
  - "Los vertebrados tienen esqueleto" — *propiedad esencial*
- Criterios para la elección
  - Qué pueden hacer eficientemente
    - Representar lo que tenemos
    - Obtener lo que buscamos (deducciones, inferencias)
  - Los gustos de cada cual
  - La representación sea consistente en todo el dominio

Alternativas de representación

- Versión 1
  - Representación implícita de ejemplares: el nombre de la clase es un símbolo de predicado unario, el argumento es el ejemplar

    Delfín(flipper)
  - Representación de la subclase *delfín* de la clase *vertebrados*

    ∀x (Delfín(x) → Vertebrado(x))
  - Representación de propiedades esenciales

    ∀x (Vertebrado(x) → Tiene-Esqueleto(x))
  - Ventajas
    - Sencillez de la representación
  - Desventajas
    - Para cualquier clase (Delfín, Elefante…)
      - crear predicados y reglas (subclases) específicas
    - Se complica el razonamiento general
    - El conocimiento implícito se deduce a través de las implicaciones.
      - Es mejor afirmar las cosas explícitamente por medio de hechos: VERSION 2

Alternativas de representación

- Versión 2
  - Representación explícita de la pertenencia de ejemplares a clases: se utiliza un símbolo de predicado binario Es_Un(ejemplar, Clase)

    Es_Un(flipper, Delfín)
  - Representación de la subclase *delfín* de la clase *vertebrados*

    ∀x (Es_Un(x, Delfín) → Es_Un(x, Vertebrado))
  - Representación de propiedades esenciales

    ∀x (Es_Un(x, Vertebrado) → Tiene_Esqueleto(x))

  - Ventajas
    - Representación explícita de pertenencia de ejemplares a clases, es_un
  - Desventajas
    - La relación de subclase sigue siendo implícita

Alternativas de representación

- Versión 3
  - Representación explícita de ejemplares y subclases utilizando pred. Es_Un

    Es_Un(flipper, Delfín)

    Es_Un(Delfín, Vertebrado)

    $\forall$x (Es_Un(x, Vertebrado) $\rightarrow$ Tiene_Esqueleto(x))

  - Ventajas
    - Representación explícita de pertenencia de ejemplares a clases
    - Representación explícita de la relación subclase
  - Desventajas
    - El sistema no puede diferenciar si *flipper* es individuo o clase
    - No puede deducirse que *Flipper* tenga esqueleto
    - Falta especificar la transitividad de la relación *Es_Un*. Hay que añadirla

      $\forall$x$\forall$y$\forall$z (Es_Un(x, y) $\wedge$ Es_Un(y, z) $\rightarrow$ Es_Un (x, z))

Alternativas de representación

- Versión 4
  - Representación explícita de ejemplares y subclases utilizando dos símbolos de predicado binarios distintos

    Ejemplar(flipper, Delfín)

    Subclase(Delfín, Vertebrado)

    $\forall x$ (Ejemplar(x, Vertebrado) $\rightarrow$ Tiene_Esqueleto(x))

  - Añadimos la transitividad (entre subclases y ejemplares)

    $\forall x \forall y \forall z$ (Ejemplar(x,y) $\wedge$ Subclase(y,z) $\rightarrow$ Ejemplar(x,z))

    $\forall x \forall y \forall z$ (Subclase(x,y) $\wedge$ Subclase(y,z) $\rightarrow$ Subclase(x,z))

Alternativas de representación

- Las cuatro versiones usan como técnica de representación a la lógica de predicados o lógica de primer orden (LPO)
  - Existen distintas posibilidades de representación dentro de la lógica (en general, esto ocurre con cualquier formalismo)
  - Algunas representaciones facilitan un tipo de razonamiento y dificultan otro

La lógica de predicados nos permite expresar conocimiento (i)

- Hemos visto como la lógica de predicados permite expresar "sentencias" del lenguaje natural:
  - Ej: "Algunos mamíferos leen a Quevedo"

Definimos SP = {mamifero^1, leerQuevedo^1}, SF = Ø;
con el significado "intuitivo":
- mamifero(X) ::= X es un mamífero
- leerQuevedo(X) ::= X lee a Quevedo

Y formalizamos:

$\exists X, (\text{mamifero}(X) \wedge \text{leerQuevedo}(X))$

  - O de otro modo:

Definimos SP = {mamifero^1, leer^2}, SF = {a};
con el significado "intuitivo":
- mamifero(X) ::= X es un mamífero
- leer (X,Y) ::= X lee a Y
- a ::= Quevedo

Y formalizamos:

$\exists X, (\text{mamifero}(X) \wedge \text{leer}(X, a))$

La lógica de predicados nos permite expresar conocimiento (II)

- Otro ejemplo: "El producto de cualquier número n por 1 es n"
  (Esta es una sentencia "matemática", pero está expresada en lenguaje natural).

Definimos SP = $\{=^2\}$, SF = $\{1^0, *^2\}$

con el significado intuitivo:

- = ::= predicado binario de igualdad (ser iguales)
- 1 ::= el "uno" de matemáticas
- * ::= el "producto" en matemáticas

Y formalizamos

$$\forall X, X * 1 = X$$

En este ejemplo se pone de manifiesto que las matemáticas están "empapadas" en la lógica (la lógica puede expresar propiedades matemáticas… incluso puede expresar cosas acerca de la lógica misma!).

Formas normales en lógica de predicados

- Para manipular predicados de manera automática (en un ordenador) nos interesa que éstos tengan un formato uniforme.
- Interesa además que este formato sea lo más simple posible
- La justificación de fondo es que necesitamos tenerlos en **forma normal** con el fin de poder utilizar el mecanismo de resolución (que veremos más adelante).
  - En particular, la forma de cláusula sólo utiliza las conectivas ¬ y $\vee$ (negación y disyunción), y necesita una sola regla de deducción
- Con esta restricción no se pierde generalidad porque existe un algoritmo de conversión a forma normal conjuntiva que permite plantear cualquier problema lógico en forma de cláusulas

Fórmulas bien formadas

- Una **fbf** se construye usando átomos y conectores del mismo modo que en lógica proposicional.
- Si w es una fbf y x es una variable, las siguientes expresiones son también fbfs
  - $(\forall x)\ w$
  - $(\exists x)\ w$

  En general (pero no siempre), la variable x forma parte de w.

  Si x forma parte de w, se suele escribir $w(x)$

- **Fbf cerrada**: Aquella en la que todas las variables están cuantificadas.
  - $(\forall x)\ [P(x) \Rightarrow R(x)]$
  - $(\exists x)\ [P(x) \Rightarrow (\exists y)(R(x,y) \Rightarrow S(f(x))]$

Nota: El **orden** en el que aparecen $\forall$, $\exists$ en la fbf es **importante**

$x,y \in \{Personas\}$

$PadreDe(x,y)$ "**x es el padre de y**" (relación binaria)

$(\forall x)(\forall y)PadreDe(x,y)$	"Todo el mundo es padre de todo el mundo"
$(\exists x)(\exists y)PadreDe(x,y)$	"Existe alguien que tiene al menos un padre"
$(\forall x)(\exists y)PadreDe(x,y)$	"Todo el mundo es padre de alguien"
$(\exists y)(\forall x)PadreDe(x,y)$	"Hay alguien que tiene a todo el mundo como padre"
$(\exists x)(\forall y)PadreDe(x,y)$	"Hay alguien que es padre de todo el mundo"
$(\forall y)(\exists x)PadreDe(x,y)$	"Todo el mundo tiene al menos un padre"

Reglas de equivalencia y reglas de inferencia

- **Reglas de equivalencia**
  - **Reglas de equivalencia de lógica proposicional**
  - **Renombramiento de variables cuantificadas.**

    El nuevo nombre debe ser diferente que el de las otras variables en la fbf
    - $(\forall x)\ w(x) \equiv (\forall y)\ w(y)$
    - $(\exists x)\ w(x) \equiv (\exists y)\ w(y)$
  - $\neg(\forall x)\ w(x) \equiv (\exists x)\ \neg w(x)$
  - $\neg(\exists x)\ w(x) \equiv (\forall x)\ \neg w(x)$

- **Reglas de inferencia**
  - **Reglas de inferencia de lógica proposicional**
  - **Instanciación del universal** (IU) [Correcta]

    $(\forall x)\ w(x) \vdash_{IU} w(A)$, donde A es alguno de los valores en el dominio de x
  - **Generalización del existencial** (GE) [Correcta]

    $w(A) \vdash_{GE} (\exists x)\ w(x)$, donde x es el símbolo de una variable cuyo dominio incluye a A

Recordamos reglas de Equivalencia

Dos **FBFs distintas** w_1, w_2 son **equivalentes** ($w_1 \equiv w_2$) cuando tienen la **misma tabla de verdad**

- **Elemento neutro:** $(w_1 \wedge V) \equiv w_1$; $(w_1 \vee F) \equiv w_1$
- **Leyes de absorción:** $(w_1 \vee (w_1 \wedge w_2)) \equiv w_1$
  $(w_1 \wedge (w_1 \vee w_2)) \equiv w_1$
- **Ley de contradicción / ley del medio excluido:**
  $(w_1 \wedge \neg w_1) \equiv F$; $(w_1 \vee \neg w_1) \equiv V$
- **Leyes de dominación:**
  $(w_1 \wedge F) \equiv F$; $(w_1 \vee V) \equiv V$
- **Idempotencia:** $(w_1 \wedge w_1) \equiv w_1$; $(w_1 \vee w_1) \equiv w_1$
- **Eliminación de la doble negación:** $\neg\neg w_1 \equiv w_1$
- **Leyes de De Morgan:**
  $\neg(w_1 \vee w_2) \equiv \neg w_1 \wedge \neg w_2$; $\neg(w_1 \wedge w_2) \equiv \neg w_1 \vee \neg w_2$
- **Conmutatividad:** $w_1 \vee w_2 \equiv w_2 \vee w_1$; $w_1 \wedge w_2 \equiv w_2 \wedge w_1$
- **Leyes asociativas:**
  $(w_1 \wedge w_2) \wedge w_3 \equiv w_1 \wedge (w_2 \wedge w_3) \equiv w_1 \wedge w_2 \wedge w_3$ [conjunción]
  $(w_1 \vee w_2) \vee w_3 \equiv w_1 \vee (w_2 \vee w_3) \equiv w_1 \vee w_2 \vee w_3$ [disyunción]
- **Leyes distributivas:**
  $w_1 \wedge (w_2 \vee w_3) \equiv (w_1 \wedge w_2) \vee (w_1 \wedge w_3)$
  $w_1 \vee (w_2 \wedge w_3) \equiv (w_1 \vee w_2) \wedge (w_1 \vee w_3)$.
- **Definición de condicional:** $w_1 \Rightarrow w_2 \equiv \neg w_1 \vee w_2$
- **Contraposición:** $w_1 \Rightarrow w_2 \equiv \neg w_2 \Rightarrow \neg w_1$
- **Def. de bicondicional:** $w_1 \Leftrightarrow w_2 \equiv (w_1 \Rightarrow w_2) \wedge (w_2 \Rightarrow w_1)$
  $\equiv (w_1 \wedge w_2) \vee (\neg w_1 \wedge \neg w_2)$

Skolemización

Un **cuantificador existencial** (∃) puede ser **eliminado** de una fbf **reemplazando** cada ocurrencia de la **variable cuantificada existencialmente** por una constante o bien una función.

- Un **objeto de Skolem (constante)**, si no hay variables cuantificadas universalmente cuyo ámbito abarque el ámbito de la variable cuyo ∃ se está eliminando.

  Ej. (∃x) w(x)

  forma de Skolem: w(Sk)

  Sk es un objeto cuya identidad desconocemos, pero que sabemos que existe

- Una **función de Skolem** cuyos argumentos son las variables cuantificadas de manera universal cuyo ámbito abarque el ámbito de la variable cuyo ∃ se está eliminando.

  Ej. "Todas las personas tienen una altura"

  (∀p)[(∃h) Altura(p,h)]

  dominio de p: Personas.

  dominio de h: Reales positivos.

  Forma de Skolem: (∀p) Altura(p,a(p))

  a(p) es una función de Skolem (desconocida, pero que sabemos que existe) que toma como argumento una persona y devuelve su altura

Metateoremas para formas de Skolem

- Metateorema SK1: La forma de Skolem de una fbf NO es equivalente a la fbf original

$$w(Sk) \models (\exists x)\ w(x) \quad \text{PERO} \quad (\exists x)\ w(x) \not\models w(Sk)$$

Ej.

$$P(A) \vee P(B) \models (\exists x)\ P(x)$$

$$\text{PERO} \quad P(A) \vee P(B) \not\models P(Sk)$$

- Metateorema SK2 (Loveland, 1978):

La forma de Skolem de un conjunto de fbfs es **equivalente inferencialmente** al conjunto original de fbfs.

Esto es, la forma de Skolem de un conjunto de fbfs es satisfacible exactamente en aquellos casos donde la base de conocimiento original es satisfacible

- Un conjunto de fbfs es satisfacible si su forma de Skolem es satisfacible.
- Un conjunto de fbfs es insatisfacible si su forma de Skolem es insatisfacible

Recordamos: modelos y satisfactibilidad

Dada una interpretación $\mathrm{I} = (\mathbb{A}, v)$, si $[[\varphi]]^{\mathbb{A}}v$ = *cierto* se dice:

- A satisface φ en el estado v o que I satisface φ , o bien
- I es un modelo de φ y se nota como:

$$\mathbf{A} \models \varphi$$

(En el caso de que $[[\varphi]]^{\mathbb{A}}v$ = *falso* se dice que I no satisface φ y se denota como $\mathbb{A} \not\models \varphi$)

- Una fórmula es satisfactible si admite un modelo, e insatisfactible en otro caso.
- Dos fórmulas φ y ψ son lógicamente equivalentes (y se escribe $\varphi \approx \psi$) si $[[\varphi]]^{\mathbb{A}}v = [[\psi]]^{\mathbb{A}}v$ para toda interpretation $\mathrm{I} = (\mathbb{A}, v)$,
- φ es lógicamente valida si $\mathrm{I} \models \varphi$ para toda interpretación I

Recordamos: Consecuencia lógica

- Dadas las fórmulas $\varphi_1, \ldots, \varphi_n$, $\psi \in L_\Sigma$ se dice que ψ es una consecuencia lógica de $\{\varphi_1, \ldots, \varphi_n\}$ y se escribe:

$$\underbrace{\{\varphi_1, \ldots, \varphi_n\}}_{\text{Premisas/hipótesis}} \models \underbrace{\psi}_{\text{Conclusión}}$$

si para toda interpretación $I = (\mathbb{A}, v)$ se tiene:

$$I \models \varphi_1, \ldots \; I \models \varphi_n \Rightarrow I \models \psi$$

FNC en lógica de primer orden

1. **Eliminar implicaciones** $\Leftrightarrow, \Rightarrow$: $w_1 \Rightarrow w_2 \equiv \neg w_1 \vee w_2$; $w_1 \Leftrightarrow w_2 \equiv (\neg w_1 \vee w_2) \wedge (\neg w_2 \vee w_1)$
2. **Reducir el ámbito de la negación** $\neg$
   - Leyes de De Morgan
     $\neg(w1 \vee w2) \equiv \neg w1 \wedge \neg w2$
     $\neg(w1 \wedge w2) \equiv \neg w1 \vee \neg w2$
   - Eliminación de negaciones dobles $(\neg\neg w \equiv w)$
   - Combinación de $\neg$ con cuantificadores
     $\neg(\forall x)\ w(x) \equiv (\exists x)\ \neg w(x)$
     $\neg(\exists x)\ w(x) \equiv (\forall x)\ \neg w(x)$
3. **Estandarizar las variables**: Renombrar las variables de tal forma que variables distintas tengan símbolos (nombres) diferentes

   $[(\forall x)\ [P(x) \Rightarrow R(x)]] \vee [(\exists x)\ P(x)] \equiv [(\forall x)\ [P(x) \Rightarrow R(x)]] \vee [(\exists y)\ P(y)]$
4. **Skolemización**: Eliminar los cuantificadores existenciales reemplazando las variables correspondientes por constantes de Skolem o funciones de Skolem.
5. **Convertir a forma prenexa** desplazando todos los cuantificadores universales al principio de la fbf

   fbf en **forma prenexa** = **Prefijo** (lista de cuantificadores)
   + **Matriz** (fórmula sin cuantificadores)
6. **Eliminar los cuantificadores universales**
7. **Usar leyes distributivas y reglas de equivalencia** de lógica proposicional para transformar la **matriz a FNC** (convertirla en una conjunción de disyunciones *(forma normal conjuntiva)*)

Ejemplo 1: Conversión a FNC

$[(\forall x)\ Q(x)] \Rightarrow$
$(\forall x)(\forall y)\ [(\exists z)\ [P(x,y,z) \Rightarrow (\forall u)\ R(x,y,u,z)]$

1. Eliminación de implicaciones $\Leftrightarrow, \Rightarrow$

$\neg[(\forall x)\ Q(x)] \vee$
$(\forall x)(\forall y)\ [(\exists z)\ [\neg P(x,y,z) \vee (\forall u)\ R(x,y,u,z)]$

2. Reducción del ámbito de las negaciones:

$[(\exists x)\ \neg Q(x)] \vee$
$(\forall x)(\forall y)\ [(\exists z)\ [\neg P(x,y,z) \vee (\forall u)\ R(x,y,u,z)]$

3. Estandarización de variables

$[(\exists w)\ \neg Q(w)] \vee$
$(\forall x)(\forall y)\ [(\exists z)\ [\neg P(x,y,z) \vee (\forall u)\ R(x,y,u,z)]$

4. Skolemización:

$\neg Q(A) \vee (\forall x,y)[\neg P(x,y,f(x,y)) \vee (\forall u) R(x,y,u,f(x,y))]$

5. Forma prenexa:

$(\forall x,y,u)[\neg Q(A) \vee \neg P(x,y,f(x,y)) \vee R(x,y,u,f(x,y))]$

6. Eliminación de los cuantificadores universales

$\neg Q(A) \vee \neg P(x,y,f(x,y)) \vee R(x,y,u,f(x,y))$

7. Conversión a FNC: la fbf ya está en FNC

Ejemplo 2: Conversión a FNC

"Todo el que ama a todos los animales es amado por alguien"

$(\forall x)[(\forall y)\{Animal(y) \Rightarrow AmaA(x,y)\} \Rightarrow (\exists y)\ AmaA(y,x)]$

1. Eliminación de implicaciones $\Leftrightarrow, \Rightarrow$

$(\forall x)[\neg(\forall y)\{\neg Animal(y) \vee AmaA(x,y)\} \vee (\exists y)\ AmaA(y,x)]$

2. Reducción del ámbito de las negaciones:

$(\forall x)[(\exists y)\{Animal(y) \wedge \neg AmaA(x,y)\} \vee (\exists y)\ AmaA(y,x)]$

3. Estandarización de variables

$(\forall x)[(\exists y)\{Animal(y) \wedge \neg AmaA(x,y)\} \vee (\exists z)\ AmaA(z,x)]$

4. Skolemización:

$(\forall x)[\{Animal(f(x)) \wedge \neg AmaA(x,f(x))\} \vee AmaA(g(x),x)]$

5. Forma prenexa: la fbf ya está en forma prenexa

6. Eliminación de cuantificadores universales

$\{Animal(f(x)) \wedge \neg AmaA(x,f(x))\} \vee AmaA(g(x),x)$

7. Conversión a FNC usando leyes distributivas y reglas de equivalencia

$\{Animal(f(x)) \vee AmaA(g(x),x)\} \wedge \{\neg AmaA(x,f(x)) \vee AmaA(g(x),x)\}$

Recordamos: Reglas de inferencia

- **Reglas de inferencia:** Reglas tipográficas que manipulan únicamente símbolos (y, que, por lo tanto, no utilizan el significado de estos símbolos, ni las tablas de verdad) y que nos permiten generar nuevas FBFs a partir de un conjunto de FBFs dado.
- **Reglas de inferencia correctas:** Reglas de inferencia en las que los modelos de conjunto de FBFs de partida son también modelos de las FBFs generadas.
- Las **reglas de equivalencia** son **reglas de inferencia correctas.** (Nota: no toda regla de inferencia lo es de equivalencia)

Conjunto de reglas de inferencia :

Sean w_1, w_2 dos FBFs

(1) *Modus ponens* : $\{w_1, w_1 \Rightarrow w_2\} \vdash_{M.P.} w_2$

(2) *Modus tollens*: $\{\neg w_2, w_1 \Rightarrow w_2\} \vdash_{M.T.} \neg w_1$

(3) Introducción de $\wedge$: $\{w_1, w_2\} \vdash_{\wedge\ INTRO} w_1 \wedge w_2$

(4) Conmutatividad de $\wedge$: $\{w_1 \wedge w_2\} \vdash_{\wedge\ CONMUTA} w_2 \wedge w_1$

(5) Eliminación de $\wedge$: $\{w_1 \wedge w_2\} \vdash_{\wedge\ ELIMIN} w_1$

(6) Introducción de $\vee$: $\{w_1\} \vdash_{\vee\ INTRO} w_1 \vee w_2$

$\{w_2\} \vdash_{\vee\ INTRO} w_1 \vee w_2$

(7) Eliminación de $\neg\neg$: $\{\neg\neg w_1\} \vdash_{\neg\neg ELIMIN} w_1$

Este conjunto de reglas de inferencia es correcto, pero no completo.

Mecanismos de inferencia

- Deducción: obtención de nuevo conocimiento *(implícito)*
- Se trata de saber si una fórmula Q es cierta conociendo
  - Los axiomas que son lógicamente válidos sea cual sea el significado de los símbolos *(tautologías)*

    $\neg F \vee F$
  - Los axiomas que son válidos sólo suponiendo ciertos significados de los símbolos *(conocimiento explícito)*
  - Las reglas de inferencia
    - Por ejemplo:

modus ponens

$$\frac{P \rightarrow Q \qquad P}{Q}$$

modus tolens

$$\frac{P \rightarrow Q \qquad \neg Q}{\neg P}$$

Ejemplo de deducción formal

- Dado un conjunto de hipótesis o premisas

  perro(milú)

  ∀x (perro(x) → animal(x))

  ∀y (animal(y) → mortal(y))

- Demostrar una conclusión

  mortal(milú)

- Pasos aplicados

  1. Aplicar instanciación universal con *x* = *milú*

     perro(milú) → animal(milú)

  2. Aplicar modus ponens

     animal(milú)

  3. Aplicar instanciación universal con *y* = *milú*

     animal(milú) → mortal(milú)

  4. Aplicar modus ponens

     mortal(milú)

Deducción por refutación

Base de conocimientos: $Q \leftarrow T$, $T \leftarrow S$, $S \leftarrow P$, P
Consulta: Q?

Paso 1.	Paso 2.	Paso 3.	Paso 4.
$Q \leftarrow T$	$T \leftarrow S$	$S \leftarrow P$	P
$\neg Q$	$\neg T$	$\neg S$	$\neg P$
---	---	---	---
$\neg T$	$\neg S$	$\neg P$	•

Paso 1. Se niega lo que se pretende demostrar

Pasos 1, 2 y 3. Aplicación reiterada de modus tolens

Paso 4. P $P\sim$ se llega a contradicción

Refutación: razonamiento por reducción al absurdo

$$\Phi \models \psi \Leftrightarrow \text{Insat } \Phi \cup \{\neg\psi\}$$

Deducción por resolución

Ejemplo: ¿a?

$b \wedge c \rightarrow a$

$d \rightarrow c$

b

d

Forma clausal

$a \vee \neg b \vee \neg c$

$c \vee \neg d$

b

d

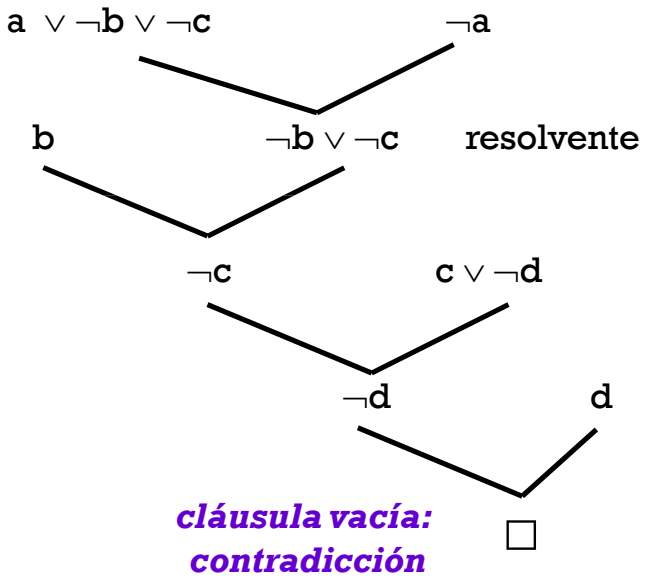
Añadir la negación de lo que queremos demostrar: $\neg a$
y probar que llegamos a una contradicción *(conjunto insatisfactible)*

Resolución

$\varphi \vee Q$ ***un literal positivo en una cláusula***

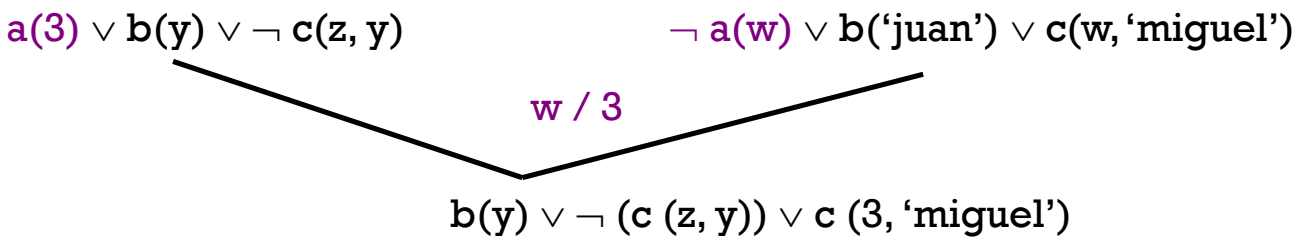
$\neg Q \vee \psi$ ***y el mismo negativo en otra***

$\varphi \vee \psi$ ***resolvente***

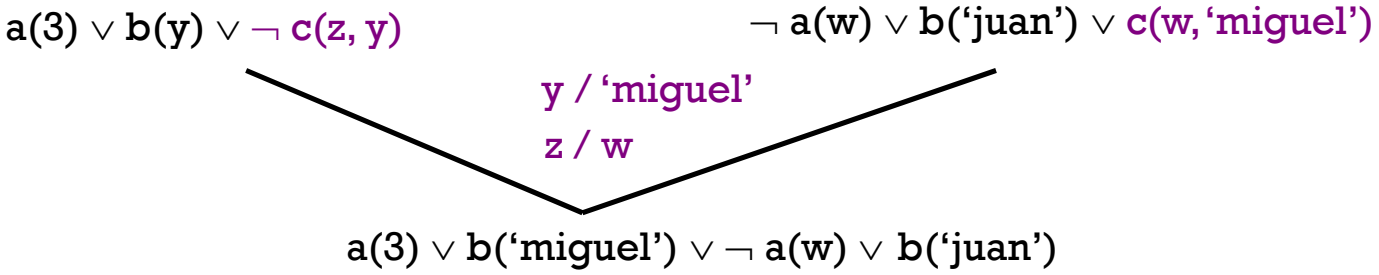


Proposiciones → Predicados (Resolución con unificación)

Resolución con unificación



U.m.g.: unificador más general posible de los literales candidatos Se aplica a las cláusulas padres antes de calcular el resolvente Deben considerarse variantes de las cláusulas (variables frescas)



Se selecciona un único par de literales complementarios

Estrategias de resolución

- No es fácil determinar en cada paso qué cláusulas resolver y con qué literales *(grado de no determinismo muy alto)*
  - Si la elección se realiza de forma sistemática, la resolución llegará a contradicción si el conjunto de cláusulas es insatisfactible
  - Así es completa, pero puede requerir mucho tiempo…
- Se utilizan diversas estrategias para acelerar el proceso
  - Indexar las cláusulas por los predicados que contienen, indicando si están o no negados, para facilitar su localización
  - Eliminar tautologías *($\varphi \vee \neg\varphi$)* y cláusulas que son *subsumidas* por otras (son implicadas por otras: $\varphi \vee \psi$ es subsumida por φ)
  - Intentar resolver con una de las cláusulas de la fórmula que estamos intentando refutar, o con alguna cláusula que se haya obtenido por resolución a partir de una de ellas *(intuición: la contradicción tiene que salir de ahí)*
  - Resolver con cláusulas que tengan un solo literal *(idea: disminuye el tamaño de las cláusulas generadas)*

Contestar a preguntas

- ¿Tiene esqueleto *flipper*?
  - Hay dos opciones para saber si algo se deduce o no de la BC

  1) Demostrar **Tiene_esqueleto(flipper)**

     ➔ añadir **¬Tiene_esqueleto(flipper)**

  2) Demostrar **¬Tiene_esqueleto(flipper)**

     ➔ añadir **Tiene_esqueleto(flipper)**

  - La opción correcta depende de cómo son los predicados de la BC

    Ejemplar(flipper, Delfín)

    Subclase(Delfín, Vertebrado)

    ∀x (Ejemplar(x, Vertebrado) → Tiene_Esqueleto(x))

    ∀x∀y∀z (Ejemplar(x,y) ∧ Subclase(y,z) → Ejemplar(x,z))

    ∀x∀y∀z (Subclase(x,y) ∧ Subclase(y,z) → Subclase(x,z))

Contestar a preguntas

- ¿Existe algún delfín?
  - Demostrar **∃x Ejemplar(x, Delfín)**
    - ➔ añadir **¬∃x Ejemplar(x, Delfín)** en forma clausal
    - ➔ **~ ∀x ¬Ejemplar(x, Delfín)** en forma clausal
    - ➔ añadir **¬Ejemplar(t, Delfín)** variante
  - Cómo preguntar depende de cómo son los predicados de la BC

    Ejemplar(flipper, Delfín)

    Subclase(Delfín,Vertebrado)

    ∀x (Ejemplar(x, Vertebrado) → Tiene_Esqueleto(x))

    ∀x∀y∀z (Ejemplar(x,y) ∧ Subclase(y,z) → Ejemplar(x,z))

    ∀x∀y∀z (Subclase(x,y) ∧ Subclase(y,z) → Subclase(x,z))
  - Si se requiere usar unificación, la resolución devolverá los valores que hacen cierta la pregunta: **{t = flipper}**

De la lógica a la programación lógica

- Unificación
- Resolución

- Sabemos:
  - Cómo representar conocimiento mediante argumentaciones y formalizar estas argumentaciones en el lenguaje de la lógica de predicados.
  - Cómo transformar las fórmulas lógicas en cláusulas.
  - Demostrar validez de una argumentación es refutar (demostrar la insatisfactibilidad) de un conjunto de fórmulas:
    - premisas + negación de la conclusión (… quizá sepamos algo de tablas de verdad?)
- Queremos saber:
  - Un método para construir refutaciones utilizando la forma clausal.
  - Una forma de plasmar ese método en un algoritmo concreto. Muchas más cosas… … pero, comencemos con la unificación

Sustitución

- Considérese la expresión w que contiene las variables $v_1, v_2, v_3, \ldots, v_n$.

  La sustitución σ es **un reemplazamiento simultáneo** de variables en w por términos.

$$w = w(v_1, v_2, v_3, \ldots, v_n)$$
$$\sigma = \{v_1 := t_1, v_2 := t_2, \ldots, v_n := t_n\}$$
$$w\sigma = w(t_1, t_2, t_3, \ldots, t_n)$$

El término t_i no puede contener a la variable v_i.
El término t_i es una instanciación de la variable v_i.
$w\sigma$ es una **instancia** de w.

- Considérese las siguientes sustituciones

$$\sigma_1 = \{x_1 := t_1, x_2 := t_2, \ldots, x_n := t_n\}$$
$$\sigma_2 = \{y_1 := s_1, y_2 := s_2, \ldots, y_m := s_m\}$$

La composición de estas sustituciones es

$$\sigma_1 \cdot \sigma_2 = \{x_1 := t_1\sigma_2, x_2 := t_2\sigma_2, \ldots, x_n := t_n\sigma_2, y_1 := s_1, y_2 := s_2, \ldots, y_m := s_m\}$$

- $w(\sigma_2 \cdot \sigma_1) = (w\sigma_2)\sigma_1$
- $(\sigma_1 \cdot \sigma_2) \cdot \sigma_3 = \sigma_1 \cdot (\sigma_2 \cdot \sigma_3)$
- $\sigma_1 \cdot \sigma_2 \neq \sigma_2 \cdot \sigma_1$ [En general]

$w = P(x, y)$, $\sigma_1 = \{x := f(y)\}$, $\sigma_2 = \{y := A\}$

$\sigma_1 \cdot \sigma_2 = \{x := f(A), y := A\}$, $\sigma_2 \cdot \sigma_1 = \{y := A, x := f(y)\}$

$w\sigma_1 = P(f(y), y)$ $w(\sigma_1 \cdot \sigma_2) = (w\sigma_1)\sigma_2 = P(f(A), A)$

$w\sigma_2 = P(x, A)$ $w(\sigma_2 \cdot \sigma_1) = (w\sigma_2)\sigma_1 = P(f(y), A)$

Sustitución: ejemplos

- Ejemplo 1: $w = A(x,y,f(z))$
  $\sigma=\{x:=D,\ y:=g(x),\ z:=C\}$
  $w\sigma = A(D,g(x),f(C))$

- Ejemplo 2: $w = B(x) \vee C(x,f(y))$
  $\sigma=\{x:=y,\ y:=g(A)\}$
  $w\sigma = B(y) \vee C(y,f(g(A)))$

- Ejemplo 3: $w = P(x,f(y),B)$
  - $\sigma_1 = \{x:=z,\ y:=w\}$
    $w\sigma_1 = P(z,f(w),B)$ [Variante alfabética]
  - $\sigma_2 = \{y:=A\}$
    $w\sigma_2 = P(x,f(A),B)$
  - $\sigma_3 = \{x:=g(z),y:=A\}$
    $w\sigma_2 = P(g(z),f(A),B)$
  - $\sigma_4 = \{x:=C,y:=A\}$
    $w\sigma_2 = P(C,f(A),B)$ [Instancia sin variables (instancia base)]

Una instancia base de una fbf es una fbf obtenida por sustitución de la fbf original que no contiene ninguna variable.

Unificación

- Se dice que la **sustitución** σ es el **unificador** de un conjunto de fbfs $\Gamma = \{w_1, w_2, \ldots, w_m\}$ cuando se cumple $w_1\sigma = w_2\sigma = \ldots = w_m\sigma$.
  - Este proceso se llama **unificación**
  - El resultado de la unificación es **único** excepto posibles variantes alfabéticas

Ej. {P(x,f(y),B), P(z,f(B),B)}

σ = {x:=A, y:=B, z:=A}

Unificación: {P(A,f(B),B)}

- **La unificación es correcta:** Dado que todas las variables están universalmente cuantificadas, la unificación es una forma particular de instanciación universal, que es una regla de inferencia correcta

- **Unificador más general (UMG)**

La sustitución μ es el unificador más general de un conjunto de fbfs $\Gamma = \{w_1, w_2, \ldots, w_m\}$,

$\mu = umg(\Gamma)$, si cualquier unificador del mismo conjunto de fbfs σ puede ser expresado como

$\sigma = \mu \cdot \sigma'$

- El conjunto de **discrepancia** de un conjunto no vacío de fbfs $\Gamma = \{w_1, w_2, \ldots, w_m\}$ es la primera subexpresión (la que está más a la izquierda) donde las fbfs en Γ no concuerdan

Ej. Γ = {P(x,f(A),B), P(x,g(A),B)}

D(Γ) = {f(A),g(A)}

Algoritmo de unificación

- **Algoritmo de unificación** (Chang & Lee 1973)

Entrada: Γ . **Salida**: $\mathrm{umg}(\Gamma)$ [unificador más general de Γ]

1. $k \leftarrow 0$; $\Gamma_k \leftarrow \Gamma$; $\sigma_k \leftarrow \varepsilon$ (sustitución vacía) (metemos en Γ todas las formulas que queremos unificar)
2. Si Γ_k tiene un solo elemento, devolver σ_k. En caso contrario continuar
3. $D_k \leftarrow$ conjunto de discrepancia de Γ_k (metemos en D el elemento k que queremos unificar)
4. Si D_k contiene (i) un término t_k
   (ii) una variable v_k que no esté en t_k

   entonces,

   $\sigma_{k+1} \leftarrow \sigma_k \cdot \{v_k := t_k\}$ (sustituyo la variable por el término, i.e. me quedo con lo más complejo)

   $\Gamma_{k+1} \leftarrow \Gamma_k\{v_k := t_k\}$ (eliminar fbfs repetidas) (actualizo las fórmulas con las sustituciones)

   En caso contrario salir devolviendo fallo [Γ no es unificable]

5. $k \leftarrow k+1$ e ir al paso 2.

Algoritmo de unificación

Ejemplo 1.

Γ = {P[f(x,g(A,y)),g(A,y)], P[f(x,z),z]}

1. $D_0 \leftarrow$ {z,g(A,y)};
   $\sigma_1 \leftarrow$ {z:=g(A,y)}, $\Gamma_1 \leftarrow$ {P[f(x,g(A,y)),g(A,y)]};

Ejemplo 2.

Γ={A(x,z,g(x,y,f(z))), A(y,f(x),w)}

1. $D_0 \leftarrow$ {x,y};
   $\sigma_1 \leftarrow$ {x:=y}, $\Gamma_1 \leftarrow$ {A(y,z,g(y,y,f(z))), A(y,f(y),w)}
2. $D_1 \leftarrow$ {z,f(y)};
   $\sigma_2 \leftarrow \sigma_1 \cdot$ {z:=f(y)};
   $\Gamma_2 \leftarrow$ {A(y, f(y),g(y,y,f(f(y)))),A(y,f(y),w)};
3. $D_2 \leftarrow$ {g(y,y,f(f(y))),w};
   $\sigma_3 \leftarrow \sigma_2 \cdot$ {w:= g(y,y,f(f(y)))};
   $\Gamma_3 \leftarrow$ {A(y,f(y),g(y,y,f(f(y))))};

Resolución en LPO

- **Una cláusula** es una disyunción de literales

  (puede ser representada como un conjunto de literales con disyunciones implícitas entre todos los literales del conjunto).

- Una fbf en **FNC** es una conjunción de cláusulas

  (puede ser representada como un conjunto de cláusulas con conjunciones implícitas entre las cláusulas del conjunto).

- **Resolución binaria entre cláusulas**

  Asumamos que los literales $\{\phi, \psi\}$ tienen un unificador más general μ.

  Entonces, del conjunto de cláusulas $\{\{\phi\} \vee \Sigma_1, \{\neg\psi\} \vee \Sigma_2\}$, es posible inferir $w = (\Sigma_1 \vee \Sigma_2)\mu$

Ej. $\Gamma = \{P(x) \vee Q(f(x)),\ R(g(y)) \vee \neg Q(f(A))\}$

$\phi = Q(f(x))$;

$\psi = Q(f(A))$;

$\mu = mgu(\{\psi, \phi\}) = \{x := A\}$

Se infiere por resolución $\{P(A) \vee R(g(y))\}$

Resolución generalizada

- La resolución binaria es **correcta** pero **no es completa para refutación**

Ej. $\Gamma = \{\neg P(x) \vee \neg P(y),\ P(z) \vee P(r)\}$

$\mu_1 = \{z := x\}$

Se infiere por resolución $\{\neg P(y) \vee P(r)\}$

De $\{\neg P(x) \vee \neg P(y),\ \neg P(y) \vee P(r)\}$

usando el umg $\mu_2 = \{r := y\}$

se infiere por resolución $\{\neg P(x) \vee \neg P(y)\}$

Todas las resoluciones binarias sólo producen variantes alfabéticas de estas cláusulas.

- Asumamos que todos los literales de la cláusula κ_1 pueden ser unificados con los literales negados de la cláusula κ_2 a través del unificador más general μ.

Entonces, del conjunto de cláusulas $\{\kappa_1 \vee \Sigma_1,\ \kappa_2 \vee \Sigma_2\}$ es posible inferir $w = (\Sigma_1 \vee \Sigma_2)\mu$

Ej. $\Gamma = \{\neg P(x) \vee \neg P(y), P(z) \vee P(r)\}$

$\mu = \{y := x,\ z := x,\ r := x\}$

Se infiere por resolución $\{\square\}$

Subsunción generalizada

- La cláusula K_1 **subsume a la cláusula** K_2 si existe una **sustitución** σ tal que

$$K_1\sigma \subset K_2$$

- Desde el punto de vista del mecanismo de **inferencia** la **cláusula subsumida**, K_2 puede ser **eliminada** de la base de conocimiento

- Ejemplo:

K_1: DisfrutaCon(Ringo,z)

K_2: ¬Placentero(Leer)∨DisfrutaCon(Ringo,Leer)

σ : {z:= Leer}

[Intuición: Cláusula K_1 es más "restrictiva" que la cláusula K_2]

Modus ponens generalizado

Asumiendo que las fbfs w_1 y w_2 tienen un unificador más general μ, se puede inferir $w_3\mu$ de w_1 y $w_2 \Rightarrow w_3$

- Ejemplo 1: ¿Es Juan malvado?

  Reglas: $(\forall x)\ [Rey(x) \wedge Avaricioso(x) \Rightarrow Malvado\ (x)]$

  Hechos: $Rey(Juan),\ Avaricioso(Juan)$

  se infiere ($\mu = \{x := Juan\}$)

  $Malvado(Juan)$

- Ejemplo2: ¿Le gustan a Ringo los cacahuetes?

  Reglas: $(\forall x)\ [Comida(x) \Rightarrow LeGusta(Ringo, x)]$

  $(\forall p, x)\ [\neg Muerto(p) \wedge ComeA(p, x) \Rightarrow Comida(x)]$

  $(\forall x)\ [ComeA(Juan, x) \Rightarrow ComeA(Jorge, x)]$

  Hechos: $ComeA(Juan, Cacahuetes),\ \neg Muerto(Juan),\ Comida(Manzana),\ Comida(Pollo)$

  De: $ComeA(Juan, Cacahuetes),\ \neg Muerto(Juan),$

  $[\neg Muerto(p) \wedge ComeA(p, x) \Rightarrow Comida\ (x)]$

  (usamos $\{p := Juan,\ x := Cacahuetes\}$)

  Se infiere: $Comida(Cacahuetes)$

  De: $[Comida(x) \Rightarrow LeGusta(Ringo, x)], Comida(Cacahuetes)$

  ($\mu = \{x := Cacahuetes\}$)

  Se infiere: **$LeGusta(Ringo, Cacahuetes)$**

Extracción de respuestas

Consideremos la Base de Conocimiento Δ.

Asumamos que necesitamos una prueba de $w = (\exists x) P(x)$

y queremos saber qué valores de x hacen que esta expresión sea consecuencia lógica de Δ.

Truco de Green (1999):

Usar refutación por resolución e incluir un literal "respuesta" en $\neg w$ para llevar la cuenta de las sustituciones realizadas.

Añadir a la BC: $(\forall x) [\neg P(x) \vee Ans(x)]$

En general: Añadir un literal $Respuesta(x_1, x_2, \ldots, x_m)$ a cada cláusula generada de la negación del teorema que se quiere demostrar.

Ojo: primero aplicar truco de Green y luego pasar a FNC.

Extracción de respuestas: ejemplo

Ejemplo:

BC: **Todos los humanos son mortales.** $(\forall x)[Humano(x) \Rightarrow Mortal(x)]$

Sócrates es humano. $Humano(Sócrates)$

¿Hay algún humano mortal? $(\exists x)[Humano(x) \wedge Mortal(x)]$

¿Quién es ese humano mortal?

Añadir a la BC: $(\forall x)[\neg Humano(x) \vee \neg Mortal(x) \vee Ans(x)]$

$\alpha = \{\neg Humano(x) \vee Mortal(x),\ Humano(Sócrates),$
$\neg Humano(y) \vee \neg Mortal(y) \vee Ans(y)\}$

$Mortal(Sócrates)$

$\neg Mortal(Sócrates) \vee Ans(Sócrates)$

$Ans(Sócrates)$

Consecuencia lógica en LPO

- **Consecuencia lógica:** La fbf w es consecuencia lógica de la BC Δ si todo modelo de Δ es también modelo de w.

$$\Delta \models w$$

- **Mecanismos de inferencia:**
  - **Proposicionalización**
    - Convertir fbfs de LPO a fbfs proposicionales eliminando todos los cuantificadores universales y existenciales.
    - Apicar a la BC proposicionalizada mecanismos de inferencia de lógica proposicional.
  - Uso de **reglas de inferencia generalizadas.**

    En particular, usar prueba mediante refutación con una regla de inferencia por resolución generalizada.

- Metateorema [Turing (1936), Church (1936)]

  "El problema de determinar si una fbf w es consecuencia lógica de la BC Δ en LPO es semi-decidible".

  Existen algoritmos tales que siempre que la expresión es consecuencia lógica, dicen SÍ en tiempo finito.

  No existen algoritmos tales que siempre que la expresión no sea consecuencia lógica, digan NO en tiempo finito.

Proposicionalización

- Solución a $\Delta_{LPO} \models w_{LPO}$ en LPO por **proposicionalización**:
  - Construir $\alpha_{LPO} = \Delta_{LPO} \wedge \neg w_{LPO}$
  - Transformar α_{LPO} a FNC ($\exists$ eliminados por Skolemización)
  - Transformar toda cláusula en α_{LPO} a cláusulas proposicionales haciendo todas las posibles asignaciones a las variables en α_{FOL} , de tal forma que sólo aparezcan términos base (términos sin variables).

Problema: Si consideramos funciones, hay **infinitos** términos base,

ej., `hijoDe(hijoDe(…hijoDe(hijoDe(`*`Pablo`*`))…))`

- Metateorema [**Herbrand** (1930)]

El conjunto de fbfs en LPO en FNC α_{LPO} es insatisfacible si hay un conjunto finito de cláusulas base (cláusulas sin variables) de α_{LPO} que es insatisfacible en cálculo proposicional.

- Algoritmo

$n \leftarrow 0$; α_{LPO}

1. $n \leftarrow n+1$
2. Crear la FNC proposicional α_n instanciando los términos en α_{LPO} usando una profundidad máxima n en las evaluaciones de funciones.

Hasta que la resolución sobre α_n produzca la cláusula vacía

Problema: El algoritmo funciona si α_{LPO} es INSAT, pero nunca para si α_{LPO} es SAT.

Refutación por resolución en LPO

Refutación por resolución generalizada

Considerar la base de conocimiento Δ y la fbf w en LPO.
¿Se cumple $\Delta \models w$?

1. Incluir la fbf negada en la BC $\alpha = \{\Delta \wedge \neg w\}$
2. Convertir a FNC
3. Aplicar resolución generalizada

   (i) Si la resolución genera la cláusula vacía (α es INSAT) entonces $\Delta \models w$

   (ii) Si la resolución no genera la cláusula vacía (α es SAT) entonces w no es consecuencia lógica de Δ.

- Si $\Delta \models w$, el algoritmo para y devuelve "Sí".
- Si $\Delta \not\models w$, el algoritmo puede no parar nunca.

La resolución es **correcta**, **completa para refutación** pero **semidecidible**.

Trucos

Normalmente,

$(\forall x)\ [w1(x) \Rightarrow w2(x)]$

$(\exists x)\ [w1(x) \wedge w2(x)]$

Ej. "Todo el mundo en la UAM es inteligente"

$(\forall x)\ [En(x,UAM) \Rightarrow Inteligente(x)]$

"Hay gente en la UAM que es inteligente"

$(\exists x)\ [En(x,UAM) \wedge Inteligente(x)]$

ERRORES COMUNES

$(\exists x)\ [En(x,UAM) \Rightarrow Inteligente(x)]$ [DEMASIADO DÉBIL]

"Hay alguien que o bien es inteligente o no está en la UAM"

$(\forall x)\ [En(x,UAM) \wedge Inteligente(x)]$ [DEMASIADO RESTRICTIVA]

"Todo el mundo está en la UAM y es inteligente"

PERO: "El mayor divisor de un entero es el propio entero"

$(\forall n)\ [Divisor(n,n) \wedge \neg(\exists m)\ [Divisor(m,n) \wedge Mayor(m,n)]]$

¿Mató la curiosidad al gato?

- Todo el que ama a los animales es amado por alguien.
- El que mata a un animal no es amado por nadie.
- Javier ama a todos los animales.
- El gato Tuno fue matado por Javier o por la curiosidad.
- ¿Mató la curiosidad al gato?

¿Mató la curiosidad al gato?

- Todo el que ama a todos los animales es amado por alguien.

  $(\forall x)[(\forall y)[Animal(y) \Rightarrow AmaA(x,y)] \Rightarrow (\exists z)\ AmaA(z,x)]$

- Cualquiera que mate a un animal no es amado por nadie.

  $(\forall x)[(\exists y)(Animal(y) \wedge MataA(x,y)) \Rightarrow (\forall z)\ \neg AmaA(z,x)]$

- Javier ama a todos los animales.

  $(\forall x)\ [Animal(x) \Rightarrow AmaA(Javier,x)]$

- El gato Tuno fue matado por Javier o por la curiosidad.

  $Gato(Tuno)$

  $MataA(Javier,Tuno) \vee MataA(Curiosidad,Tuno)$

- Un gato es un animal

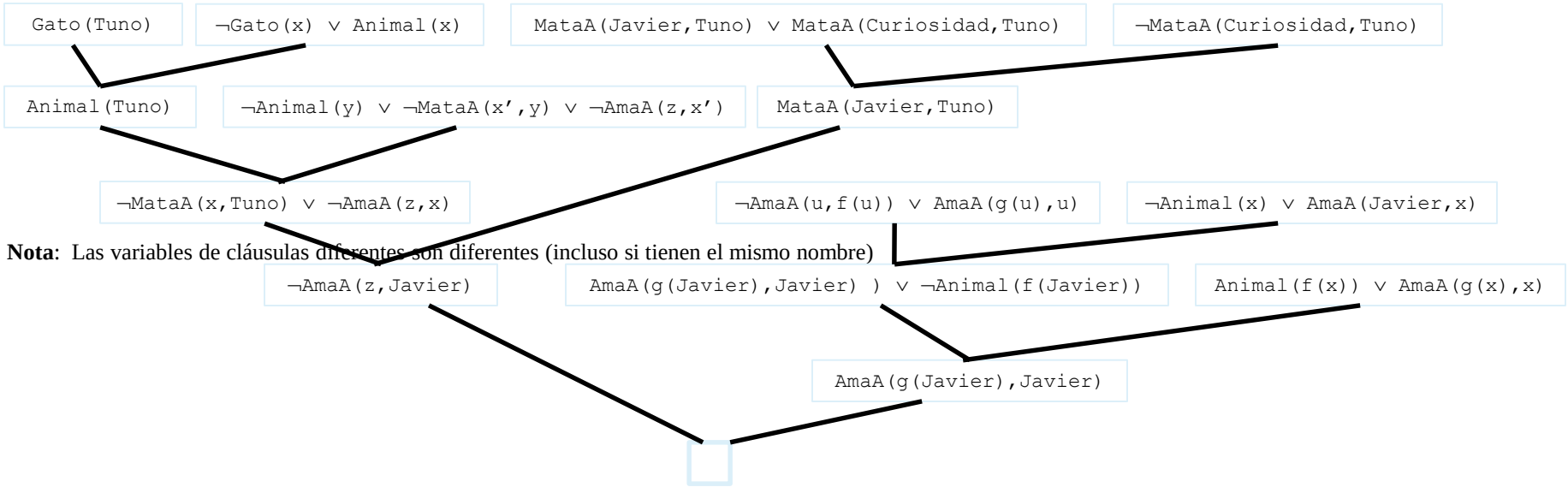
  $(\forall x)\ Gato(x) \Rightarrow Animal(x)$

- ¿Mató la curiosidad a Tuno?

  $MataA(Curiosidad,Tuno)$???

¿Mató la curiosidad al gato?

A. Animal(f(x)) ∨ AmaA(g(x),x)
B. ¬AmaA(u,f(u)) ∨ AmaA(g(u),u)
C. ¬Animal(y) ∨ ¬MataA(x,y) ∨ ¬AmaA(z,x)
D. ¬Animal(x) ∨ AmaA(Javier,x)
E. MataA(Javier,Tuno) ∨ MataA(Curiosidad,Tuno)
F. Gato(Tuno)
G. ¬Gato(x) ∨ Animal(x)
H. ¬MataA(Curiosidad,Tuno)



Nota: Las variables de cláusulas diferentes son diferentes (incluso si tienen el mismo nombre)

Ontología para conjuntos

- **Ontología** para **teoría de conjuntos** (vocabulario de objetos, predicados, funciones)
  - Objetos:
    - Conjunto vacío : $\{\}$
    - Elementos del conjunto: A, B, C, ...
  - Predicados: $Conjunto^1$, $Miembro^2(\in)$, $Subconjunto^2(\subset)$, $Igual^2(=)$
  - Funciones: $adjoin^2$ $(\{|\})$, $unión^2(\cup)$, $intersección^2(\cap)$

- **Axiomas** de la teoría de conjuntos:

1. $\forall s[Conjunto(s) \Leftrightarrow (s=\{\}) \vee (\exists x,s')[Conjunto(s') \wedge (s=\{x|s'\})]]$
2. $\neg(\exists x,s)[\{x|s\}=\{\}]$
3. $(\forall x,s)[x \in s \Leftrightarrow s=\{x|s\}]$
4. $(\forall x,s)[x \in s \Leftrightarrow (\exists y,s')[s=\{y|s'\} \wedge (x=y \vee x \in s')]]$
5. $(\forall s_1,s_2)[s_1 \subset s_2 \Leftrightarrow (\forall x)[x \in s_1 \Rightarrow x \in s_2]]$
6. $(\forall s_1,s_2)[s_1=s_2 \Leftrightarrow (s_1 \subset s_2 \wedge s_2 \subset s_1)]$
7. $(\forall x,s_1,s_2)[x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)]$
8. $(\forall x,s_1,s_2)[x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)]$

Ontología para listas

- Ontología para listas
  - **Objetos constantes:** `Nil, A,B,C... (átomos)`
  - **Predicados:** `Find`^{2}`, Atom`^{1}`, Listp`^{1}` , Null`^{1}
  - **Funciones:** `cons`^{2}`, append`^{2}`, first`^{1}`, rest`^{1}

- ¿Axiomas? [en casa: codifica los axiomas para caracterizar las propiedades de listas en]

Números naturales

- Ontología para números naturales
  - Objeto constante: 0
  - Predicados
    - Test de número natural: $\mathtt{NumNat}^1$
    - Predicado igualdad: $\mathtt{Igual}^2$ $(=)$
  - Funciones
    - Función sucesor: $\mathtt{sucesor}^1$
    - Función adición: $\mathtt{suma}^2$ $(+)$

- Axiomas de Peano

$$\mathtt{NumNat}(0)$$
$$(\forall n)[\mathtt{NumNat}(n) \Rightarrow \mathtt{NumNat}(\mathtt{sucesor}(n))]$$
$$(\forall n)[0 \neq \mathtt{sucesor}(n)]$$
$$(\forall m,n)[m \neq n \Rightarrow \mathtt{sucesor}(m) \neq \mathtt{sucesor}(n)]$$
$$(\forall n)[\mathtt{NumNat}(n) \Rightarrow \mathtt{suma}(n,0)=n]$$
$$(\forall m,n)[\mathtt{NumNat}(m) \wedge \mathtt{NumNat}(n) \Rightarrow \mathtt{suma}(\mathtt{sucesor}(m),n)=\mathtt{sucesor}(\mathtt{suma}(m,n))]$$

Nota: El principio de **inducción** sólo puede ser formulado en **lógica de segundo orden**, donde las relaciones y funciones de lógica de primer orden se tratan como objetos.

Por ello, se pueden hacer afirmaciones sobre ellas (ej. se puede escribir un predicado de segundo orden que especifica cuándo una relación de primer orden es transitiva).

Predicado igualdad

- **Igualdad:** Predicado especial que permite expresar que dos términos diferentes se refieren al mismo objeto del mundo real.
  - Igual(padreDe(Juan), Pedro)
    [también: padreDe(Juan)=Pedro)
    Significado: "Pedro es el padre de Juan"
  - $(\exists x,y)[Hermano\ (x,Jose) \wedge Hermano(y,Jose) \wedge \neg Igual(x,y)]$
    $(también: (\exists x,y)[Hermano(x,Jose) \wedge Hermano(y,Jose) \wedge (x \neq y)])$
    Significado: "Jose tiene al menos dos hermanos"

- **Axiomatización de la igualdad**
  - **Reflexiva:** $(\forall x)[x=x]$
  - **Simétrica:** $(\forall x,y)[x=y \Rightarrow y=x]$
  - **Transitiva:** $(\forall x,y,z)[[x=y] \wedge [y=z] \Rightarrow x=z]$
  - **Reglas de sustitución:** Los Igual pueden ser sustituidos por Igual en relaciones y funciones.
    $(\forall x,y)[x=y \Rightarrow f_i^{(1)}(x)=f_i^{(1)}(y)];\ i=1,2,...$
    $(\forall x,y)[x=y \Rightarrow P_i^{(1)}(x)=P_i^{(1)}(y)];\ i=1,2,...$
    $(\forall x,y,z,t)\ [(x=z) \wedge (y=t) \Rightarrow f_i^{(2)}(x,y)=f_i^{(2)}(z,t)];\ i=1,2,...$
    $(\forall x,y,z,t)\ [(x=z) \wedge (y=t) \Rightarrow P_i^{(2)}(x,y)=P_i^{(2)}(z,t)];\ i=1,2,...$
    ......

Evaluación de expresiones

- Para algunos predicados y funciones en vez de usar inferencia es más sencillo y eficiente **evaluar expresiones** que contengan **términos base**.
  - Predicado de igualdad:
    Evaluar `¬(2222=4444)`
  - Comparaciones Booleanas:
    Evaluar `(2222<=4444)`
  - Funciones aritméticas:
    Evaluar `(2222+4444)`

- En inferencia con fbfs en FNC
  - Cláusulas cuya evaluación es
    - `Falso` hacen que el proceso de inferencia termine (con una contradicción)
    - `Verdadero` pueden ser eliminadas de la Base de Conocimiento.
  - Literales cuya evaluación es
    - `Falso` pueden ser eliminados de la cláusula.
    - `Verdadero` nos permiten eliminar la cláusula de la Base de Conocimiento.