Inferencia en Lógica de Primer Orden II

BLOQUE TEMÁTICO:	**Representación del Conocimiento y Razonamiento**
TEMA 3.2:	**Representación y Razonamiento mediante Lógica**
LECCIÓN 17:	**Inferencia en Lógica de Predicados II**

DESARROLLO DEL TEMA:

1. Encadenamiento hacia delante y hacia atrás.
2. Programación Lógica.
3. Resolución.

Especificación Universal (EU)

Podemos inferir cualquier sentencia obtenida por sustitución de la variable por un término base (un término sin variables).
Denotamos el resultado de aplicar una sustitución θ a una sentencia α mediante $SUST(\theta, \alpha)$.
Cada especificación de una sentencia cuantificada universalmente es implicada por ella:

$$\frac{\forall v \alpha}{SUST(\{v/g\}, \alpha)}$$

para cualquier variable v y término base g.

Ejemplo, $\forall x Rey(x) \wedge Codicioso(x) \Rightarrow Malvado(x)$ infiere cualquiera de las siguientes sentencias:
$Rey(Juan) \wedge Codicioso(Juan) \Rightarrow Malvado(Juan)$
$Rey(Ricardo) \wedge Codicioso(Ricardo) \Rightarrow Malvado(Ricardo)$
$Rey(Padre(Juan)) \wedge Codicioso(Padre(Juan)) \Rightarrow Malvado(Father(Juan))$
$\vdots$

Especificación Existencial (EE)

Para cualquier sentencia α, variable v, y símbolo constante k, *que no aparezca en ninguna otra parte de la base de conocimiento*:

$$\frac{\exists v \alpha}{SUST(\{v/k\}, \alpha)}$$

Ejemplo, $\exists x Corona(x) \wedge SobreCabeza(x, Juan)$ implic

$$Corona(C_1) \wedge SobreCabeza(C_1, Juan)$$

mientras que C_1 sea un nuevo símbolo constante, llamado Constante de Skolem.

Otro ejemplo: de $\exists x\ d(x^y)/dy = x^y$, obtenemos

$$d(e^y)/dy = e^y$$

mientras que e sea un símbolo constante nuevo.

Unificación

UNIFICAR$(\alpha, \beta) = \theta$ si $\alpha\theta = \beta\theta$

El algoritmo *UNIFICA* toma dos sentencias y devuelve un **unificador** para ellas, si éste existe.

Supongamos que tenemos una petición $Conoce(Juan, x)$: ¿A quién conoce Juan?

p	q	θ
$Conoce(Juan, x)$	$Conoce(Juan, Juana)$	{x/Juana}
$Conoce(Juan, x)$	$Conoce(y, OJ)$	{x/OJ,y/Juan}
$Conoce(Juan, x)$	$Conoce(y, Madre(y))$	{y/Juan,x/Madre(Juan)}
$Conoce(Juan, x)$	$Conoce(x, OJ)$	fallo

Modus Ponens Generalizado (MPG)

$$\frac{p_1{}',\quad p_2{}',\ \ldots,\ p_n{}',\quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

donde $p_i{}'\theta = p_i\theta$ para todo i.

p_1' *es Rey*(*Juan*)	p_1 *es Rey*(x)
p_2 *es* Codicioso(y)	p_2 *es* Codicioso(x)
θ *es* $\{x/Juan, y/Juan\}$	q *es Malvado*(x)
$q\theta$ *es* Malvado(Juan)	

Ejemplo de base de conocimiento

La ley dice que para un Americano es un crimen vender armas a naciones hostiles. Cierto país, enemigo de América, tiene algunos misiles y todos sus misiles fueron vendidos por un Coronel americano.

Probar que el coronel es un criminal.

Ejemplo de base de conocimiento

... para un americano es un crimen vender armas a naciones hostiles: $Americano(x) \wedge Arma(y) \wedge Vende(x, y, z) \wedge Hostil(z) \Rightarrow Criminal(x)$

País ... tiene algunos misiles, es decir, $\exists\, x\; Tiene($***País***,$x) \wedge Misil(x)$:

$Tiene($***País***,$M_1)$ y $Misil(M_1)$

... todos los misiles del país le fueron vendidos por el coronel.

$\forall x Misil(x) \wedge Tiene($***País***,$x) \Rightarrow Vende(Coronel, x,$***País***$)$

Los misiles son armas:

$Misil(x) \Rightarrow Arma(x)$

Ejemplo de base de conocimiento

Un enemigo de américa cuenta como hostil:

$Enemigo(x,\textit{América}) \Rightarrow Hostil(x)$

El Coronel, que es americano, ...

$Americano(Coronel)$

El País, un enemigo de América ...

$Enemigo(\textit{País},\textit{América})$

Algoritmo sencillo de encadenamiento hacia delante

función PREGUNTA-EHD-LPO (BC,α) devuelve una sustitución o *falso*
  entradas: BC, un conjunto de cláusulas positivas de primer orden
    α, la petición, una sentencia atómica
  variables locales: *nuevas*, las nuevas sentencias inferidas en cada iteración
 repetir hasta *nuevo* está vacío
  nuevo $\leftarrow \{\}$
  para cada sentencia *r* en la BC, hacer
   $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ ESTANDARIZAR-VAR(r)
   para cada θ tal que SUST$(\theta, p_1 \wedge \ldots \wedge p_n) = SUST(p'_1 \wedge \ldots \wedge p'_n)$
      para algún $p'_1, \ldots, p'_n$ en BC
    $q' \leftarrow SUST(\theta, q)$
    si q' no es el renombramiento de una sentencia de BC o de *nuevo*,
    entonces hacer
     añadir q' a *nuevo*
     $\phi \leftarrow UNIFICA(q', \alpha)$
     si ϕ no es *fallo* entonces devolver ϕ
  añadir *nuevo* a BC
 devolver *falso*

Ejemplo de EHD

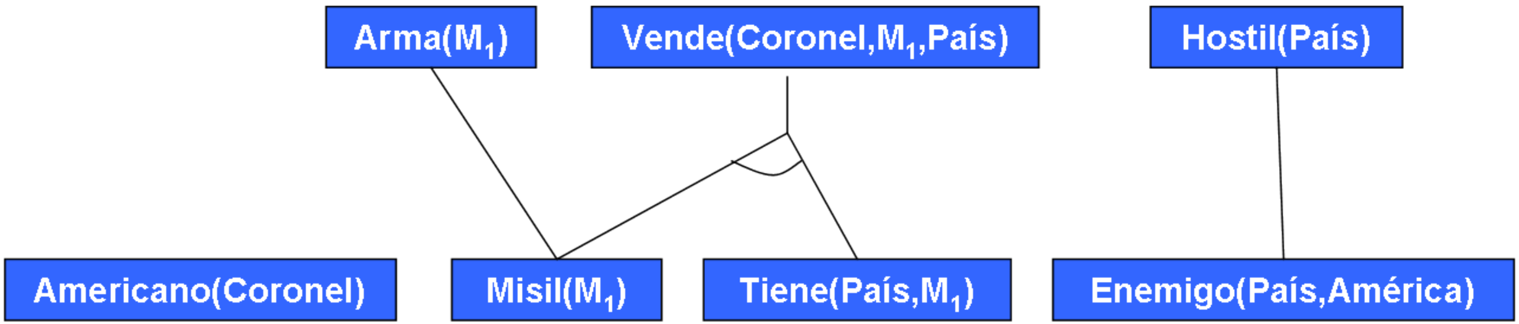
Usaremos nuestro problema sobre el crimen para ilustrar cómo funciona el algoritmo PREGUNTA-EHD-LPO.
Las sentencias de implicación son:

- $Americano(x) \wedge Arma(y) \wedge Vende(x, y, z) \wedge Hostil(z) \Rightarrow Criminal(x)$
- $\forall x Misil(x) \wedge Tiene($**País**$, x) \Rightarrow Vende(Coronel, x,$**País**$)$
- $Misil(x) \Rightarrow Arma(x)$
- $Enemigo(x,$**América**$) \Rightarrow Hostil(x)$

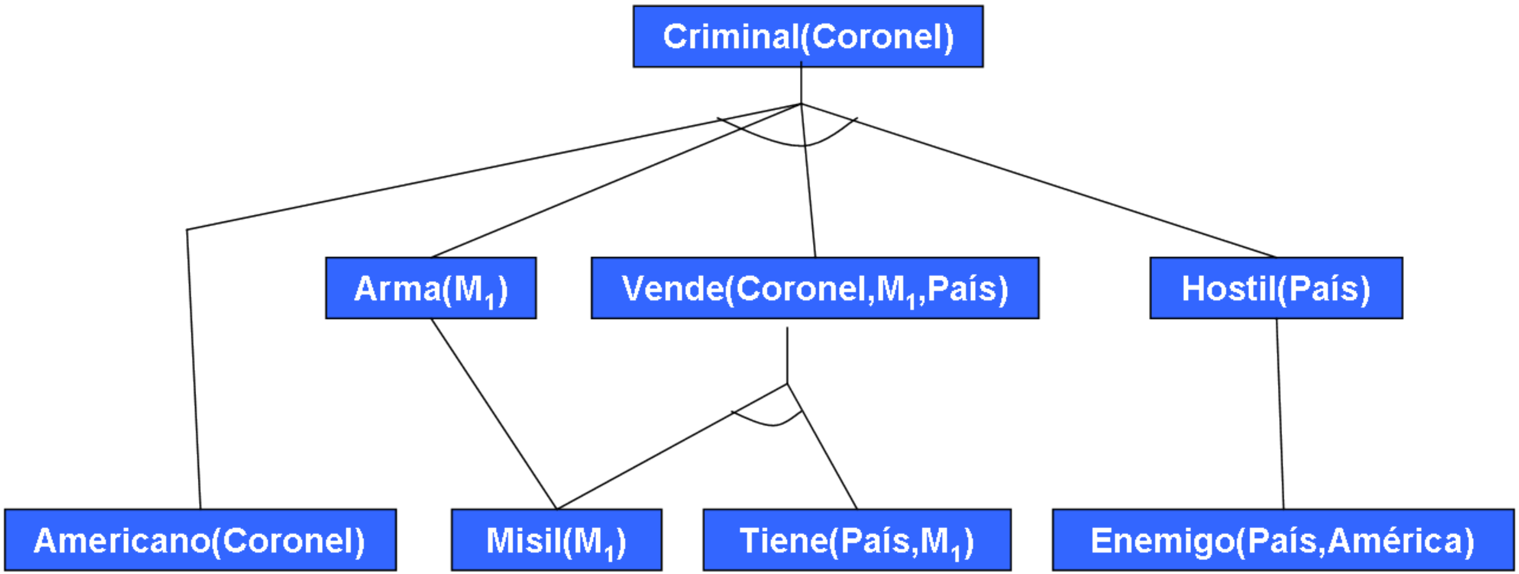
Ejemplo de EHD

Americano(Coronel) | Misil(M_1) | Tiene(País,M_1) | Enemigo(País,América)

Ejemplo de EHD



Ejemplo de EHD



Algortimo sencillo de Encadenamiento Hacia Atrás

función PREGUNTA-EHA-LPO (BC,objetivos,θ) devuelve un conjunto de sustituciones

entradas: *BC*, una base de conocimientos

objetivos, una lista de conjuntores que forman la petición (θ ya aplicada)

θ, la sustitución inicial, inicialmente la sustitución vacía

variables locales: *respuestas*, un conjunto de sustitucioines inicialmente vacío.

si *objetivos* está vacío entonces devolver θ

$q' \leftarrow SUST(\theta, PRIMERO($*objetivos*$))$

para cada sentencia *r* en BC

donde ESTANDARIZAR-VAR(*r*) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$

and $\theta' \leftarrow UNIFICA($*q*, *q'*$)$ tiene éxito

nuevos_objetivos $\leftarrow [p_1, \ldots, p_n | RESTO($*objetivos*$)]$

respuestas $\leftarrow$ PREGUNTA-EHA-LPO(BC, *nuevos_objetivos*, COMPÓN(θ', θ))

$\cup$ *respuestas*

devolver *respuestas*

Composición, COMPÓN

El algoritmo de EHA utiliza la composición de sustituciones.

COMPÓN($\theta 1, \theta 2$) es la sustitución cuyo efecto es idéntico a aplicar cada sustitución en orden. Es decir,

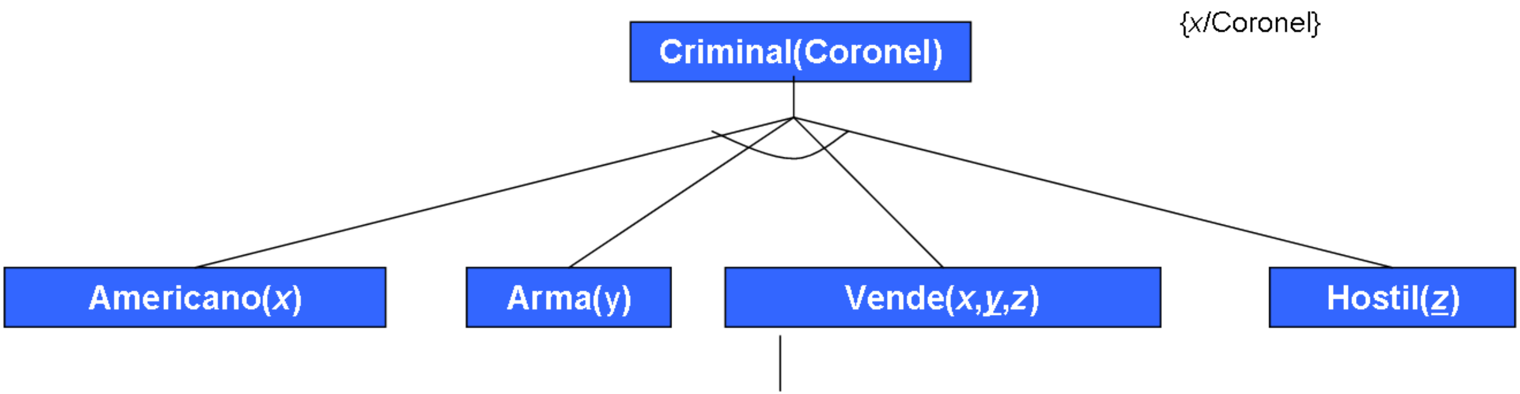
SUST(COMPÓN($\theta 1, \theta 2$),p) = SUST($\theta 2$, SUST($\theta 1$,p))

Ejemplo de Encadenamiento Hacia Atrás

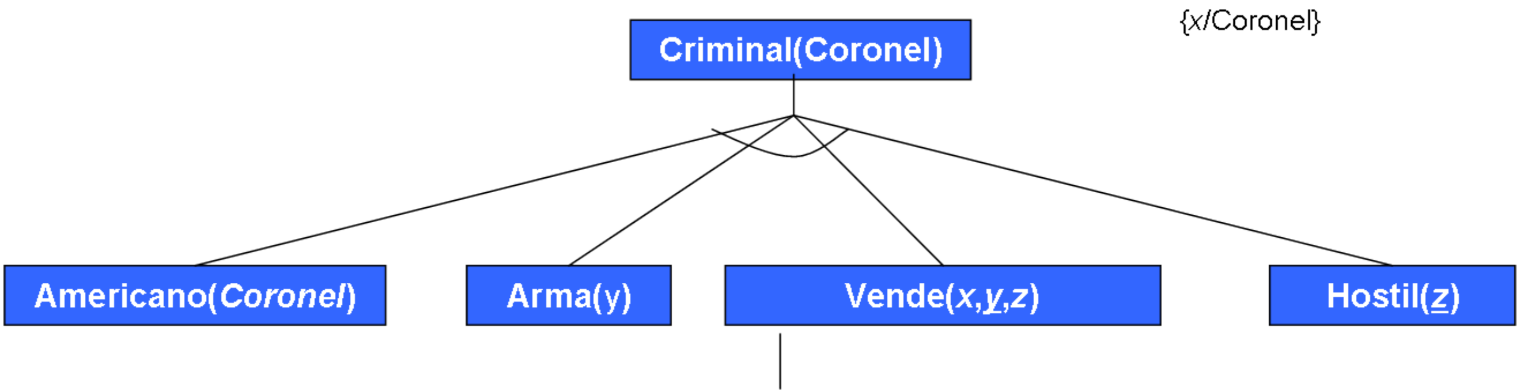
Criminal(Coronel)

- $Americano(x) \wedge Arma(y) \wedge Vende(x,y,z) \wedge Hostil(z) \Rightarrow Criminal(x)$

Ejemplo de Encadenamiento Hacia Atrás

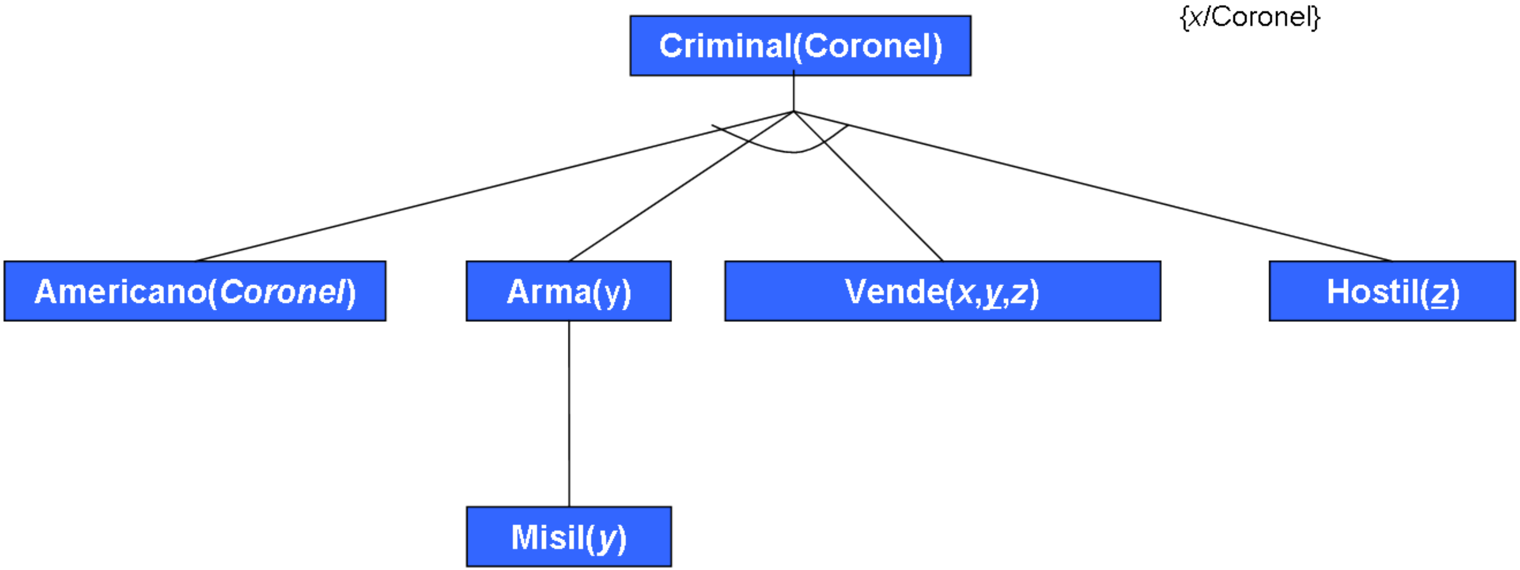


Ejemplo de Encadenamiento Hacia Atrás

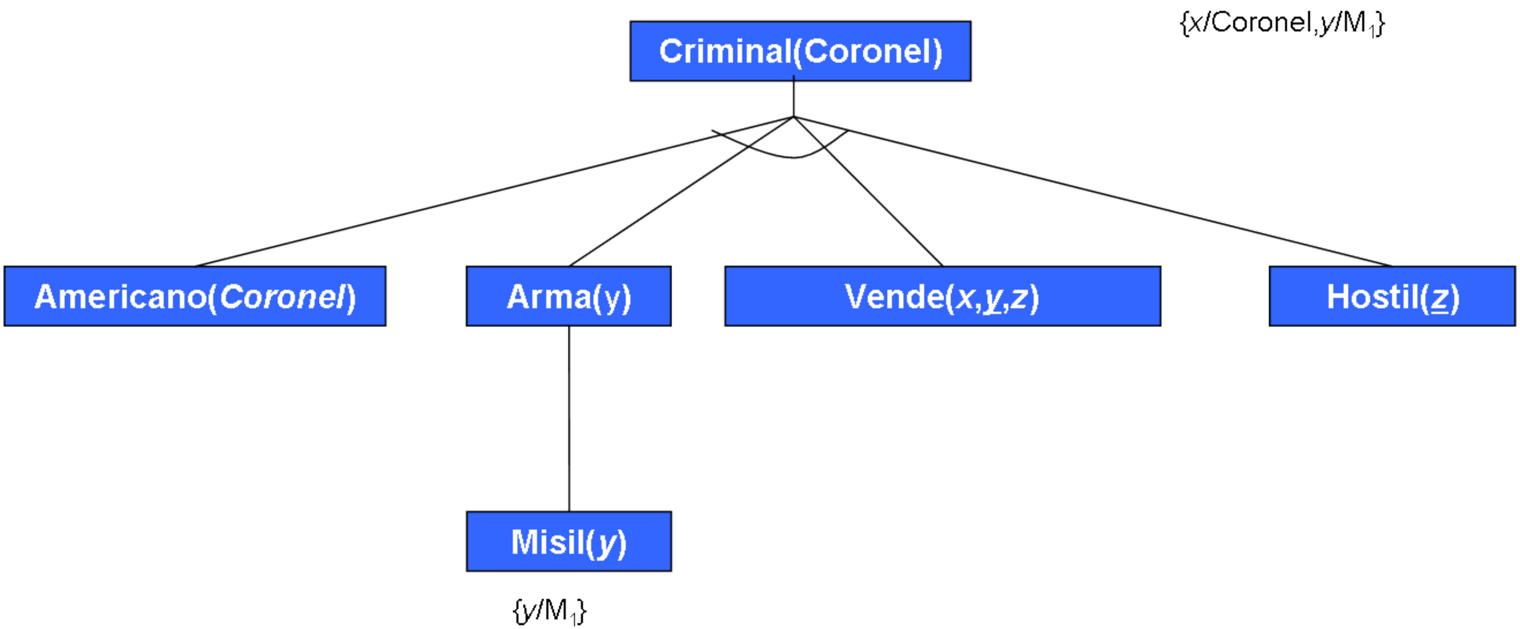


- $Misil(x) \Rightarrow Arma(x)$

Ejemplo de Encadenamiento Hacia Atrás

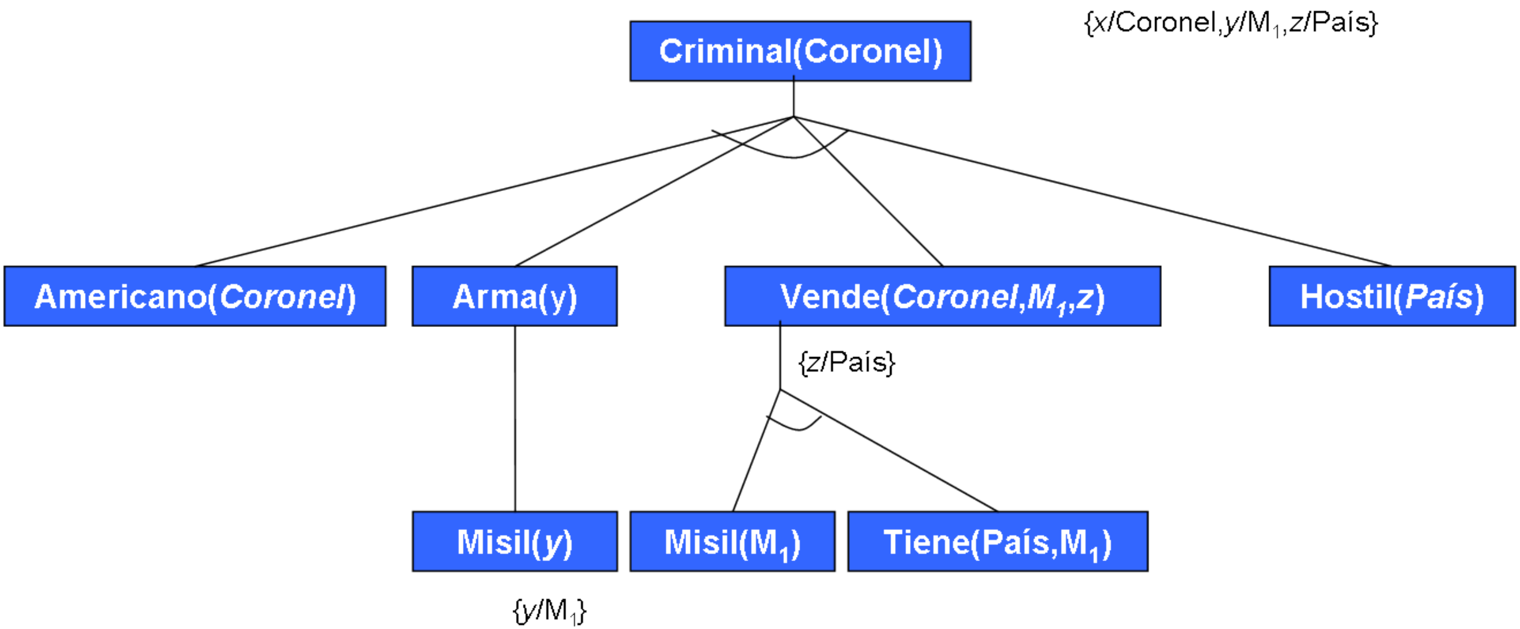


Ejemplo de Encadenamiento Hacia Atrás



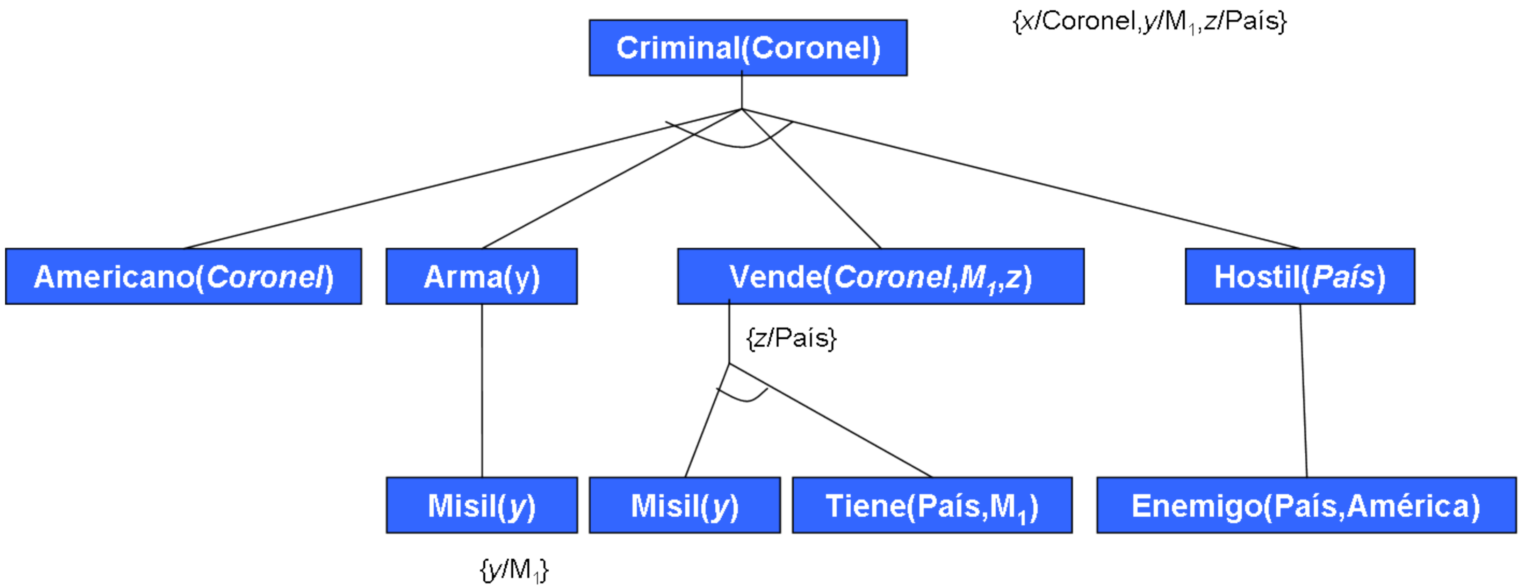
- $\forall x Misil(x) \wedge Tiene(\textit{\textbf{País}}, x) \Rightarrow Vende(Coronel, x, \textit{\textbf{País}})$

Ejemplo de Encadenamiento Hacia Atrás



- $Enemigo(x,\textit{América}) \Rightarrow Hostil(x)$

Ejemplo de Encadenamiento Hacia Atrás



Propiedades del EHA

- El EHA, tal como lo hemos escrito, es claramente un algoritmo de búsqueda primero en profundidad. Por tanto, sus requerimientos de espacio son lineales respecto al tamaño de la demostración.
- Además, el EHA (a diferencia del EHD) sufre algunos problemas con los estados repetidos y la incompletitud. Es incompleto debido a los bucles infinitos.
- Es ineficiente debido a los subobjetivos repetidos.
- Ampliamente usado (sin mejoras!) para la programación lógica.

Programación Lógica

Computación como inferencia sobre BCs lógicas.

	Programación Lógica	Programación Ordinaria
1.	Identificar el problema	Identificar el problema
2.	Recopilar información	Recopilar información
3.	–	Calcular la solución
4.	Codificar la información en BC	Solución del programa
5.	Codificar instancias del problema como hechos	Codificar instancias del problema
6.	Preguntar peticiones	Aplicar el programa a los datos
7.	Encontrar hechos falsos	Eliminar errores del procedimient

Debería ser más fácil depurar $Capital(NewYork, US)$ que $x := x + 2$!

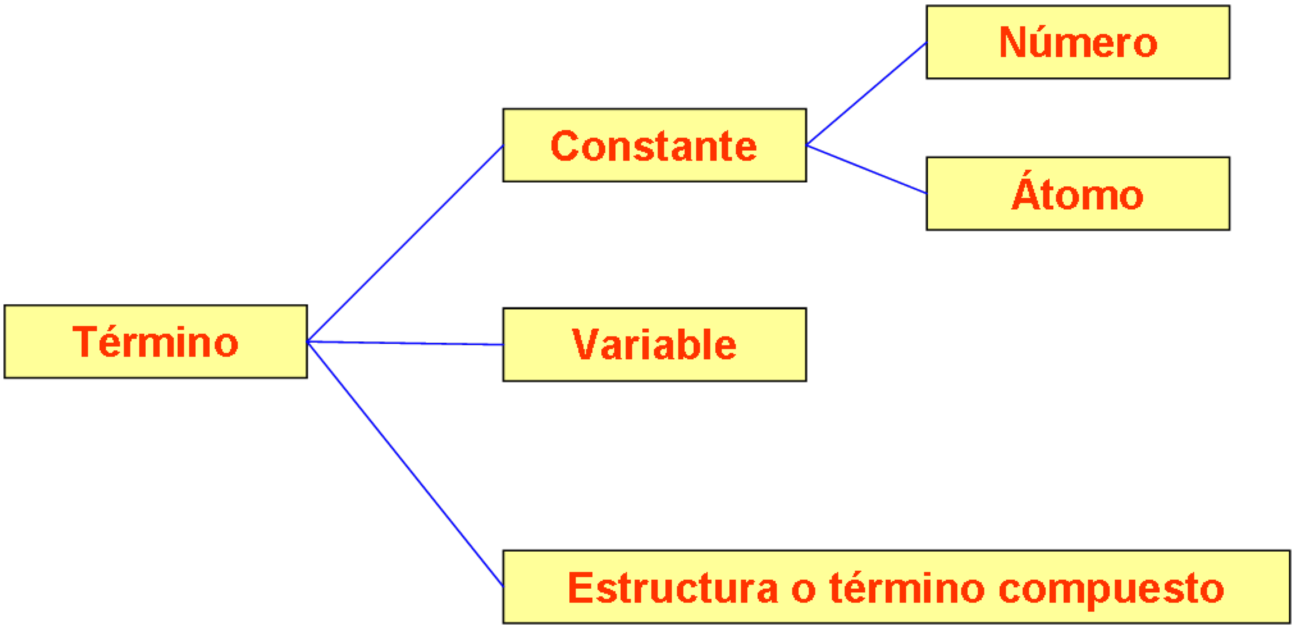
Sistemas Prolog

Prolog es el lenguaje de programación más utilizado.
Razonamiento según lógica de primer orden (cláusulas de Horn).
Programa = conjunto de cláusulas positivas escritas en una notación algo diferente a la estándar de la lógica de primer orden.
Medidas de eficiencia: LIPS (inferencias lógicas por segundo).
Estructura:

- Declaración de hechos sobre objetos y relaciones.
- Definición de reglas sobre dichos objetos y relaciones.
- Planteamiento de consultas.

Estos apartados se subdividen en cláusulas (= instrucciones).
Prolog utiliza las letras mayúsculas en las variables y las minúsculas en las constantes.

Prolog: unidades básicas



Prolog: Hechos

- Cláusulas simples, formadas por predicados con argumentos instanciados: nombre_pred (*arg1*,*arg2*, ...,arg3)
- Los argumentos son términos.
- La aridad es el número de argumentos
- Los predicados se denotan *nombre/aridad*: nombre_pred/n
- Se pueden anidar términos generando estructuras:

  posee (juan, libro (ulises, autor(james, joyce), 3124))

femenino(maría)
masculino(raúl)
progenitor (maría, raúl)

Prolog: Consultas

- Meta desde la que se infiere una respuesta.
- Proceso de inferencia busca en la base de hechos algo que unifique la meta, devolviendo: éxito: Yes o fracaso: No.

?- progenitor(maría, raúl) Yes
?- progenitor(raúl, maría) No

Sistemas Prolog

Las cláusulas están escritas con la cabeza precediendo al cuerpo; «:-» se utiliza para las implicaciones a la izquierda, las comas separan los literales en el cuerpo, y el punto indica el final de la sentencia:

criminal(x) :- americano(x), arma(y), vende(x, y, z), hostil(z).

El Prolog incluye una sintaxis para la notación de listas y la artimética. Por ejemplo: unir(X,Y,Z), que tiene éxito si la lista Z es el resultado de unir las listas X e Y.

unir([],Y,Y); unir una lista vacía a una lista Y genera la misma lista Y.

Podemos realizar la petición unir(A,B,[1,2]): ¿qué listas se pueden unir para obtener [1,2]? Obtenemos las soluciones:
A = [], B = [1,2]
A = [1], B = [2]
A = [1,2], B = []

Resolución

El tema de la existencia de procedimientos de demostración completos concierne directamente a los matemáticos.

1930: **Kurt Gödel** demostró el primer teorema de la completitud para la lógica de primer orden, probando que una sentencia implicada tiene una desmostración finita.

1931: **Kurt Gödel** demostró el aún más famoso teorema de la incompletitud, que muestra que un sistema lógico que incluye el principio de inducción (sin el que muy pocas de las matemáticas discretas se pueden construir) es necesariamente incompleto.

A pesar del teorema de Gödel, los demostradores de teoremas basados en la resolución han sido utilizados para demostrar teoremas matemáticos.

Formas Normales Conjuntivas (FNC) en LPO

Al igual que en lógica proposicional, la resolución en Lógica de Primer Orden (LPO) requiere que las sentencias estén en la Forma Normal Conjuntiva (FNC), es decir, una conjunción de cláusulas, donde cada cláusula es disyunción de literales.

Por ejemplo:

$$\forall x\ \mathit{Americano}(x) \land \mathit{Arma}(y) \land \mathit{Vende}(x,y,z) \land \mathit{Hostil}(z) \Rightarrow \mathit{Criminal}(x)$$

se transforma a FNC como:

$$\neg\ \mathit{Americano}(x) \lor \neg \mathit{Arma}(y) \lor \neg \mathit{Vende}(x,y,z) \lor \neg \mathit{Hostil}(z) \lor \mathit{Criminal}(x)$$

Cada sentencia en lógica de primer orden se puede convertir a una sentencia en FNC que es equivalente inferencialmente.

Formas Normales Conjuntivas (FNC) en LPO

El procedimiento para la conversión a FNC es similar al del caso proposicional, con la diferencia de que ahora es necesario eliminar los cuantificadores universales.
Veámoslo con el siguiente ejemplo:

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

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

Formas Normales Conjuntivas (FNC) en LPO

Eliminación de las implicaciones

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

Anidar las $\neg$

$\neg\forall x\ p$ se convierte en $\exists x\ \neg p$

$\neg\exists x\ p$ se convierte en $\forall x\ \neg p$

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

Formas Normales Conjuntivas (FNC) en LPO

Estandarizar las variables: para las sentencias del tipo ($\forall x$ *P*(x)) ∨ ($\exists x$ *Q*(x)) que utilizan el mismo nombre de variable dos veces, se cambia una de las dos para evitar confusiones.

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

Skolemizar: borrar los cuantificadores existenciales mediante su eliminación.

En el caso más sencillo se aplica la regla de Especificación Existencial.

$$\forall x\ [\mathit{Animal}(A) \wedge \neg \mathit{Ama}(x, A)] \vee [\mathit{Ama}(B, x)]$$

donde se obtiene un significado totalmente erróneo; la sentencia dice que todo el mundo o no puede amar al animal A o es amado por la entidad B. Por tanto, lo que queremos es obtener las entidades de Skolem que dependen de x:

$$\forall x\ [\mathit{Animal}(F(x)) \wedge \neg \mathit{Ama}(x, F(x))] \vee [\mathit{Ama}(G(x), x)]$$

F y G son **funciones de Skolem**, que dependen de las variables cuantificadas universalmente cuyo ámbito abarca a los cuantificadores existenciales.

Formas Normales Conjuntivas (FNC) en LPO

Eliminar los cuantificadores universales: En este punto, en el que todas las variables que quedan deben estar cuantificadas universalmente, podemos eliminar los cuantificadores universales.

$$[\mathit{Animal}(F(x)) \wedge \neg \mathit{Ama}(x, F(x))] \vee [\mathit{Ama}(G(x), x)]$$

Distribuir la $\wedge$ respecto a la $\vee$:

$$[\mathit{Animal}(F(x)) \vee \mathit{Ama}(G(x), x)] \wedge [\neg \mathit{Ama}(x, F(x)) \vee \mathit{Ama}(G(x), x)]$$

La regla de inferencia de Resolución

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{SUST(\theta, \ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

donde $UNIFICA(\ell_i, \neg m_j) = \theta$.

Por ejemplo, podemos resolver las dos cláusulas:

[*Animal*$(F(x))$ $\vee$ *Ama*$(G(x), x)$] y [$\neg$*Ama*(u, v) $\vee$ $\neg$*Mata*(u, v)]

eliminando los literales complementarios *Ama*$(G(x), x)$ y $\neg$*Ama*(u, v), con el unificador $\theta = \{u/G(x), v/x\}$, para producir la cláusula resolvente

[*Animal*$(F(x))$ $\vee$ $\neg$*Mata*$(G(x), x)$]

Demostración del ejemplo del crimen

La resolución demuestra que $BC \models \alpha$ probando que BC y $\neg\alpha$ es insatisfactible. Las sentencias en FNC son:

- $\neg$ *Americano*$(x) \vee \neg$ *Arma*$(y) \vee \neg$ *Vende*$(x, y, z) \vee \neg$ *Hostil*$(z) \vee$ *Criminal*(x)
- $\neg$ *Misil*$(x) \vee \neg$ *Tiene*(*País*,x)$\vee$ *Vende*$(Coronel, x,$*País*$)$
- $\neg$ *Enemigo*$(x,$*América*$)\vee$ *Hostil*(x)
- $\neg$ *Misil*$(x)\vee$ *Arma*(x)
- *Tiene*(*País*,M_1)
- *Misil*(M_1)
- *Americano*$(Coronel)$
- *Enemigo*(*País*,*América*)

Demostración del ejemplo del crimen