

# Metodología de Programación I

## Resolución-SLD

Dr. Alejandro Guerra-Hernández

Departamento de Inteligencia Artificial  
Facultad de Física e Inteligencia Artificial  
aguerra@uv.mx  
<http://www.uv.mx/aguerra>

Maestría en Inteligencia Artificial 2009



Universidad Veracruzana

# Ejemplo

- ▶ Consideren el siguiente programa definitivo  $\Delta$ :

*orgullosos/1*

*orgullosos(X) ← padre(X, Y), recién\_nacido(Y).*

*padre(X, Y) ← papa(X, Y).*

*padre(X, Y) ← mama(X, Y).*

*papa(juan, marta).*

*recién\_nacido(marta).*

- ▶ Interpretamos la meta  $\leftarrow \textit{orgullosos}(Z)$ . como ¿Alguien está orgulloso?



# Metas

- ▶ En realidad la meta  $\leftarrow orgullososo(Z)$ .
- ▶ Es una abreviatura de  $\forall Z \neg orgullososo(Z)$
- ▶ Que a su vez es equivalente a  $\neg \exists Z orgullososo(Z)$ .
- ▶ Que se lee como “Nadie está orgulloso”. La respuesta negativa a la pregunta original ¿Alguién está orgulloso?
- ▶ Si demostramos que ese enunciado es falso en todo modelo de  $\Delta$ , entonces sabremos que  $\Delta \models \exists Z orgullososo(Z)$ . Pero eso solo respondería *si* a la pregunta original.
- ▶ El objetivo en realidad es encontrar una **substitución**  $\theta$  tal que el conjunto  $\Delta \cup \{\neg orgullososo(Z)\}\theta$  sea insatisfacible, lo que equivale a que  $\Delta \models orgullososo(Z)\theta$ .



## Razonamiento: se inicia por la meta

- ▶ El punto inicial de nuestro razonamiento es asumir la meta  $G_0$ 
  - Para cualquier  $Z$ ,  $Z$  no está orgulloso.
- ▶ La inspección del programa  $\Delta$  revela que una regla describe una condición para que alguien esté orgulloso:

$$\text{orgullosos}(X) \leftarrow \text{padre}(X, Y), \text{recien\_nacido}(Y).$$

- ▶ Lo cual es lógicamente equivalente a:

$$\forall (\neg \text{orgullosos}(Z) \Rightarrow \neg (\text{padre}(X, Y) \wedge \text{recien\_nacido}(Y)))$$



# Estrategia: para generar nuevas metas

- ▶ Partiendo de:

$$\forall(\neg orgulloso(Z) \Rightarrow \neg(\text{padre}(X, Y) \wedge \text{recien\_nacido}(Y)))$$

- ▶ Al renombrar  $X$  por  $Z$ , eliminar el cuantificador universal y usar *modus ponens* con respecto a  $G_0$ , obtenemos  $G_1$ :

$$\neg(\text{padre}(Z, Y) \wedge \text{recien\_nacido}(Y))$$

- ▶ o su equivalente:

$$\leftarrow \text{padre}(Z, Y), \text{recien\_nacido}(Y).$$

- ▶  $G_1$  que es verdadera en todo modelo  $\Delta \cup \{G_0\}$ .



# Estrategia: y resolverlas

- Ahora solo queda probar que  $\Delta \cup \{G_1\}$  es no satisfacible. Observen que  $G_1$  es equivalente a la fbf:

$$\forall Z \forall Y (\neg \text{padre}(Z, Y) \vee \neg \text{recien\_nacido}(Y))$$

- Se puede probar que  $G_1$  es no satisfacible para  $\Delta$ , si en todo modelo de  $\Delta$  hay una persona que es padre de un recién nacido. Verificamos primero si hay padres con estas condiciones. El programa contiene la cláusula:

$$\text{padre}(X, Y) \leftarrow \text{papa}(X, Y).$$

- Que es equivalente a:

$$\forall (\neg \text{padre}(X, Y) \Rightarrow \neg \text{papa}(X, Y))$$

- Por lo que  $G_1$  se reduce a  $G_2$ :

$$\leftarrow \text{papa}(Z, Y), \text{recien\_nacido}(Y).$$



# Estretega: recursiva

- ▶ El programa declara que *juan* es padre de *marta*:

*papa(juan, marta).*

- ▶ Así que sólo resta probar que “*marta* no es una recién nacida” no se puede satisfacer junto con  $\Delta$ :

$\leftarrow$  *recien\_nacido(marta).*

- ▶ pero el programa contiene el hecho:

*recien\_nacido(marta).*

- ▶ equivalente a  $\neg$ *recien\_nacido(marta)*  $\Rightarrow$   $\square$
- ▶ lo que conduce a una refutación  $\square$ .



# Resumiendo

- ▶ Para probar la existencia de algo:
  - ▶ Suponer lo **opuesto**
  - ▶ y usar **modus ponens** y la regla de **eliminación del cuantificador universal**,
  - ▶ para encontrar un contra ejemplo al supuesto.



# Unificador

- ▶ La meta es ahora un conjunto de átomos a ser probados.
- ▶ Para ello, se seleccionó una fbf atómica de la meta  $p(s_1, \dots, s_n)$  y una cláusula del programa con la forma  $p(t_1, \dots, t_n) \leftarrow A_1, \dots, A_n$
- ▶ para encontrar una instancia común de  $p(s_1, \dots, s_n)$  y  $p(t_1, \dots, t_n)$ , es decir, una substitución  $\Theta$  que hace que  $p(s_1, \dots, s_n)\Theta$  y  $p(t_1, \dots, t_n)\Theta$  sean idénticos. Tal substitución se conoce como **unificador**.
- ▶ La nueva meta se construye reemplazando el átomo seleccionado en la meta original, por los átomos de la cláusula seleccionada, aplicando  $\Theta$  a todos los átomos obtenidos de esta manera.



# Resolución-SLD

- ▶ La regla de inferencia **principio de resolución SLD** para programas definitivos, combina *modus ponens*, *eliminación del cuantificador universal* y en el paso final un *reductio ad absurdum*.
- ▶ Si se prueba en  $k$  pasos que la meta definida en cuestión no puede satisfacerse, probamos que:

$$\leftarrow (A_1, \dots, A_m)\Theta_1 \dots \Theta_k$$

- ▶ es una instancia que no puede satisfacerse. De manera equivalente, que:

$$\Delta \models (A_1 \wedge \dots \wedge A_m)\Theta_1 \dots \Theta_k$$



# Observaciones

- ▶ Esta computación **no es determinista**: cualquier átomo de la meta puede ser seleccionado y pueden haber varias cláusulas del programa que unifiquen con el átomo seleccionado.
- ▶ Pueden existir unificadores alternativos para dos átomos. Esto sugiere que es posible construir muchas soluciones (algunas veces, una cantidad infinita de ellas).
- ▶ Por otra parte, es posible también que el átomo seleccionado no unifique con ninguna cláusula en el programa. Esto indica que no es posible construir un contra ejemplo para la meta definida inicial.
- ▶ La computación puede caer en un **ciclo** y de esta manera no producir solución alguna.



# Substitución

## Definición (Substitución)

Una *substitución*  $\theta$  es un conjunto finito de la forma:

$$\{X_1/t_1, \dots, X_n/t_n\}, \quad (n \geq 0)$$

donde las  $X_i$  son variables, *distintas* entre si, y los  $t_i$  son términos. Decimos que  $t_i$  *substituye* a  $X_i$ . La forma  $X_i/t_i$  se conoce como *ligadura* de  $X_i$ . La *substitución*  $\theta$  se dice se dice *de base* (grounded) si cada término  $t_i$  es un término base (no incluye variables).

- ▶ La *substitución vacía* se conoce como *substitución de identidad* y se denota por  $\epsilon$ .



# Restricción

- ▶ La **restricción** de una substitución  $\theta$  sobre un conjunto de variables  $Var$  es la substitución  $\{X/t \in \theta \mid X \in Var\}$ .

## Ejemplo

*$\{Y/X, X/g(X, Y)\}$  y  $\{X/a, Y/f(Z), Z/(f(a), X_1/b)\}$  son substituciones. La restricción de la segunda substitución sobre  $\{X, Z\}$  es  $\{X/a, Z/f(a)\}$ .*



# Casos

## Definición (Caso)

Sea  $\theta = \{X_1/t_1, \dots, X_n/t_n\}$  una substitución y  $\alpha$  una expresión. Entonces  $\alpha\theta$ , el **caso** (instance) de  $\alpha$  por  $\theta$ , es la expresión obtenida al substituir simultáneamente  $X_i$  por  $t_i$  para  $1 \leq i \leq n$ .

- ▶ Las **expresiones** son términos, literales ó conjunciones y disyunciones de literales.
- ▶ Si  $\alpha\theta$  es una expresión de base, se dice que es un **caso base** y se dice que  $\theta$  es una substitución de base para  $\alpha$ .
- ▶ Si  $\Sigma = \{\alpha_1, \dots, \alpha_n\}$  es un conjunto finito de expresiones, entonces  $\Sigma\theta$  denota  $\{\alpha_1\theta, \dots, \alpha_n\theta\}$ .



# Ejemplo

- ▶ Sea  $\alpha$  la expresión  $p(Y, f(X))$  y
- ▶ sea  $\theta$  la substitución  $\{X/a, Y/g(g(X))\}$ .
- ▶ La ocurrencia de  $\alpha$  por  $\theta$  es  $\alpha\theta = p(g(g(X)), f(a))$ .
- ▶ Observen que  $X$  e  $Y$  son simultáneamente remplazados por sus respectivos términos, lo que implica que  $X$  en  $g(g(X))$  no es afectada por  $X/a$ .



# Composición de substituciones

## Definición (Composición)

Sean  $\theta = \{X_1/s_1, \dots, X_m/s_m\}$  y  $\sigma = \{Y_1/t_1, \dots, Y_n/t_n\}$  dos substituciones. Consideren la secuencia:

$$X_1/(s_1\sigma), \dots, X_m/(s_m\sigma), Y_1/t_1, \dots, Y_n/t_n$$

Si se borran de esta secuencia las ligaduras  $X_i/s_i\sigma$  cuando  $X_i = s_i\sigma$  y cualquier ligadura  $Y_j/t_j$  donde  $Y_j \in \{X_1, \dots, X_m\}$ . La substitución consistente en las ligaduras de la secuencia resultante es llamada **composición** de  $\theta$  y  $\sigma$ , se denota por  $\theta\sigma$ .



# Ejemplo

- ▶ Sea  $\theta = \{X/f(Y), Z/U\}$  y  $\sigma = \{Y/b, U/Z\}$ .
- ▶ Construimos la secuencia de ligaduras  $X/(f(Y)\sigma), Z/(U)\sigma, Y/b, U/Z$  lo cual es  $X/f(b), Z/Z, Y/b, U/Z$ .
- ▶ Al borrar la ligadura  $Z/Z$  obtenemos la secuencia  $X/f(b), Y/b, U/Z = \theta\sigma$ .



# Ocurrencia

## Definición (Ocurrencia)

Sean  $\theta$  y  $\sigma$  dos substituciones. Se dice que  $\theta$  es una *ocurrencia* de  $\sigma$ , si existe una substitución  $\gamma$ , tal que  $\sigma\gamma = \theta$ .

- ▶ La substitución  $\theta = \{X/f(b), Y/a\}$  es una ocurrencia de  $\sigma = \{X/f(X), Y/a\}$ , puesto que  $\sigma\{X/b\} = \theta$ .



# Propiedades de las substituciones

- ▶ Sea  $\alpha$  una expresión, y sea  $\theta$ ,  $\sigma$  y  $\gamma$  substituciones. Las siguientes relaciones se cumplen:
  1.  $\theta = \theta\epsilon = \epsilon\theta$
  2.  $(\alpha\theta)\sigma = \alpha(\theta\sigma)$
  3.  $(\theta\sigma)\gamma = \theta(\sigma\gamma)$



# Unificación

- ▶ Uno de los pasos principales en el cómputo de una meta, es que dos fbf atómicas se vuelvan sintácticamente equivalentes. Este proceso se conoce como **unificación** y posee una solución algorítmica.

## Definición (Unificador)

Sean  $\alpha$  y  $\beta$  términos. Una substitución  $\theta$  tal que  $\alpha$  y  $\beta$  sean idénticos ( $\alpha\theta = \beta\theta$ ) es llamada **unificador** de  $\alpha$  y  $\beta$ .

- ▶ Por ejemplo:
  1.  $unifica(\text{conoce}(\text{juan}, X), \text{conoce}(\text{juan}, \text{maria})) = \{X/\text{maria}\}$
  2.  $unifica(\text{conoce}(\text{juan}, X), \text{conoce}(Y, Z)) = \{Y/\text{juan}, X/Z\}$



# Unificador más general (MGU)

## Definición (Generalidad entre substituciones)

Una substitución  $\theta$  se dice *más general* que una substitución  $\sigma$ , si y sólo si existe una substitución  $\gamma$  tal que  $\sigma = \theta\gamma$ .

## Definición (MGU)

Un unificador  $\theta$  se dice el *unificador más general (MGU)* de dos términos, si y sólo si  $\theta$  es más general que cualquier otro unificador entre esos términos.



# Forma resuelta y MGU

## Definición (Forma resuelta)

*Un conjunto de ecuaciones  $\{X_1 = t_1, \dots, X_n = t_n\}$  está en **forma resuelta**, si y sólo si  $X_1, \dots, X_n$  son variables distintas que no ocurren en  $t_1, \dots, t_n$ .*

- ▶ Sea  $\{X_1 = t_1, \dots, X_n = t_n\}$  un conjunto de ecuaciones en forma resuelta. Entonces  $\{X_1/t_1, \dots, X_n/t_n\}$  es un MGU (idempotente) de la forma resuelta.



# Equivalencia

## Definición (Equivalencia en conjuntos de ecuaciones)

*Dos conjuntos de ecuaciones  $E_1$  y  $E_2$  se dicen **equivalentes**, si tienen el mismo conjunto de unificadores.*

- ▶ La definición puede usarse como sigue: para computar el MGU de dos términos  $\alpha$  y  $\beta$ , primero intente transformar la ecuación  $\{\alpha = \beta\}$  en una forma resuelta equivalente. Si esto falla, entonces  $mgu(\alpha, \beta) = \text{fallo}$ . Sin embargo, si una forma resuelta  $\{X_1 = t_1, \dots, X_n = t_n\}$  existe, entonces  $mgu(\alpha, \beta) = \{X_1/t_1, \dots, X_n/t_n\}$ .



# Algoritmo de unificación

```

1: function UNIFICA( $E$ )
2:   repeat
3:      $(s = t) \leftarrow \text{seleccionar}(E)$ 
4:     if  $f(s_1, \dots, s_n) = f(t_1, \dots, t_n)$  ( $n \geq 0$ ) then
5:       reemplazar  $(s, t)$  por  $s_1 = t_1, \dots, s_n = t_n$ 
6:     else if  $f(s_1, \dots, s_m) = g(t_1, \dots, t_n)$  ( $f/m \neq g/n$ ) then
7:       return(fallo)
8:     else if  $X = X$  then
9:       remover la  $X = X$ 
10:    else if  $t = X$  then
11:      reemplazar  $t = X$  por  $X = t$ 
12:    else if  $X = t$  then
13:      if subtermino( $X, t$ ) then
14:        return(fallo)
15:      else reemplazar todo  $X$  por  $t$ 
16:      end if
17:    end if
18:  until No hay accion posible para  $E$ 
19: end function

```

▷  $E$  es un conjunto de ecuaciones



# Ejemplo 1

- ▶ El conjunto  $\{f(X, g(Y)) = f(g(Z), Z)\}$  tiene una forma resuelta, pues:

$$\begin{aligned}\{f(X, g(Y)) = f(g(Z), Z)\} &\Rightarrow \{X = g(Z), g(Y) = Z\} \\ &\Rightarrow \{X = g(Z), Z = g(Y)\} \\ &\Rightarrow \{X = g(g(Y)), Z = g(Y)\}\end{aligned}$$



## Ejemplo 2

- ▶ El conjunto  $\{f(X, g(X), b) = f(a, g(Z), Z)\}$  no tiene forma resuelta, puesto que:

$$\Rightarrow \{X = a, g(X) = g(Z), b = Z\}$$

$$\Rightarrow \{X = a, g(a) = g(Z), b = Z\}$$

$$\Rightarrow \{X = a, a = Z, b = Z\}$$

$$\Rightarrow \{X = a, Z = a, b = Z\}$$

$$\Rightarrow \{X = a, Z = a, b = a\}$$

$$\Rightarrow \textit{fallo}$$



## Ejemplo 3

- ▶ El conjunto  $\{f(X, g(X)) = f(Z, Z)\}$  no tiene forma resuelta, puesto que:

$$\Rightarrow \{X = Z, g(X) = Z\}$$

$$\Rightarrow \{X = Z, g(Z) = Z\}$$

$$\Rightarrow \{X = Z, Z = g(Z)\}$$

$\Rightarrow$  *fallo*



# Consideraciones

- ▶ Este algoritmo termina y regresa una forma resuelta equivalente al conjunto de ecuaciones de su entrada, o bien regresa fallo si la forma resuelta no existe.
- ▶ Sin embargo, el computar  $subtermino(X, t)$  (**verificación de ocurrencia**) hace que el algoritmo sea altamente ineficiente.
- ▶ El standard ISO Prolog (1995) declara que el resultado de la unificación es no decidible.
- ▶ Al eliminar la verificación de ocurrencia es posible que al intentar resolver  $X = f(X)$  obtengamos  $X = f(f(X)) \dots = f(f(f \dots))$ .



# Formalizando

- ▶ El método de razonamiento descrito informalmente al inicio de esta sesión, puede resumirse con la siguiente regla de inferencia:

$$\frac{\forall \neg(\alpha_1 \wedge \dots \wedge \alpha_{i-1} \wedge \alpha_i \wedge \alpha_{i+1} \wedge \dots \wedge \alpha_m) \quad \forall(\beta_0 \leftarrow \beta_1 \wedge \dots \wedge \beta_n)}{\forall \neg(\alpha_1 \wedge \dots \wedge \alpha_{i-1} \wedge \beta_1 \wedge \dots \wedge \beta_n \wedge \alpha_{i+1} \wedge \dots \wedge \alpha_m)\theta}$$

- ▶ donde:

1.  $\alpha_1, \dots, \alpha_m$  son fbf atómicas.
2.  $\beta_0 \leftarrow \beta_1, \dots, \beta_n$  es una cláusula definitiva en el programa  $\Delta$  ( $n \geq 0$ ).
3.  $MGU(\alpha_i, \beta_0) = \theta$ .



# Observaciones

- ▶ La regla tiene dos premisas: una meta y una cláusula definitivas.
- ▶ Cada una por su cuenta están cuantificadas universalmente, por lo que el alcance de los cuantificadores es disjunto.
- ▶ Solo hay un cuantificador universal para la conclusión. Por lo tanto, se requiere que el conjunto de variables en las premisas sea disjunto.
- ▶ Puesto que todas las variables en las premisas están cuantificadas, es siempre posible **renombrar** las variables de la cláusula definitiva para cumplir con esta condición.





# Usando la resolución-SLD

- ▶ El punto de partida es una meta definida  $G_0$ :

$$\leftarrow \alpha_1, \dots, \alpha_m \quad (m \geq 0)$$

- ▶ Una submeta  $\alpha_j$  será seleccionada. Una nueva meta  $G_1$  se construye al seleccionar una cláusula del programa  $\beta_0 \leftarrow \beta_1, \dots, \beta_n$  ( $n \geq 0$ ) cuya cabeza  $\beta_0$  unifica con  $\alpha_j$ , resultando en  $\theta_1$ .  $G_1$  tiene la forma:

$$\leftarrow (\alpha_1, \dots, \alpha_{j-1}, \beta_1, \dots, \beta_n, \dots, \alpha_m)\theta_1$$

- ▶ Ahora es posible aplicar el principio de resolución a  $G_1$  para obtener  $G_2$ , y así sucesivamente.



# Terminación

- ▶ El proceso puede terminar o no. Hay dos situaciones donde no es posible obtener  $G_{i+1}$  a partir de  $G_i$ :
  1. cuando la submeta seleccionada no puede ser resuelta (no es unificable con la cabeza de una cláusula del programa).
  2. cuando  $G_i = \square$  (meta vacía = **f**).



# Derivación-SLD

## Definición (Derivación-SLD)

Sea  $G_0$  una meta definitiva,  $\Delta$  un programa definitivo y  $\mathcal{R}$  una función de selección. Una **derivación SLD** de  $G_0$  (usando  $\Delta$  y  $\mathcal{R}$ ) es una secuencia finita o infinita de metas:

$$G_0 \overset{\alpha_0}{\rightsquigarrow} G_1 \dots G_{n-1} \overset{\alpha_{n-1}}{\rightsquigarrow} G_n$$

- ▶  $\alpha_j \in \Delta$ . Las variables en  $\alpha_j$  se renombran con subíndices  $i$ .



# Substitución computada

- ▶ Cada derivación SLD nos lleva a una secuencia de MGUs  $\theta_1, \dots, \theta_n$ . La composición

$$\theta = \begin{cases} \theta_1\theta_2 \dots \theta_n & \text{si } n > 0 \\ \epsilon & \text{si } n = 0 \end{cases}$$

de MGUs se conoce como la **substitución computada** de la derivación.



# Ejemplo

- ▶ Consideren la meta definida  $\leftarrow orgullososo(Z)$  y el programa del inicio de esta clase. Entonces

$$G_0 = \leftarrow orgullososo(Z).$$

$$\alpha_0 = orgullososo(X_0) \leftarrow padre(X_0, Y_0), recién\_nacido(Y_0).$$

- ▶ La unificación de  $orgullososo(Z)$  y  $orgullososo(X_0)$  nos da el MGU  $\theta_1 = \{X_0/Z\}$ . Asumamos que nuestra función de selección es tomar la **submeta más a la izquierda**:

$$G_1 = \leftarrow padre(Z, Y_0), recién\_nacido(Y_0).$$

$$\alpha_1 = padre(X_1, Y_1) \leftarrow papa(X_1, Y_1).$$

$$\text{con } \theta_2 = \{X_1/Z, Y_1/Y_0\}.$$



# Ejemplo

- ▶ La derivación continua como sigue:

$$G_2 = \leftarrow \text{papa}(Z, Y_0), \text{recien\_nacido}(Y_0).$$

$$\alpha_2 = \text{papa}(\text{juan}, \text{marta}).$$

$$G_3 = \leftarrow \text{recien\_nacido}(\text{marta}).$$

$$\alpha_3 = \text{recien\_nacido}(\text{marta}).$$

$$G_4 = \square$$

- ▶ la substitución computada para esta derivación es:

$$\begin{aligned} \theta_1\theta_2\theta_3\theta_4 &= \{X_0/Z\}\{X_1/Z, Y_1/Y_0\}\{Z/\text{juan}, Y_0/\text{marta}\}\epsilon \\ &= \{X_0/\text{juan}, X_1/\text{juan}, Y_1/\text{marta}, \\ &\quad Z/\text{juan}, Y_0/\text{marta}\} \end{aligned}$$



# Refutación-SLD y derivación fallida

## Definición (Refutación SLD)

*Una derivación SLD finita:*

$$G_0 \overset{\alpha_0}{\rightsquigarrow} G_1 \dots G_{n-1} \overset{\alpha_{n-1}}{\rightsquigarrow} G_n$$

donde  $G_n = \square$ , se llama *refutación SLD* de  $G_0$ .

## Definición (Derivación fallida)

*Una derivación de la meta  $G_0$  cuyo último elemento no es la meta vacía y no puede resolverse con ninguna cláusula del programa, es llamada *derivación fallida*.*



# Arbol-SLD

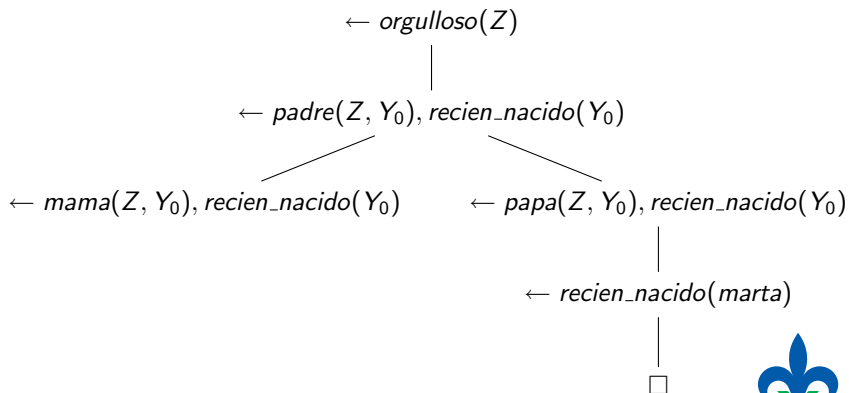
## Definición (Arbol-SLD)

Sea  $\Delta$  un prog. definitivo,  $G_0$  una meta definitiva, y  $\mathcal{R}$  una función de selección. El *árbol-SLD* de  $G_0$  (usando  $\Delta$  y  $\mathcal{R}$ ) es un árbol etiquetado, posiblemente infinito, que cumple con:

- ▶ La raíz del árbol está etiquetada por  $G_0$ .
- ▶ Si el árbol contiene un nodo etiquetado como  $G_i$  y existe una cláusula renombrada  $\alpha_i \in \Delta$  tal que  $G_{i+1}$  es derivada de  $G_i$  y  $\alpha_i$  via  $\mathcal{R}$ , entonces el nodo etiquetado como  $G_i$  tiene un hijo etiquetado  $G_{i+1}$ . El arco entre ambos es  $\alpha_i$ .



# Ejemplo



# Propiedades de la resolución-SLD

## Definición (Consistencia)

Sea  $\Delta$  un programa definitivo,  $\mathcal{R}$  una función de selección, y  $\theta$  una substitución de respuesta computada a partir de  $\Delta$  y  $\mathcal{R}$  para una meta  $\leftarrow \alpha_1, \dots, \alpha_m$ . Entonces  $\forall((\alpha_1 \wedge \dots \wedge \alpha_m)\theta)$  es una *consecuencia lógica* del programa  $\Delta$ .

## Definición (Compleción)

Sea  $\Delta$  un programa definitivo,  $\mathcal{R}$  una función de selección y  $\leftarrow \alpha_1, \dots, \alpha_m$  una meta definitiva. Si  $\Delta \models \forall((\alpha_1 \wedge \dots \wedge \alpha_m)\sigma)$ , entonces existe una refutación de  $\leftarrow \alpha_1, \dots, \alpha_m$  vía  $\mathcal{R}$  con una substitución de respuesta computada  $\theta$ , tal que  $(\alpha_1 \wedge \dots \wedge \alpha_m)\sigma$  es un caso de  $(\alpha_1 \wedge \dots \wedge \alpha_m)\theta$ .



# Bibliografía



U. Nilsson and J. Maluszynski.  
*Logic, Programming and Prolog.*  
John Wiley & Sons Ltd, 2nd edition, 2000.

