

Programación para la Inteligencia Artificial

Programación Lógica

Dr. Alejandro Guerra-Hernández

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
*Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097*
mailto:aguerra@uv.mx
<https://www.uv.mx/personal/aguerra/pia>

Maestría en Inteligencia Artificial 2023



Universidad Veracruzana

Objetivo

- ▶ A la programación lógica le concierne el uso de la Lógica de Primer Orden para **representar** y **resolver** problemas.
- ▶ Usamos una lógica de primer orden **restringida** a **cláusulas de Horn** y **resolución-SLD** [4, 2].
- ▶ Revisaremos algunos **conceptos preliminares** de la Lógica de Primer Orden, necesarios para **formalizar** los conceptos citados.



Representación

- ▶ Cuando describimos situaciones de nuestro interés, usamos **enunciados declarativos**.
- ▶ Expresiones del lenguaje natural que son o bien **verdaderas**, o bien **falsas** (a diferencia de interrogativas, admirativas, etc.).
- ▶ Las **proposiciones** representan hechos que se dan o no en la realidad.
- ▶ La Lógica de Primer Orden tienen un compromiso ontológico más fuerte [5], donde la realidad implica además, **objetos** y **relaciones** entre ellos.



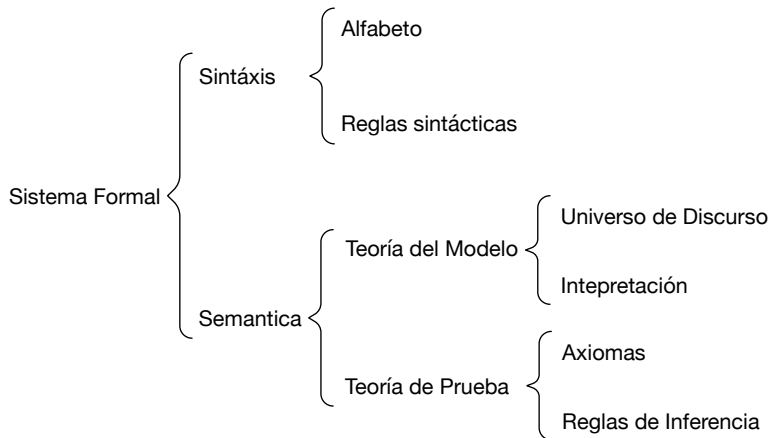
Razonamiento

- ▶ Consideren los siguientes enunciados declarativos:
 1. Julia es madre y Luis es hijo de Julia.
 2. Toda madre ama a sus hijos.
- ▶ Conociéndolas es posible inferir:
 3. Julia ama a Luis.
- ▶ Para ello asumimos un conjunto de **reglas de inferencia**.
- ▶ **Ejemplo:** *Modus Ponens*

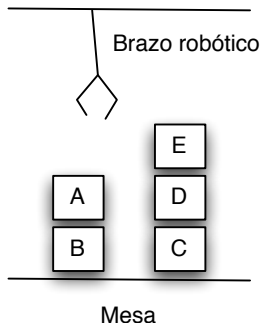
$$\frac{P, P \rightarrow Q}{Q} \quad (M.P.)$$



Componentes de un Sistema Formal



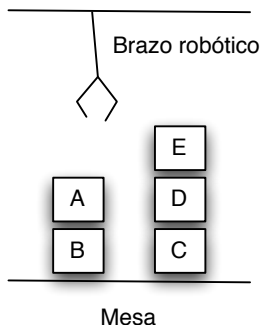
El mundo de los bloques [1]



- ▶ Necesitaremos **símbolos** para:
 - ▶ Objetos
 - ▶ Relaciones
 - ▶ Funciones
 - ▶ Variables



Universo de discurso



- ▶ Se denota como \mathcal{U} y es el conjunto de todos **objetos** sobre los cuales queremos expresarnos.

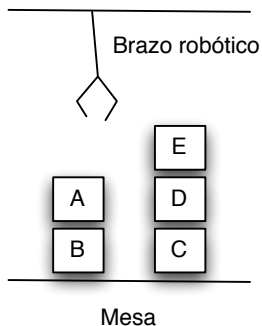
- ▶ Para el mundo de los bloques:

$$\mathcal{U} = \{a, b, c, d, e, \text{brazo}, \text{mesa}\}$$

- ▶ Seguiré la notación usada por Prolog, las **constantes** son cadenas que inician con minúscula.



Funciones



- ▶ Son relaciones especiales entre los miembros de \mathcal{U} . Mapean un conjunto de objetos de entrada a un **objeto único** de salida.
- ▶ La función **parcial sombrero** = $\{(b, a), (c, d), (d, e)\}$:

$$\text{sombrero}(b) \mapsto a$$

- ▶ **Base funcional** es el conjunto de todas las funciones consideradas.



Predicados

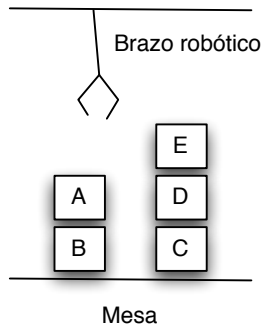
- ▶ Denotan **relaciones** entre los miembros de \mathcal{U} . Mapean a **falso** o **verdadero**.
- ▶ El predicado $sobre = \{(a, b), (d, c), (e, d)\}$:

$$sobre(a, b) \mapsto true$$

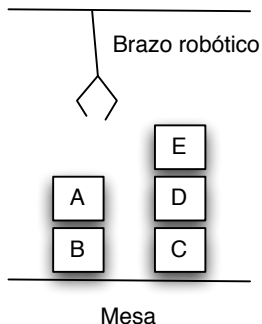
- ▶ El predicado $libre = \{a, e\}$:

$$libre(d) \mapsto false$$

- ▶ **Base relacional** es el conjunto de todos los predicados considerados.



Predicados y Funciones



- ▶ $sobre = base = \{(a, b), \dots\}$
- ▶ Los predicados pueden verse como funciones **booleanas**:

$$sobre(a, b) \mapsto true$$

- ▶ Las funciones en este contexto, tienen su codominio en algún subconjunto de \mathcal{U} :

$$base(a) \mapsto b$$



Extensión de un predicado

- ▶ Para universos de discurso **finitos**, existe un límite superior en el número posible de predicados n -arios.
- ▶ Para un universo de discurso de **cardinalidad** b , existen b^n distintas n -tuplas.
- ▶ Cualquier predicado n -ario es un subconjunto de estas b^n tuplas.
- ▶ Por lo tanto, un predicado n -ario debe corresponder a uno de **máximo** $2^{(b^n)}$ conjuntos posibles.



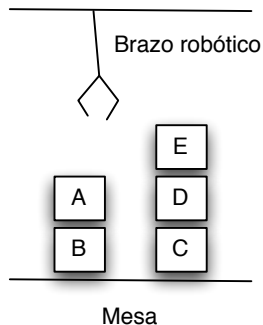
Variables y cuantificadores

- ▶ Toman valores de \mathcal{U} .
- ▶ El cuantificador “**para todo**” (\forall) expresa hechos acerca de todos los objetos en \mathcal{U} , sin necesidad de enumerarlos:

$$\forall X \text{ libre}(X) \Leftarrow \dots$$

- ▶ El cuantificador “**existe**” (\exists) expresa la existencia de un objeto en \mathcal{U} con cierta propiedad en particular:

$$\forall X \exists Y \text{ ocupado}(X) \Leftarrow \text{sobre}(Y, X)$$



Alfabeto de la Lógica de Primer Orden

Const El conjunto de símbolos de constantes;

Var El conjunto de símbolos de variables;

Pred El conjunto de símbolos de predicados;

Func El conjunto de símbolos funcionales ($Const \subset Func$);

↯ El operador unario de negación;

∨ El operador binario de disyunción;

∀ El símbolo de cuantificación universal;

() Los paréntesis.



Reglas sintácticas (términos)

1. Si $\alpha \in Const$ entonces $\alpha \in Term$;
2. Si $\alpha \in Var$, entonces $\alpha \in Term$;
3. Si $\alpha/n \in Func$, entonces $\alpha(\phi_1, \dots, \phi_n) \in Term$ si y sólo si $\phi_{1 \leq i \leq n} \in Term$.
4. Ninguna otra expresión es un término.

▶ Ejemplos:

- ▶ $a, b, c, mesa, \dots$
- ▶ $X, Y, Z, Cubo1, Cubo2, \dots$
- ▶ $base(b), sombrero(X), \dots$



Reglas sintácticas (fórmulas bien formadas)

1. Si $\alpha/n \in Pred$, entonces $\alpha(\phi_0, \dots, \phi_n) \in \mathcal{L}_{FOL}$ si y sólo si $\phi_i \in Term, i = 0 \dots n$;
2. Si $\alpha \in \mathcal{L}_{FOL}$, entonces $\neg\alpha \in \mathcal{L}_{FOL}$;
3. Si $\alpha \in \mathcal{L}_{FOL}$ y $\beta \in \mathcal{L}_{FOL}$, entonces $(\alpha \vee \beta) \in \mathcal{L}_{FOL}$
4. Si $\alpha \in \mathcal{L}_{FOL}$ y $X \in Vars$ es una variable que ocurre en α , entonces $\forall X \alpha \in \mathcal{L}_{FOL}$
5. Nada más es una fórmula bien formada (fbf).

▶ Ejemplos:

- ▶ $sobre(a, b), libre(X), ama(X, hijo(X)), \dots$
- ▶ $\neg libre(a), \neg libre(Y), \dots$
- ▶ $sobre(X, Y) \wedge \neg libre(X), \dots$
- ▶ $\forall X ama(X, hijo(X)), \dots$



Definiciones auxiliares

Conjunción. $(\alpha \wedge \beta) =_{def} \neg(\neg\alpha \vee \neg\beta)$;

Implicación material. $(\alpha \rightarrow \beta) =_{def} (\neg\alpha \vee \beta)$;

Equivalencia material. $(\alpha \equiv \beta) =_{def} ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha))$;

Falso. $f =_{def} \neg\alpha \wedge \alpha$;

Verdadero. $t =_{def} \neg f$

Cuantificador existencial. $\exists X \alpha =_{def} \neg(\forall X \neg\alpha)$



Términos BNF

- ▶ Un término se define como:

$$t ::= x \mid c \mid f(t, \dots, t)$$

donde $x \in Var$; $c \in Func$ tal que $|c| = 0$; y $f \in Func$ tal que $|f| > 0$.



Fórmulas bien formadas BNF

- ▶ Las fbf del lenguaje de la Lógica de Primer Orden se construye a partir de las variables *Var*, los funtores *Func* y los predicados *Pred* como sigue:

$$\phi ::= P(t_1, \dots, t_n) \mid \neg(\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\forall x \phi) \mid (\exists x \phi)$$

donde $P \in Pred$ es un símbolo de predicado de aridad $n \geq 1$; t_i denota términos; y $x \in Var$.



Notación extra

- ▶ En una fbf de la forma $\forall X \alpha$, se dice que la fbf α está bajo el **alcance** del cuantificador $\forall X$.
- ▶ En tal caso, se dice que la ocurrencia de X en α está **acotada**, en caso contrario se dice que la ocurrencia de la variable es **libre**.
- ▶ **Ejemplo:** En $\exists X \text{ sobre}(X, Y)$ la variable X está acotada, mientras que Y está libre.
- ▶ Un término sin variables se conoce como **término de base**.
- ▶ **Ejemplo:** $\text{sobre}(a, b)$.



Interpretación

- ▶ Para expresar que al menos hay un bloque que no tiene nada encima, escribimos: $\exists X \text{ bloque}(X) \wedge \text{libre}(X)$.
- ▶ Cuando usamos cuantificadores siempre tenemos en mente al \mathcal{U} , en este caso $\{a, b, c, d, e, \text{brazo}, \text{mesa}\}$.
- ▶ Una **interpretación** de esta expresión es un subconjunto de \mathcal{U} tal que los miembros de ese subconjunto satisfacen el **significado esperado** de la expresión. En este caso $\{a, e\}$.



Teoría del modelo

- ▶ Para obtener un **modelo** para el lenguaje \mathcal{L}_{FOL} formamos la tupla $M = \langle D, V \rangle$, donde D es el universo de discurso y la interpretación V , con respecto a D , es una función que satisface las siguientes propiedades:
 - ▶ Si $\alpha \in Const$, entonces $V(\alpha) = \alpha$;
 - ▶ Si $\alpha/n \in Pred$, tal que $n \geq 1$, entonces $V(\alpha) \subseteq D^n$.
- ▶ Algunas veces la expresión $V(\alpha)$ se abrevia α^V .
- ▶ **Ejemplos:**
 - ▶ $libre^V \subseteq D = \{a, b, c, d, e\}$.
 - ▶ $sobre^V \subseteq D \times D$.



Interpretación para el mundo de bloques

$$a^V = a$$

$$b^V = b$$

$$c^V = c$$

$$d^V = d$$

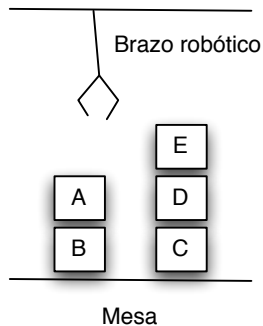
$$e^V = e$$

$$\text{sobre}^V = \{(a, b), (e, d), (d, c)\}$$

$$\text{enLaMesa}^V = \{b, c\}$$

$$\text{libre}^V = \{a, e\}$$

$$\text{porEncima}^V = \{(a, b), (e, d), (e, c), (d, c)\}$$



Asignación de variables y términos

- ▶ Decimos que U es una **asignación de variables** basada en el modelo $M = \langle D, V \rangle$ si para todo $\alpha \in Var$, $U(\alpha) \in D$.
- ▶ La **asignación de términos** T , dadas la interpretación V y la asignación de variables U , es un mapeo de términos a objetos del universo de discurso que se define como sigue:
 - ▶ Si $\alpha \in Const$, entonces $T_{VU}(\alpha) = V(\alpha)$.
 - ▶ Si $\alpha \in Var$, entonces $T_{VU}(\alpha) = U(\alpha)$.
 - ▶ Si $\alpha \in Term$ y es de la forma $\alpha(\phi_1, \dots, \phi_n)$; y $V(\alpha) = g$; y $T_{VU}(\phi_i) = x_i$, entonces $T_{VU}(\alpha(\phi_1, \dots, \phi_n)) = g(x_1, \dots, x_n)$.



Satisfacción

- ▶ El hecho de que el enunciado α se **satisfaga** bajo una interpretación V y una asignación U , se escribe:

$$\models_V \alpha[U]$$

- ▶ Entonces podemos escribir $M \models V_U(\alpha)$ para expresar que α es verdadera en el modelo $M = \langle D, V \rangle$ cuando las variables en α toman valores de acuerdo a la asignación U .
- ▶ **Ejemplo:** $M \models V_U(\text{sobre}(X, b))$ si $X \setminus a \in U$.



Satisfacción:

- Dado un modelo $M = \langle D, V \rangle$ y una asignación de términos T_{VU} :
1. $\models_V (\phi = \psi)[U]$ si y sólo si $T_{VU}(\phi) = T_{VU}(\psi)$.
 2. $\models_V \phi(\tau_1, \dots, \tau_n)[U]$ si y sólo si $(T_{VU}(\tau_1), \dots, T_{VU}(\tau_n)) \in \phi^V$.
 3. $\models_V (\neg\phi)[U]$ si y sólo si $\not\models_V \phi[U]$.
 4. $\models_V (\phi \wedge \psi)[U]$ si y sólo si $\models_V \phi[U]$ y $\models_V \psi[U]$.
 5. $\models_V (\phi \vee \psi)[U]$ si y sólo si $\models_V \phi[U]$ o $\models_V \psi[U]$.
 6. $\models_V (\phi \rightarrow \psi)[U]$ si y sólo si $\not\models_V \phi[U]$ o $\models_V \psi[U]$.
 7. $\models_V (\forall v\phi)[U]$ si y sólo si para todo $d \in D$ es el caso que $\models_V \phi[W]$, donde $\nu^W = d$ y $\mu^W = \mu^U$ para $\mu \neq v$.
 8. $\models_V (\exists v\phi)[U]$ si y sólo si para algún $d \in D$ es el caso que $\models_V \phi[W]$, donde $\nu^W = d$ y $\mu^W = \mu^U$ para $\mu \neq v$.



Definiciones complementarias

- ▶ Si una interpretación V satisface a un enunciado α para toda asignación de variables, se dice que V es un **modelo** de α .
- ▶ Un enunciado se dice **satisfacible** si existe alguna interpretación y asignación de variables que lo satisfaga.
- ▶ De otra forma, se dice que el enunciado es **insatisfacible**.
- ▶ Se dice que una fbf α es **válida**, si y sólo si se satisface en toda interpretación y asignación de variables.
- ▶ Las fbf válidas lo son en virtud de su **estructura lógica**, por lo que no proveen información acerca del dominio descrito.
- ▶ **Ejemplo:** $p(X) \vee \neg p(X)$ es una fbf válida.



Mi mamá me ama

- ▶ Retomemos el ejemplo de la introducción:
 1. Toda madre ama a sus hijos.
 2. Julia es madre y Luis es hijo de Julia.
 3. Julia ama a Luis.

- ▶ Se puede formalizar como:
 1. $\forall X \forall Y \text{ madre}(X) \wedge \text{hijo_de}(Y, X) \rightarrow \text{ama}(X, Y)$
 2. $\text{madre}(\text{julia}) \wedge \text{hijo_de}(\text{luis}, \text{julia})$
 3. $\text{ama}(\text{julia}, \text{luis})$



Reglas de inferencia

- ▶ La inferencia puede verse como un proceso de manipulación de fbf, donde a partir las **premisas**, se producen las **conclusiones**.

Modus Ponens:

$$\frac{\alpha \quad \alpha \rightarrow \beta}{\beta} \quad (\rightarrow E)$$

Eliminación de cuantificador universal:

$$\frac{\forall X \alpha(X)}{\alpha(t)} \quad (\forall E)$$

Introducción de conjunción:

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta} \quad (\wedge I)$$



Universidad Veracruzana

Ejemplo de derivación

- ▶ Inicio:
 1. $\forall X \forall Y \text{madre}(X) \wedge \text{hijo_de}(Y, X) \implies \text{ama}(X, Y)$
 2. $\text{madre}(\text{julia}) \wedge \text{hijo_de}(\text{luis}, \text{julia})$
- ▶ Al aplicar la eliminación de cuantificador universal ($\forall E$) a (1) obtenemos:
 3. $\forall Y (\text{madre}(\text{julia}) \wedge \text{hijo_de}(Y, \text{julia}) \implies \text{ama}(\text{julia}, Y))$
- ▶ Al aplicar nuevamente ($\forall E$) a (3) obtenemos:
 4. $\text{madre}(\text{julia}) \wedge \text{hijo_de}(\text{luis}, \text{julia}) \implies \text{ama}(\text{julia}, \text{luis})$
- ▶ Finalmente, al aplicar Modus Ponens a (2) y (4):
 5. $\text{ama}(\text{julia}, \text{luis})$



Solidez y robustez

- ▶ Un conjunto de reglas de inferencia se dice **sólido** (*sound*), si para todo conjunto de fbf cerradas (sin ocurrencia de variables libres) Δ y cada fbf cerrada α , siempre que $\Delta \vdash \alpha$ se tiene que $\Delta \models \alpha$.
- ▶ Las reglas de inferencia se dicen **completas** si $\Delta \vdash \alpha$ siempre que $\Delta \models \alpha$.



Enunciados declarativos

- ▶ Describen relaciones **positivas** entre elementos de \mathcal{U} :
 - ▶ Incondicionadas (**hechos**) y Condicionadas (**reglas**).

1. Antonio es hijo de Juan.

2. Ana es hija de Antonio.

3. Juan es hijo de Marcos.

4. Alicia es hija de Juan.

5. El nieto de una persona es el hijo del hijo de esa persona.

1. $hijo_de(antonio, juan)$

2. $hijo_de(ana, antonio)$

3. $hijo_de(juan, marcos)$

4. $hijo_de(alicia, juan)$

5. $\forall X \forall Y (nieto_de(X, Y) \leftarrow \exists Z (hijo_de(\bar{Z}, Y) \wedge hijo_de(X, Z)))$



Formas alternativas para una regla

- ▶ La fórmula:

$$\forall X \forall Y (\text{nieto_de}(X, Y) \leftarrow \exists Z (\text{hijo_de}(Z, Y) \wedge \text{hijo_de}(X, Z)))$$

- ▶ Se puede escribir como:

- ▶ $\forall X \forall Y (\text{nieto_de}(X, Y) \vee \neg \exists Z (\text{hijo_de}(Z, Y) \wedge \text{hijo_de}(X, Z)))$

- ▶ $\forall X \forall Y (\text{nieto_de}(X, Y) \vee \forall Z \neg (\text{hijo_de}(Z, Y) \wedge \text{hijo_de}(X, Z)))$

- ▶ $\forall X \forall Y \forall Z (\text{nieto_de}(X, Y) \vee \neg (\text{hijo_de}(Z, Y) \wedge \text{hijo_de}(X, Z)))$

- ▶ $\forall X \forall Y \forall Z (\text{nieto_de}(X, Y) \leftarrow (\text{hijo_de}(Z, Y) \wedge \text{hijo_de}(X, Z)))$

- ▶ Con **equivalencias** como:

- ▶ $\alpha \rightarrow \beta \equiv \neg \alpha \vee \beta$ ó

- ▶ $\forall X \alpha \equiv \neg \exists X \neg \alpha.$



Literales

- ▶ Una **literal** es un átomo o la negación de un átomo.
- ▶ Una **literal positiva** es un átomo.
- ▶ Una **literal negativa** es la negación de un átomo.
- ▶ Ejemplos:
 - ▶ $hijo_de(juan, marcos)$.
 - ▶ $\neg hijo_de(juan, alicia)$.
- ▶ Son los bloques de construcción α_i en:

$$\alpha_0 \leftarrow \alpha_1 \wedge \cdots \wedge \alpha_n \quad (n \geq 0)$$



Cláusulas

- ▶ Una **cláusula** es una disyunción finita de cero o más literales.
- ▶ Una cláusula se dice **definitiva**, si tiene exactamente una literal positiva.

$$\alpha_0 \vee \neg\alpha_1 \vee \cdots \vee \neg\alpha_n \quad (n \geq 0)$$

- ▶ Lo cual es equivalente a la forma general de fbf que nos interesaba:

$$\alpha_0 \leftarrow \alpha_1 \wedge \cdots \wedge \alpha_n \quad (n \geq 0)$$

- ▶ La cláusula vacía (sin literales) se denota como \square y es equivalente a $\square \leftarrow \blacksquare$ (en otra notación alternativa $\perp \leftarrow \top$).



Hechos y reglas revisitados

- ▶ En $\alpha_0 \leftarrow \alpha_1 \wedge \cdots \wedge \alpha_n$ ($n \geq 0$), tenemos que:
 - ▶ Si $n = 0$, la literal α_0 será positiva, por lo que la cláusula definitiva será un **hecho**.
 - ▶ Si $n > 0$ la cláusula definitiva toma la forma de una **regla**, donde α_0 es la **cabeza** de la regla; y la conjunción $\alpha_1 \wedge \cdots \wedge \alpha_n$ su **cuerpo**.
- ▶ Una forma restringida de **cuantificación**:

$$\forall X \forall Y (nieto_de(X, Y) \vee \neg \exists Z (hijo_de(Z, Y) \wedge hijo_de(X, Z))).$$



Programa y Meta Definitivos

- ▶ Un **programa definitivo** es un conjunto finito de cláusulas definitivas.
- ▶ Una cláusula sin literales positivas es una **meta definitiva**.

$$\leftarrow \alpha_1 \wedge \cdots \wedge \alpha_n \quad (n \geq 1)$$



Ejemplo

- Considere las siguientes consultas:

Consulta

Meta definitiva

¿Es Ana hija de Antonio?

← *hijo(ana, antonio)*

¿Quién es nieto de Ana?

← *nieto(X, ana)*

¿De quién es nieto Antonio?

← *nieto(antonio, X)*

¿Quién es nieto de quién?

← *nieto(X, Y)*



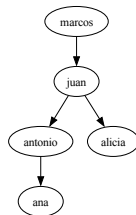
Respuestas obtenidas

Programa:

1. $hijo_de(antonio, juan)$
2. $hijo_de(ana, antonio)$
3. $hijo_de(juan, marcos)$
4. $hijo_de(alicia, juan)$
5. $\forall X \forall Y (nieto_de(X, Y) \leftarrow \exists Z (hijo_de(Z, Y) \wedge hijo_de(X, Z)))$

Metas:

- a. $\leftarrow hijo_de(ana, antonio)$
- b. $\leftarrow nieto_de(X, ana)$
- c. $\leftarrow nieto_de(antonio, X)$
- d. $\leftarrow nieto_de(X, Y)$



Respuestas:

- a. *true.*
- b. *false.*
- c. $X = marcos.$
- d. $X = ana \ Y = juan,$
 $X = antonio \ Y = marcos,$
 $X = alicia \ Y = marcos.$



Cláusulas de Horn

- ▶ Una **cláusula de Horn** es una cláusula ó una meta definitivas.
- ▶ La cláusula vacía \square es una meta definitiva y, por lo tanto, una cláusula de Horn.
- ▶ Las cláusulas de Horn implican **restricciones expresivas**: No podemos expresar $p(a) \vee p(b)$
- ▶ Debido a su estructura restringida, las cláusulas de Horn son **más fáciles de manipular** que las cláusulas generales.



Significado lógico de las metas

- ▶ El significado lógico de las metas puede explicarse recordando que las variables que ocurren en una meta, están cuantificadas **existencialmente**:

$$\neg \exists X_1 \dots X_m (\alpha_1 \wedge \dots \wedge \alpha_n)$$

- ▶ Prolog tratará de encontrar términos $t_1 \dots t_m$ tales que al remplazar X_i por t_i ($1 \leq i \leq m$) en la fbf $(\alpha_1 \wedge \dots \wedge \alpha_n)$, ésta se **satisface**.
- ▶ Una demostración por **contra ejemplo**.



Conocimiento positivo

- ▶ Los programas definitivos solo pueden expresar **conocimiento positivo**.
- ▶ Hechos y reglas nos dicen que objetos del universo de discurso **están** en una relación, pero no nos dicen cuales no.
- ▶ Al usar programas definitivos, no es posible construir **descripciones contradictorias**, i.e., conjuntos de fbf no satisfacibles.
- ▶ Todo programa definitivo **tiene un modelo**.
- ▶ ¿Cual es ese modelo?



Modelos e interpretaciones (recordatorio)

- ▶ Sea α una fbf y V una interpretación. V es un **modelo** de α si $\models_V \alpha$.
- ▶ Sea Δ un conjunto finito de fbf y V una interpretación. V es un modelo de Δ si $\models_V \alpha$ para toda $\alpha \in \Delta$.
- ▶ Una clase interesante de interpretaciones para los programas definitivos se conoce como **interpretaciones de Herbrand**.



Universo y Base de Herbrand

- ▶ Sea L un alfabeto extraído de un programa definitivo Δ , donde $|Const| \geq 1$.
- ▶ El **Universo de Herbrand** (U_L) es el conjunto de todos los términos formados con las constantes y funtores de L .
- ▶ La **Base de Herbrand** (B_L) es el conjunto de todos los átomos que pueden formarse con los predicados y los términos en U_L .



Ejemplo: impar

- ▶ Sea $\Delta = \{\text{impar}(s(0)), \text{impar}(s(s(X))) \leftarrow \text{impar}(X)\}$. Entonces:
 - ▶ $U_L = \{0, s(0), s(s(0)), s(s(s(0))), \dots\}$
 - ▶ $B_L = \{\text{impar}(0), \text{impar}(s(0)), \text{impar}(s(s(0))), \dots\}$



Interpretación de Herbrand

- ▶ Sea Δ un programa definitivo y L el alfabeto compuesto por los símbolos de Δ .
- ▶ V es una interpretación de Herbrand de Δ , si y sólo si:
 - ▶ El dominio de V es U_L .
 - ▶ Para cada constante $c \in L$, $c^V = c$.
 - ▶ Para cada functor $f/n \in L$, $f^V : U_L^n \mapsto U_L$.
 - ▶ Para cada predicado $p/n \in L$, $p^V \subseteq U_L^n$.



Modelo de Herbrand

- ▶ Sea Δ un programa definitivo; L el alfabeto compuesto por los símbolos en Δ ; y V una interpretación de Herbrand de Δ .
- ▶ Si V es un modelo de toda fbf en Δ , se dice que es un **modelo de Herbrand** de Δ .



Ejemplo: impar

- ▶ Sea $\Delta = \{\text{impar}(s(0)), \text{impar}(s(s(X))) \leftarrow \text{impar}(X)\}$.
- ▶ Una posible modelo de este programa es

$$\text{impar}^V = \{s(0), s(s(s(0)))\}$$



Resultados de interés

- ▶ Una interpretación de Herbrand V es un **modelo** de $\forall\alpha$, si α es verdadera en V para todas las posibles asignaciones de las variables en α .
- ▶ Para las cláusulas definitivas:

$$\Delta \models \alpha$$

donde α es una fbf atómica, si y sólo si todo modelo de Herbrand de Δ es también un modelo de α .



Programa definitivo extendido con meta definitiva

- ▶ Sea Δ un programa definitivo y γ una meta definitiva. Sea L un alfabeto compuesto por los símbolos en el programa y la meta definitivos.
- ▶ Si V' es un modelo de $\Delta \cup \{\gamma\}$, entonces $V = \{\alpha \in B_L \mid \models_{V'} \alpha\}$ es un **modelo de Herbrand** de $\Delta \cup \{\gamma\}$.



Consistencia

- ▶ Sea Δ un programa definitivo y α una cláusula definitiva. Sea $\Delta' = \Delta \cup \{\neg\alpha\}$. Entonces $\Delta \models \alpha$ si y sólo si Δ' no tiene modelo de Herbrand.
- ▶ Esto es, si Δ' es no satisfacible, lo cual es cierto sólo si Δ' no tiene modelos y por lo tanto, **no tiene** modelo de Herbrand.



Intersección de modelos de Herbrand

- ▶ Sea M una familia no vacía de modelos de Herbrand de un programa definitivo Δ . Entonces la **intersección** $V = \bigcap M$ es un modelo de Herbrand de Δ .
- ▶ Al tomar la intersección de los modelos de Herbrand de un programa definitivo (todos tienen al menos un modelo, e.g., B_Δ), obtenemos el **modelo mínimo de Herbrand**.



Ejemplo

- ▶ Sea Δ el programa definitivo $\{masculino(adan), femenino(eva)\}$ con su interpretación obvia.
- ▶ Δ tiene los siguientes **modelos de Herbrand**:
 1. $\{masculino(adan), femenino(eva)\}$
 2. $\{masculino(adan), femenino(eva), masculino(eva)\}$
 3. $\{masculino(adan), femenino(eva), femenino(adan)\}$
 4. $\{masculino(adan), femenino(eva), masculino(eva), femenino(adan)\}$
- ▶ El modelo mínimo es el único que corresponde con el modelo pretendido del programa, pero esto **no siempre es el caso**.
- ▶ ¿Cuándo lo es?



Consecuencia Lógica

- ▶ El modelo mínimo de Herbrand de un programa definitivo Δ , M_Δ , es el conjunto de todas las **consecuencias lógicas atómicas de base** del programa. Esto es: $M_\Delta = \{\alpha \in B_\Delta \mid \Delta \models \alpha\}$.
- ▶ La prueba (Ver Nilsson y Maluszynski [3], pág. 29) pasa por demostrar que $M_\Delta \supseteq \{\alpha \in B_L \mid \Delta \models \alpha\}$ y $M_\Delta \subseteq \{\alpha \in B_\Delta \mid \Delta \models \alpha\}$.



Operador de consecuencia inmediata

- ▶ Sea $base(\Delta)$ el conjunto de todos los casos posibles de cláusulas de base en Δ . T_Δ es una función sobre las interpretaciones de Herbrand de Δ definida como sigue:

$$T_\Delta(V) = \{\alpha_0 \mid \alpha_0 \leftarrow \alpha_1, \dots, \alpha_n \in base(\Delta) \wedge \{\alpha_1, \dots, \alpha_n\} \subseteq V\}$$

- ▶ Se puede demostrar que existe un **punto fijo** $T_\Delta(V) = V$ y que V es idéntica al modelo mínimo de Herbrand de Δ .



Notación

- ▶ Existe una notación estándar para denotar a los miembros de esta secuencia de interpretaciones construidas a partir de Δ :

$$\begin{aligned}T_{\Delta} \uparrow 0 &= \emptyset \\T_{\Delta} \uparrow (i + 1) &= T_{\Delta}(T_{\Delta} \uparrow i) \\T_{\Delta} \uparrow n &= \bigcup_{i=0}^{\infty} T_{\Delta} \uparrow i\end{aligned}$$

- ▶ el conjunto construido de esta manera es idéntico al modelo mínimo de Herbrand de Δ .



Ejemplo

- ▶ Tomando Δ como el programa de *impar*/1 tenemos:

$$T_{\Delta} \uparrow 0 = \emptyset$$

$$T_{\Delta} \uparrow 1 = \{\textit{impar}(s(0))\}$$

$$T_{\Delta} \uparrow 2 = \{\textit{impar}(s(0)), \textit{impar}(s(s(s(0))))\}$$

$$\vdots$$

$$T_{\Delta} \uparrow m = \{\textit{impar}(s^n(0)) \mid n \in \{1, 3, 5, \dots\}\}$$



Programas y metas definitivos

- ▶ Consideren el siguiente **programa definitivo** Δ :

papa(juan, marta).

recien_nacido(marta).

orgullosa(X) ← padre(X, Y), recien_nacido(Y).

padre(X, Y) ← papa(X, Y).

padre(X, Y) ← mama(X, Y).

- ▶ ¿Cómo interpretamos la siguiente **meta definitiva**?

\leftarrow *orgullosa(Z).*



Metas

- ▶ La meta $\leftarrow orgullososo(Z)$. es una abreviatura de $\forall Z \neg orgullososo(Z)$
- ▶ Que a su vez es equivalente a $\neg \exists Z orgullososo(Z)$.
- ▶ Si ese enunciado es contradictorio en Δ , entonces sabemos que $\Delta \models \exists Z orgullososo(Z)$. Eso respondería *true*, a la meta.
- ▶ La otra parte de la respuesta es encontrar una **substitución** θ (asignación de variables) t.q. el conjunto $\Delta \cup \{\neg orgullososo(Z)\theta\}$ sea insatisfacible, i.e., $\Delta \models orgullososo(Z)\theta$.
- ▶ **Ejemplo:** $\Delta \cup \neg orgullososo(Z)\{Z/juan\}$



Razonamiento

- ▶ Asumimos la meta G_0 – Para todo Z , Z no está orgulloso.
- ▶ La inspección del programa Δ revela que una regla describe una condición para que alguien esté orgulloso:

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

- ▶ Lo cual es lógicamente equivalente a:

$$\forall (\neg \text{orgullosa}(X) \rightarrow \neg (\text{padre}(X, Y) \wedge \text{recien_nacido}(Y)))$$



Estrategia

- ▶ Partiendo de:

$$\forall(\neg orgullosa(X) \rightarrow \neg(padre(X, Y) \wedge recién_nacido(Y)))$$

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

$$\neg(padre(Z, Y) \wedge recién_nacido(Y))$$

- ▶ o su equivalente:

$$\leftarrow padre(Z, Y), recién_nacido(Y).$$

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



Resolución

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

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

- G_1 no es satisfacible en Δ , si en todo modelo de Δ hay una persona que es padre de un recién nacido.
- Revisemos la primer condición:

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

- Por lo que G_1 se reduce a G_2 :

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



Estrategia 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 Δ :

\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.



Regla de inferencia

- La resolución puede resumirse con la siguiente regla de inferencia:

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

- 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 Δ ($n \geq 0$).
3. $MGU(\alpha_j, \beta_0) = \theta$.



Observaciones

- ▶ La regla tiene dos **premisas**: una meta y una cláusula definitivas.
- ▶ 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.



S de selección

- ▶ Una meta definitiva puede incluir **varias** fbf atómicas que unifican con la cabeza de alguna cláusula en el programa.
- ▶ En este caso, es deseable contar con un mecanismo **determinista** para seleccionar un átomo α_j a unificar.
- ▶ Se asume una **función de selección** para las sub-metas de la meta definitivas.
- ▶ **Ejemplo.** De izquierda a derecha.



Usando la resolución-SLD

- ▶ El punto de partida es una **meta definitiva** G_0 :

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

- ▶ La **sub-meta** α_1 será seleccionada.
- ▶ Una **nueva meta** G_1 se construye con una cláusula del programa $\beta_0 \leftarrow \beta_1, \dots, \beta_n$ ($n \geq 0$) tal que $\theta_1 = MGU(\alpha_1, \beta_0)$.
- ▶ G_1 tiene la forma:

$$\leftarrow (\beta_1, \dots, \beta_n, \alpha_2, \dots, \alpha_m)\theta_1$$

- ▶ 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 sub-meta seleccionada no puede ser resuelta (no unifica con la cabeza de alguna cláusula del programa).
 2. Cuando $G_i = \square$.
- ▶ En cualquier otro caso la resolución-SLD se sigue aplicando.



Derivación-SLD

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

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

- ▶ $\alpha_i \in \Delta$. Las variables en α_i 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**:

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

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

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

- ▶ 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 .

- ▶ 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**.

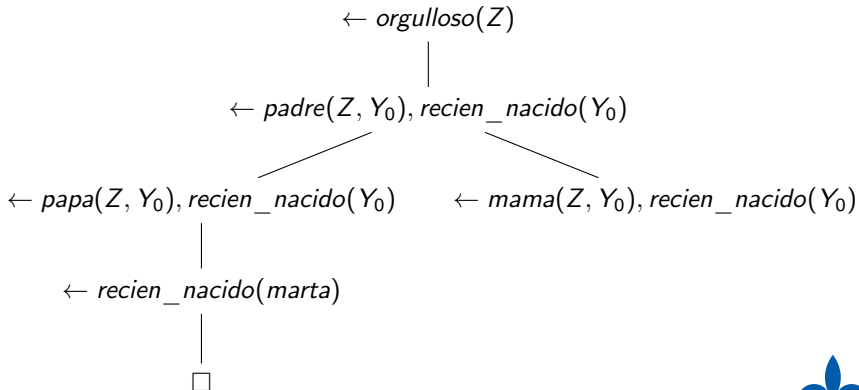


Arbol-SLD

- ▶ Sea Δ un programa definitivo, G_0 una meta definitiva, y \mathcal{R} una función de selección.
- ▶ El **árbol-SLD** de G_0 (usando Δ 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 $\alpha_j \in \Delta$ tal que G_{i+1} es derivada de G_i y α_j via \mathcal{R} , entonces el nodo etiquetado como G_i tiene un hijo etiquetado G_{i+1} . El arco entre ambos es α_j .



Ejemplo



Observaciones

- ▶ Esta computación no necesariamente es **determinista** si \mathcal{R} es justa, i.e., cualquier átomo de la meta y cláusula del programa pueden ser seleccionados.
- ▶ Pueden existir unificadores **alternativos** para dos átomos.
- ▶ Por otra parte, es posible también que el átomo seleccionado **no unifique** con ninguna cláusula en el programa.
- ▶ La computación puede caer en un **ciclo** y de esta manera no producir solución alguna.



Propiedades de la resolución-SLD

- Correctez.** Sea Δ un programa definitivo, \mathcal{R} una función de selección, y θ una sustitución de respuesta computada a partir de Δ y \mathcal{R} para una meta $\leftarrow \alpha_1, \dots, \alpha_m$. Entonces $\forall((\alpha_1 \wedge \dots \wedge \alpha_m)\theta)$ es consecuencia lógica del programa Δ .
- Completez.** Sea Δ 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 sustitución de respuesta computada θ , tal que $(\alpha_1 \wedge \dots \wedge \alpha_m)\sigma$ es un caso de $(\alpha_1 \wedge \dots \wedge \alpha_m)\theta$.



Substitución

- ▶ Una **substitución** θ 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 θ se dice se dice **de base** (*grounded*) si cada término t_i es un término base.

- ▶ La **substitución vacía** se conoce como **substitución de identidad** y se denota por ϵ .



Caso (expresiones)

- ▶ Sea $\theta = \{X_1/t_1, \dots, X_n/t_n\}$ una sustitución y α una expresión. Entonces $\alpha\theta$, el **caso** (*instance*) de α bajo θ , es la expresión obtenida al sustituir simultáneamente X_i por t_i para $1 \leq i \leq n$.
- ▶ Si $\alpha\theta$ es una expresión de base, se dice que es un **caso base** y se dice que θ es una sustitución de base para α .
- ▶ 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 α la expresión $p(Y, f(X))$ y
- ▶ la substitución $\theta = \{X/a, Y/g(g(X))\}$.
- ▶ El caso de α bajo θ es $\alpha\theta = p(g(g(X)), f(a))$.
- ▶ Observen que X e Y son remplazados **simultáneamente** por sus respectivos términos, i.e., X en $g(g(X))$ no es afectada por X/a .



Composición de sustituciones

- ▶ Sean $\theta = \{X_1/s_1, \dots, X_m/s_m\}$ y $\sigma = \{Y_1/t_1, \dots, Y_n/t_n\}$ dos sustituciones. 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 $Y_j \in \{X_1, \dots, X_m\}$, obtenemos la **composición** de θ y σ denotada 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
 $X/f(b), Y/b, U/Z = \theta\sigma$.



Propiedades de las sustituciones

- ▶ Sea α una expresión, y sean θ , σ y γ sustituciones.
- ▶ 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.
- ▶ Sean α y β términos. Una substitución θ tal que $\alpha\theta = \beta\theta$ es llamada **unificador** de α y β .
- ▶ $unifica(conoce(juan, X), conoce(Y, Z)) = \{Y/juan, X/Z\}$



Unificador más general (MGU)

- ▶ Una sustitución θ se dice **más general** que una sustitución σ , ssi existe una sustitución γ tal que $\sigma = \theta\gamma$.
- ▶ Un unificador θ se dice el **unificador más general** (MGU: *Most General Unifier*) de dos términos, ssi θ es más general que cualquier otro unificador entre esos términos.



Forma resuelta y MGU

- ▶ 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 .
- ▶ Para computar el MGU de dos términos α y β , primero intente transformar la ecuación $\{\alpha = \beta\}$ en una forma resuelta equivalente.
- ▶ Si esto falla, entonces $mgu(\alpha, \beta) = \text{fallo}$.
- ▶ 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 acción 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 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 estándar 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 $X = f(f(X)) \dots = f(f(f \dots))$.



Referencias I

- [1] MR Genesereth y NJ Nilsson. *Logical Foundations for Artificial Intelligence*. Palo Alto, CA, USA: Morgan Kauffman Publishers, Inc., 1987.
- [2] RA Kowalski y D Kuehner. "Linear Resolution with Selection Function". En: *Artificial Intelligence* 2.3/4 (1971), págs. 227-260.
- [3] U Nilsson y J Maluszynski. *Logic, Programming and Prolog*. 2nd. John Wiley & Sons Ltd, 2000.
- [4] JA Robinson. "A Machine-Oriented Logic based on the Resolution Principle". En: *Journal of the ACM* 12.1 (1965), págs. 23-41.
- [5] SJ Russell y P Norvig. *Artificial Intelligence: A Modern Approach*. Third. Prentice Hall Series in Artificial Intelligence. USA: Prentice Hall, 2009.

