

## **Modelli minimi e programmazione logica**

Marco Piastra

# Clausole di Horn in $L_{PO}$

- Definizione quasi identica al caso proposizionale

Forma a clausole (della skolemizzazione di un insieme di enunciati)

In ciascuna clausola occorre al massimo un atomo in forma positiva

## Fatti, regole e goal

**Fatti:** clausola con un singolo atomo in forma positiva

$\{Umano(socrate)\}, \{Pyramid(x)\}, \{Sorella(alba, madreDi(paolo))\}$

**Regole:** clausola di due o più atomi, uno in forma positiva

$\{Umano(x), \neg Filosofo(x)\},$

$\forall x (Filosofo(x) \rightarrow Umano(x))$

$\{\neg Femmina(x), \neg Genitore(k(x),x), \neg Genitore(k(y),y), Sorella(x,y)\}$

$\forall x \forall y ((Femmina(x) \wedge \exists z (Genitore(z,x) \wedge Genitore(z,y))) \rightarrow Sorella(x,y))$

$\{\neg Above(x,y), On(x,k(x))\}, \{\neg Above(x,y), On(j(y),y)\}$

$\forall x \forall y (Above(x,y) \rightarrow (\exists z On(x,z) \wedge \exists v On(v,y)))$

**Goal:** clausola di atomi in forma negativa

$\{\neg Umano(socrate)\}$

$\{\neg Sorella(alba,x), \neg Sorella(x,paola)\}$

Negazione di  $\exists x (Sorella(alba,x) \wedge Sorella(x,paola))$

# Universo e base di Herbrand

## ■ Termini e atomi di Herbrand

Dato un linguaggio  $L_{PO}$

Un **termine** di Herbrand è un *termine base* (*ground term* = che non contiene variabili)

Esempi:

$f(a), g(a,b), g(f(a),b), g(f(a),g(b,c)), g(f(a),g(f(b),c)), \dots$

Un **atomo** di Herbrand è un *atomo base* (*ground atom* = che non contiene variabili)

Esempi:

$P(f(a)), P(g(a,b)), Q(g(f(a),b), g(f(a),g(b,c))), \dots$

## ■ Universo e base di Herbrand

L'**universo** di Herbrand è l'insieme di tutti i termini di Herbrand

Esempio:

$U_H \equiv \{f(a), g(a,b), g(f(a),b), g(f(a),g(b,c)), g(f(a),g(f(b),c)), \dots\}$

La **base** di Herbrand è l'insieme di tutti gli atomi di Herbrand

Esempio:

$B_H \equiv \{P(f(a)), P(g(a,b)), Q(g(f(a),b), g(f(a),g(b,c))), \dots\}$

# Modelli di Herbrand

## ▪ **Struttura di Herbrand per $L_{PO}$**

Una struttura  $\langle \mathbf{U}_H, v_H \rangle$  tale che

$$\forall c \in \text{Cost}(L_{PO}), v_H(c) = c$$

$$\forall t \in \mathbf{U}_H, v_H(t) = t$$

## ▪ **Interpretazione $v_H$ di Herbrand**

Un qualsiasi **sottoinsieme** della base di Herbrand  $B_H$

$$v_H \equiv \{ P(a), P(f(b)), P(c), Q(a, g(b, c)), Q(b, c) \dots \} \quad (\text{solo formule atomiche chiuse})$$

$$v_H \subseteq B_H$$

## ▪ **Modello di Herbrand**

$$\varphi \in \text{Atom}(L_{PO}), \langle \mathbf{U}_H, v_H \rangle [s] \models \varphi \quad \text{sse } \varphi \in v_H$$

$$\varphi \in \text{Atom}(L_{PO}), \langle \mathbf{U}_H, v_H \rangle [s] \models \neg \varphi \quad \text{sse } \varphi \notin v_H$$

$$\langle \mathbf{U}_H, v_H \rangle [s] \models \neg \varphi \quad \text{sse } \langle \mathbf{U}_H, v_H \rangle [s] \not\models \varphi$$

$$\langle \mathbf{U}_H, v_H \rangle [s] \models \varphi \rightarrow \psi \quad \text{sse } (\langle \mathbf{U}_H, v_H \rangle [s] \not\models \varphi \text{ o } \langle \mathbf{U}_H, v_H \rangle [s] \models \psi)$$

$$\langle \mathbf{U}_H, v_H \rangle [s] \models \forall x \varphi \quad \text{se per ogni } c \in \text{Cost}(L_{PO}) \text{ si ha } \langle \mathbf{U}_H, v_H \rangle [s](x:c) \models \varphi$$

# Clausole di Horn e modelli di Herbrand

- **Teorema di Herbrand (in forma generale)**

Data una teoria di enunciati universali  $\Sigma$ ,  
 $H(\Sigma)$  ha un modello sse  $\Sigma$  ha un modello

- Corollario (forma a clausole di Horn)

Sia  $\Gamma$  un insieme di clausole di Horn, le seguenti affermazioni sono equivalenti:

- $\Gamma$  è soddisfacibile
- $\Gamma$  ha un modello di Herbrand  
(Notare: si afferma che  $\Gamma$  ha un modello di Herbrand, non  $H(\Gamma)$ )

Non vale in generale: solo se  $\Gamma$  è un insieme clausole di Horn

In questa forma (finita), è quasi una procedura effettiva ...

# Clausole di Horn e modelli di Herbrand

## ■ Corollario del teorema di Herbrand

Sia  $\Gamma$  un insieme di clausole di Horn, le seguenti affermazioni sono equivalenti:

- $\Gamma$  è soddisfacibile
- $\Gamma$  ha un modello di Herbrand

Non vale in generale: solo se  $\Gamma$  è un insieme clausole di Horn

## ■ **Modello minimo** di Herbrand

Il modello minimo  $M_\Gamma$  è l'intersezione di tutti i modelli di Herbrand  $M_i$  di  $\Gamma$ :

$$M_\Gamma \equiv \bigcap_{M_i} M_i$$

## ■ Teorema (van Emden e Kowalski, 1976)

Sia  $\Gamma$  un insieme di clausole di Horn e  $\varphi$  un atomo base, le seguenti affermazioni sono equivalenti:

- $\Gamma \models \varphi$
- $\varphi \in M_\Gamma$  (L'unione degli atomi base  $\varphi$  che sono conseguenza logica di  $\Gamma$  coincide con  $M_\Gamma$ )
- $\varphi$  è derivabile da  $\Gamma$  tramite risoluzione con unificazione

# Programmi e modello minimo

## ■ Teorema (Apt e van Emden, 1982)

Sia  $\Pi$  un **programma** (= un insieme di clausole di Horn).

Applicata a  $\Pi$ , la procedura di risoluzione genera il modello minimo  $M_\Pi$

La procedura termina se  $M_\Pi$  è finito (raggiungimento del *punto fisso*)

Esempio:

$$\Pi \equiv \{ \{Umano(x), \neg Filosofo(x)\}, \{Mortale(x), \neg Umano(x)\}, \\ \{Filosofo(socrate)\}, \{Filosofo(platone)\}, \{Filosofo(aristotele)\} \}$$

Applicando la procedura di risoluzione in modo esaustivo, si ottiene:

$$M_\Pi \equiv \{ \{Mortale(x), \neg Filosofo(x)\}, \\ \{Filosofo(socrate)\}, \{Filosofo(platone)\}, \{Filosofo(aristotele)\}, \\ \{Umano(socrate)\}, \{Umano(platone)\}, \{Umano(aristotele)\}, \\ \{Mortale(socrate)\}, \{Mortale(platone)\}, \{Mortale(aristotele)\} \}$$

(assomiglia alla generazione di un database, *implicitamente descritto* da  $\Pi$ ...)

# Programmi e goal

Un dimostratore di teoremi, applicato ad un programma logico  $\Pi$ , risponde solo a domande del tipo “ $\Pi \models \phi$  ?”

Si rammenti che, se  $\Pi \models \phi$ , allora  $\Pi \cup \{\neg\phi\}$  è insoddisfacibile

- Un sistema di programmazione logica è in grado di generare un particolare sottoinsieme di  $M_\Pi$

Un goal  $\{\neg\alpha_1, \neg\alpha_2, \dots, \neg\alpha_m\}$ , dove occorrono le variabili  $x_1, x_2, \dots, x_m$  equivale all'enunciato  $\forall x_1 \forall x_2 \dots \forall x_n (\neg\alpha_1 \vee \neg\alpha_2 \vee \dots \vee \neg\alpha_m)$  che equivale a  $\neg\exists x_1 \exists x_2 \dots \exists x_n (\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_m)$

Un sistema di programmazione logica genera tutte le **sostituzioni**

$[x_1/t_1, x_2/t_2, \dots, x_n/t_n]$  tali per cui  $\Pi \cup \{\neg(\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_m)[x_1/t_1, x_2/t_2, \dots, x_n/t_n]\}$  è insoddisfacibile

(vale a dire  $\Pi \models (\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_m)[x_1/t_1, x_2/t_2, \dots, x_n/t_n]$ )

(vale a dire  $(\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_m)[x_1/t_1, x_2/t_2, \dots, x_n/t_n] \in M_\Pi$ )

Il goal agisce da filtro, caratterizzando il sottoinsieme di  $M_\Pi$

*Goal diverso, sottoinsieme diverso*



# Esempio

- Un programma logico  $\Pi$ :

$$\Pi \equiv \{ \{Umano(x), \neg Filosofo(x)\}, \{Mortale(y), \neg Umano(y)\}, \\ \{Filosofo(socrate)\}, \{Filosofo(platone)\}, \{Filosofo(aristotele)\} \}$$

$$\phi \equiv \exists x Mortale(x)$$

$$\neg\phi \equiv \neg\exists x Mortale(x)$$

$$\equiv \forall x \neg Mortale(x)$$

$$\equiv \{ \neg Mortale(x) \} \quad (\text{goal in forma di clausola di Horn})$$

Applicando la procedura di risoluzione in modo esaustivo

Si ottengono le sostituzioni:

$$\Sigma \equiv \{ [x/socrate], [x/platone], [x/aristotele] \}$$

Assomiglia alla query su un database, *implicito* ...

# Risoluzione SLD

- Un metodo per la risoluzione di programmi

*S*: *selection function*, una funzione di selezione degli atomi da unificare

*L*: *linear resolution*, risoluzione lineare, cioè in sequenza

*D*: *definite clause*, clausole di Horn con esattamente un letterale positivo

- Descrizione

Programma (*definite clauses*: regole + fatti):  $\Pi$

Regole:  $\beta \vee \neg\gamma_1 \vee \neg\gamma_2 \vee \dots \vee \neg\gamma_n$

Fatti:  $\delta$

Goal:  $\neg\alpha_1 \vee \neg\alpha_2 \vee \dots \vee \neg\alpha_k$

Caratteristiche della procedura:

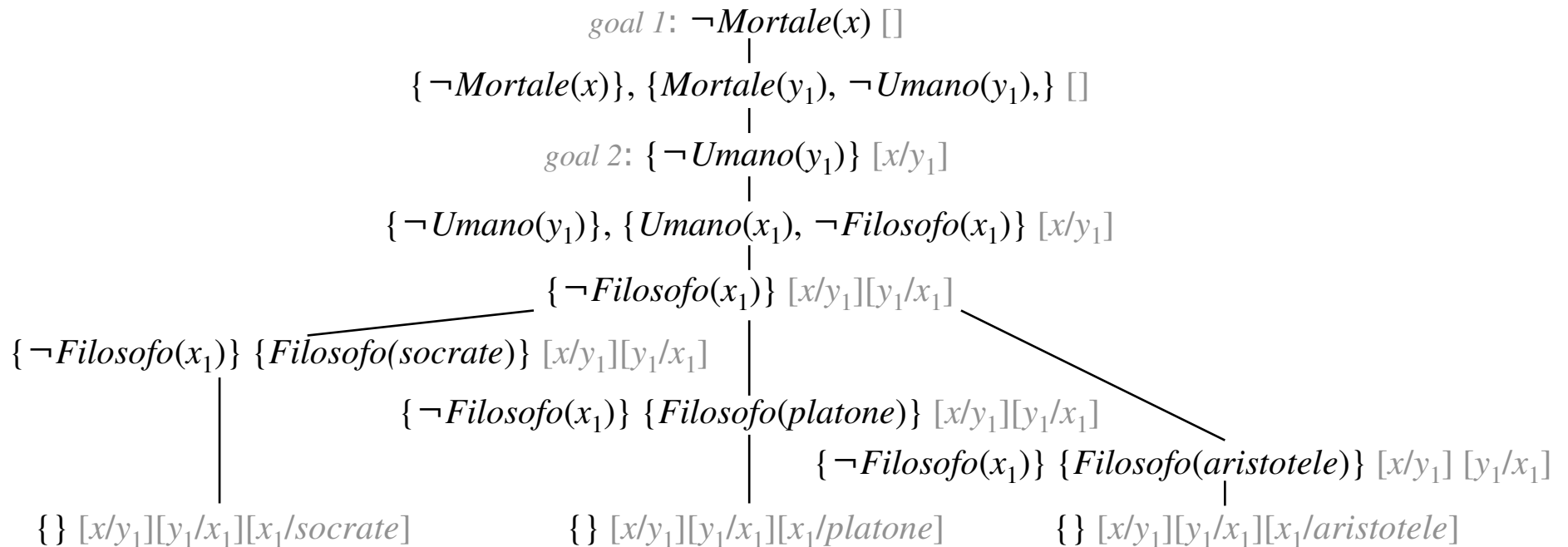
- I goal vengono considerati secondo l'ordine definito dalla *selection function*
- Per ciascun goal  $\neg\alpha_i$  viene tentata la risoluzione (con unificazione) di tutte le regole (o fatti) che hanno un letterale positivo compatibile (*esplorazione delle alternative*)
- Le risposte sono le assegnazioni che permettono di derivare la clausola vuota
- L'insieme delle risposte è un sottoinsieme di  $M_{\Pi}$

# Alberi SLD

## ■ Una traccia del metodo di risoluzione SLD

Esempio:

$\Pi \equiv \{ \{Umano(x), \neg Filosofo(x)\}, \{Mortale(y), \neg Umano(y)\},$   
 $\{Filosofo(socrate)\}, \{Filosofo(platone)\}, \{Filosofo(aristotele)\} \}$   
 $goal \equiv \{ \neg Mortale(x), \neg Umano(x) \}$       "Chi è mortale ed umano?"

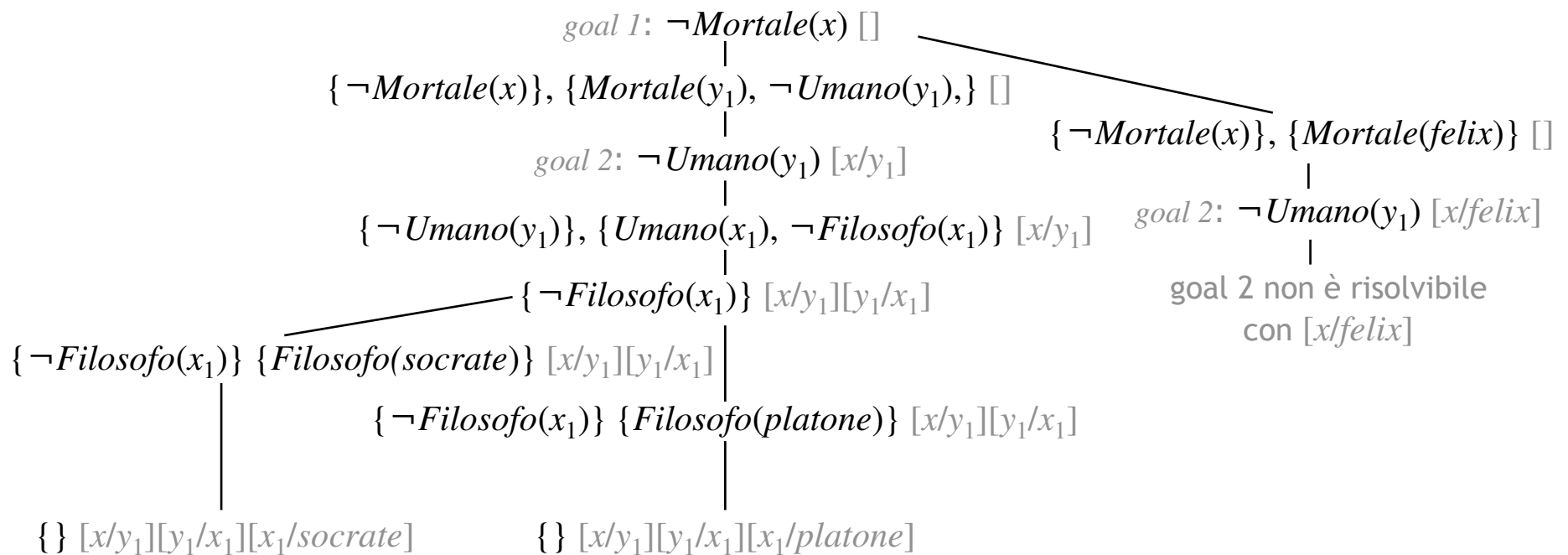


# Esempio

- Non tutti i rami SLD si chiudono con successo

$\Pi \equiv \{ \{Umano(x), \neg Filosofo(x)\}, \{Mortale(y), \neg Umano(y)\},$   
 $\{Filosofo(socrate)\}, \{Filosofo(platone)\}, \{Mortale(felix)\} \}$

$goal \equiv \{ \neg Mortale(x), \neg Umano(x) \}$       "Chi è mortale ed umano?"

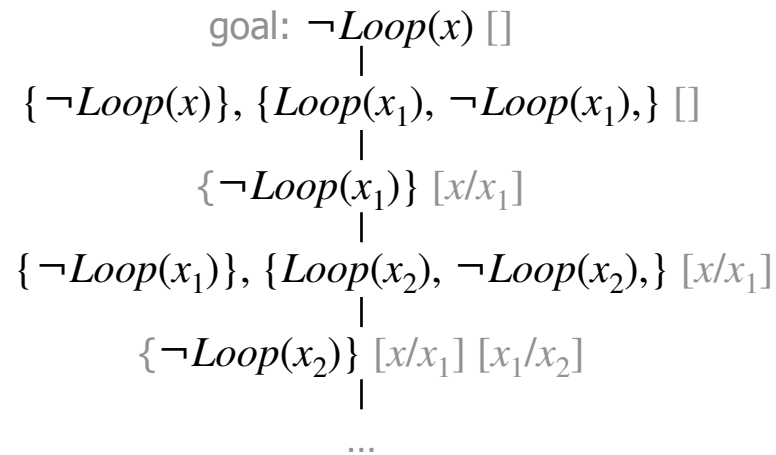


# Esempio

- Non tutti gli alberi SLD sono finiti

$\Pi \equiv \{\{Loop(x), \neg Loop(x)\}\}$

$goal \equiv \{\neg Loop(x)\}$



# SLD e programmazione logica

## ▪ Insieme delle risposte

Insieme di tutte le sostituzioni complete delle variabili, nei rami dell'albero SLD che si chiudono con successo (= con una clausola vuota)

## ▪ Metodo effettivo (*semantica procedurale*)

*Selection function* delle clausole

Si usa (quasi) sempre la *leftmost sub-goal first*, con sostituzione del *sub-goal*

*Strategia di esplorazione* delle alternative

- in **ampiezza** (*breadth-first*)
- in **profondità** (*depth-first*)

Il metodo SLD con selezione in *ampiezza* è **completo** (si dice anche SLD **fair**)

Trova tutti i rami finiti (con successo o meno) dell'albero SLD  
(= *procedura completa di semi-decisione per  $\Pi \models \phi$* , con  $\Pi$  e  $\phi$  a clausole)

In pratica si utilizza la selezione in *profondità*

(Il metodo SLD non è completo - può divergere anche quando  $\Pi \models \phi$ )

# Risoluzione SLD in Prolog

## Metodo effettivo

*Selection function: leftmost sub-goal first*

Esplorazione *depth-first* delle alternative

Si esplora una sola alternativa alla volta, e si risparmia memoria (*backtracking*)

E' una strategia **incompleta**:

Un ramo divergente impedisce di trovare tutte le risposte dei rami 'alla destra'

Scelta tra risoluzioni alternative

(= *ordine di esplorazione dei sotto-alberi*)

Ordine di definizione della clausola applicata

( $\approx$  quella che compare prima nel file)

Il metodo SLD *depth-first* non troverà la risposta  $\sigma_l$

