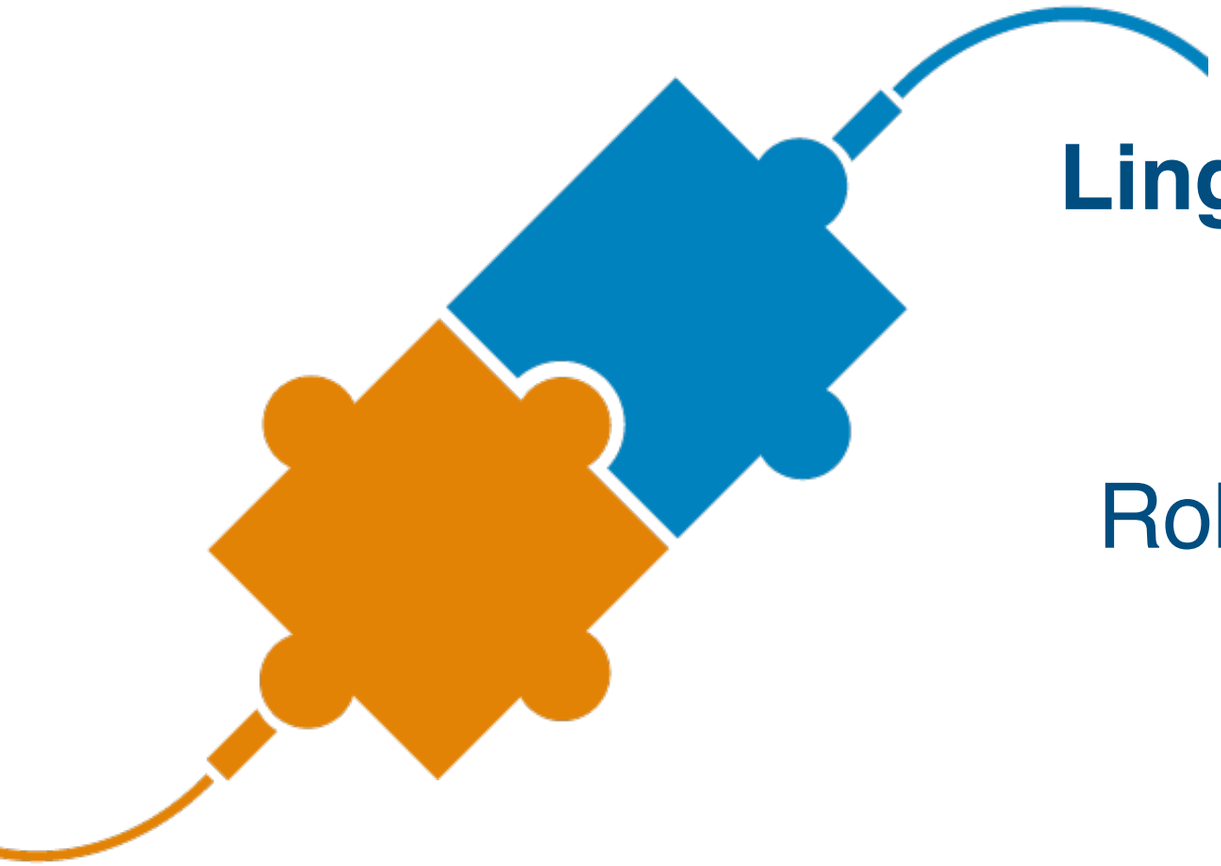


Linguaggi di Programmazione

Roberta Gori

Systemi Logici 2.2

Regole di Inferenza

Regole di inferenza

premesse (una, nessuna, molte)

$$\frac{x_1 \quad \cdots \quad x_n}{y}$$

conclusioni (una)

se le premesse sono valide, allora anche la conclusione è valida

$x_1, ..., x_n, y$ sono formule

ogni variabile è (implicitamente) quantificata universalmente

l'istanza di una regola è ottenuta applicando una sostituzione ρ a $x_1, ..., x_n, y$

Assiomi

senza premesse

$$\frac{}{y}$$

la conclusione e' valida

conclusione (sempre valida, e' un fatto)

Istanza di una regola

$$(prod)\frac{\mathsf{E}_0 \longrightarrow n_0 \quad \mathsf{E}_1 \longrightarrow n_1}{\mathsf{E}_0 \otimes \mathsf{E}_1 \longrightarrow n} \; n = n_0 \cdot n_1$$

un istanza di prod

$$(prod)\frac{1 \longrightarrow 1 \quad 1 \oplus 2 \longrightarrow 3}{1 \otimes (1 \oplus 2) \longrightarrow 3} \; 3 = 1 \cdot 3$$

c’e anche un altra istanza di (*prod*)!

$$(prod)\frac{1 \longrightarrow 3 \quad 1 \oplus 2 \longrightarrow 5}{1 \otimes (1 \oplus 2) \longrightarrow 15} \; 15 = 3 \cdot 5$$

ma sara’ difficile che le premesse siano valide

Altre istanze

$$(prod)\,\frac{\mathsf{E}_0 \longrightarrow n_0 \quad \mathsf{E}_1 \longrightarrow n_1}{\mathsf{E}_0 \otimes \mathsf{E}_1 \longrightarrow n} \; n = n_0 \cdot n_1$$

un istanza di (*prod*)

$$(prod)\,\frac{\mathsf{E} \otimes 2 \longrightarrow k \quad \mathsf{E} \oplus 1 \longrightarrow 3}{(\mathsf{E} \otimes 2) \otimes (\mathsf{E} \oplus 1) \longrightarrow 3k}$$

possiamo avere la stessa variabile

Sistema Logico

$$R = \left\{ \frac{}{z}, \frac{x_1 \quad \cdots \quad x_n}{y}, \ldots \right\}$$

Un sistema logico e' un insieme di assiomi e regole di inferenza

se una regola di inferenza contiene alcune variabili, assumiamo che tutte le sue istanze siano in R

Una derivazione

Dato un sistema logico R, una derivazione in R, si scrive

$$d \Vdash_R y$$

dove

- o $d = (\frac{}{y})$ è un assioma di R;
- oppure $d = (\frac{d_1,\ldots,d_n}{y})$ ed esistono n derivazioni $d_1 \Vdash_R x_1, \ \ldots\ldots, \ d_n \Vdash_R x_n$ e $(\frac{x_1,\ldots,x_n}{y})$ è una regola di inferenza di R.

una derivazione e' un albero di prova (le cui foglie sono assiomi)

Esempio

S ::= ϵ | (S) | S S

$$R = \left\{ \frac{}{\epsilon \in \mathcal{L}}, \frac{S \in \mathcal{L}}{(\ S\ ) \in \mathcal{L}}, \frac{S_0 \in \mathcal{L} \quad S_1 \in \mathcal{L}}{S_0\ S_1 \in \mathcal{L}} \right\}$$

$$d \Vdash_R (()(())) \in \mathcal{L}$$

$$(()(())) \in \mathcal{L}$$

Esempio

$$R = \left\{ \frac{}{\mathsf{N} \longrightarrow n}, \frac{\mathsf{E}_0 \longrightarrow n_0 \quad \mathsf{E}_1 \longrightarrow n_1}{\mathsf{E}_0 \oplus \mathsf{E}_1 \longrightarrow n_0 + n_1}, \frac{\mathsf{E}_0 \longrightarrow n_0 \quad \mathsf{E}_1 \longrightarrow n_1}{\mathsf{E}_0 \otimes \mathsf{E}_1 \longrightarrow n_0 \cdot n_1} \right\}$$

$$d = \dfrac{\dfrac{\dfrac{}{1 \longrightarrow 1} \quad \dfrac{}{2 \longrightarrow 2}}{(1 \oplus 2) \longrightarrow 3} \quad \dfrac{\dfrac{}{3 \longrightarrow 3} \quad \dfrac{}{4 \longrightarrow 4}}{(3 \oplus 4) \longrightarrow 7}}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21}$$

$$d \Vdash_R (1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$$

Teoremi

Dato un sistema logico *R*, un teorema di *R* e' scritto nel seguente modo:

$$\Vdash_R y \qquad\qquad \exists d.\ d \Vdash_R y$$

dove *y* e' una formula tale che possiamo trovare una derivazione per *y* in *R*

L'insieme di tutti i teoremi di R è denotato con I_R.

$$I_R \triangleq \{\, y \mid \Vdash_R y \,\}$$

Notazione inline

$$d = \dfrac{\dfrac{\dfrac{}{1 \longrightarrow 1} \quad \dfrac{}{2 \longrightarrow 2}}{(1 \oplus 2) \longrightarrow 3} \quad \dfrac{\dfrac{}{3 \longrightarrow 3} \quad \dfrac{}{4 \longrightarrow 4}}{(3 \oplus 4) \longrightarrow 7}}{(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21}$$

$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$

$\nwarrow \quad (1 \oplus 2) \longrightarrow 3 \;,\; (3 \oplus 4) \longrightarrow 7$

$\nwarrow \quad 1 \longrightarrow 1 \;, 2 \longrightarrow 2 \;,\; (3 \oplus 4) \longrightarrow 7$

$\nwarrow \quad 2 \longrightarrow 2 \;,\; (3 \oplus 4) \longrightarrow 7$

$\nwarrow \quad (3 \oplus 4) \longrightarrow 7$

$\nwarrow \quad 3 \longrightarrow 3 \;, 4 \longrightarrow 4 \qquad \nwarrow \quad 4 \longrightarrow 4 \qquad \nwarrow \quad \square$

derivazione goal-oriented

niente da provare

Backtracking

$(1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow 21$

derivazione
goal-oriented
(depth-first)

$\nwarrow \quad (1 \oplus 2) \longrightarrow 7\,,\ (3 \oplus 4) \longrightarrow 3$

$\nwarrow \quad 1 \longrightarrow 1\,, 2 \longrightarrow 6\ ,\ (3 \oplus 4) \longrightarrow 3$

$\nwarrow \quad 2 \longrightarrow 6\ ,\ (3 \oplus 4) \longrightarrow 3$

fallimento! dobbiamo tornare indietro (backtrack) fino all'ultima scelta e riprovare

$\nwarrow \quad 1 \longrightarrow 2\,, 2 \longrightarrow 5\ ,\ (3 \oplus 4) \longrightarrow 3$

fallimento! dobbiamo tornare indietro (backtrack) fino all'ultima scelta e riprovare

...

alternativamente possiamo esplorare tutte le alternative in parallelo
(breadth-first vs depth-first)

Programmazione Logica

PROLOG

Il Prolog è un linguaggio di programmazione dichiarativo semplice e potente, basato sulla logica dei predicati del primo ordine.

PROgrammation en LOGique

[’70] (Univ. Marseilles) A. Colmerauer, P. Roussel, R. Kowalski (aimed at processing natural (French) language)

Ogni psichiatra è una persona.
Ogni persona che analizza è malata.
Jacques è uno psichiatra di Marsiglia.
Jacques è una persona?
Dove si trova Jacques?
Jacques è malato?

Si. A Marseille. Non so.

```
TOUT PSYCHIATRE EST UNE PERSONNE.
CHAQUE PERSONNE QU’IL ANALYSE, EST MALADE.
JACQUES EST UN PSYCHIATRE A *MARSEILLE.
EST-CE QUE *JACQUES EST UNE PERSONNE?
OU EST *JACQUES?
EST-CE QUE *JACQUES EST MALADE?
OUI. A MARSEILLE. JE NE SAIS PAS.
```

Algoritmo

algoritmo = logica + controllo

	cosa	come
	(descrizione del problema)	(passi per arrivare alla soluzione)
	clausole di Horn	risoluzione
	database PROLOG	interprete PROLOG

Formule

$X = \{x, y, ...\}$ un insieme di variabili

$\Sigma = \{\Sigma_n\}_n$ una segnatura di simboli di funzioni $c, f, g, ...$

$\Pi = \{\Pi_n\}_n$ una segnatura di simboli di predicato $p, q, ...$

formula atomica

$$a = p(t_1, ..., t_n) \qquad p \in \Pi_n \qquad t_1, ..., t_n \in T_{\Sigma,X}$$

formula

$$a_1, ..., a_n$$

una congiunzione (anche vuota) di formule atomiche

Esempio

$X = \{\mathsf{N}, \mathsf{E}, \mathsf{E}_0, \mathsf{E}_1, ..., n, n_0, n_1, ...\}$ un insieme di variabili

$\Sigma_0 = \{0, 1, 2, ..., 0, 1, 2, ...\}$ un insieme di costanti

$\Sigma_2 = \{\_ \oplus \_, \_ \otimes \_\}$ un insieme di operatori binari (infissi)

$\Pi_2 = \{\_ \longrightarrow \_\}$ simbolo di predicato binario (infisso)

formula atomica $\mathsf{E} \oplus 2 \longrightarrow 5$

formula $\mathsf{E} \oplus 2 \longrightarrow 5, \mathsf{E} \otimes 7 \longrightarrow n$

Programmi Logici

Clausole di Horn

una formula atomica (la TESTA)

$h \ :- \ r$

una formula (il CORPO)

← Implicazione logica

$a \ :- \ a_1, ..., a_n$ simile a $\dfrac{a_1 \ \cdots \ a_n}{a}$

un insieme (o una lista) di clausole di Horn

Un programma logico L

$$L = \left\{ \begin{array}{l} \cdots \\ h :- r. \\ \cdots \end{array} \right\}$

Refutazione di un goal

Voglio dimostrare che la formula $a_1, ..., a_n$ è una conseguenza logica (puo' essere derivata dagli assiomi e dalle regole del sistema)

Aggiungiamo all'insieme delle regole la negazione della formula che vogliamo dimostrare

$$? - a_1, ..., a_n \qquad \text{Goal}$$

$$false \leftarrow a_1, \ldots, a_n$$

e cerchiamo di ottenere una contraddizione

Un esempio

```
vive(anna, roma).
vive(mario, milano).
vive(giovanni, torino).
vive(maria, roma).
vive(paolo, milano)

vive_citta(X, Y) :- vive(X, Z), vive(Y, Z), X \= Y.


% Goal
% “Anna vive a Roma?"
?- vive_citta(anna, milano)

% Goal
% "Chi vive nella stessa città di Anna?"
?- vive_citta(anna, X)

% Goal
% "Chi vive a Torino?"
?- vive_citta(X, torino)


?- % Goal
% "Chi vive nella stessa città?"
?- vive_citta(X, Y)
```

Applicazioni

un programma logico serve a rispondere alla seguente domanda:
data una formula g (goal) che vogliamo dimostrare, quali sono le istanze di g valide?

$$?- \quad (1 \oplus 2) \otimes (3 \oplus 4) \longrightarrow n \ , \ n \oplus \mathsf{E} \longrightarrow 26$$

Risoluzione SLD

Idea: ridurre iterativamente il goal iniziale g
applicando una delle clausole di Horn in L
a una delle formule atomiche del goal;

ogni applicazione calcola un unificatore più generale (mgu),
sostituisce la formula selezionata con il corpo della clausola
selezionata e applica l' mgu al nuovo obiettivo

$$
\begin{array}{lll}
?- \; g & \nwarrow_{\sigma_1} & g_1 \\
 & \nwarrow_{\sigma_2} & g_2 \\
 & \nwarrow_{\sigma_3} & \ldots \\
 & \nwarrow_{\sigma_m} & \square
\end{array}
$$

allora $g\sigma_1\sigma_2...\sigma_m$ è un teorema

Risoluzione SLD

$?- a_1, ... \boxed{a_i} ..., a_k$

L

$\dots$

$\boxed{h} :- b_1, ..., b_n$

$\dots$

a_i e h unificano!!

Ripeti i passi seguenti finche' ci sono formule atomiche nel goal

1. seleziona una formula atomica a_i nel goal (per esempio prendi la prima da sinistra);
2. seleziona una clausola Horn $h : -b_1, ..., b_n \in L$ la cui testa unifica con a_i;
3. sia σ l'unificatore piú generale (mgu) tra h e a_i, $a_i\sigma = h\sigma$;
4. sostituisci a_i nel goal con il corpo della clausola $b_1, ..., b_n$;
5. applica la sostituzione σ a tutto il goal ottenuto: $(a_1, ..., a_{i-1}, b_1, ..., b_n, a_{i+1}, ..., a_k)\sigma$.

Attenzione

le formule atomiche in un goal possono condividere variabili: la sostituzione deve essere applicata a tutto il goal per propagare l'informazione

$$(a_1, ..., b_1, ..., b_n, ..., a_k)\sigma$$

$$a_1, ..., (b_1, ..., b_n)\sigma, ..., a_k$$

Attenzione

la stessa clausola può essere riutilizzata molte volte:
ogni volta le sue variabili devono essere rinominate (prima dell'unificazione) con nuovi identificatori per evitare clashes

Ripeti i passi seguenti finche' ci sono formule atomiche nel goal

1. seleziona una formula atomica a_i nel goal;
2. seleziona una clausola Horn $h \overset{\rho}{:-} b_1, ..., b_n \in L$;
3. sia ρ un renaming per le variabili in $vars(h :- b_1, ..., b_n)$;
4. $(h :- b_1, ..., b_n)\rho$ è chiamata *variante* dell'originale;
5. sia σ l' mgu tra $h\rho$ e a_i, $a_i\sigma = (h\rho)\sigma$;
6. sostituisci a_i con $(b_1, ..., b_n)\rho$;
7. applica la sostituzione σ: $(a_1, ..., a_{i-1}, (b_1, ..., b_n)\rho, a_{i+1}, ..., a_k)\sigma$.

ρ ridenominazione delle variabili

Attenzione

nelle sostituzioni, solo le variabili che appaiono nel goal sono rilevanti per la risposta

$$\sigma : X \to T_{\Sigma,X}$$

$$Y \subseteq X$$

$$\sigma_{|Y}(x) \triangleq \begin{cases} \sigma(x) & \text{if } x \in Y \\ x & \text{otherwise} \end{cases}$$

riportiamo solo questa informazione parziale

$$a_1, ..., a_i, ..., a_k \ \nwarrow_{\widehat{\sigma}} \ (a_1, ..., (b_1, ..., b_n)\rho, ..., a_k)\sigma$$

$$\widehat{\sigma} \triangleq \sigma_{|vars(a_1,...,a_k)}$$

Risposte Calcolate

$$
\begin{array}{lll}
?- \; g & \nwarrow_{\widehat{\sigma}_1} & g_1 \\
 & \nwarrow_{\widehat{\sigma}_2} & g_2 \\
 & \nwarrow_{\widehat{\sigma}_3} & \ldots \\
 & \nwarrow_{\widehat{\sigma}_m} & \square
\end{array}
$$

$$\sigma = \widehat{\sigma}_1 \cdot \widehat{\sigma}_2 \cdot \widehat{\sigma}_3 \cdots \widehat{\sigma}_m$$

la sostituzione e' chiamata risposta calcolata (computed answer substitution) e viene calcolata come composizione delle risposte calcolate ad ogni passo

$$t(\sigma_1 \cdot \sigma_2) \stackrel{\triangle}{=} t\sigma_1\sigma_2 = \sigma_2(\sigma_1(t)) = (\sigma_2 \circ \sigma_1)(t)$$

Esempio

$\Sigma_0 = \{0, ...\}$ $\quad \Pi_3 = \{\mathsf{sum}, ...\}$

$\Sigma_1 = \{\mathsf{s}, ...\}$

somma codificata come predicato

$\mathsf{sum}(x, y, z)$ significa $x + y = z$

L

(un fatto)

$\mathsf{sum}(0, y, y).$

$\mathsf{sum}(\mathsf{s}(x), y, \mathsf{s}(z)) :- \mathsf{sum}(x, y, z).$

in PROLOG le clausole terminano con il punto

2+2=?

il goal $?- \mathsf{sum}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(\mathsf{s}(0)), n).$

L

$\mathsf{sum}(0, y, y).$

$\mathsf{sum}(\mathsf{s}(x), y, \mathsf{s}(z)) :- \mathsf{sum}(x, y, z).$

$\{\mathsf{sum}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(\mathsf{s}(0)), n) \stackrel{?}{=} \mathsf{sum}(0, y', y')\}$ fallisce

$\{\mathsf{sum}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(\mathsf{s}(0)), n) \stackrel{?}{=} \mathsf{sum}(\mathsf{s}(x_1), y_1, \mathsf{s}(z_1))\}$ ha successo

$$\sigma_1 = [x_1 = \mathsf{s}(0)\,,\ y_1 = \mathsf{s}(\mathsf{s}(0))\,,\ n = \mathsf{s}(z_1)]$$
$$\widehat{\sigma}_1 = [n = \mathsf{s}(z_1)]$$

$\widehat{\sigma}_1 = [n = \mathsf{s}(z_1)]$

$$\mathsf{sum}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(\mathsf{s}(0)), n) \ \nwarrow_{\widehat{\sigma}_1} \ (\,\mathsf{sum}(x_1, y_1, z_1)\,)\sigma_1$$
$$= \mathsf{sum}(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), z_1)$$

1+2=?

nuovo goal $\mathsf{sum}(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), z_1)$.

L

$\mathsf{sum}(0, y, y)$.

$\mathsf{sum}(\mathsf{s}(x), y, \mathsf{s}(z)) :- \mathsf{sum}(x, y, z)$.

$\{\mathsf{sum}(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), z_1) \stackrel{?}{=} \mathsf{sum}(0, y', y')\}$ fallisce

$\{\mathsf{sum}(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), z_1) \stackrel{?}{=} \mathsf{sum}(\mathsf{s}(x_2), y_2, \mathsf{s}(z_2))\}$ ha successo

$$\sigma_2 = [x_2 = 0\,,\ y_2 = \mathsf{s}(\mathsf{s}(0))\,,\ z_1 = \mathsf{s}(z_2)]$$
$$\widehat{\sigma}_2 = [z_1 = \mathsf{s}(z_2)]$$

$\widehat{\sigma}_1 = [n = \mathsf{s}(z_1)]$
$\widehat{\sigma}_2 = [z_1 = \mathsf{s}(z_2)]$

$$\mathsf{sum}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(\mathsf{s}(0)), n) \nwarrow_{\widehat{\sigma}_1} \mathsf{sum}(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), z_1) \nwarrow_{\widehat{\sigma}_2} (\,\mathsf{sum}(x_2, y_2, z_2)\,)\sigma_2 = \mathsf{sum}(0, \mathsf{s}(\mathsf{s}(0)), z_2)$$

0+2=?

nuovo goal $\mathsf{sum}(0, \mathsf{s}(\mathsf{s}(0)), z_2).$

L

$\mathsf{sum}(0, y, y).$

$\mathsf{sum}(\mathsf{s}(x), y, \mathsf{s}(z)) :- \mathsf{sum}(x, y, z).$

$\{\mathsf{sum}(0, \mathsf{s}(\mathsf{s}(0)), z_2) \stackrel{?}{=} \mathsf{sum}(0, y_3, y_3)\}$ ha successo

$\sigma_3 = [y_3 = \mathsf{s}(\mathsf{s}(0)) \, , \, z_2 = \mathsf{s}(\mathsf{s}(0))]$

$\widehat{\sigma}_3 = [z_2 = \mathsf{s}(\mathsf{s}(0))]$

$\mathsf{sum}(\mathsf{s}(\mathsf{s}(0)), \mathsf{s}(\mathsf{s}(0)), n)$

$\nwarrow_{\widehat{\sigma}_1}$ $\mathsf{sum}(\mathsf{s}(0), \mathsf{s}(\mathsf{s}(0)), z_1)$

$\nwarrow_{\widehat{\sigma}_2}$ $\mathsf{sum}(0, \mathsf{s}(\mathsf{s}(0)), z_2)$

$\nwarrow_{\widehat{\sigma}_3}$ $\square$

$\widehat{\sigma}_1 = [n = \mathsf{s}(z_1)]$
$\widehat{\sigma}_2 = [z_1 = \mathsf{s}(z_2)]$
$\widehat{\sigma}_3 = [z_2 = \mathsf{s}(\mathsf{s}(0))]$

$2 + 2 = 4$!

$\widehat{\sigma}_1 \cdot \widehat{\sigma}_2 \cdot \widehat{\sigma}_3 = [n = \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))))]$

1+n=3?

il goal $?- \mathsf{sum}(\mathsf{s}(0), n, \mathsf{s}(\mathsf{s}(\mathsf{s}(0))))$.

L

$\mathsf{sum}(0, y, y).$

$\mathsf{sum}(\mathsf{s}(x), y, \mathsf{s}(z)) :- \mathsf{sum}(x, y, z).$

$\{\mathsf{sum}(\mathsf{s}(0), n, \mathsf{s}(\mathsf{s}(\mathsf{s}(0)))) \stackrel{?}{=} \mathsf{sum}(0, y', y')\}$ fallisce

$\{\mathsf{sum}(\mathsf{s}(0), n, \mathsf{s}(\mathsf{s}(\mathsf{s}(0)))) \stackrel{?}{=} \mathsf{sum}(\mathsf{s}(x_1), y_1, \mathsf{s}(z_1))\}$ ha successo

$$\sigma_1 = [x_1 = 0\,,\ y_1 = n\,,\ z_1 = \mathsf{s}(\mathsf{s}(0))]$$
$$\widehat{\sigma}_1 = [\,]$$

$\mathsf{sum}(\mathsf{s}(0), n, \mathsf{s}(\mathsf{s}(\mathsf{s}(0))))$ $\nwarrow_{\widehat{\sigma}_1}$ $(\,\mathsf{sum}(x_1, y_1, z_1)\,)\sigma_1$ $\quad \widehat{\sigma}_1 = [\,]$

$= \mathsf{sum}(0, n, \mathsf{s}(\mathsf{s}(0)))$

0+n=2?

nuovo goal $\mathsf{sum}(0, n, \mathsf{s}(\mathsf{s}(0)))$

L

$\mathsf{sum}(0, y, y).$

$\mathsf{sum}(\mathsf{s}(x), y, \mathsf{s}(z)) :- \mathsf{sum}(x, y, z).$

$\{\mathsf{sum}(0, n, \mathsf{s}(\mathsf{s}(0))) \stackrel{?}{=} \mathsf{sum}(0, y_2, y_2)\}$ ha successo

$\sigma_2 = [y_2 = \mathsf{s}(\mathsf{s}(0)) \, , \, n = \mathsf{s}(\mathsf{s}(0))]$

$\widehat{\sigma}_2 = [n = \mathsf{s}(\mathsf{s}(0))]$

$\mathsf{sum}(\mathsf{s}(0), n, \mathsf{s}(\mathsf{s}(\mathsf{s}(0)))) \quad \nwarrow_{\widehat{\sigma}_1} \quad \mathsf{sum}(0, n, \mathsf{s}(\mathsf{s}(0)))$

$\nwarrow_{\widehat{\sigma}_2} \quad \square$

$\widehat{\sigma}_1 = [\,]$

$\widehat{\sigma}_2 = [n = \mathsf{s}(\mathsf{s}(0))]$

$\widehat{\sigma}_1 \cdot \widehat{\sigma}_2 = [n = \mathsf{s}(\mathsf{s}(0))]$

$(1 + n = 3) \Rightarrow n = 2\,!$

Jumping creatures

Assuming that:

1. All jumping creatures are green
2. All small jumping creatures are Martians
3. All green Martians are intelligent
4. Ngtrks is small and green
5. Pgvdrk is a jumping Martian

Who is intelligent?

Jumping creatures

Assuming that:

1. All jumping creatures are green
2. All small jumping creatures are Martians
3. All green Martians are intelligent
4. Ngtrks is small and green
5. Pgvdrk is a jumping Martian

Who is intelligent?

```
green(X) :- jumping(X).
martian(X) :- small(X) , jumping(X).
intelligent(X) :- green(X) , martian(X).
small(ngtrks).
green(ngtrks).
jumping(pgvdrk).
martian(pgvdrk).
```

?– intelligent(W).

intelligent(W)

↖ green(W) , martian(W)

↖ jumping(W) , martian(W)

↖ martian(pgvdrk)

↖ □ $[W = \text{pgvdrk}]$

martian(ngtrks)

↖ small(ngtrks) , jumping(ngtrks)

↖ jumping(ngtrks)

fail!

Antenati

Supponiamo di avere un predicato parent(x,y) che ci dice che x e' genitore di y.

- Definiamo un predicato sibling(x,y) che e' vero quando x e y hanno un genitore in comune

```
sibling(x,y):- parent (z,x),parent(z,y).
```

- Definiamo un predicato cousin(x,y) che e' vero quando x e y sono cugini

```
cousin(x,y):- parent(z,x),parent(v,y), sibling(z,v).
```

- Definiamo un predicato ancestor(x,y) che e' vero quando x e' un antenato diretto (padre, nonno, bisnonno…) di y

```
ancestor(x,y):- parent(x,y).
ancestor(x,y):- parent(x,z),ancestor(z,y).
```

Supponendo i seguenti fatti per parent

```
parent(alice,bob).
parent(alice,carl).
parent(bob, ella).
parent(carl,francisco).
```

```
ancestor(x,y):- parent(x,y).
ancestor(x,y):- parent(x,z),ancestor(z,y).
 sibling(x,y):- parent (z,x),parent(z,y).
  cousin(x,y):- parent(z,x),parent(v,y),
                 sibling(z,v).
```

Posso derivare i seguenti goal?

```
?- sibling(ella, francisco).
?- sibling(ella, diana).
?- cousin(ella, francisco).
?- cousin(ella, diana).
?- ancestor(alice,ella).
?- ancestor(carl,ella).
```

Esercizio 1

Sia $\Sigma_0 = \{0\}$ e $\Sigma_1 = \{s\}$, estendere il programma logico che definisce il predicato sum $\in \Pi_3$ (visto in classe) per definire:

1. un predicato prod $\in \Pi_3$ per calcolare il prodotto di due numeri;

2. un predicato pow $\in \Pi_3$ per calcolare la potenza;

3. un predicato div $\in \Pi_2$ per dire se il primo argomento divide il secondo.

Esercizio 2

Data la sintassi dell'Es. 1, risolvete i problemi di unificazione seguenti

1. *G1* = {prod(s(x), y, s(z)) ?= prod(y, z, x)}

2. *G2* = {pow(x, s(y), x) ?= pow(s(y), z, z)}

3. *G3* = {div(x, s(y)) ? = div(z, x) , div(y, s(z)) ? = div(u, s(u))}

Esercizio 3

Dati i programmi logici dell'Es. 1, scrivere alcune possibili derivazioni per i goal seguenti :

1. somma(x, s(0), s(s(0)))

2. prod(s(s(0)), y, s(s(0)))

3. div(z, s(s(0))

Esercizio da consegnare
(per mail entro la prox settimana)

Un albero binario T è vuoto (nullo)
oppure è composto da un valore radice e due successori, che sono essi stessi alberi binari.

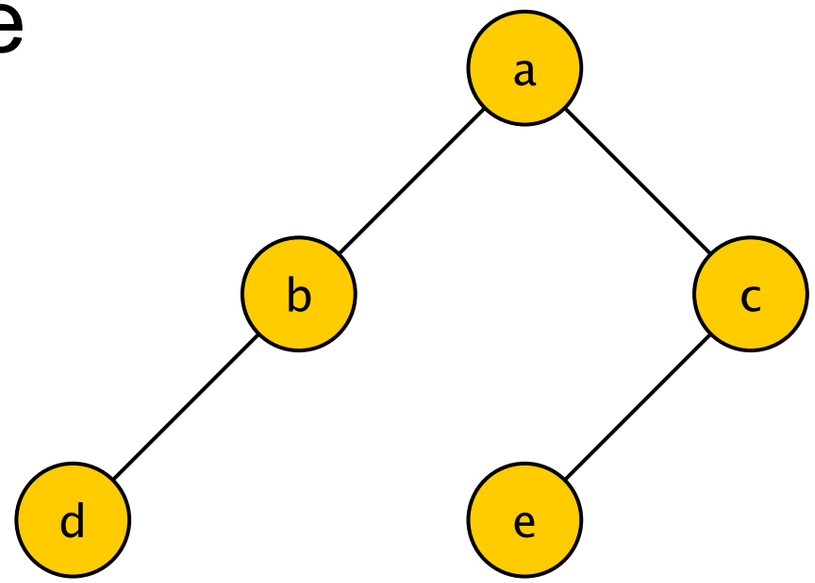
T è simmetrico se è possibile tracciare una linea verticale attraverso il nodo radice e quindi il sottoalbero destro è l'immagine speculare del sottoalbero sinistro (siamo interessati solo alla struttura, i valori non sono rilevanti).

1. Data la firma qui sotto, scrivi le clausole di Horn per verificare se un albero è l'immagine speculare di un altro (definisci il predicato mirror)
2. Poi estendi il codice per controllare se un albero è simmetrico (definisci il predicato sum).

$\Sigma_0 = \{\mathsf{nil}\}$ $\Sigma_3 = \{\mathsf{node}\}$ $\Pi_1 = \{\mathsf{sym}\}$ $\Pi_2 = \{\mathsf{mirror}\}$

Esercizio

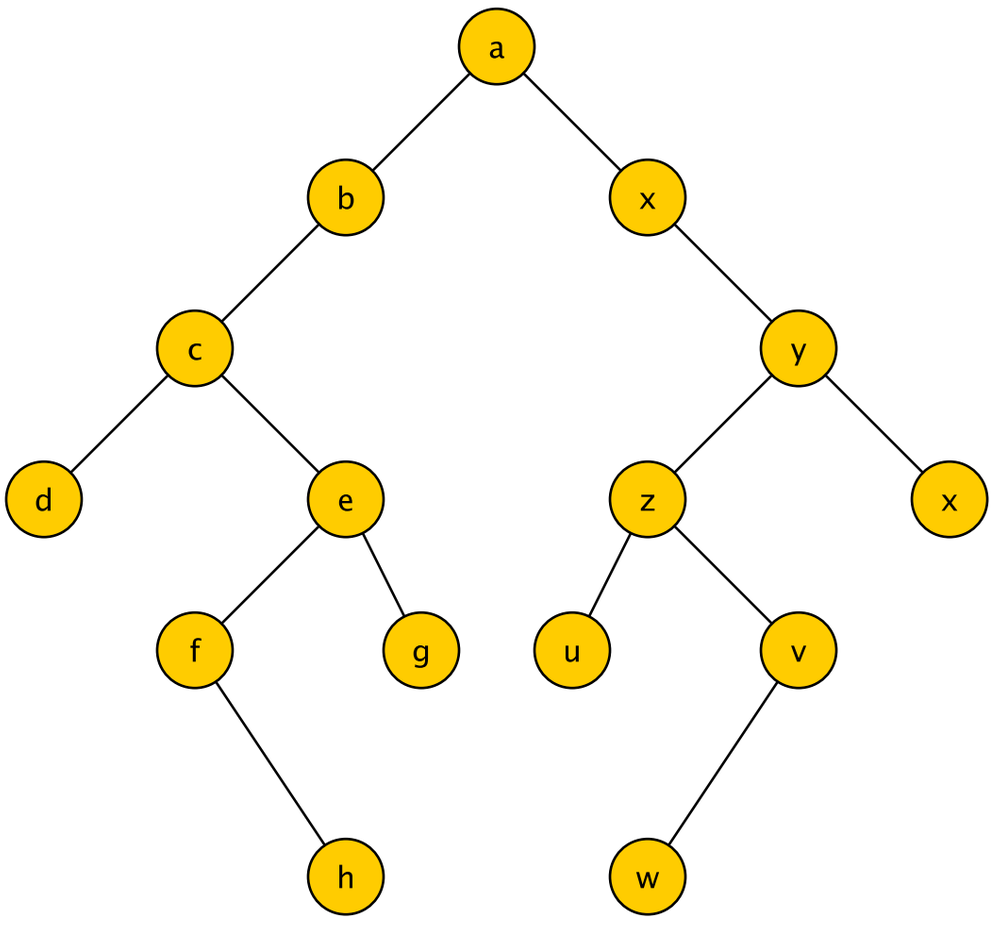
esempio:
alberi come
termini



node(a , node(b , node(d , nil , nil) ,
nil) ,
node(c , node(e , nil , nil) ,
nil))

Esercizio

un esempio di albero simmetrico



e uno che non e' simmetrico