

Prodromi di Prolog

Modulo *Frontiere dei sistemi multimediali*

2013-2014

Andrea Omicini

<mailto:andrea.omicini@unibo.it>

<http://andreaomicini.apice.unibo.it>

Prima della Programmazione Logica

- I primi edifici formali: aritmetica e geometria
- Aristotele e l'inferenza logica
- Il concetto di dimostrazione da Euclide a Cauchy
- Russell e il programma di Hilbert
- Gödel e i limiti delle teorie formalizzate
- Robinson e la dimostrazione automatica
- Da Kowalski a Colmerauer: Prolog, un linguaggio per la PROgrammazione LOGica

La rappresentazione della conoscenza

- logica e altri formalismi
- approccio dichiarativo (Aristotele)
- il dominio del discorso

Teorie del prim'ordine: cenni

- alfabeto e linguaggio del prim'ordine
 - correttezza, sintassi
- assiomi e regole d'inferenza
 - verità, “semantica”
- che rapporto tra un linguaggio logico e la logica del prim'ordine?
- che cos'è la logica e cos'è il controllo?
- Algoritmo: logica + controllo
 - il mito?
- Algoritmi equivalenti:
 $A1 = L + C1$
 $A2 = L + C2$

Rappresentazione del dominio del discorso in

- Sintassi
 - costanti, variabili, funzioni
 - termini
 - *termini ground*
 - *Universo di Herbrand*
 - predicati
 - atomi (formule atomiche)
 - *atomi ground*
 - *Base di Herbrand*
 - simboli non interpretati
 - $p(a)$
 - $\sin(0)$
- Semantica
 - pre-interpretazione
 - interpretazione

Uniformità dei simboli

- Prolog non distingue simboli per funzioni e predicati

`mamma(francesca, giuseppe)`

- è un termine o un atomo?
 - `mamma` simbolo di funzione o di predicato?
- Problema o vantaggio?
 - meta-interpreti: programmi come dati

Le clausole

- Letterali: $A, \neg A$
- Clausola: disgiunzione di letterali
 - $\forall X_1, \dots, \forall X_k (A_1 \vee \dots \vee A_n \vee \neg B_1 \vee \dots \vee \neg B_m)$
 - $\neg(F \wedge F') = \neg F \vee \neg F'$
 - $\forall X_1, \dots, \forall X_k (A_1 \vee \dots \vee A_n \vee \neg(B_1 \wedge \dots \wedge B_m))$
 - $F \leftarrow F' = F \vee \neg F'$
 - $\forall X_1, \dots, \forall X_k ((A_1 \vee \dots \vee A_n) \leftarrow (B_1 \wedge \dots \wedge B_m))$
 - $A_1, \dots, A_n \leftarrow B_1, \dots, B_m$
- Clausola definita: $n = 1$
 - Clausola unitaria: $m = 0$
- Goal definito: $n = 0$
- Clausola di Horn: $n = 0$ oppure $n = 1$

Il Principio di Risoluzione per le Clausole Generali

- Clausole genitrici
 - $C1 = A1 \vee \dots \vee An$
 - $C2 = B1 \vee \dots \vee Bm$
- Se
 - $Ai = \sim p(t1, \dots, tk)$, $Bj = p(t1', \dots, tk')$, oppure
 - $Ai = p(t1, \dots, tk)$, $Bj = \sim p(t1', \dots, tk')$
- e
 - $\vartheta = \text{mgu}(p(t1, \dots, tk), p(t1', \dots, tk'))$
- allora possiamo derivare da C1 e C2 la clausola (risolvente)
$$C3 = [A1 \vee \dots \vee Ai-1 \vee Ai+1 \vee \dots \vee An \vee \overset{\uparrow}{\sigma} B1 \vee \dots \vee Bj-1 \vee Bj+1 \vee \dots \vee Bm]\vartheta$$

Dimostrazione per contraddizione (1)

- Se l'insieme di clausole $H \cup \{\sim F\}$ è insoddisfacibile, mentre H è soddisfacibile, allora F segue logicamente da H
- Se da un insieme di clausole posso derivare la contraddizione logica, esso è insoddisfacibile
- Se, applicando la risoluzione, arrivo al risolvente vuoto, questo rappresenta la contraddizione logica ($\perp$): da

$C1 = \sim p(t1, \dots, tk), C2 = p(t1', \dots, tk'),$ oppure

$C1 = p(t1, \dots, tk), C2 = \sim p(t1', \dots, tk'),$ con

$\vartheta = mgu(p(t1, \dots, tk), p(t1', \dots, tk')),$ deriviamo

- $C3 = \perp$

Dimostrazione per contraddizione (2)

- Intuitivamente
 - derivo due atomi in contraddizione tra loro
 - derivo la contraddizione logica
 - non ci sono interpretazioni valide (modelli) per l'insieme di clausole $H \cup \{\sim F\}$
 - l'insieme $H \cup \{\sim F\}$ è insoddisfacibile
 - F è una conseguenza logica di H

Programma Prolog – introduzione informale

- Un programma Prolog è un insieme di clausole di Horn, che rappresentano:
 - fatti, riguardanti gli oggetti in esame e le relazioni che intercorrono
 - regole sugli oggetti e sulle relazioni
 - Se... allora...
 - goal (clausole senza testa), sulla base di conoscenza definita

Esempio

- “Due individui sono colleghi se lavorano presso la stessa ditta”

```
collega(X,Y) :- lavora(X,Z), lavora(Y,Z), X \== Y.
```

```
lavora(emp1,ibm).
```

```
lavora(emp2,ibm).
```

```
lavora(emp3,txt).
```

```
lavora(emp4,olivetti).
```

```
lavora(emp5,txt).
```

```
?- collega(X,Y).
```

```
X=emp1      Y=emp2;
```

```
X=emp2      Y=emp1;
```

```
X=emp3      Y=emp5;
```

```
X=emp5      Y=emp3;
```

```
no.
```

Esempio

- R1. $\text{sum}(0, X, X)$.
- R2. $\text{sum}(s(X), Y, s(Z)) :- \text{sum}(X, Y, Z)$.
 - non esiste interpretazione dei simboli sum e s
 - i numero interi sono rappresentati dalla struttura successore
 - si utilizza la ricorsione
 - esistono molte possibili interrogazioni per lo stesso database (programma).
- Possibili interrogazioni
 - ?- $\text{sum}(s(0), s(s(0)), Y)$.
 - ?- $\text{sum}(s(0), Y, s(s(s(0))))$.
 - ?- $\text{sum}(X, Y, s(s(s(0))))$.
 - ?- $\text{sum}(X, Y, Z)$.
 - ?- $\text{sum}(X, Y, s(s(s(0))))^3, \text{sum}(X, s(0), Y)$.

Prova di un goal

- Un goal viene provato provando i singoli letterali da sinistra verso destra.
 - :- collega(X,Y), persona(X), persona(Y)
- Un goal dato da un singolo letterale (goal atomico) viene provato confrontandolo con le teste delle clausole contenute nella base di conoscenza.
- Se esiste una sostituzione per cui il confronto ha successo:
 - se la clausola con cui si unifica è una clausola unitaria (fatto), la prova termina
 - se è una clausola condizionale (regola), ne viene provato il body
- Se non esiste una sostituzione¹⁴, il goal fallisce.

Esempio

`append([ ], X, X) .`

`append([X|Z], Y, [X|T]) :- append(Z, Y, T) .`

?- `append([a,b],[c,d],[a,b,c,d]) .`

?- `append([a,b],Y,[a,b,c,d]) .`

?- `append(X,[c,d],[a,b,c,d]) .`

?- `append(X,Y,[a,b,c,d]) .`

?- `append(X,Y,Z) .`

PIU' FORMALMENTE

- Linguaggio Prolog: caso particolare del paradigma di Programmazione Logica
- SINTASSI: un programma Prolog è costituito da un insieme di **clausole definite** della forma

(c11) $A.$ $\longrightarrow$ **FATTO o ASSERZIONE**

(c12) $A :- B_1, B_2, \dots, B_n.$ $\longrightarrow$ **REGOLA**

(c13) $:- B_1, B_2, \dots, B_n.$ $\longrightarrow$ **GOAL**

- In cui A e B_i sono formule atomiche
- A : **testa** della clausola
- $B_1, B_2, \dots, B_n$: **body** della clausola
- Il simbolo “,” indica la congiunzione; il simbolo “:-” l’implicazione logica in cui A e’ il conseguente e $B_1, B_2, \dots, B_n$ l’antecedente

PIU' FORMALMENTE

- Una **formula atomica** è una formula del tipo

$$p(t_1, t_2, \dots, t_n)$$

in cui p è un **simbolo predicativo** e $t_1, t_2, \dots, t_n$ sono **termini**

- Un **termine** è definito ricorsivamente come segue:
 - le costanti (numeri interi/floating point, stringhe alfanumeriche aventi come primo carattere una lettera minuscola) sono termini
 - le variabili (stringhe alfanumeriche aventi come primo carattere una lettera maiuscola oppure il carattere “_”) sono termini.
 - $f(t_1, t_2, \dots, t_k)$ è un termine se “ f ” è un simbolo di funzione (operatore) a k argomenti e $t_1, t_2, \dots, t_k$ sono termini. $f(t_1, t_2, \dots, t_k)$ viene detta struttura

NOTA: le costanti possono essere viste come simboli funzionali a zero argomenti.

ESEMPI

- COSTANTI: $a, \text{pippo}, aB, 9, 135, a92$
- VARIABILI: $x, x1, \text{Pippo}, _pippo, _x, _$
 - la variabile $_$ prende il nome di variabile anonima
- TERMINI COMPOSTI: $f(a), f(g(1)), f(g(1), b(a), 27)$
- FORMULE ATOMICHE: $p, p(a, f(x)), p(y), q(1)$
- CLAUSOLE DEFINITE:
 $q.$
 $p: -q, r.$
 $r(Z).$
 $p(X) :- q(X, g(a)).$
- GOAL:
 $:-q, r.$
- Non c'è distinzione tra costanti, simboli funzionali e predicativi.

INTERPRETAZIONE DICHIARATIVA

- Le variabili all'interno di una clausola sono quantificate universalmente
- per ogni asserzione (fatto)

$$p(t_1, t_2, \dots, t_m) .$$

se $x_1, x_2, \dots, x_n$ sono le variabili che compaiono in $t_1, t_2, \dots, t_m$ il significato è: $\forall x_1, \forall x_2, \dots, \forall x_n (p(t_1, t_2, \dots, t_m))$

- per ogni regola del tipo

$$A : - B_1, B_2, \dots, B_k .$$

se $y_1, y_2, \dots, y_n$ sono le variabili che compaiono solo nel body della regola e $x_1, x_2, \dots, x_n$ sono le variabili che compaiono nella testa e nel corpo, il significato è:

$$\forall x_1, \forall x_2, \dots, \forall x_n, \forall y_1, \forall y_2, \dots, \forall y_n ((B_1, B_2, \dots, B_k) \rightarrow A)$$

$$\forall x_1, \forall x_2, \dots, \forall x_n ((\exists y_1, \exists y_2, \dots, \exists y_n (B_1, B_2, \dots, B_k)) \rightarrow A)$$

INTERPRETAZIONE DICHIARATIVA

- ESEMPI

`padre(X,Y)` “x è il padre di y”

`madre(X,Y)` “x è la madre di y”

`nonno(X,Y) :- padre(X,Z) , padre(Z,Y) .`

“per ogni x e y, x è il nonno di y se esiste z tale che x è padre di z e z è il padre di y”

`nonno(X,Y) :- padre(X,Z) , madre(Z,Y) .`

“per ogni x e y, x è il nonno di y se esiste z tale che x è padre di z e z è la madre di y”

ESECUZIONE DI UN PROGRAMMA

- Una computazione corrisponde al tentativo di dimostrare, tramite la risoluzione, che una formula segue logicamente da un programma (è un teorema).
- Inoltre, si deve determinare una sostituzione per le variabili del goal (detto anche “query”) per cui la query segue logicamente dal programma.
- Dato un programma P e la query:

$:- p(t_1, t_2, \dots, t_m) .$

se $x_1, x_2, \dots, x_n$ sono le variabili che compaiono in $t_1, t_2, \dots, t_m$ il

significato della query è: $\exists x_1, \exists x_2, \dots, \exists x_n p(t_1, t_2, \dots, t_m)$ e

l'obiettivo è quello di trovare una sostituzione

$\sigma = \{x_1/s_1, x_2/s_2, \dots, x_n/s_n\}$

dove s_i sono termini tale per cui $P \models [p(t_1, t_2, \dots, t_m)]\sigma$

SCHEMA RIASSUNTIVO

