

Prolog – Prodromi



Sistemi distribuiti LS

2003/2004

Prof. Andrea Omicini

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*

- *ne parleremo meglio nel corso regolare...*

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 un linguaggio logico

- *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*
 - $\equiv \forall X_1, \dots, \forall X_k (A_1 \vee \dots \vee A_n \vee \neg B_1 \vee \dots \vee \neg B_m)$
 - ☑ $\neg(F \wedge F') = \neg F \vee \neg F'$
 - $\equiv \forall X_1, \dots, \forall X_k (A_1 \vee \dots \vee A_n \vee \neg(B_1 \wedge \dots \wedge B_m))$
 - ☑ $F \leftarrow F' = F \vee \neg F'$
 - $\equiv \forall X_1, \dots, \forall X_k ((A_1 \vee \dots \vee A_n) \leftarrow (B_1 \wedge \dots \wedge B_m))$
 - $A_1, \dots, A_n \leftarrow B_1, \dots, B_m$
- *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*
$$\vartheta = mgu(p(t1, \dots, tk), p(t1', \dots, tk'))$$
- *allora possiamo derivare da C1 e C2 la clausola (risolvente)*
$$C3 = [A1 \vee \dots \vee Ai-1 \vee Ai+1 \vee \dots \vee An \vee B1 \vee \dots \vee Bj-1 \vee Bj+1 \vee \dots \vee Bm] \vartheta$$

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 ($\square$): 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

$$C3 = \square$$

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 – Intro Informale



- *Un programma PROLOG e' 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), div(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
```

```
. . .
```

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 numeri interi sono rappresentati dalla struttura successore*
- *si utilizza la ricorsione*
- *esistono molte possibili interrogazioni per lo stesso database (programma).*
- *POSSIBILI INTERROGAZIONI:*
 - ?- $\text{sum}(s(0), s(s(0)), Y) .$*
 - ?- $\text{sum}(s(0), Y, s(s(s(0)))) .$*
 - ?- $\text{sum}(X, Y, s(s(s(0)))) .$*
 - ?- $\text{sum}(X, Y, Z) .$*
 - ?- $\text{sum}(X, Y, s(s(s(0)))), \text{sum}(X, s(0), Y) .$*

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 sostituzione, 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) .`