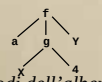
Dati e strutture in Prolog

Sistemi intelligenti distribuiti LS
2004/2005
Prof. Andrea Omicini

I termini

- *Strutture ad albero*
  `f(a,g(X,4),Y)`



- *I simboli di funzione rappresentano i nodi dell'albero*
- *Le costanti rappresentano le foglie*
- *Le variabili*
  - *a singolo assegnamento*
- *consentono di "raffinare" una struttura ad albero*
  - *assegnando progressivamente alle variabili i valori determinati dalla sostituzione calcolata*
- *Attenzione alla struttura e ai nomi!!*

Le liste

- *Le liste rappresentano una particolare struttura ad albero "degenere"*
- *con una notazione ad hoc*
  - *il funtore "ufficiale" sarebbe il* `.`*, operatore binario applicato alla testa e alla coda della lista*
  - `[]` *è la costante nil, che rappresenta la lista vuota*
- *Ma le liste le avete già viste con Natali*
  - *quindi non simulate ignoranza…*

Gli atomi

- *Gli atomi rappresentano relazioni tra entità del dominio*
  - *possono essere usati per manipolare gli alberi*
  - *insieme all'unificazione*
- *Esempio: automa a stati finiti non deterministico*

Esempio: la rappresentazione dell'automa

```
finale(s3).
transizione(s1,a,s1).
transizione(s1,a,s2).
transizione(s1,b,s1).
transizione(s2,b,s3).
transizione(s3,b,s4).
silente(s2,s4).
silente(s3,s1).
```

Esempio: il funzionamento dell'automa

```
accetta(Stato,[]) :- finale(Stato).
accetta(Stato,[Input|RestoInput]) :-
    transizione(Stato,Input,NuovoStato),
    accetta(NuovoStato,RestoInput).
accetta(Stato,Input) :-
    silente(Stato,NuovoStato),
    accetta(NuovoStato,Input).
```

Esempio: semplici query

```
?- accetta(s1,[a,a,a,b]).
yes
?- accetta(S,[a,b]).
S = s1;
S = s3;
no
?- accetta(s1,[X1,X2,X3]).
X1 = a, X2 = a, X3 = b;
X1 = b, X2 = a, X3 = b;
no
?- Input = [_,_,_], accetta(s1,Input).
Input = [a,a,b];
Input = [b,a,b];
no
```

Interpretazione di programmi Prolog come basi di dati

- *L'insieme dei fatti ground nella teoria Prolog è la base di dati*
- *In più, posso rappresentare i rapporti tra le diverse relazioni*
  - *ovvio esempio: le relazioni di parentela*
    - *la base di dati è rappresentata dalle relazioni ground genitore figlio*
    - *le regole rappresentano il significato relativo delle diverse relazioni*
  - *Il goal Prolog è la query alla base dati*
    - *in generale, la programmazione logica è più espressiva e potente della programmazione relazionale*
    - *che però è ovviamente più efficiente su ampie masse di dati*
- *Esempio per elettronici: pagg. 33-36 Bratko*
- *Esercizio concettuale: algebra relazionale e Prolog*
  - *pagg. 42-43 Bratko*