Lezione 13

PROLOG II
Programmi aperti
Ricorsione di Coda e difference lists

Elenco caratteristiche prolog

- **Costanti:** **a)** simboli atomici (iniziano con una minuscola, cifra, …)
  b) '….', ad esempio 'X'
- **Commenti**: /* …… */
- **Direttive:** :- Body.
- Direttiva per includere un programma: :- include(*nomefile*)
- **Operatori**

  op(precedenza,associazione,simbolo)

Associazioni:

- fx : unario, non associativo (analogo xf)
  esempio: op(200, fx, il):
  posso scrivere: il cane ma non: il il cane
- fy : unario, associativo (analogo yf)
  esempio: op(200, fy, s):
  posso scrivere: s s s 0 al posto di s(s(s(0)))
- xfx : binario, non associativo
  esempio: op(200, fx, '<-'):
  posso scrivere: A <- B, ma non A <- B <- C
- xfy : binario, associativo a destra
  esempio: op(200, xfy, &):
  posso scrivere: A & B & C al posto di A & (B & C)
- yfx : binario, associativo a sinistra
  esempio: op(200, fx, '°'):
  posso scrivere: 1 ° 0 ° 1 ° 1 al posto di ((1 ° 0) ° 1) ° 1

Uguaglianza sintattica:

- X == Y , X \== Y

- **Aritmetica:**
  - Costanti intere 53, -5, ..
  - Costanti reali: 2.5, …
  - Operazioni: +, *, -, // (intera), / (reale), mod
  - Predicati: =:=, =\=, <, >, >=, <=
  - Quando un'espressione (termine o predicato) E è valutata con sostituzione corrente σ, Eσ dev'essere ground
  - **Assegnamento/test:** X is E; in questo caso:
    - Eσ dev'essere ground;
    - Xσ dev'essere una costante numerica o una variabile

- **Liste:**
  - Notazione [ ], [Testa|Coda] ed abbreviazioni
    - [a,b,c|L], …
  - Predefinite: append, union, intersection, member

- **Controllo:** !
- Esempio: la negazione come fallimento con il cut:

```
not(A) :- A, !, fail.
not(_).
```

- **Disgiunzione**, indicata con ; :

```
A :- Body1.
A :- Body2.
```

equivale logicamente a

```
A :- Body1 ; Body2.
```


Controllo: if C then A else B si scrive C -> A ; B

```
H :- (C -> A ; B).
```

equivale a

```
H :- C, !, A.
H :- B.
```

Si ha una *differenza nel backtracking*:
con il cut influisce su tutte le clausole che definiscono il predicato di H, mentre C -> A ; B influisce solo localmente.

ESEMPIO: le Tabelline per la somma binaria

```
tab(C1,C2,CR,R ° S) :-
        C1 =\= C2 ->  (CR = 0 -> R ° S = 0 ° 1;
                                 R ° S = 1 ° 0);
                      (C1=0  -> R ° S = 0 ° CR;
                                 (CR = 0 ->  R ° S = 1 ° 0;
                                             R ° S = 1 ° 1 )).
```


I programmi aperti: Esempio della somma

- Concludiamo l'esempio della somma con un programma aperto;
  - Un programma in cui dei predicati, detti aperti, non sono definiti da nessuna clausola;
  - Un programma aperto P può essere chiuso in molti modi differenti, includendolo in contesti che definiscono in modi diversi i predicati aperti di P
- Nell'esempio che segue:
  - se definiamo i predicati aperti *tab* e *cifra* in un sistema a k cifre, ottenete un sommatore in base k.

Vedere il programma commentato nei programmi allegati alla lezione

```
sum(X,Y,R) :- cifra(X),  !,
              addCifra(Y,X,R).
sum(X,Y,R) :- cifra(Y),  !,
              addCifra(X,Y,R).
sum(X ° C1, Y ° C2, U) :- sumR(X ° C1, Y ° C2, 0, U).


sumR(X ° C1, Y ° C2, R, V ° S) :- !, tab(C1,C2,R, C ° S),
                                    sumR(X,Y,C,V).
sumR(X,Y,R,U) :- sum(X,Y,V),
                 addCifra(V,R,U).

addCifra(X,0,X) :- !.
addCifra(X ° C1, C2, V ° S) :- !,  tab(C1, C2, 0, C ° S),
                                   addCifra(X, C, V) .
addCifra(C1, C2, R) :- tab(C1, C2, 0, R).
```


Esercizio 1: completare per ottenere il sommatore binario.

Esercizio 2: completare per ottenere un sommatore in base 1000, dove le cifre sono numeri fra 0 e 999; ad esempio 1 ° 54 ° 321 significa 1.054.321.

Differenza e ricorsione di coda: esempio della reverse

rev([ ],[ ]).
rev([T|C], R) :- rev(C,RC), append(RC,[T],R).

- Come passare a ricorsione di coda? Passare dall'operazione cumulativa iterata alla sua inversa, in questo caso da append a diff :
  - diff(X,Y) lista X a cui tolgo la sottolista finale Y
    diff([a,b,c,d],[c,d]) = [a,b]
  - diff è l'inversa sinistra di append:
    diff(X,X) = [ ] e append(diff(X,Y),Y) = X

revd([ ], diff(X,X)).
revd([T|C], diff(X,Y)) :- revd(C, diff(X,[T|Y])).

- Infatti
  revd([b,c,d],diff([**d,c,b**,a],[a])) ← revd([c,d], diff([**d,c**,b,a],[b,a]))

= aggiungere b al minuendo ← Togliere b dal sottraendo

L'idea: partire con il sottraendo nullo, mediante la chiamata revd(L,diff(X,[])); usare il sottranedo come accumulatore: ogni chiamata ricorsiva toglie la resta da L e la aggiunge al sottraendo; quando L è vuoto, la clausola revd([], diff(X,X)) unifica minuendo e sottraendo e il risultato passa nel minuendo.

rev(L,X) :- revd(L,diff(X,**[]**)).
revd([**T|C**], diff(X,**Y**)) :- !, revd(**C**, diff(X,[**T|Y**])).
revd([ ], diff(X,**X**)).

Modo: revd(**In**, diff(Out,**In**))

Differenza e ricorsione di coda: altro esempio

pow(X,0,1).
pow(X,N, Z) :- N > 0, M is N-1, pow(X,M,U), Z is U*X.

- Come passare a ricorsione di coda? Passare dall'operazione cumulativa iterata alla sua inversa, in questo caso da * a / :

powd(X,0,A/A).
powd(X,N,A/B) :- N > 0, M is N-1, powd(X,M,A/(B*X)).

- Infatti $X^{N-1} = A/(B*X) \rightarrow X^N = X*X^{N-1} = X*(A/(B*X)) = A/B$

- Partendo da powd(X,N,A/1), B unifica con 1 e diventa un accumulatore che scarica il risultato finale in A alla fine, raggiunta la base A/A; il programma è:

pow(X,N,Z) :- powd(X,N,A/1).
powd(X,0,A/A).
powd(X,N,A/B) :- N > 0, M is N-1, C is B*X, powd(X,M,A/D).

Uso delle difference lists

- Quando si voglia trattare con sequenze di dati di lunghezza variabile.
- L'idea delle difference lists (per accumulare un risultato) si applica spesso per ottenere la ricorsione di coda. Solitamente l'operatore astrattt diff nonè esplicitamente usato, anziché diff(X,Y) si usano direttamente i due argomenti X,Y

rev(L,X) :- rev(L,X,[]).
rev([T|C], X,Y) :- !, rev(C, X,[T|Y]).
rev([ ], X,X).

Esempio

- Un semplice pianificatore, vedere programmi allegati

Esercizio

- 3 missionari e 3 cannibali devono attraversare un fiume con una barca di al più 2 posti. Su ciascuna delle due rive non devono mai esserci più cannibali che missionari. Come fare?
- A) determinare una rappresentazione degli stati
  - s(Ms,Cs,B) con
    - Ms = missionari su riva sinistra
    - Cs = cannibali su riva sinistra
    - B = s (barca a sinistra) o B = d (barca a destra)

B) determinare i passi:
s(Ms,Cs,s) → s(Ns,Ds,d) sotto quali condizioni?
s(Ms,Cs,d) → s(Ns,Ds,s) sotto quali condizioni?
C) Definire gli stati iniziali: sarà s(3,3,s)
D) Definire i goals: sarà s(0,0,d)
E) Usare pianificatore, definendo passo e diverso