COMPITO DI INTELLIGENZA ARTIFICIALE

18 Gennaio 2002

Esercizio 1
Si consideri il seguente problema di soddisfacimento di vincoli:

$$A^2+B^2 <= 9,\ A+C > 5,\ D = B-1,\ A = D+1$$

dove le variabili spaziano da 0 a 5. Si trovi una soluzione utilizzando il Forward Checking, con euristica per la selezione della variabile *First Fail*.

Esercizio 2
Si consideri il seguente programma Prolog:

```
p(X) :- call(X), !.
p(X) :- X =..L, append(L,[V],T), Y=..T, p(Y).
```

e le seguenti definizioni standard dei predicati `append/3` e `member/2`:

```
append([],X,X).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).

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

Si rappresenti l'albero di derivazione (SLD) corrispondente alla seguente query:

```
:- p(member(1)).
```

Esercizio 3
Si traducano le seguenti frasi nella logica dei predicati del primo ordine, poi in forma a clausole:
- Qualunque bimbo durante il giorno gioca oppure guarda la tv (ex-or)
- Mattina, pomeriggio e sera sono periodi del giorno; la mattina precede il pomeriggio ed il pomeriggio la sera.
- Quando un bimbo è con la mamma non guarda la tv
- Elena è una bimba
- Elena la sera è con la mamma
- Se un bimbo gioca, vuol dire che nel periodo precedente ha guardato la tv (cioè, se in un periodo il bambino gioca ed esiste un periodo precedente, allora in quel periodo ha guardato la tv).

Si usi poi il principio di risoluzione per dimostrare il quale periodo del giorno Elena guarda la tv.

Esercizio 4
Si scriva un metainterprete Prolog che cambi l'ordine di selezione dei letterali nel body delle clausole. In particolare, il metainterprete, prima dell'unificazione, deve contare nella testa della clausola:

- il numero *N* di parametri numerici o che contengono un termine numerico (ad esempio f(d(X,3)) contiene un termine numerico)
- il numero *P* di parametri non numerici (ossia che non soddisfano le suddette condizioni).

Se *N* è maggiore o uguale a *P* allora il body della clausola verrà risolto con una strategia left most, mentre in caso contrario right most.

Ad esempio, nella clausola:

```
p(X,1,q(2,X)) :- r, s(X).
```

il body viene risolto con strategia left-most, infatti ci sono due parametri numerici (1 e q(2,X)) ed uno non numerico (X). Nella clausola:

```
p(X,q(Y,Z),3) :- s(Y), s(Z).
```

il body viene risolto con strategia right-most, in quanto ci sono due parametri non numerici(X e q(Y,Z)) ed uno numerico (3).

SOLUZIONE

Esercizio 1

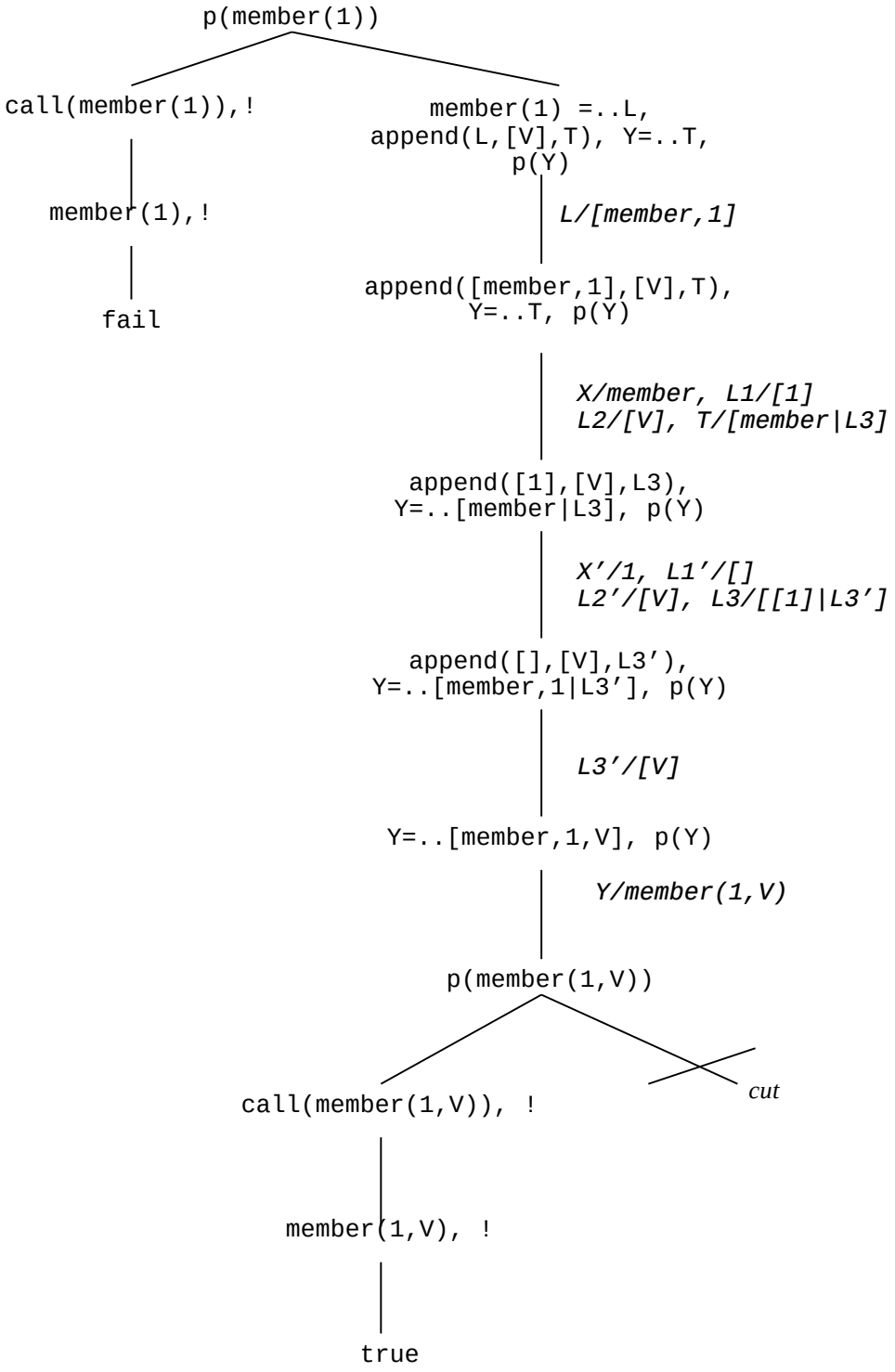
$A^2+B^2 <= 9$, A+C > 5, D = B-1, A = D+1

	Fase	Dom A	Dom B	Dom C	Dom D
	inizio	0..5	0..5	0..5	0..5
1	labeling A	**0**	0..3	∅	∅
2	backtracking	0..5	0..5	0..5	0..5
3	labeling A	**1**	0..2	5	0
4	labeling C	1	0..2	**5**	0
5	labeling D	1	1	5	**0**

Soluzione: A = 1, B=1, C=5, D = 0.

Esercizio 2

L'albero di derivazione SLD è il seguente:



Esercizio 3

In logica:

∀ X ∀ Y bimbo(X) ∧ giorno(Y) -> gioca(X,Y) ex-or guarda_tv(X,Y)
giorno(mattina)
giorno(pomeriggio)
giorno(sera)
precede(mattina, pomeriggio)
precede(pomeriggio, sera)
∀ X ∀ Y bimbo(X) ∧ giorno(Y) ∧ insieme(X,Y,mamma) -> ~guarda_tv(X,Y)

bimbo(elena)
insieme(elena, sera, mamma)
∀B, ∀X, ∀Y bimbo(B) ∧ giorno(X) ∧ gioca(B,X) and precede(Y,X) -> guarda_tv(B,Y)
(La frase può essere scritta in maniera equivalente:
∀B, bimbo(B) -> [∃X gioca(B,X) ∧ ∃Y precede(Y,X) ⇒ guarda_tv(B,Y)])

Goal: ∃ Y giorno(Y) and guarda_tv(elena,Y)

Trasformazione in clausole:
C1: ~ bimbo(X) ∨ ~giorno(Y) ∨ gioca(X,Y) ∨ guarda_tv(X,Y)
C2: ~ bimbo(X) ∨ ~ giorno(Y) ∨ ~gioca(X,Y) ∨ ~guarda_tv(X,Y)
C3: giorno(mattina)
C4: giorno(pomeriggio)
C5: giorno(sera)
C6: precede(mattina, pomeriggio)
C7: precede(pomeriggio, sera)
C8: ~ bimbo(X) ∨ ~giorno(Y) ∨ ~insieme(X,Y,mamma) ∨ ~guarda_tv(X,Y)
C9: bimbo(elena)
C10: insieme(elena,sera,mamma)
C11: ~bimbo(B) ∨ ~giorno(X) ∨ ~gioca(B,X) ∨ ~precede(Y,X) ∨ guarda_tv(B,Y)
Goal negato: ~ giorno(Y) ∨ ~guarda_tv(elena,Y)

Applicando il Principio di Risoluzione:
C12 = GN + C11: ~ bimbo(elena) ∨ ~giorno(Y) ∨ ~giorno(X) ∨ ~gioca(elena,X) ∨ ~precede(Y,X)
C13 = C12 + C7 : ~bimbo(elena) ∨ ~giorno(pomeriggio) ∨ ~giorno(sera) ∨ ~gioca(elena,sera)
C14 = C13 + C1: ~bimbo(elena) ∨ ~giorno(sera) ∨ guarda_tv(elena, sera)
C15 = C14 + C8: ~bimbo(elena) ∨ ~giorno(sera) ∨ ~insieme(elena, sera, mamma)
C16 = C15 + C9 + C5 + C10: []

Esercizio 4

```
solve(true):- !.
solve((A,B)):- !, solve(A), solve(B).
solve(G):- functor(G,P,A),
 functor(T,P,A),
 clause(T,B),
 conta(T,Num,NoNum),
 ordina_body(Num,NoNum,B, Bord),
 solve(Bord).

conta(T,Num,NoNum):-
      T=..[_|Arg],
      num(Arg,0,Num,0,NoNum).

%num(+ListaParam,+NumericiIn,-NumericiOut,
%     +NonNumericiIn,-NonNumericiOut)
num([],Num,Num,NoNum,NoNum).
num([Arg|L],NumIn,NumOut,NoNumIn,NoNumOut):-
            number(Arg),!,
            NumT is NumIn + 1,
            num(L,NumT,NumOut,NoNumIn,NoNumOut).
```

```
num([Arg|L],NumIn,NumOut,NoNumIn,NoNumOut):-
      compound(Arg),!, Arg =..[_|Arg1],
      % Riutilizzo num/5 per verificare se ci sono
      % dei parametri numerici
      num(Arg1,0,Numeric,0,_),
      (Numeric > 0
            -> NumT is NumIn +1, NoNumT = NoNumIn
            ;  NumT = NumIn, NoNumT is NoNumIn+1),
      num(L,NumT,NumOut,NoNumT,NoNumOut).
num([_|L],NumIn,NumOut,NoNumIn,NoNumOut):-
             NoNumT is NoNumIn + 1,
             num(L,NumIn,NumOut,NoNumT,NoNumOut).

ordina_body(Num,NoNum,B,B):- Num >= NoNum, !.
ordina_body(_Num,_NoNum,B,Bord):- reverse_conv(B,Bord).

% Inverte una lista con parentesi tonde.
% Utilizza il predicato predefinito che agisce sulle
% liste con parentesi quadre
reverse_conv(A,B) :-
      convert(A,La),
      reverse(La,Lrev),
      convert(B,Lrev).

% Converte la lista con le parentesi tonde in quella con
% le parentesi quadre e viceversa.
% Nota che, per funzionare in modo bidirezionale, nella
% prima clausola bisogna accettare solo le liste con
% parentesi quadre che hanno almeno 2 elementi, altrimenti
% non verrebbe selezionata la seconda
convert((H,T),[H,A|T1]) :- !, convert(T,[A|T1]).
convert(X,[X]).
```