Examen final Prolog / Juin 1994 — ESSI2

Michel Rueher

Documents autorisés: *supports (et notes) de cours de TD*

1 Arbre de recherche et arbre de preuves — 3 points

Construire l'arbre de recherche et exhiber les arbres de preuves pour la requête `plus(X,Y,s(s(s(zero))))` et le le programme ci-dessous:

```
plus(zero,X,X).
plus(s(X),Y,s(Z)) :-
     plus(X,Y,Z).}
```

2 Programme `mybagof/3` — 5 points

- Expliquer le rôle du `fail` dans la première clause du programme ci-dessous :

```
mybagof(O,P,L):-
     P,
     add_mylist(O),
     fail.

mybagof(_,_,L) :- !,
     retract(mylist(L-[])).
```

```
add_mylist(O) :-
      retract(mylist(L-S)),
      S=[O|S1],
      assert(mylist(L-S1))

add_mylist(O) :-
      not(mylist(_)), !
      L = [O|S],
      assert(mylist(L-S)).
```

- Construire l'arbre de recherche pour les requêtes `mybagof(I,p(I),L)` avec le "jeu d'essais" suivant :

```
p(b).      p(a).      p(c).      p(a).      p(d).
```

3 Générateur de plans pour robots — 7 points

On voudrait déplacer trois cubes A, B et Ç disposés sur les emplacements 1, 2 et 3 (situation **Init**) pour obtenir la configuration finale suivante (situation **Final**). On dispose pour cela d'un petit robot qui sait déplacer les cubes: quand il reçoit la commande `transfert(X,I,J)`, le robot enlève le cube `X` au sommet de la pile à la position `I` et le pose au sommet de la pile à la position `J`.

Ecrivez le prédicat `plan_robot(P,N)` où `P` est une suite de commandes pour déplacer les cubes de la configuration initiale à la configuration finale.

```
                                                       ---
                                                      | A |
 ---                                                   ---
| A |                                                 | B |
 ---       ---                                         ---
| C |     | B |                                       | C |
 ---       ---                                         ---
--------------------------          ------------------------
  1         2         3               1         2         3

          (Init)                               (Final)
```

Exemple:

```
| ?- plan_robot(P,5).

P = [transfert(A,1,2),transfert(C,1,3),
     transfert(A,2,1),transfert(B,2,3),transfert(A,1,3)]

true
```

4 Programme `coupure/3` — 5 points

Expliquez le rôle du prédicat `coupable/1` dans le programme ci dessous.

```
coupure1([V1,C1,V2|Suite],[V1,-,C1|Reste]):-
     voyelle(V1),
     consonne(C1),
     voyelle(V2),
     coupure1([V2|Suite],Reste).

coupure1([V1,C1,C2|Suite],[V1,-,C1|Reste]):-
     voyelle(V1),
     consonne(C1),
     consonne(C2),
     non_separable(C1,C2),
     coupure1([C2|Suite],Reste).

coupure1([C1,C2,C3|Suite],[C1,-,C2|Reste]):-
     consonne(C1),
     consonne(C2),
     consonne(C3),
     non_separable(C2,C3),
     coupure1([C3|Suite],Reste).

coupure1([L1|Suite],[L1|Reste]):-
     not coupable([L1|Suite]),
     coupure1(Suite,Reste).

coupable([L1,L2,L3|_]):-
     voyelle(L1),
     consonne(L2),
     voyelle(L3).
coupable([L1,L2,L3,L4|_]):-
     voyelle(L1),
     consonne(L2),
     consonne(L3),
     voyelle(L4) .
coupable([L1,L2,L3|_]):-
     consonne(L1),
     consonne(L2),
     consonne(L3).

coupure1([],[]).
```

```
coupure1([V1,C1,C2,V2|Suite],[V1,C1,-,C2|Reste]):-
          voyelle(V1),
          consonne(C1),
          consonne(C2),
          not non_separable(C1,C2),
          voyelle(V2),
          coupure1([V2|Suite],Reste).

coupure1([C1,C2,C3|Suite],[C1,C2,-|Reste]):-
          consonne(C1),
          consonne(C2),
          consonne(C3),
          not non_separable(C2,C3),
          coupure1([C3|Suite],Reste).
```