Master d'Ingéniérie Informatique de Paris 7
M1: Prolog et programmation par contraintes

Examen partiel du 8 Novembre 2005 - Durée: 1h45

Documents autorisés; le barème est donné à titre indicatif.

Exercice 1 (**5 points**) Pour chaque requête de la liste suivante, indiquer la réponse de l'interpreteur Prolog, sans motiver votre choix, ou bien donner une réponse à la ligne ..., si vous pensez que toutes les réponses précedentes sont fausses:

1. `X is 9+2, Y = X*2.`

   (a) `[ INSTANTIATION ERROR- in arithmetic: expected bound value ]`

   (b) `X = 11, Y = 11*2`

   (c) `X = 11, Y = 9`

   (d) ...

2. `[A,B,C]=[3,D,u], B is 77, B > A.`

   (a) `[ INSTANTIATION ERROR- in arithmetic: expected bound value ]`

   (b) `no`

   (c) `A = 3, B = D = 77, C = u`

   (d) ...

3. `[A,B,C]=[3,D,u], B is 77, not(B > A).`

   (a) `[ INSTANTIATION ERROR- in arithmetic: expected bound value ]`

   (b) `no`

   (c) `A = 3, B = D = 77, C = u`

   (d) ...

4. `[A,B,C]=[3,7|L],length(L,X).`

   (a) `A = 3, B = 7, L = C, X = 1`

   (b) `no`

   (c) `A = 3, B = 7, L = [[C]], X = 1`

   (d) ...

5. `[A|[B|C]]=[[[]]].`

   (a) `A = [[]], B = C = []`

   (b) `no`

   (c) `A = B = C = []`

   (d) ...

Exercice 2 (**10 points**)

1. Écrire un prédicat `insert(+Entier,+ListeArgument,-ListeResultat)` pour insérer un entier donné dans une liste d'entiers donnée, que l'on suppose triée, de telle façon que la liste résultat reste triée.

   Exemple d'utilisation:

```
  ?- insert(7,[3,5,13],R).
R = [3,5,7,13] ?
yes
```

2. Écrire un prédicat `tri(+ListeArgument,-ListeResultat)` pour trier une liste d'entiers. L'algorithme à utiliser est celui du "tri par insertion":

   - La liste vide est triée.
   - Pour trier la liste `[N|L]`, on commence par trier `L`, puis on insère `N` dans le résultat en utilisant `insert`.

   Exemple d'utilisation:

```
  ?- tri([5,3,13,1],R).
R = [1,3,5,13] ?
yes
```

3. rappel: le terme `[P,D|R]` dénote une liste dont le premier élément est `P`, le deuxième est `D` et le reste est la liste `R`.

   Écrire un prédicat `trie(+ListeArgument)` qui réussit si l'argument est une liste triée, echoue sinon.

   Exemples d'utilisation:

```
  ?- trie([6,13,90]).
yes
  ?- trie([4,2]).
no
```

4. Si $l = [n_1, ..., n_k]$ est un liste d'entiers de longueur k, alors la *liste des intervals* de l est la liste d'entier $[n_2 - n_1, n_3 - n_2, ..., n_k - n_{k-1}]$, de longueur $k - 1$ (pour $k \leq 1$, la liste des intervals est vide).

   Écrire un prédicat `intervals(+ListeArgument,-ListeResultat)` pour calculer la liste des intervals d'une liste donnée.

   Exemple d'utilisation:

```
  ?- intervals([7,9,5,47],R).
R = [2,-4,42] ?
yes
```

5. On dit q'une liste d'entiers l est *parfaite* si

- l est vide, ou bien
- l est triée et la liste des intervals de l est parfaite (cette définition récursive est bien fondée car, si l n'est pas vide, la liste des intervals de l est strictement plus courte que l).

Écrire un prédicat `parfaite(+ListeArgument)` qui réussit si l'argument est une liste parfaite, echoue sinon.

Exemples d'utilisation:

```
   ?- parfaite([1,3,7]).
yes
   ?- parfaite([1,5,7]).
no
```

Exercice 3 (**5 points**) On considère le programme:

```
a(1).
a(2).
b(3).
b(4).
c(5).
c(6).
test_cut(R):-a(X),b(Y),c(Z),R is X*Y*Z.
```

1. Quelles sont les réponses données par l'interpreteur Prolog à la requête
   `test_cut(X).`
   dans l'ordre?

2. Pour changer cet ensemble de réponses, on s'autorise à modifier le programme: **l'unique modification autorisée est l'insertion d'une coupure dans le corps de la règle qui définit** `test_cut`. Pour chaque ensemble de réponses de la liste suivante donner la version modifiée du programme qui produit exactement cet ensemble à la requête `test_cut(X).`, si une telle version existe.

   (a) 15,18.

   (b) 15,18,20,24.

   (c) 15,18,30,36.

   (d) 15.