M1 Informatique – Université Paris Cité

Programmation Logique et par Contraintes

Examen partiel du 7 novembre 2024 - Durée: 2 heures

Documents autorisés; le barème est donné à titre indicatif.

La lisibilité et les commentaires du code seront pris en compte dans l'évaluation.

Exercice 1 (**6 points**) Pour chacune des requêtes suivantes, donner le résultat renvoyé par l'interpréteur Prolog (sans justifier). Dans les cas où plusiers résultats sont renvoyées par des retours en arrière succéssifs, les indiquer tous, séparées par de ";". Le prédicat `mem/2` est défini par les deux règles suivantes:

```
mem(X,[X|_]):-!.
mem(X,[_|L]):-mem(X,L).
```

1. `Q=..[+,X,Y], X is 3, Y is 4, Q>6.`
2. `Q=..[*,X,Y], X is 2, Y is 3, Q==6.`
3. `member(X,[1,2]),!,member(Y,[1,2]).`
4. `mem(X,[1,2]),mem(Y,[1,2]).`
5. `setof(L,mem(3,L),H),length(H,K).`
6. `setof(L,member(3,L),H),length(H,K).`
7. `not(not(((L=[1];L=[1,1]),length(L,K),K>1))).`
8. `not(not(((L=[1];L=[1,1]),!,length(L,K),K>1))).`

Exercice 2 (**6 points**) La *description* d'une liste d'entiers non négatifs $l = [n_1, ..., n_k]$ est la liste d'entiers non négatifs h de longueur $1 + max\{n_i : 1 \leq i \leq k\}$ telle que, pour tout i allant de 0 au $max\{n_i : 1 \leq i \leq k\}$, $h[i]$ est le nombre d'occurrences de i dans l. Par exemple, la description de $[2, 3, 0, 0]$ est $[2, 0, 1, 1]$, la description de $[2, 0, 1, 1]$ est $[1, 2, 1]$ et la description de $[1, 2, 1]$ est $[0, 2, 1]$. Dans les points qui suivent, vous utiliserez d'éventuels prédicats auxiliaires que vous définirez, ainsi que les prédicats spécifiés aux points précédents, même si vous ne les avez pas écrits.

1. Écrire un prédicat `desc(+liste,-desc)` qui implémente la définition ci-dessus. On aura par exemple:

```
[eclipse 1]: desc([2,0,1,1],L).
L = [1,2,1]
Yes
[eclipse 2]: desc([1,2,1],L).
L = [0, 2, 1]
Yes
```

Un *cycle de descriptions* est une séquences de listes $l_1, l_2, \ldots, l_n$ telles que, pour tout $0 < i < n$ l_{i+1} est la description de l_i et $l_1 = l_n$. Une liste l est *cyclique* s'il existe un cycle de descriptions $l, ..., l$. Par exemple, $[1, 1, 1]$ est cyclique, vu que $[1, 1, 1], [0, 3], [1, 0, 0, 1], [2, 2], [0, 0, 2], [2, 0, 1], [1, 1, 1]$ est une cycle de descriptions.

2 Écrire un prédicat `is_cyclic(+liste)` qui réussit si `liste` est cyclique.

Exercice 3 (**8 points**) Soit **J** un jeu combinatoire et `move(+position1,-position2)` le prédicat qui implémente la règle du jeu, c'est-à-dire que l'appel `move(X,Y)` génère toutes les positions *atteignables en un coup* à partir de la position `X` (dans la suite de l'exercice, on on appelera "attegnables à partir de `X`" ces positions atteignables en un coup à partir de `X`), les unes après les autres .

Par exemple, si **J** est le jeu soustractif 1-2-3 vu en cours, on obtient :

```
[eclipse 1]: move(17,Y).
Y = 16
Yes (solution 1, maybe more) ? ;
Y = 15
Yes (solution 2, maybe more) ? ;
Y = 14
Yes (solution 3)
```

Le prédicat suivant implémente la stratégie de "force brute" pour **J** :

```
wins(X):-move(X,Y),not(wins(Y)).
```

Considérons les prédicats `play1(+pos1,-pos2)` et `play2(+pos1,-pos2)`, qui utilisent cette stratégie pour jouer au jeu **J**, définis comme suit:

```
play1(X,Y):-move(X,Y),not(wins(Y)),!.

play2(X,Y):- (move(X,Y),not(wins(Y)),!) ; (move(X,Y),!).
```

1. Si **J** est le jeu soustractif 1-2-3, quels sont les résultats des appels `play1(4,Y)` et `play2(4,Y)` respectivement? Justifiez vos réponses.

Le but de cet exercice est de définir un prédicat `play3(+pos1,-pos2)` qui, parmi les positions atteignables à partir de `pos1`, renvoie celle qui minimise la *fonction de Sprague-Grundy* du jeu **J**. Cette fonction est définie récursivement sur l'ensemble des positions de **J** de la manière suivante :

$$sg(p) = mex\{sg(p') : p' \text{ est atteignable à partir de } p \text{ dans le jeu } \mathbf{J}\}$$

où mex désigne la fonction qui associe à un ensemble P de nombres naturels le plus petit nombre naturel qui n'appartient pas à P (en particulier, $mex(\emptyset) = 0$). Dans les points qui suivent, notamment 3 et 4, vous utiliserez le prédicat `move` de **J**, d'éventuels prédicats auxiliaires que vous définirez, ainsi que les prédicats spécifiés aux points précédents, même si vous ne les avez pas écrits.

2 Écrire un prédicat `mex(+liste_entiers,-entier)` qui implémente la fonction mex. Par exemple :

```
[eclipse 2]: mex([],N).
N = 0
Yes
[eclipse 3]: mex([7,0,1],N).
N = 2
Yes
```

3 Écrire un prédicat `sg(+position,-entier)` qui implémente la fonction de Sprague-Grundy pour le jeu **J**. Par exemple, si **J** est le jeu soustractif 1-2-3 :

```
[eclipse 4]: sg(0,N).
N = 0
Yes (0.00s cpu)
[eclipse 5]: sg(5,N).
N = 1
Yes
```

4 Écrire un prédicat `play3(+pos1,-pos2)` qui choisit, parmi les positions atteignables à partir de `pos1` dans le jeu **J**, une position `pos2` dont la valeur de Sprague-Grundy est minimale. Si `pos1` est une position finale, l'appel `play3(pos1,_)` échoue. Par exemple, si **J** est le jeu soustractif 1-2-3 :

```
[eclipse 6]: play3(17,Y).
Y = 16
Yes
[eclipse 7]: play3(4,Y).
Y = 1
Yes
[eclipse 8]: play3(0,Y).
No
```