Université Paris 7 - DEUG-MIAS Année 2004-2005

TD de *Logique et Circuits* n° 5

Définitions inductives: les listes

Exercice 1 L'ensemble des listes d'entiers naturels $\mathcal{L}(\mathbb{N})$ est la clotûre inductive de l'ensemble de règles suivantes :

$$\frac{}{nil \in \mathcal{L}(\mathbb{N})} \qquad \frac{n \in \mathbb{N} \text{ et } l \in \mathcal{L}(\mathbb{N})}{cons(n,l) \in \mathcal{L}(\mathbb{N})}$$

Définir les fonctions $longueur, max, min : \mathcal{L}(\mathbb{N}) \to \mathbb{N}$.

Soit $\mathcal{R}_1 \subseteq \mathcal{L}(\mathbb{N}) \times \mathcal{L}(\mathbb{N})$ le relation définie par $(l_1, l_2) \in \mathcal{R}_1$ si $longueur(l_1) \leq longueur(l_2)$.

1. Prouver que $\mathcal{R}_1$ est un préordre.
2. Quel est l'ensemble ordonné canoniquement associé à $\mathcal{R}_1$?
3. L'ordre de cet ensemble ordonné est-il
   - total?
   - bien fondé?

Repondre aux mêmes questions relativement à la relation $\mathcal{R}_2$ définie par $(l_1, l_2) \in \mathcal{R}_2$ si $max(l_1) \leq max(l_2)$ et $min(l_1) \geq min(l_2)$.

Exercice 2

1. Calculer les ensembles inductifs définis par les règles suivantes :

   (a)

   $$\frac{}{nil \in \mathcal{A}} \qquad \frac{n, m \in \mathbb{N} \text{ et } l \in \mathcal{A}}{cons(2n, cons(2m+1, l)) \in \mathcal{A}}$$

   (b)

   $$\frac{}{nil \in \mathcal{B}} \qquad \frac{l \in \mathcal{B}}{cons(longueur(l), l) \in \mathcal{B}}$$

2. Donner une définition inductive de l'ensemble de listes d'entiers dont chaque élément est la somme de tous les éléments suivants.

Exercice 3 En OCaml, la liste vide est notée `[]` et la concaténation d'un élément `a` et d'une liste `l` est notée `a::l`. Ainsi, la liste formée des éléments x, y, z est notée (par exemple) `x::y::z::[]` ou `[x;y;z]`.

1. Écrire une fonction `concat elem liste` concaténant un élément et une liste.
2. Écrire une fonction testant si une liste est vide en complétant le code suivant

```
let est_vide liste=
  match liste with
    [] -> ...
  |a::l-> ...
;;
```

3. Écrire une fonction récursive `let rec affiche liste` affichant les éléments d'une liste d'entiers. **Note :** L'expression `print_int x` affiche l'entier `x`.
4. Tester (à la main) cette fonction avec la liste `[2;3;4]`.
5. Écrire une fonction récursive `let rec affiche_inverse liste` affichant les éléments d'une liste d'entiers dans l'ordre inverse. **Note :** Pour afficher les éléments dans l'ordre inverse, il suffit d'inverser l'appel récursif et l'affichage.
6. Tester (à la main) cette fonction avec la liste `[2;3;4]`.
7. Écrire une fonction récursive `let rec affiche_palindrome liste` affichant les éléments d'une liste d'entiers dans l'ordre puis dans l'ordre inverse. Cette fonction affichera `2 3 4 4 3 2` sur l'entrée `[2;3;4]`.
8. Tester (à la main) cette fonction avec la liste `[2;3;4]`.
9. Écrire une fonction récursive `let rec est_triée liste` testant si une liste est triée pour l'ordre standard `<`.
10. Tester (à la main) cette fonction avec la liste `[2;3;4]` et `[2;4;3]` par exemple.

Exercice 4 On souhaite écrire une fonction `ordre` de type `'a list -> 'a list -> bool` décrivant l'ordre dictionnaire entre listes. On adopte la convention que la liste vide est plus petite que toute autre suite.

1. Écrire une fonction `let ordre_0 liste1 liste2` testant si le premier élément de la `liste1` est plus petit que celui de la `liste2`. Si une des suites est vide, la fonction renvoie une erreur.
2. Écrire une fonction récursive `let rec ordre liste1 liste2` testant si `liste1` est plus petite que `liste2`.
3. Tester.