Université Paris 7 - DEUG-MIAS Année 2004-2005

TP de *Logique et Circuits* n° 4

Définitions inductives

Exercice 1 Le but de cet exercice est de travailler sur des listes.

1. Définir une fonction `somme` renvoyant la somme d'une liste d'entiers.
2. Tester la fonction `somme`.
3. Définir une fonction `max_liste` renvoyant le maximum des éléments d'une liste.
4. Quel est le type de la fonction `max_liste`?
5. Tester la fonction `max_liste`.
6. Définir une fonction `est_triée` qui teste si une liste est triée ou non.
7. Quel est son type ?
8. La tester sur `["aa";"bb";"cc"]`, `["cc";"bb"]` et `[37;45;75]`.
9. Définir une fonction `un_sur_deux` qui prend une liste et renvoie la liste formée des éléments $1, 3, 5, \ldots$ de la liste. Par exemple `un_sur_deux [45;65;32;48;456]` renverra `[45;32;456]`.
10. Tester la fonction `un_sur_deux`.

Exercice 2 On veut manipuler des expressions arithmétiques. Pour cela on définit un type

```
type expr =
    Constante of int
  | Plus of expr * expr
  | Fois of expr * expr
  | Moins of expr;;
```

1. Définir une expression `e1` de type `expr` pour représenter $6 \times (6 + 1)$.
2. Écrire une fonction `affiche_entiers` qui prend une expression et affiche la liste des entiers qui apparaissent dans sa définition.
3. Évaluer `affiche_entiers e1`.
4. Écrire une fonction `entiers` qui prend une expression et renvoie la liste des entiers qui apparaissent dans sa définition. La concaténation de deux listes `l1` et `l2` s'écrit `l1@l2`.
5. Calculer `entiers e1`.

6. On cherche à définir la fonction d'évaluation.
   (a) Quel type doit avoir une fonction d'évaluation ?
   (b) Que doit faire la fonction d'évaluation sur l'expression `Constante 12` ?
   (c) Que doit faire la fonction d'évaluation sur l'expression `Plus(e1,e2)` ?
   (d) Définir une fonction `évalue` évaluant une expression.
   (e) Évaluer `e1`.
7. Définir une fonction `égal exp1 exp2` testant l'égalité de deux expressions, c'est-à-dire si deux expressions représentent le même entier.

Exercice 3 Cet exercice propose l'écriture de deux fonctions pour inverser une liste.

1. Écrire une fonction `reverse` qui prend une liste et renvoie la liste inversée. Pour cela, `reverse liste` concaténera la liste inverse de la queue de `liste` et la tête de `liste`.
2. Pour essayer d'optimiser la fonction précédente, on peut définir une fonction `reverse_linéaire` avec deux arguments de la manière suivante :
   - `reverse_linéaire liste accumulateur` observe son argument `liste`. Si `liste` est non vide, elle empile la tête de `liste` sur `accumulateur` et fait un appel récursif.
   - Si `liste` est vide, elle renvoie `accumulateur`.

   Pour inverser une liste `l`, on évaluera `reverse_linéaire l []`. Écrire cette fonction.
3. Tester ces deux fonctions.