Définitions par induction

Florian Bourse

1 Expressions arithmétiques

On définit en OCaml une expression arithmétique comme suit, avec un exemple de valeur :

Expressions arithmétiques

```ocaml
type arith = N of int | Add of arith * arith | Mult of arith * arith;;

let exemple = Add (Mult (Mult (Add (N 52, N 53), N 52),
 Add (N 52, N 53)), Mult (Add (N 52, N 53), N 52));;
```

Question 1. Écrire une fonction `nombres : arith -> int` qui compte le nombre de nombres dans une expression arithmétique.

Question 2. Écrire une fonction `operateurs : arith -> int` qui compte le nombre d'opérateurs dans une expression arithmétique.

Question 3. Que peut-on dire des résultats des deux fonctions précédentes sur une même expression arithmétique ? le démontrer.

Question 4. Écrire une fonction `eval : arith -> int` qui évalue une expression arithmétique et donne son résultat.

Question 5. Écrire des fonctions `prefixe : arith -> string`, `postfixe : arith -> string` et `infixe : arith -> string` qui permettent de transformer une expression arithmétique en un chaîne de caractères qui la représente en notation préfixe, postfixe ou infixe.

On souhaite à présent ajouter une variable x pour obtenir des expressions littérales :

Expressions littérales à opérateurs binaires

```ocaml
type expr_bin = N of int | Var | Add of expr_bin * expr_bin
        | Mult of expr_bin * expr_bin;;
```

Question 6. Écrire une fonction `eval_eb : expr_bin -> int -> int` qui prend en entrée une expression littérale et un entier n et qui renvoie le résultat de l'expression si la variable x vaut n.

Question 7. En réalité, les expressions que l'on manipule ici sont des polynômes. Écrire une fonction `degree : expr_bin -> int` qui calcule le degré d'un polynôme représenté par une expression arithmétique.

Question 8. Écrire une fonction `egales : expr_bin -> expr_bin -> bool` qui teste l'égalité entre deux expressions littérales.

Nous souhaitons à présent obtenir la liste des coefficients du polynôme. Procédons par étapes. Premièrement, nous allons autoriser nos opérateurs $+$ et $\times$ à opérer sur un nombre arbitraire d'opérandes :

Expressions littérales

```
type expr = N of int | Var | Add of expr list | Mult of expr list;;
```

Question 9. Écrire une fonction `regroupe : expr_bin -> expr` qui transforme une expression littérale utilisant des opérateurs binaires en expression littérale qui alterne les additions et les multiplications.

Question 10. Écrire une fonction `developpe : expr -> expr` qui transforme une expression littérale en une expression littérale composée d'une somme de produits.

Question 11. Écrire une fonction `coeffs : expr -> int list` qui donne la liste des coefficients du polynôme représenté par une expression littérale.

2 Ensembles de mots construits par induction

Dans cet exercice, on demande de donner les règles de construction qui permettent de définir par induction un ensemble A, mais on entend par là que l'ensemble des termes construits à partir de ce règles serait en bijection avec A.

Question 12. Donner les règles de construction qui permettent de construire l'ensemble des mots de la forme $<^n \mid >^n$ pour $n \in \mathbb{N}$ c'est-à-dire avec n caractères $<$, un caractère $|$, et enfin n caractères $>$.

Question 13. Donner les règles de construction qui permettent de construire l'ensemble des mots de la forme $<^n \mid >^m$ pour $(n, m) \in \mathbb{N}^2$ c'est-à-dire avec n caractères $<$, un caractère $|$, et enfin m caractères $>$.

Question 14. Donner les règles de construction qui permettent de construire l'ensemble des mots de la forme $<^n \mid >^m$ pour $(n, m) \in \mathbb{N}^2$ avec $n < m$.

Question 15. Donner les règles de construction qui permettent de construire l'ensemble des mots sur $\Sigma = \{a, b\}$ (qu'on notera Σ^* dans la suite de l'exercice).

Question 16. Donner la définition de ℓ_a qui à un mot de Σ^* associe son nombre de a. Définir de même ℓ_b.

Question 17. Donner les règles de construction qui permettent de construire l'ensemble suivant. Justifier que votre proposition est correcte.

$$\mathcal{L} = \{u \in \Sigma^* \mid \ell_a(u) = \ell_b(u) + 1, \text{ et pour tout } v \text{ préfixe strict de } u, \ell_a(v) < \ell_b(v) + 1\}$$

Question 18. Que représentent ces mots ? Avec quel ensemble inductif déjà vu en cours $\mathcal{L}$ est-il en bijection.