Programmation fonctionnelle

Répétition 5

22 mars 2007

Correction des exercices proposés

1. Ecrire une fonction `compose-fgf` qui prend comme argument une fonction f et renvoie une fonction qui prend comme argument une fonction g et qui renvoie la fonction $f \circ g \circ f$.

2.$^{\star\clubsuit}$ Ecrire une fonction `compose-fgab` qui prend comme argument deux fonctions f et g, ainsi que deux entiers a et b et renvoie la fonction

$$\underbrace{f \circ \ldots f}_{a} \circ \underbrace{g \circ \ldots g}_{b}$$

3.$^{\star\clubsuit}$ Ecrire une fonction `compose-fa` qui prend comme argument une fonction f et deux entier a, b et renvoie la fonction

$$x \mapsto \underbrace{f \circ \ldots f}_{a}(b^x)$$

4.$^{\star\clubsuit}$ Ecrire un prédicat `prime?` qui détermine si un entier strictement positif est premier ou non.

5.$^{\star\clubsuit}$ Un carré magique de dimension n est un tableau de taille $n \times n$ dans lequel chaque nombre de 1 à $n \times n$ apparaît une fois, et dont les sommes des lignes, des colonnes et des diagonales sont égales. Ecrire un prédicat magic? qui vérifie si un carré de nombres est magique ou non. On représente un carré de nombres de dimension `n` par une liste de `n` listes contenant chacune `n` éléments.

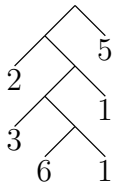
`(magic? '((4 9 2) (3 5 7) (8 1 6)))` $\Rightarrow$ `#t`

`(magic? '((4 4 7) (5 3 7) (8 1 6)))` $\Rightarrow$ `#f`

Exercices sur les arbres

6.$^{\star}\clubsuit$ Ecrire une fonction `depth-first` qui prend comme argument un arbre binaire complet — chaque nœud a 0 (une feuille) ou 2 fils (un nœud interne) — dont uniquement les feuilles sont étiquetées, et renvoie la liste des étiquettes des feuilles, obtenue par un parcours en profondeur d'abord, et de gauche à droite, de l'arbre.

Par exemple, `depth-first` appliquée à l'arbre



renvoie la liste `(2 3 6 1 1 5)`.

On choisira et spécifiera une représentation adéquate des arbres.

Variante: même exercice avec un arbre non plus binaire mais avec un nombre quelconque de fils, dont tous les nœuds sont étiquetés.

7.$\clubsuit$ On considère des arbres binaires complets (chaque nœud a exactement 0 ou 2 fils) dont les feuilles sont étiquetées par des symboles atomiques. Les nœuds internes ne sont pas étiquetés.

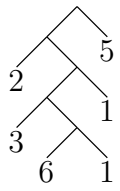
Soit `a` un tel arbre. Simplifier `a` consiste à supprimer dans cet arbre les feuilles redondantes. Ainsi tout nœud ayant deux fils étiquetés par le même symbole est remplacé par ce symbole et ainsi de suite.

Ecrire une fonction `simplify` qui prend pour argument un arbre binaire complet `a` et qui retourne l'arbre binaire complet simplifié correspondant.

Exercices proposés

8.* Ecrire une fonction `breadth-first` qui prend comme argument un arbre binaire complet — chaque nœud a 0 (une feuille) ou 2 fils (un nœud interne) — dont uniquement les feuilles sont étiquetées, et renvoie la liste des étiquettes des feuilles, obtenue par un parcours en largeur d'abord, et de gauche à droite, de l'arbre.

Par exemple, `breadth-first` appliquée à l'arbre



renvoie la liste `(5 2 1 3 6 1)`.

On choisira et spécifiera une représentation adéquate des arbres.

Variante: même exercice avec un arbre non plus binaire mais avec un nombre quelconque de fils, dont tous les nœuds sont étiquetés.

9.$^{\star\clubsuit}$ Un entier naturel est *parfait* s'il est égal à la somme de ses diviseurs positifs propres. Ecrire un prédicat `perfect?` déterminant si son argument est un nombre parfait.

Exemple: Le nombre 28 est parfait parce que $28 = 1 + 2 + 4 + 7 + 14$; le nombre 32 n'est pas parfait parce que $32 \neq 1 + 2 + 4 + 8 + 16$.