

INFO0054 - Programmation fonctionnelle

Répétition 5: Les spécificités de la programmation fonctionnelle

Jean-Michel BEGON

23 Mars 2017

Les listes variables d'arguments

Exercice 1.

Que renvoie l'évaluation des expressions suivantes :

(+) (*) (list)

Exercice 2.

Ecrire une fonction `my+` à un nombre variable d'arguments et qui renvoie la somme de ces éléments.

Closure et factory

Exercice 3.

Définir les fonctions `symmetrize` et `anti-symmetrize`, qui prennent comme argument une fonction $f : \mathbb{R} \rightarrow \mathbb{R}$ et qui renvoient respectivement les fonctions

$$f' : \mathbb{R} \rightarrow \mathbb{R} \quad x \mapsto \frac{f(x) + f(-x)}{2}$$

et

$$f' : \mathbb{R} \rightarrow \mathbb{R} \quad x \mapsto \frac{f(x) - f(-x)}{2}$$

Définir ensuite une fonction `func-op` à trois arguments, un opérateur `op` de $\mathbb{R} \times \mathbb{R} \rightarrow R$, et deux fonctions unaires `f` et `g`. `func-op` renvoie la fonction unaire `h` telle que

$$\forall x \in \mathbb{R} : h(x) = \text{op}(f(x), g(x))$$

Redéfinir ensuite `symmetrize` et `anti-symmetrize` à partir de `func-op`.

Exercice 4.

Définir une fonction `compose-n` qui renvoie la fonction unaire donnée en argument `n` fois composée avec elle-même.

Variante :

Écrire 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 $f \circ g \circ f$.

Variante 2 :

Écrire une fonction `compose-fgab` qui prend comme argument deux fonctions f et g , ainsi que deux entiers a et b et renvoie la fonction

$$\underbrace{f \circ \dots \circ f}_a \circ \underbrace{g \circ \dots \circ g}_b$$

Variante 3 :

Écrire une fonction `compose-fa` qui prend comme argument une fonction f et deux entiers a, b et renvoie la fonction

$$x \mapsto \underbrace{f \circ \dots \circ f}_a(b^x)$$

Les listes variables d'arguments et closure

Exercice 5.

Écrire une fonction `linear-map-factory` à un nombre variable d'arguments réels et qui renvoie une fonction prenant une liste de réel en argument et renvoyant le produit scalaire entre cette liste et les arguments encapsulés.

Exercice 6.

Écrire une fonction `mymap` qui prend une fonction $f : D_1 \times \dots \times D_n \rightarrow Y$ ($n \geq 1$) ainsi que n listes l_1, \dots, l_n telles que $|l_i| = m$ et les éléments de l_i appartiennent à D_i ($1 \leq i \leq n$) et qui renvoie une liste ys de m éléments telle que $ys_j = f(l_1^{(j)}, \dots, l_n^{(j)})$ ($1 \leq j \leq m$).

`(mymap (lambda (x y) (+ (* x 2) y)) '(1 2 3) '(1 1 1)) ==> '(3 5 7)`

Exercice 7.

Écrire une fonction `curry` qui prend deux arguments, une fonction $f : D_1 \times \dots \times D_n \rightarrow Y$ et un élément $d_1 \in D_1$ et qui renvoie une fonction $h : D_2 \times \dots \times D_n \rightarrow Y$ telle que $(h d_2 \dots d_n) = (f d_1 d_2 \dots d_n)$.

CPS

Exercice 8.

Écrire une fonction `sqrt*` qui à tout entier strictement positif n associe le nombre

$$\sqrt{n + \sqrt{n-1 + \dots + \sqrt{1}}}$$

de manière classique (récursion directe) et en CPS.

Exercice 9.

Écrire une fonction `sqrt*-inv` qui à tout entier strictement positif n associe le nombre

$$\sqrt{1 + \sqrt{2 + \dots + \sqrt{n}}}$$

en CPS.

Les structures de données

Exercice 10.

Proposer une implémentation d'une file à l'aide de deux listes.