

Série d'exercices #3

IFT-2035

September 22, 2009

Micro optimiseur

Soit le code Haskell ci-dessous:

```
data Exp = Enum Int          -- Une constante
         | Eplus Exp Exp     -- e1 + e2
         | Etimes Exp Exp    -- e1 * e2

optimize :: Exp -> Exp
```

Exp est un type qui représente des expressions simples incluant uniquement des opérateurs arithmétiques. Écrire la fonction *optimize* qui va essayer de simplifier une expression en éliminant les multiplications par 1 et 0, ainsi que les additions à 0.

Sans variables

La programmation *point-free* ou *sans variable* consiste en un style de programmation fonctionnelle où l'on combine des fonctions pour en construire d'autres plutôt que d'utiliser la forme λ . Les fonctions de base sont généralement appelées des *combineurs*. En utilisant les fonctions prédéfinies telles que $(.)$ (composition de fonctions, habituellement notée \circ), $(=)$, $(/)$, **not** ainsi que les fonctions ci-dessous:

```
flip f x y = f y x
map2 f (x:xs, y:ys) = f x y : map2 f (xs, ys)
map2 f _ = []
```

mais sans utiliser λ (ni un λ implicite caché dans une définition, bien sûr), définir les fonctions suivantes:

```
pas_zero 2 ==> True      pas_zero 0 ==> False
moitie 13 ==> 6.5
somme [1,4,8] ==> 13
produit_scalaire ([1,2,3], [2,3,4]) ==> 20
```

Que vaut: `flip (.) moitie sqrt 18 ?`

Des fonctions pour table

Définir en Haskell des fonctions pour gérer des tables associatives sans utiliser de constructeur (tels que `(:)` ou des types algébriques). Plus précisément, définir:

<code>empty = \x -> error "Not found"</code>	Une table vide
<code>add x v t</code>	Ajouter un lien $x \mapsto v$ à la table <code>t</code>
<code>lookup t x</code>	Renvoie la valeur <code>v</code> liée à <code>x</code> dans <code>t</code>

De telle manière que:

```
t = add "x" 1 (add "y" 2 empty)
lookup t "y" ==> 2
lookup t "z" ==> <error>
```

Vu que les constructeurs de données ne peuvent pas être utilisés, les tables seront nécessairement représentées par des fermetures (il faudra utiliser des fonctions d'ordre supérieur).