

Série d'exercices #2

IFT-2035

September 15, 2009

Quicksort

Implanter en Haskell une variante de *quicksort* pour des listes d'entiers. En clair, trier une liste comme suit:

1. choisir un élément, que l'on nommera le pivot.
2. partitionner la liste en deux sous-listes d'éléments plus petits resp. plus grands que le pivot.
3. trier récursivement les deux sous-listes.
4. combiner les listes triées et le pivot en une liste triée.

Le type sera: $quicksort :: [Int] \rightarrow [Int]$.

Il faudra peut-être définir une ou plusieurs fonctions auxiliaires.

L'opération de concaténation de deux listes s'écrit `++` en Haskell:

$$[1, 2] ++ [4, 5, 6] \equiv (++) [1, 2] [4, 5, 6] \rightsquigarrow^* [1, 2, 4, 5, 6]$$

Finalement, généraliser la fonction de tri précédente pour pouvoir l'appliquer à des listes quelconques (pas seulement `[Int]`), en passant un argument supplémentaire qui indique l'opération de comparaison à utiliser.

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.

Nombres de Church

Il est possible d'encoder les nombres naturels en utilisant seulement des fonctions. Soient les déclarations suivantes:

$$\begin{aligned} \text{zero} &= \lambda f \rightarrow \lambda x \rightarrow x \\ \text{succ } n &= \lambda f \rightarrow \lambda x \rightarrow f (n f x) \\ \text{nat2int } n &= n (\lambda x \rightarrow x + 1) 0 \end{aligned}$$

1. Inférer le type de chacune des 3 expressions.
I.e. donner non seulement le type, mais aussi une justification pas à pas de pourquoi le type que vous avez trouvé est en effet correct.
2. Montrer les étapes de l'évaluation de l'expression:

$$\text{trois} = \text{succ} (\text{succ} (\text{succ } \text{zero}))$$

I.e. appliquer la règle de β -réduction (appel de fonction) autant que possible jusqu'à obtenir une valeur.

3. Passer alors cette valeur à *nat2int*, i.e. écrire les étapes de l'évaluation de l'expression:

$$\text{nat2int } \text{trois}$$

Note: pour éviter de s'embrouiller il peut-être utile de systématiquement renommer les arguments f et x pour qu'ils soient uniques, surtout dans le calcul de *trois* pour que chaque *succ* utilise des noms de variable différents (e.g. f_1, f_2, \dots).