Série d'exercices #5

IFT-2035

May 18, 2021

5.1 Une classe de conteneurs

Soit la classe de types suivante:

```
class Conteneur a where
  size    :: a → Int
  subset  :: a → a → Bool
  reverse :: a → a
```

Où `reverse` renvoie un conteneur dont les éléments sont dans l'ordre inverse.

1. Donner une définition pour l'instance `Integer` (on n'utilisera pas les nombres négatifs), où chaque bit représente la présence du nombre correspondant, donc $11 = 2^0 + 2^1 + 2^3$ représente l'ensemble $\{0, 1, 3\}$.

2. Donner une définition pour l'instance `[Int]`. Pour vous aider vous pouvez utiliser la fonction prédéfinie `elem :: Eq a ⇒ [a] → [a] → Bool`.

3. Donner une définition pour l'instance `Maybe a`, qui ne peut contenir qu'un élément au maximum.
   Rappel: `data Maybe a = Nothing | Just a`

5.2 Évaluateur

```
type Var = String

-- Expressions du code source en forme ASA.
data Exp = Enum Int               -- Une constante
         | Evar Var               -- Une variable
         | Elambda Var Exp        -- Une fonction
         | Ecall Exp Exp          -- Un appel de fonction

-- Valeurs renvoyées.
data Val = Vnum Int               -- Un nombre entier
         | Vfun Var Exp           -- Une fonction

elookup x ((x1,v1):env) =
    if x == x1 then v1 else elookup x env

eval env (Enum n) = Vnum n
eval env (Evar x) = eval (elookup x env)
eval env (Elambda arg body) = Vfun arg body
eval env (Ecall fun actual) =
    case eval env fun of
      Vfun formal body -> eval ((formal,actual):env) body
```

Soit les déclarations ci-dessus utilisées pour un interpréteur:

1. Donner le type des deux fonctions. Vérifier que le code est typé correctement et corriger les éventuelles erreurs.

2. Cet évaluateur implante-t-il l'appel par valeur ou par nom?
   Indiquer quelle partie du code vous indique clairement la réponse.
   Le changer pour qu'il implante l'autre genre de passage d'arguments.

3. Cet évaluateur implante-t-il la portée lexicale ou dynamique?
   Indiquer quelle partie du code vous indique clairement la réponse.
   Le changer pour qu'il implante l'autre genre de portée.

Points bonus:

- Discuter de l'influence de l'ordre d'évaluation de Haskell sur vos réponses aux points 1 et 2. Comment faut-il changer ce code pour que l'évaluateur reproduise les règles de portée utilisées par le langage Haskell (i.e. que votre évaluateur implante la portée dynamique si Haskell utilisait la portée dynamique, et qu'il implante la portée statique si Haskell utilisait la portée statique).

5.3 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:

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

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 (i.e. il faudra utiliser des fonctions d'ordre supérieur).