Série d'exercices #$3\frac{1}{2}$

IFT-2035

September 19, 2024

Un petit évaluateur

Soit les déclarations suivantes pour un mini-langage d'expressions:

```
type Var = String

-- Expressions du code source en forme ASA.
data Exp = Enum Int           -- Une constante
         | Evar Var           -- Une variable
         | Elet Var Exp Exp   -- Une expr "let x = e1 in e2"
         | Ecall Exp Exp      -- Un appel de fonction
-- Valeurs renvoyées.
data Val = Vnum Int             -- Un nombre entier
         | Vprim (Val → Val)    -- Une primitive
```

Les fonctions prédéfinies sont les quatre opérations arithmétiques, liées aux variables "+", "-", "*", et "/", respectivement. Ces fonctions prennent deux arguments qui sont passés de manière currifiée. Par exemple une expression telle que "let $x = 3$ in $x + 4$" est représentée par la structure suivante de type *Exp*:

```
sampleExp = Elet "x" (Enum 3)
                 (Ecall (Ecall (Evar "+") (Evar "x")) (Enum 4))
```

L'environnement initial `env0` prédéfini les quatre fonctions:

```
mkPrim :: (Int → Int → Int) → Val
mkPrim f = Vprim (λ(Vnum x) → Vprim (λ(Vnum y) → Vnum (f x y)))

-- L'environnement initial qui contient toutes les primitives.
type Env = [(Var, Val)]
env0 :: Env
env0 = [("+", mkPrim (+)), ("-", mkPrim (-)),
        ("*", mkPrim (*)), ("/", mkPrim div)]
```

Écrire la fonction *eval* qui prend un environnement qui décrit les variables liées (et leur valeur) ainsi qu'une expression et qui renvoie le résultat de l'évaluation de l'expression. I.e.:

```
eval :: Env → Exp → Val
```

de sorte que `eval env0 sampleExp` renvoie `Vnum 7`.