Laboratorul 7: MicroHaskell

MicroHaskell

Fișierul microHaskell.hs conține un mini-limbaj funcțional, împreună cu semantica lui denotațională.

```
type Name = String

data  Value  =  VBool Bool
        |VInt Int
        |VFun (Value -> Value)
        |VError

data  Hask  = HTrue | HFalse
        |HIf Hask Hask Hask
        |HLit Int
        |Hask :==: Hask
        |Hask :+:  Hask
        |HVar Name
          |HLam Name Hask
          |Hask :$: Hask
        deriving (Read, Show)
```

1) Completați funcția de evaluare a unei expresii de tip `Hask`.

Definim comanda:

```
run :: Hask -> String
run pg = showV (hEval pg [])
```

Astfel, `run pgm` va întoarce rezultatul evaluării (rulării) programului `pgm`.

2.1) Scrieți mai multe programe si rulați-le pentru a vă familiariza cu sintaxa.

2.2) Adăugați operația de înmulțire pentru expresii, cu precedență mai mare decât a operației de adunare. Definiți semantica operației de înmulțire.

2.3) Folosind funcția `error`, înlocuiți acolo unde este posibil valoarea `VError` cu o eroare care să precizeze motivul apariției erorii.

2.4) Adăugați expresia `HLet Name Hask Hask` ca alternativă în definirea tipului

`Hask`. Semantica acestei expresii este cea uzuală: `HLet x ex e` va evalua `e` într-un mediu în care `x` are valoarea lui `ex` în mediul curent. De exemplu, dacă definim

h1 = HLet “x” (HLit 3) ((HLit 4) :+: HVar “x”)

atunci `run h1` va întoarce “7”.

3) Făcând o copie a programului, înlocuiți tipul funcției `hEval :: Hask -> HEnv -> Value` cu `hEval :: Hask -> HEnv -> Maybe Value` inlocuind aruncarea erorilor cu utilizarea elementului `Nothing`.

4) La fel ca exercitiul 3, doar ca rezultatul functiei va fi de tip Either String Value, unde varianta de eroare va fi de tip Left cu un mesaj string corespunzator.