プログラム言語特論レポート02

author : M1 丹野雄太
number : 4555235023
date : 2024-01-23
repository : https://github.com/yttnn/haskell_tokuron/tree/main/program_lang_report02

目次

必須1

- すみません、まったく分かりませんでした！！
- 今回の課題、一つも解けなかったです...

必須2

(a)

- これは書くだけなので省略...

(b) globalEnvを作る

- ガチで分からなかったので、ChatGPT先輩とペアプログラミングした
  - builtInFnsを評価するのとかは、思いつかなかった
  - foldrとか完全に忘れてた

```
globalEnv :: Env
globalEnv = foldr addBuiltInFn emptyEnv builtinFns
  where addBuiltInFn (name, def) env = updateEnv name (evalBuiltInFn def
env) env

evalBuiltInFn :: [Char] -> Env -> Val
evalBuiltInFn def env = expval (parseProg def) env
```

- これで試してみたが、`fourthPower`や`evenp`、`oddp`が実行時エラーになった
  - builtInFnsの順番に依存している状態になっていそうなので直す
  - それが下のエラー

```
ghci> ev "evenp 4"
*** Exception: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/base/GHC/List.hs:1644:3 in base:GHC.List
  errorEmptyList, called at libraries/base/GHC/List.hs:87:11 in
base:GHC.List
  badHead, called at libraries/base/GHC/List.hs:83:28 in base:GHC.List
  head, called at ./Parser.hs:227:19 in main:Parser
```

- `parseProg = fst . head . pProg . lexProg`を見ると、関数合成しているので、pProgの戻り値が空になっているのが原因か？
- `ghci> globalEnv`すると、同じ感じになる
    - そもそも、globalEnvが壊れている気がするが、直し方分からない...

```
ghci> globalEnv
[("square",VClosure "x" (Bexpr Mul (Var "x") (Var "x"))
[("fourthPower",VClosure "x" (Apply (Var "square") (Apply (Var "square")
(Var "x"))) [("abs",VClosure "x" (If (Rexpr LessThan (Var "x") (Num 0))
(Bexpr Sub (Num 0) (Var "x")) (Var "x")) [("evenp",*** Exception:
Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/base/GHC/List.hs:1644:3 in base:GHC.List
  errorEmptyList, called at libraries/base/GHC/List.hs:87:11 in
base:GHC.List
  badHead, called at libraries/base/GHC/List.hs:83:28 in base:GHC.List
  head, called at ./Parser.hs:227:19 in main:Parser
```

任意3-1

Paser.hs改造

- `NewParser.hs`がなさそうなので、Parserも自分で書いてみる！
    - `pFun`でラムダ式をパースしているので、ここを参考に、`pSFun`を実装する
    - BNFの通りに、`pFormal`を作り、それを`pVariable`があった位置に置く
    - `pFormal`がやることは、`@`があることをチェックして、`pVariable`を呼ぶこと
    - 実装は、`pFun`を見よう見まねでやる
        - `pSeq`, `pSeqSnd`の使い分けがよくわからない（やりたいことは、パーサの結合？だと思うが。。）
        - とりあえず、パーサのコード見た感じ、`pSeqSnd`を使えばよさそうな気がするためそうする
        - `pFun`は`pExpr1`の定義のところにいるので、ここに`pSFun`も登録しておく（よくわかってない）

```
-- Parser.hs
pExpr = pFoldAlt [pExpr1, pFun, pSFun, pLet, pLetrec, pIf]
~~~~~~~~~~~~~~
pFormal :: Parser Token Token
pFormal = (pLiteral "@" `pSeqSnd` pVariable)
~~~~~~~~~~~~~~
pSFun :: Parser Token Expr
pSFun =
  (pLiteral "lambda" `pSeqSnd` pFormal `pSeq` pLiteral "." `pSeqSnd` pExpr)
  `pUsing` uncurry SFun
```

- これで、一応それっぽくパースできるようになった

```
ghci> parseProg "lambda @x . x * x"
SFun "x" (Bexpr Mul (Var "x") (Var "x"))
```

CBNs.hs改造

- `expval (Apply ..)`に、`VStrictClosure`の対応をすればよさそうなので、やってみる
- 変更したコード
  - Strictの時だけ、そのまま評価することが目的のため、`SFun`のときに特別なことをするようにした（かった）
  - whereの中に`s_newenv`などのstrictの時に使うものと、`v_newenv`などの非strictのときに使うものを書いた
  - 本当は、strictと非strictで別に分ける必要があるが、どう書けばいいか分からなかったので、とんでもないコードになっている

```
expval (SFun x e) env = VStrictClosure x e env
expval (Apply e1 e2) env
  | e2 == SFun = f `seq` s `seq` expval s_body s_newenv
  | otherwise  = f `seq` v `seq` expval v_body v_newenv
  where f = expval e1 env
        v = delay e2 env
        s = expval e2 env
        VClosure v_x v_body v_env' = f
        VStrictClosure s_x s_body s_env' = s
        v_newenv = updateEnv v_x v v_env'
        s_newenv = updateEnv s_x (Evaled s) s_env'
```

- コンパイルできなかった
  - ApplyはExprを引数に持つからいいのでは？と思ったが違うことが分かった
  - `Variable -> Expr -> Expr`がどこから来ているのかが分からない
  - これが分からない原因は、Haskell力の無さ

```
CBNs.hs:84:11: error:
    • Couldn't match expected type ‘Expr’
                  with actual type ‘Variable -> Expr -> Expr’
```

感想

- 受けてよかった＆学部生のころにやりたかったなと思いました
- ただ、Haskellは自分の処理能力の限界を越えている気がしました...
  - 概念は理解できたが、全く書けなかったです
  - 手続き的でないので、そもそもの考え方を変える必要がありそうかなという印象です
- 型がドキュメントだったり、高度な抽象化ができるのは独特で興味深かったです

- ここまで書けないと悔しい＆数学と絡みを理解したいので、もう少し勉強しようかなと思います
- ありがとうございました