2015 年 9 月

2015-09-23 (Wed)

GHC 7.12 では、-fext-core オプションないよとか言われたので、別の PC に入っている 7.6.3 でためす。

```
module Sample4 where
f True = "true message"
f False = "false message"
```

まず、-ddump-simpl の結果：

```
==================== Tidy Core ====================
Result size of Tidy Core = {terms: 8, types: 5, coercions: 0}

Sample4.f :: GHC.Types.Bool -> [GHC.Types.Char]
[GblId, Arity=1]
Sample4.f =
  \ (ds_deM :: GHC.Types.Bool) ->
    case ds_deM of _ {
      GHC.Types.False -> GHC.CString.unpackCString# "false message";
      GHC.Types.True -> GHC.CString.unpackCString# "true message"
    }
```

次に、-fext-core の結果：

```
%module main:Sample4
  main:Sample4.f :: ghczmprim:GHCziTypes.Bool ->
                    ghczmprim:GHCziTypes.ZMZN ghczmprim:GHCziTypes.Char =
    \ (dsdeM::ghczmprim:GHCziTypes.Bool) ->
        %case (ghczmprim:GHCziTypes.ZMZN ghczmprim:GHCziTypes.Char) dsdeM
        %of (wildX3::ghczmprim:GHCziTypes.Bool)
          {ghczmprim:GHCziTypes.False ->
             ghczmprim:GHCziCString.unpackCStringzh
             ("false message"::ghczmprim:GHCziPrim.Addrzh);
           ghczmprim:GHCziTypes.True ->
             ghczmprim:GHCziCString.unpackCStringzh
```

```
            ("true message"::ghczmprim:GHCziPrim.Addrzh)};
```

これ、普通っぽく書き直すと、

```
f :: Bool -> [Char]
f = \(x :: Bool) ->
 %case ([Char]) x %of (_ :: Bool)
  { False -> unpackCString ("false message" :: Addrzh);
    True -> unpackCString ("true message" :: Addrzh)};
```

って感じか。% case (aty) exp %of vbind alt ; alt の aty には、case 式の型、 vbind には、exp を束縛する変数束縛が書かれるということか (この例では、束縛したところで用いられないので wildcard になってる)。

2015-09-18 (Fri)

[Bunny] Pattern

去年の 7 月にいちど書いた Pattern.hs を元に書き直す。つぎは subst。去年とちがって、Typing.Expr に対してきちんと作用するように書き直す必要あり。

2015-09-14 (Mon)

[Bunny] Core

Core 言語むけのデータ型を定義した (src/Core.hs)。そろそろ deriving Show だけではつらいので、wl-pprint を使った pretty printing[*1]を導入する。

2015-09-08 (Tue)

[Bunny] 型クラスのメンバ関数の扱い

Main.>>= が unbound identifier になっている件について。

多相をどうするのかという話をちゃんと考えないといけないのだけど、ひとまずはトップレベルの関数宣言にしてあげればいいはずだよね。

たとえば、次のようなことをすると、multiple declarations になるので、メンバ関数はそのモジュールのトップレベル（？）の名前になる。

```
a >> b = a + b

class Hoge h where
```

[*1] 正確には prettier printing

```
  (>>) :: h -> h -> h
```

問題は、今回の Main.>>= には型宣言しかなくて、関数定義がないので、結局 unbound になるだろう点。

これは、とりあえず tqd_t.hs のなかで (>>=) = undefined と仮の定義を書いてやりすごそう。後に、コンパイラが暗黙のうちにこれにあたる定義を暗黙に足してやればいいように思う。

なおした → d8ed8e1088d21e6919ba89880104166268160d82[*2]

まだまだ tqd.hs の型推論できるまでの道のりは遠いように思っていたのだけど、とりあえず通ってしまった。

ひとまず、`sample/drive_semant.exe < testcases/tqd_t.hs` の結果ファイル: tqd_infered.txt[*3]

tqd_t.hs は、バグ・未実装回避のための修正をこちらに書いておこうと思って用意したものだったんだけど、修正はそんなに多くはならなかった。

```
$ diff testcases/tqd_t.hs <(cat testcases/tqd.hs testcases/minlib.hs )
8c8
< -- main :: IO ()
---
> main :: IO ()
25d24
<   (>>=) = error ">>= is not defined."
```

一個目は、`types do not unify: (TCon (Tycon "IO" (Kfun Star Star)),TCon (Tycon "Main.IO" (Kfun Star Sta`
となってしまうのでコメントアウトしたやつ。二個目は今日の件。

今日はこんなとこかなー。型推論の結果を見ないといけないな。

いきなり大きな結果を見るのもおっくうなので、ちっさい結果もおいておこう。

sample1.hs: main = putStrLn "Hello, world!"

=>

```
[([],[[("Main.main",[([],Ap (Var "Prim.putStrLn") (Lit (LitStr "Hello, world!")))])]])]
["Main.main" :>: Forall [] ([] :=> TAp (TCon (Tycon "Main.IO" (Kfun Star Star))) (TCon (Tycon "()" S
```

sample2.hs: main = putStrLn s where s = "Hello, world!"

=>

```
[([],[[("Main.main",[([],Let ([],[[("Main.l0.l0.s",[([],Lit (LitStr "Hello, world!"))])]]) (Ap (Var
```

[*2] https://github.com/unnohideyuki/bunny/commit/d8ed8e1088d21e6919ba89880104166268160d82

[*3] http://uhideyuki.sakura.ne.jp/files2015/tqd_infered.txt

```
["Main.main" :>: Forall [] ([] :=> TAp (TCon (Tycon "Main.IO" (Kfun Star Star))) (TCon (Tycon "()" S
```

そりゃそうと、自分で、自分がつくったこの Wiki の記法忘れててやばい。整形済テキストの書き方がわからなかった[*4]。

いじょ。

2015-09-07 (Mon)

[Bunny] 型推論つづき

コミットメッセージに日本語打てそうとか思って適当に書いてたら文字化けしまくった。

今日やったのは、

- preludeClasses で、”Show” だったクラス名を ”Prim.Show” に変更
- ”context reduction” が出るようになったので、 preludeClasses に Prim.Show のインスタンスを追加
- unbound identifier: Prim.error が出るようになったので、preDefined に追加

この調子でパリパリ直していこうかなと思ったんだけど、つぎは `unboud identifier: Main.>>=` が出るようになってしまった。これは、クラス定義に書いてあるのだけど、それがクラス環境に適切に反映されていないということで、preDefine しましょうという話じゃないな。

というわけで、今日はひとまず、これくらい。約一時間だ。

2015-09-06 (Sun)

[Bunny] 型推論つづき

まだ tqd.hs の型推論が完了するところまでいっていない。

今日やったこと：

- preludeClasses が空っぽだったのを直した。今度は unbound identifier: Prim.putStrLn とでた。
- PreDefined.hs にて、Prim.putStrLn の Assumption を追加。types do not unify になった。

あとで直す：- FlexibleContexts オプションはあとではずす（まだ意味がわかってない）

2015-09-05 (Sat)

X205AT にいくつかインストールするのだ。すでに初日にいれたのが、chrome, xyzzy, そして cygwin 上の w3m くらい。

今日いれるのは、

- Haskell Platform 7.10.2a
- Git 2.5.1 for Windows (git bash)
- Putty 0.65

[*4] Markdown と異なる書き方を選んだので

くらいかな。Ruby はとりあえずいれないでおこう。

tanakh さんの[*5]を見ながら、chocolatey なるものをインストールしてみる。

chocolatey install emacs ののち、haskell-mode も無事にインストールできた。ほっ。

[Bunny] ひさしぶりの再開

ひさしぶりすぎる。

$ sample/drive_semant.exe < <(cat testcases/tqd.hs testcases/minlib.hs) > /dev/null drive_semant.exe: qname not found: >

こうなるのは、プログラムはただしくて、Ord クラスを宣言してないからのような気がする。

~/ghc-7.10.2/libraries/ghc-prim/GHC/Classes.hs

minlib.hs に適当な関数宣言を追加したら、こうなった。

drive_semant.exe: insts: Ord

エラーは Typing.hs のなか。Class 環境に Ord がないよといってるみたい。

つづく

2015-09-04 (Fri)

軽いし、起動も早いので、ぱっと開いて使えるな。この Wiki に書く機会も増やせるといい。

2015-09-05 (Sat)

ASUS X205TA が来た。

注文してあった ASUS X205TA Red が来た。軽い！これはすごいな。鞄にいれても「あー重。あ、今日はノート PC 入ってるんだった」とかならない。

おっさんなので、さっそく CTRL キーと CAPS キーをいれかえ。ただし、ctrl2cap インストールしたらキーボードが利かなくなって焦った。ソフトキーボードをつかって、なんとかアンインストールして[*6]、その後、レジストリを書き換える方法を実施。

やっぱり、もとの配列のままじゃ無理そうだ。

キーボードの使い心地は悪くはない。もちろん、HHKB とは比較にならないけど、持ち運びやすさを勘案すれば十分おつりがくる。

意図せずタッチパッドに触ってしまうことがなく、それにまつわるストレスもない。

[*5] http://tanakh.jp/posts/2013-05-23-windows-setup.html

[*6] コマンドラインプロンプトから操作しないといけないので、ソフトキーボードなかったら詰んでるところだった