ひとり Advent Calendar 2016 年 11 月 11 日

これは、ひとり Advent Calendar 2016 6 日目の記事です。

多相（Ad-hoc 多相のほう）については、関数の多重定義の仕組みや、単相関数内で多相関数を呼ぶ際の扱い（その場で辞書をつくってわたす）までは一応動いている。だが、そうでない多相関数の扱い、つまり、辞書をうけとって、その辞書を他の多相関数に渡すような関数における辞書渡しの処理が未実装となっている。

未実装はいいんだけど、実装済みの部分についても、あやふやな理解のまま書いて、書いてからしばらくたっているので、いまどうなっているのか、あんまり覚えていない。今日は、そのあたりの確認をした。

まず、確認したのが、多相関数に対する辞書渡しが、どのように「未実装」なのかについて。これは、DictPass の最後の方にあった：

```
tcBind :: Bind -> Bind
tcBind (Rec bs) = Rec $ map tcbind bs
  where
  tcbind (v@(TermVar _ ([] :=> t)), e) = (v, tcExpr e t)
  tcbind b@((TermVar _ (q :=> _)), e) = b -- #overloaded#
```

多相関数への束縛については何もしないという非常に単純なことになってた。ないす、おれ。

で、この `#overloaded#` というやつ、自分で書いておいて、おぼろげにしか覚えていなかったので、コード見ながら思い出した内容を以下にメモっておく。

これは、型クラス定義における各メソッド定義を表す「プレースホルダ」である。型クラス定義におけるメソッド定義は、基本的に名前と型のみからなるので、それを保持している。（デフォルト定義は別途扱っている。class table (ctab) だったかな）

`#overloaded# Prelude.> "Prelude.Ord"` の形 (App Term (App Term Lit)) をとる。

で、このプレースホルダは、ずっとこのまま CORE -> STG とそのまま伝播していって、CodeGen で「トランポリン」関数として出力されることになる。だから、Core での辞書渡し形式への変換 (DictPass にて行ってる）において、こいつをほったらかし[*1]にするのは正しいのだけど、多相関数をまるごとほったらかすのは間違っていた。

そこで、一時的に tcBind を修正して、ここでの分類を以下のようにした：

- 単相関数（一応動いている）
- 多相関数
  - `#overloaded#` 式（現状まま、なにもしなくてよい）
  - それ以外の多相関数（★未実装）

ほんとは、`#overloaded#` 以外の関数は、多相も単相も統一的に扱うことが出来そうだし、それが望ましいはずだが、いま単相関数の処理が一応動いているのだから、それを温存したまま進めて、あとできれいにする

[*1] 「ほったらかす」、「ほったらかしにする」は、「放っておく」というくらいの意味です。

方針で。

そうなると、あとは（★未実装）と書いた、「それ以外の多相関数」に対処すれば良いのだが…もうひとつ、大きな問題がある。インスタンスのメソッド。

実は、IO の >> がうまく動いているように見えるのが謎だったのだけど、これは、ほんとは多相関数なのに、いま束縛定義側では単相として扱われていて、呼び出し側では多相に見えていたせいで、結果的にたまたまうまくいっていただけだった[*2]。

呼び出し側では >> に辞書渡しされ、渡された側では、なにもきにせず >>= を呼んでいた、という。(まだなんか怪しい気もする）

そういうわけで、多相の一応の完成までには、

1. インスタンスのメソッドをちゃんと多相として扱う
2.「それ以外の多相関数（★未実装)」の扱い

の二つが必要。

なお、(★未実装）にとりかかるために小さいサンプルが欲しいと思って test8b.hs を書いているときに、あらたなバグも発見してしまった。(`Issue#2016#Nov001`)

よし、がんばろう。12/15 に間に合うんかな〜。

[*2] これは勘違いでした。11/14 に追記しています。