2018 年 1 月

2018-01-15 (Mon)

Firemacs いれた。

この自分専用 Wiki とかの文章編集を、ブラウザの Text まどでやるのはつらいので、w3m + xyzzy なんかをつかっていたんだけど、ふと思い立って、Chrome 拡張でいいのがないか探してみた…んだけど、Chrome 拡張ではいまいちいいのが見つからず。

どうも、Firemacs がいいらしいので、ひさしぶりに Firefox をインストールした。なんか、最新の Firefox には対処中とのことで、ESR をインストールして、firemacs をインストール。

わー、良い。とても良い。

2018-01-10 (Wed)

[Bunny] パターンマッチコンパイル周りのコード整理にとりかかる前に…

ぼちぼち、ユーザ定義型などの実装にとりかかろうと考えて、data 宣言の取り込みをきれいにすることを考えていた。data 宣言は、型推論向けの仮定 (`[Assump]`) だけでなく、パターンマッチのための arity, constructors のために蓄積されていく必要がある。

そこで、Efficient Compilation of pattern-matching を読み直しつつコードの見直しをしていこうとし始めた（昨日あたりから)。

ふと、この論文にのっている mappairs や nodups は処理できるんかなと試してみたら…だめじゃん！ ふつうの map にあたる mymap サンプルでもだめ。

- DictPass.tyapp での型変数生成が fresh じゃなかったのがだめでした(済)→ これで mymap, mappairs はクリア
- Rename.renPat が ListExpr 未対応（済）

でも、まだ nodup はだめ。こちらは、ガードが未対応なんだと思う。

そこで、ガードを用いない nodup0 を用意してみたけど、そちらも、辞書渡し形式のところでだめっぽい。

```
$ ./test-compile.sh testcases/nodup0.hs
source file: testcases/nodup0.hs
dst dir: /nodup0
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc.exe: Error: dictionary not found: Prelude.==, (Tyvar ".t23" Star,[(Tyvar ".t26"
CallStack (from HasCallStack):
  error, called at src\DictPass.hs:168:32 in main:DictPass
```

調査なんかは明日以降にするとして、もしかして型推論がしくってるのか？と思って –ddump-assump で確

認してみたが、nodup 関数の型推論はあっているっぽい。(nodups :: Eq a => [a] -> [a] にあたる内容)

```
Main.nodups :: Forall [Star] ([IsIn "Prelude.Eq" (TGen 0)] :=> ([g0] -> [g0]))
```

というわけで、型推論はあってるけど辞書渡しがうまくないということで継続。Case の scrut の処理がまだなのが怪しいな。

2018-01-06 (Sat)

[Bunny] リファクタリング＆コードウォーク

いちばん長大だった Semant にここ数日とりかかっていて、そういえば、これを書いていたときは色々悩みながらで混乱していたり、Haskell 的な書き方がいまいちだったりで、今見ると結構書き直すところがあった。そうすると多少は行数も減るのだが、やっぱり 900 行は長すぎる。書き直しでこれが半分以下になるわけではない。

そこで、Semant は、リネーミング部とトポロジカルソート部に分割することにした。リネーミング部は、それでもまだ長かったので、さらに RenUtil と Rename の 2 つにわけた。

各モジュールは 500 行以内を目安にしよう。そうなると Typing がオーバしているが、これは分割したくないので特例でおっけいしておきたい。

```
$ ./showlogs.sh
src/Absyn.hs                    151 lines       0 warnings      0 hint
src/BindGrouping.hs             164 lines       0 warnings      0 hint
src/CodeGen.hs                  469 lines       0 warnings      2 hints
src/CompilerOpts.hs             91 lines        0 warnings      0 hint
src/Core.hs                     44 lines        0 warnings      0 hint
src/DDumpAssump.hs              15 lines        0 warnings      0 hint
src/DDumpCore.hs                15 lines        0 warnings      0 hint
src/Desugar.hs                  18 lines        0 warnings      0 hint
src/DictPass.hs                 221 lines       0 warnings      0 hint
src/NameMangle.hs               27 lines        0 warnings      0 hint
src/Pattern.hs                  121 lines       0 warnings      0 hint
src/PreDefined.hs               182 lines       0 warnings      0 hint
src/Rename.hs                   496 lines       0 warnings      0 hint
src/RenUtil.hs                  213 lines       0 warnings      0 hint
src/Semant.hs                   50 lines        0 warnings      0 hint
src/STG.hs                      78 lines        0 warnings      0 hint
src/Symbol.hs                   39 lines        0 warnings      0 hint
src/TrCore.hs                   257 lines       0 warnings      1 hint
```

```
src/TrSTG.hs                    38 lines        0 warnings      0 hint
src/Types.hs                    47 lines        0 warnings      0 hint
src/Typing.hs                   644 lines       0 warnings      0 hint
app/Main.hs                     110 lines       0 warnings      0 hint
----
                                3490 lines      0 warnings      3 hints
```

かつては無視していたコンパイラ警告に対処していて気付いたんだけど、パタンを網羅していないといって怒られるのは、データ定義が適切でないのが本質的な問題のようだ。

たとえば、クラス宣言だけに作用するような関数を書いたら、「宣言はほかにもあるのに網羅してないよ」となる。これは、それぞれ異なる処理が行われる、つまり、本質的に異なるものをまぜたデータ型を定義しているのがよろしくないのかなと思う。

いまは、ワイルドカードでうけて error "must not occur" とかしているのだが、こんなその場しのぎ対処ではなくて、Absyn の定義みなおそう。

あと、この調子でやっていくと、ある程度までは急激にコードがよくなっていくのだけど、ある程度以上は「凝りすぎ」になってしまう傾向にあるので、そうならないよう注意した方がよさそう。ずーっと同じ目でみるより、すこし経ってから見た方かいいというのもあるし。

cwalk1 ブランチの目標期限をきめて、ちゃちゃっと次に進みたい。

あ、そうだ、書き直していて「おっ」と思ったことを、忘れぬうちにメモっておこう。

- 無駄にオレオレ再帰しているのは、ことごとく書き直し対象になった。map, concat など（そのモナド版 mapM なども）をうまく使った方がいい。
  - 再帰呼び出しは強力すぎるので、繰り返し毎に処理が独立しているなら map でそうとわかるように表示する意味もある（もちろん、記述が簡潔になる点も大事）
- Renaming モナドの状態にいれる必要のないものをいれていたケースもあり。単なるアキュムレータみたいなのはいれたくない。その状態により、そのうえでの計算が変わるようなものをいれる。
  - やたら似たような一時変数のネーミングで苦慮していたのも、オレオレ再帰の廃止や、無駄な一時変数の削除（一回しか使わないなら、変数にいれないで済ます）などで解決してしまった。

ほかにもあった気がしたけど…。やっぱり、データ型の階層みなおす必要を実感できたのが一番大きいかな。