047: showpair0.hs がランタイムに IntegerShowFunc: must not occur

↑ up

- issued: 2020-04-25
- 分類： A サンプルコードが fail
- status: Closed (2020-04-25)

概要

list や pair, triple などの Show クラス処理に取り掛かるまえに、複数の辞書をうけとる関数のコンパイル状況をためそうとして、不具合に遭遇。

showpair0.hs:

```
f (a, b) = "(" ++ show a ++ "," ++ show b ++ ")"

main = putStrLn $ f (1, 'a')
```

これの core があやしい：

```
(Main.main :: (Prelude.IO ())) =
    ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ()))) $
      ((((Main.f :: ([Prelude.Show t2,Prelude.Show t3] :=> ((((,) t3) t2) -> [Prelude.Char])))
            ${Prelude.Integer Prelude.Show})
          ${Prelude.Char Prelude.Show})
        (((Prelude.(,) :: (t4 -> (t5 -> (((,) t4) t5))))
            (1 :: ([Prelude.Num t6] :=> t6)))
          'a')))

(Main.f :: ([Prelude.Show t21,Prelude.Show t22] :=> ((((,) t22) t21) -> [Prelude.Char]))) =
    \(Main.f.DARG0 :: ^^c3^^84) (Main.f.DARG1 :: ^^c3^^84) ->
        \(_Main.f.U1 :: ([Prelude.Show t7,Prelude.Show t8] :=> (((,) t8) t7))) ->
            case (_Main.f.U1 :: ([Prelude.Show t7,Prelude.Show t8] :=> (((,) t8) t7))) (_Main.f.U1b
                Prelude.(,) (_Main.f.U2 :: t9) (_Main.f.U3 :: t10) :: (t9 -> (t10 -> (((,) t9) t10))
                    (((Prelude.++ :: ([t11] -> ([t11] -> [t11])))
                        "(")
                      (((Prelude.++ :: ([t12] -> ([t12] -> [t12])))
                          (((Prelude.show :: ([Prelude.Show t13] :=> (t13 -> [Prelude.Char])))
```

```
          (Main.f.DARG0 :: ^^c3^^85))
        (_Main.f.U2 :: t9)))
    (((Prelude.++ :: ([t16] -> ([t16] -> [t16])))
        ",")
      (((Prelude.++ :: ([t17] -> ([t17] -> [t17])))
          (((Prelude.show :: ([Prelude.Show t18] :=> (t18 -> [Prelude.Char])))
              (Main.f.DARG0 :: ^^c3^^85))
            (_Main.f.U3 :: t10)))  ")"))))
```

- Main.f において、Main.f.DARG0 が二か所で用いられている（片方は DARG1 であるべき）
- Main.f の型：([Prelude.Show a,Prelude.Show b] :=> ((((,) b) a) -> [Prelude.Char])：a, b 逆になってない？

Main.f.DARG0 の検索部

```
       lookupDict (k, tv) (((c, tv'), i):d')
-        | k == c  || k `elem` super ce c = Just i
-        | otherwise = Nothing
-      lookupDict _ _ = Nothing
+        | tv == tv' && (k == c  || k `elem` super ce c) = Just i
+        | otherwise = lookupDict (k, tv) d'
+      lookupDict _ [] = Nothing
```

いままで、同じクラスの述語に対応していなかったので、k ==c (たとえば ”Prelude.Show” == ”Prelude.Show” のみ成立で対応付けていても偶々 OK だった。これを、ただしく型変数の一致、かつ、クラス名が一致またはスーパークラスに一致すれば OK と変えた。

いままでは、型変数 y は用いられていなかったわけで、正しく用いるようにしたおかげで、let (TVar y) = case apply s (TVar x) of ... のパターンが [a] に未対処だったのが露見。ひとまず、lib/Prelude から Show [a] に関する部分をコメントアウトした。

```
bunnyc: Irrefutable pattern failed (TAp (TCon (Tycon "Prelude.[]" (Kfun Star Star))) (TVar (Tyvar "t
```

これでも、まだ showpair0.hs はエラーする。今度は、二つの辞書を Main.f に渡す呼び出し側における順序があやしい。

つぎに呼び出し側の不具合調査と修正が必要だが、いったん master branch にマージする（make check は通っている状態）。

呼び出し側で逆順だった件

findApplyDict の内部関数 mkdicts は、結果を逆順に返していたので、それを利用するときに逆順にすべきだった。当初 `foldl App e dicts` だったのを `foldl App e (reverse dicts)` でもよかったのだが、foldr を使う形に修正：

```
--- a/compiler/src/DictPass.hs
+++ b/compiler/src/DictPass.hs
@@ -235,8 +235,8 @@ tcExpr e@(Var (TermVar n (qv :=> t'))) qt -- why ignore qs?
                                                 ++ n ++ ", " ++ show (x,n2,y,itvars))
                                 Just v' -> mkdicts qs (Var v' : ds)
                 mkdicts _ _ = error "mkdicts: must not occur"
-          dicts <- mkdicts qv []
-          return (foldl App e dicts)
+          dicts <- mkdicts qv [] -- mkdicts returns dictionaries in reverse order
+          return (foldr (flip App) e dicts)
             {-
             where
               appliedQv :: [Expr] -> TC (Maybe (Qual Type))
```

showpair0.hs は sample161 とする。

クローズ。