092: gcdlcm.hs が通らなくなった

↑ up

- issued: 2020-05-23
- 分類： A サンプルコードが fail
- status: Closed (2020-05-27)

概要

073 の対処により、これまで通っていたテストが通らなくなってしまったもののひとつ。

lib/Prelude.hs の lcm がエラーするようになったので、これをコメントアウトしたため、lcm を呼び出していたプログラムは当然ながらコンパイルできなくなった。

gcdlcm.hs は、083 で sample219.hs として追加されたもの。

調査ログ

2020-05-27 (Wed)

pss の保持と、lookupDictArg の処理があっておらず、ネストされた Let 式において、辞書の受け渡しに失敗していた。

tcbind では、↓★部で辞書引数をうけとる関数の名前 n を保持している。そして、その二行したで、一つ外側の pss' と連結。（これがよくない）

```
tcBind :: Bind -> ClassEnv -> Maybe TcState -> (Bind, TcState)
tcBind (Rec bs) ce maybest = tbloop st0 bs []
  where
    tbloop :: TcState -> [(Var, Expr)] -> [(Var, Expr)] -> (Bind, TcState)
    tbloop st []     res = (Rec (reverse res), st)
    tbloop st (b:bs) res = let (ve, st') = tcbind b st
                           in tbloop st' bs (ve:res)

    st0 = case maybest of
            Just st' -> st'
            Nothing  -> mkTcState ce [] nullSubst 0

    tcbind :: (Var, Expr) -> TcState -> ((Var, Expr), TcState)
    tcbind (v@(TermVar n qt@(qs :=> t)), e) st
      | isOVExpr e = ((v, e), st)
```

```
      | otherwise =
        let pss = (zip qs (repeat n)) -- ★
            pss' = tcPss st
            st' = st{tcPss=(pss++pss')}
            (e', st'') = runState (tcExpr e qt) st'
            num = tcNum st''
        in
          if null qs then ((v, e'), st{tcNum=num})
          else ((v, Lam (mkVs n qs) e'), st{tcNum=num})
    tcbind _ _ = error "tcbind: must not occur."
```

lookupDictArg では、連結された pss+pss' 全体に通し番号を振っている（↓※部）ので、各関数における正しい引数番号を再現できていなかった。

```
lookupDictArg :: (Id, Tyvar) -> TC (Maybe Var)
lookupDictArg (c, y) = do
  s <- getSubst
  pss <- getPss
  ce <- getCe
  let d = zip (map (\((IsIn i t), _) -> (i, apply s t)) pss) [(0::Int)..] -- ※
      lookupDict (k, tv) (((c, tv'), i):d')
        | tv == tv' && c `isin` k = Just i
        | otherwise = lookupDict (k, tv) d'
      lookupDict _ [] = Nothing
      c1 `isin` c2 = (c1 == c2)|| (or $ map (`isin` c2) (super ce c1))
      ret = case lookupDict (c, TVar y) d of
        Nothing -> Nothing
        Just j  -> let (_, n) = pss !! j
                   in Just $ TermVar (n ++ ".DARG" ++ show j) ([] :=> TGen 100)
  return ret
```

修正内容

pss を格納するときに、番号も含めて生成してしまうことにした：

```
+++ b/compiler/src/DictPass.hs
@@ -46,7 +46,8 @@ tcBind (Rec bs) ce maybest = tbloop st0 bs []
     tcbind (v@(TermVar n qt@(qs :=> t)), e) st
```

```
       | isOVExpr e = ((v, e), st)
       | otherwise =
-        let pss = (zip qs (repeat n))
+        let ds = zipWith (++) (repeat (n ++ ".DARG")) (map show [0..])
+            pss = (zip qs ds)
             pss' = tcPss st
             st' = st{tcPss=(pss++pss')}
             (e', st'') = runState (tcExpr e qt) st'
```

lookupDictArg 側では、単純にそれを用いる：

```
@@ -203,16 +204,15 @@ lookupDictArg (c, y) = do
   s <- getSubst
   pss <- getPss
   ce <- getCe
-  let d = zip (map (\((IsIn i t), _) -> (i, apply s t)) pss) [(0::Int)..]
-      lookupDict (k, tv) (((c, tv'), i):d')
-        | tv == tv' && c `isin` k = Just i
+  let d = map (\((IsIn i t), n) -> ((i, apply s t), n)) pss
+      lookupDict (k, tv) (((c, tv'), s):d')
+        | tv == tv' && c `isin` k = Just s
         | otherwise = lookupDict (k, tv) d'
       lookupDict _ [] = Nothing
       c1 `isin` c2 = (c1 == c2)|| (or $ map (`isin` c2) (super ce c1))
       ret = case lookupDict (c, TVar y) d of
         Nothing -> Nothing
-        Just j  -> let (_, n) = pss !! j
-                   in Just $ TermVar (n ++ ".DARG" ++ show j) ([] :=> TGen 100)
+        Just s  -> Just $ TermVar s ([] :=> TGen 100)
   return ret
```

これで、lib/Prelude.hs における lcm 定義はエラーしなくなり、gcdlcm.hs, gcd2lcm2.hs ともに通るようになった。

- gcdlcm.hs -> sample227.hs