085: oddInt.hs がランタイムで abend

↑ up

- issued: 2020-05-18
- 分類： A サンプルコードが fail
- status: Closed (2020-05-27)

概要

lib/Prelude.hs に追加した、even, odd は、内部であやまって Num t => t が Integer に特殊化されてしまっていて、Int の引数をあたえると abend する。

```
(Prelude.even :: ([Prelude.Integral t1408] :=> (t1408 -> Prelude.Bool))) =
    \(Prelude.even.DARG0 :: ^^c3^^84) ->
        \(_Prelude.even.U1 :: ([Prelude.Integral t1402] :=> t1402)) ->
            ((((Prelude.== :: ([Prelude.Eq t1404] :=> (t1404 -> (t1404 -> Prelude.Bool))))
                  ${Prelude.Integer Prelude.Eq})
                ((((Prelude.rem :: ([Prelude.Integral t1405] :=> (t1405 -> (t1405 -> t1405))))
                      (Prelude.even.DARG0 :: ^^c3^^85))
                  (_Prelude.even.U1 :: ([Prelude.Integral t1402] :=> t1402)))
                 (((Prelude.fromInteger :: ([Prelude.Num t1406] :=> (Prelude.Integer -> t1406)))
                     (Prelude.even.DARG0 :: ^^c3^^85))
                   (2 :: Prelude.Integer))))
              (((Prelude.fromInteger :: ([Prelude.Num t1407] :=> (Prelude.Integer -> t1407)))
                  (Prelude.even.DARG0 :: ^^c3^^85))
                (0 :: Prelude.Integer)))
```

調査ログ

2020-05-20 (Wed)

gcd, lcm でも同じ問題が発現。gcdlcm2.hs

2020-05-21 (Thu)

oddInt については原因判明

概要にあげた core では、Prelude.Eq t1404 にあたる型変数が Prelude.Integral t1408 に一致せず、lookupDict が Nothing を返していた。（その結果 defaulting が実施され Integer に）

原因は、Eq が Integral に一致せず、Integral の supers のどれにも一致しなかったため。Eq は Integral の super の super なので、再帰的に検査しなければならなかった。

```
@@ -197,10 +197,10 @@ lookupDictArg (c, y) = do
   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' && (k == c  || k `elem` super ce c) = Just i
+        | 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
```

また、CodeGen も同様に、super の super も処理する必要があった。

```
--- a/compiler/src/CodeGen.hs
+++ b/compiler/src/CodeGen.hs
@@ -8,7 +8,7 @@ import           Symbol
 import           Typing                     (ClassEnv (..), super)

 import           Control.Monad.State.Strict
-import           Data.List                  (find, intercalate)
+import           Data.List                  (find, intercalate, nub)
 import           Data.List.Split            (splitOn)
 import qualified Data.Map                   as Map
 import           Data.Maybe                 (fromJust, fromMaybe)
@@ -486,7 +486,8 @@ emitInsts dest dicts ((qin, qcn):ctab) ce = do
   let ms = ddMethods $ fromJust $ find ((== qcn). ddId) dicts
       msM = map mangle ms
       pdname = cls2dictNameM qcn
-      supers = super ce qcn
+      f c = c : concatMap f (super ce c)
+      supers = nub $ concatMap f (super ce qcn)
       sdname = map cls2dictNameM supers
```

```
dname = cls2dictNameM $ qin ++ "@" ++ qcn
mname = modname qin
```

これによって、oddInt.hs は通るようになった ⇒ sample220.hs

だが、gcdlcm2.hs はまだダメ。こちらも、lookupDict が Nothing を返しているのだが、クラスではなく tv が不一致であるらしく、原因は別のところにあるようだ。

2020-05-27 (Wed)

092 対処により、gcdlcm2.hs も通るようになった。

- gcdlcm2.hs -> sample228.hs