009: nodup0.hs で dictionary not found

↑ up

- issued: 2020-04-09
- 分類： A サンプルコードが fail
- status: Closed (2020-04-13)

現象

nodup0.hs をコンパイルすると、以下のように dictionary not found となって失敗する。

```
$ ./test-compile.sh testcases/nodup0.hs
source file: testcases/nodup0.hs
dst dir: /nodup0
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: Error: dictionary not found: Main.nodups, (Tyvar "t19" Star,[(Tyvar "t20" Star
CallStack (from HasCallStack):
  error, called at src/DictPass.hs:193:32 in main:DictPass
```

nodup0.hs の内容は以下の通り：

```
nodups [] = []
nodups [x] = [x]
nodups (y:x:xs) = if y == x
                  then nodups (x:xs)
                  else y : nodups (x:xs)

main = putStrLn $ nodups "331223"
```

調査ログ

2020-04-10 (Fri)

f.hs: 自己参照がだめなのかなと思って、小さくした例でためす。やはりエラー。

```
f []     = 0
f (x:xs) = 1 + f xs

main = putStrLn $ show $ f "abcde"
```

f2.hs: パターンマッチ変換に難があるのかもと、初めから case 式で書いてみたが、結果は同じ：

```
f xs = case xs of
  []     -> 0
  (x:xs) -> 1 + f xs

main = putStrLn $ show $ f "abcde"
```

自己参照関数の扱いが未だだったかなぁ、と思ったがそんなはずはなく。なにしろ、qsort が既に動いている。

```
qsort :: Ord a => [a] -> [a]
qsort []     = []
qsort (x:xs) = qsort smaller ++ [x] ++ qsort larger
  where
    smaller = [a | a <- xs, a <= x]
    larger  = [b | b <- xs, b > x]

-- main :: IO ()
main =
  do let helo = "Hello, World!"
     putStrLn helo
     putStrLn.show $ qsort [3, 1, 4, 1, 5, 9, 2, 6, 5]
     putStrLn $ show $ qsort helo
```

(main の型シグネチャをつけるとエラーした、これは別の issue へ)

g.hs：これはいける：

```
g x = x + x

main = putStrLn $ show $ g 1
```

うまくいっているケースと、そうでない場合、それぞれの型を比べてみると、もしかすると、多相関数で、かつ、多相でない型変数を含む型の処理がうまくないのかもしれない。

- `f :: Num t => [t1] -> t` – NG
- `qsort :: Ord a => [a] -> [a]` – OK.
- `g :: Num a => a -> a` – OK

2020-04-11 (Sat)

＃＃＃ その 1

f.hs (↓に再掲）では、main 中い現れる `f "abcde"` の型が決まらずにエラーしている。

```
f []     = 0
f (x:xs) = 1 + f xs

main = putStrLn $ show $ f "abcde"
```

この項の型は、曖昧なのだが、default 則で解決済のはず。ためしに、この項を変数束縛してみると、通る。

fx.hs:

```
f []     = 0
f (x:xs) = 1 + f xs

x = f "abcde"

main = putStrLn $ show x
```

現状の実装では、Typing モジュールによる型推論結果は、モジュール内の全変数に関する [Assump] の形でしか後段に引き継いでいないのが原因と思われる。

まずは、この実装を改善する必要がある ⇒ 013

・・・と思ったが、Typing を大きく変えないと難しそうなので、DictPass で冗長な Type checking をするのはアリ、継続の方向でいきたい。

不具合をなおす、Defaulting は追加する、など。

その２：怪しい箇所

nodup0.hs の –ddump-core0 結果をみると、Main.l2.l0.F の型が既におかしいような…

```
source file: --ddump-core0
```

```
dst dir: /

---- ddumpCore ----
(Main.main :: (Prelude.IO ())) =
    ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ()))) $
      ((Main.nodups :: ([Prelude.Eq t2] :=> ([t2] -> [t2])))
        "331223"))

(Main.nodups :: ([Prelude.Eq t26] :=> ([t26] -> [t26]))) =
    \(_Main.nodups.U1 :: ([Prelude.Eq t3] :=> [t3])) ->
        case (_Main.nodups.U1 :: ([Prelude.Eq t3] :=> [t3])) (_Main.nodups.U1b :: ([Prelude.Eq t3] :
            Prim.: (_Main.nodups.U2 :: t6) (_Main.nodups.U3 :: [t6]) :: (t6 -> ([t6] -> [t6])) ->
                case (_Main.nodups.U3 :: [t6]) (_Main.nodups.U3b :: [t6]) of
                    Prim.: (_Main.nodups.U3 :: t12) (_Main.nodups.U4 :: [t12]) :: (t12 -> ([t12] ->
                        let
                            (Main.l2.l0.F :: (Bool -> [v131])) =
                                \(_Main.l2.l0.F.U1 :: Bool) ->
                                    case (_Main.l2.l0.F.U1 :: Bool) (_Main.l2.l0.F.U1b :: Bool) of
                                        Prim.True  :: Bool ->
                                            ((Main.nodups :: ([Prelude.Eq t19] :=> ([t19] -> [t19]))
                                              (((Prim.: :: (t20 -> ([t20] -> [t20])))
                                                  (_Main.nodups.U3 :: t12))
                                                (_Main.nodups.U4 :: [t12])))
                                        Prim.False  :: Bool ->
                                            (((Prim.: :: (t13 -> ([t13] -> [t13])))
                                                (_Main.nodups.U2 :: t6))
                                              ((Main.nodups :: ([Prelude.Eq t15] :=> ([t15] -> [t15]
                                                (((Prim.: :: (t16 -> ([t16] -> [t16])))
                                                    (_Main.nodups.U3 :: t12))
                                                  (_Main.nodups.U4 :: [t12])))))



                        in
                          ((Main.l2.l0.F :: (Bool -> [v131]))
                            (((Prelude.== :: ([Prelude.Eq t23] :=> (t23 -> (t23 -> Bool))))
                                (_Main.nodups.U2 :: t6))
                              (_Main.nodups.U3 :: t12)))
                    Prim.[]  :: [t8] ->
                        (((Prim.: :: (t9 -> ([t9] -> [t9])))
```

```
                          (_Main.nodups.U2 :: t6))
                        (Prim.[] :: [t11]))

            Prim.[]  :: [t4] ->
                (Prim.[] :: [t5])
```

これは、DictPass より前に実施されているので、まずそっちを直さないと。

2020-04-12 (Sun)

Main.10.12.F は、Rename において case 式が let 式に変換されるときに生じるのだが、この変換がしくっているようにも思えなかった。

ためし、最初から let 式で書いたもの（以下、nosub0b.hs) でも同様。

```
nodups []       = []
nodups [x]      = [x]
nodups (y:x:xs) = let
  f True  = nodups (x:xs)
  f False = y : nodups (x:xs)
  in
    f (x == y)

main = putStrLn $ nodups "331223"
```

Typing が被疑箇所なのか…なぁ。

この件、すこし寝かせておいて、ほかの事を片付けていこうかな。

2020-04-13 (Mon)

023 のために作成した、mycompare4.hs で少しわかったかもしれない。

mycompare4.hs:

```
mycompare x y = let
  f True  = EQ
  f False = let
    g True  = LT
    g False = GT
    in g (x <= y)
  in f (x == y)
```

```
main = do
  print (mycompare 1 2)
  print (mycompare 100 2)
  print (mycompare 'a' 'a')
```

これも、

```
bunnyc: Error: dictionary not found: Prelude.<=, (Tyvar "t13" Star,"Prelude.Ord",TVar (Tyvar "t12" S
CallStack (from HasCallStack):
  error, called at src/DictPass.hs:193:32 in main:DictPass
```

といってコケるので、同様のエラーなのだが、ここで –ddump-assump でみると、f, g の型は、たしかにこれであっている。

- `Main.l0.l0.l1.l0.g :: (Bool -> Prelude.Ordering)`
- `Main.l0.l0.f :: (Bool -> Prelude.Ordering)`

そうか、自由変数が多相の場合があるので、自由変数に関する辞書も生成しないといけないんだ。

・・・ということは、辞書渡し変換は、クロージャ変換後にすればよい？

⇒ クロージャ変換すればいいというものではなさそう（nodup0bx.hs)。let をリネームする際の ”level” で、外側の辞書にアクセスできるようにすればよいのかな。

⇒ tcBind において、トップレベルでは新しく環境をつくるが、let binding はネストをあつかえるように変更、解決。nodup0, nodup0b については通るようになったので、sample141, 142 とした。

nodup0bx はエラー、mycompare4 も、こちらはランタイムで abend してしまっていた。いずれも別のインシデントで扱う（それぞれ 024, 023）とし、本件はクローズ。