078: instanceshow.hs で unbound identifier: Main.show

↑ up

- issued: 2020-05-16
- 分類： A サンプルコードが fail
- status: Open

概要

Main 中で Show クラスのインスタンスを定義出来なかった問題で、lib/Prelude.hs における renaming monad の情報がひきつがれていなかったのは、RenUtil.putCdicts で辞書を上書きしているためだった。

それを修正：

```
--- a/compiler/src/RenUtil.hs
+++ b/compiler/src/RenUtil.hs
@@ -84,10 +84,11 @@ getCDicts = do
   st <- get
   return $ rnCdicts st

-putCDicts :: [DictDef] -> RN()
-putCDicts dicts= do
+appendCDicts :: [DictDef] -> RN()
+appendCDicts dicts= do
   st <- get
-  put st{rnCdicts = [(ddId d, d) | d <-dicts]}
+  let cdicts0 = rnCdicts st
+  put st{rnCdicts = cdicts0 ++ [(ddId d, d) | d <-dicts]}

 lookupCDicts :: Id -> RN DictDef
 lookupCDicts n = do
diff --git a/compiler/src/Semant.hs b/compiler/src/Semant.hs
index b7cec3c..8b51236 100644
--- a/compiler/src/Semant.hs
+++ b/compiler/src/Semant.hs
@@ -24,7 +24,7 @@ renProg m = do
   let bgs' = toBg ctbs
       as2 = map (\(n, scm, _) -> n :>: scm) $ fst $ head bgs'
   let dicts = map (trCdecl modid) cds
```

```
-  putCDicts dicts
+  appendCDicts dicts
   (itbs, ctab) <- renInstDecls ids
   tbs <- renDecls ds
```

そうすることで、少し先にすすんだが、今度は別のエラーがでるようになった：

```
$ ./tcheck testcases/instanceshow.hs
# 1. test-compile
source file: testcases/instanceshow.hs
dst dir: /instanceshow
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: unbound identifier: Main.show
```

2020-06-08 (Mon)

インスタンス宣言にあるメソッド定義が scanValueDecl2 のなかで renameVar されて、show -> Main.show の辞書が登録されてしまっているのが悪さをしていた。

そこで、仮対処として、インスタンス宣言における scanValueDecl2 による辞書更新を巻き戻すようにした：

```
scanInstDecl2 (A.InstDecl ctx t ds) = do
    lv_save <- getLvs
    dss' <- mapM (\(A.VDecl d) -> scanValueDecl2 d) ds
    putLvs lv_save
    let ds' = map A.VDecl $ concat dss'
    return (A.InstDecl ctx t ds')
```

これで先にすすむようになったが、どこかの fromJust でこけている。

既存のテストに悪影響はないようだ。make check は通る。

2020-06-09 (Tue)

fromJust がコケていたのは、CodeGen.emitInsts においてだった。インスタンスのコードを出力する際に、スーパークラスの辞書定義を参照しているが、Prelude で定義されたクラスの辞書をひきついでいなかったため失敗していた。

ひきつぐように変更することで、instanceshow.hs は通るようになった。なお、deriving Show を適用しよ

うとすると NG, 定義を少しかえたものは通った。

- instanceshow.hs : ok sample279.hs
- instanceshow2.hs: NG
- instanceshow3.hs: ok sample280.hs
- instanceshow4.hs: NG

NG だったものは別 issue にして先に進めてはどうかな。

2020-10-16 (Fri)

いくつか別件で不具合を直しているので、本 issue で残っていたテストについて再確認してみた。

instanceshow2.hs は、エラー。`src/Rename.hs:(249,15)-(263,46): Non-exhaustive patterns in function deriv`
となる。

instanceshow4.hs はコンパイル、実行はできるのだが、表示結果が違う。

bunny による表示結果：

```
Hoge 10
Fuga "nine"
X Fuga "nine"
```

runhaskell による表示結果：

```
Hoge 10
Fuga "nine"
X (Fuga "nine")
```

Show（や、Read）をきちんと「本物」にしていく必要あり。

2020-11-06 (Fri)

Haskell98 の deriving Show に関する章[*1] をみながら、Show の定義を書いてみた（instanceshow4b.hs):

```
data Hoge a b = Hoge a
              | Fuga b

instance (Show a, Show b) => Show (Hoge a b) where
```

[*1] https://www.sampou.org/haskell/report-revised-j/derived.html

```
  showsPrec d (Hoge x) = showParen (d > 10) showStr
    where showStr = showString "Hoge " . showsPrec 11 x
  showsPrec d (Fuga y) = showParen (d > 10) showStr
    where showStr = showString "Fuga " . showsPrec 11 y

data X a = X a

instance (Show a) => Show (X a) where
  showsPrec d (X x) = showParen (d > 10) showStr
    where showStr = showString "X " . showsPrec 11 x

x, y :: Hoge Int String
x = Hoge 10
y = Fuga "nine"

main = do
  putStrLn $ show x
  putStrLn $ show y
  putStrLn $ show (X y)
```

まだ、見よう見まねで書いているだけで、precedence の数字がどのように使われるのか、ちゃんと理解してないのだけど、導出される Show インスタンスの定義をこんな風にしてやればいいのだろうなと。このコードは、現状のコンパイラでも期待通りの印字をする。

2020-12-17 (Thu)

instanceshow2.hs:

```
data Hoge = Hoge Integer
          | Fuga [Char]
          deriving Show

x = Hoge 10
y = Fuga "nine"

main = do
  putStrLn $ show x
  putStrLn $ show y
```

bunny 0.9.0 での実行結果（エラー）:

```
$ bunny testrun instanceshow2.hs
/home/unno/bunny/0.9.0/bin/bunnyc -d ./jout/instanceshow2 --xno-implicit-prelude /home/unno/bunny/0.
/home/unno/bunny/0.9.0/bin/bunnyc -d ./jout/instanceshow2 --xlibrary-path /home/unno/bunny/0.9.0/lib
bunnyc: src/Rename.hs:(256,15)-(270,46): Non-exhaustive patterns in function deriveShowDecl

testrun: failed to compile instanceshow2.hs
```

instanceshow4.hs:

```
data Hoge a b = Hoge a
              | Fuga b
              deriving Show

data X a = X a deriving Show


x, y :: Hoge Int String
x = Hoge 10
y = Fuga "nine"

main = do
  putStrLn $ show x
  putStrLn $ show y
  putStrLn $ show (X y)
```

bunny 0.9.0 での実行結果（期待通りでない）:

```
$ runhaskell instanceshow4.hs
Hoge 10
Fuga "nine"
X (Fuga "nine")
$ bunny testrun instanceshow4.hs
/home/unno/bunny/0.9.0/bin/bunnyc -d ./jout/instanceshow4 --xno-implicit-prelude /home/unno/bunny/0.
/home/unno/bunny/0.9.0/bin/bunnyc -d ./jout/instanceshow4 --xlibrary-path /home/unno/bunny/0.9.0/lib
Hoge 10
```

```
Fuga "nine"
X Fuga "nine"
```

2021-03-12 (Fri)

方針

まず、instanceshow4.hs の方を対処する。つまり、優先度におうじて括弧をつける showsprec の導出を先に。

こちら[*2] も参考にするのと、中値演算子で値コンストラクタが定義されているケースにも対処したい。

そのためには、カバーするつもりのテストを先に書くのがいいのかもしれない。

上のリンク先にある Tree がいいかな。

なお、derived instance のメソッド生成は scandecl1 で行っており、その時点では中値タイプの showPrec に必要な優先順位の情報がでそろっていない。なので、この時点では `GETPREC(":^:")` のような形で directive として埋め込んでおいて、後段の renExp 時（このときには必要な情報は環境にはいっている）に解決するようにしてはどうか。

このあとの作業プラン

- genDerivedShow 関数を別ファイルに切り出す
- instanceshow4 に対応できるレベルにこれを改造する（中値タイプには未対応）
- 中値タイプの値構築子が出現するテストケースを作成する（red になる）
- Absyn に GETPREC を追加
- 新しいテストにも対応するよう改造

2021-10-24 (Sun)

全件状況確認。

まず、instanceshow2.hs の方。コンパイルエラーになる：

```
unno@unno-FMVD70GN7G ~/work/bissues/078
$ cat instanceshow2.hs
data Hoge = Hoge Integer
          | Fuga [Char]
          deriving Show

x = Hoge 10
y = Fuga "nine"
```

[*2] https://www.haskell.org/onlinereport/haskell2010/haskellch11.html#x18-18600011.4

```
main = do
  putStrLn $ show x
  putStrLn $ show y
unno@unno-FMVD70GN7G ~/work/bissues/078
$ runhaskell instanceshow2.hs
Hoge 10
Fuga "nine"
unno@unno-FMVD70GN7G ~/work/bissues/078
$ ~/prj/bunny/compiler/bin/bunny testrun instanceshow2.hs
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/instanceshow2 --xno-implicit-prelude /home/unno/p
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/instanceshow2 --xlibrary-path /home/unno/prj/bunn
bunnyc: src/Rename.hs:(253,15)-(267,46): Non-exhaustive patterns in function deriveShowDecl

testrun: failed to compile instanceshow2.hs
```

つぎに、instanceshow4.hs は、エラーにはならないが show 結果がただしくない（括弧がつかない）。

```
unno@unno-FMVD70GN7G ~/work/bissues/078
$ cat instanceshow4.hs
data Hoge a b = Hoge a
              | Fuga b
              deriving Show

data X a = X a deriving Show


x, y :: Hoge Int String
x = Hoge 10
y = Fuga "nine"

main = do
  putStrLn $ show x
  putStrLn $ show y
  putStrLn $ show (X y)
unno@unno-FMVD70GN7G ~/work/bissues/078
$ runhaskell instanceshow4.hs
Hoge 10
Fuga "nine"
X (Fuga "nine")
```

```
unno@unno-FMVD70GN7G ~/work/bissues/078
$ ~/prj/bunny/compiler/bin/bunny testrun instanceshow4.hs
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/instanceshow4 --xno-implicit-prelude /home/unno/p
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/instanceshow4 --xlibrary-path /home/unno/prj/bunn
Hoge 10
Fuga "nine"
X Fuga "nine"
```