010: sigdoc1.hs で variable not found

↑ up

- issued: 2020-04-09
- 分類： A サンプルコードが fail
- status: Closed (2020-04-10)

現象

sigdoc1.hs をコンパイルすると、Variable not found: Main.l0.l0.s となって失敗：

```
$ ./test-compile.sh testcases/sigdoc1.hs
source file: testcases/sigdoc1.hs
dst dir: /sigdoc1
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: Variable not found: Main.l0.l0.s
, fromList []
, [fromList []]
CallStack (from HasCallStack):
  error, called at src/CodeGen.hs:282:33 in main:CodeGen
```

調査ログ

2020-04-09 (Thu)

CodeGen でコケているので、型推論は通っている。途中の中間言語をダンプしてみてみる。

まずは、sigdoc1.hs の場合：

```
$ ./test-compile.sh --ddump-core testcases/sigdoc1.hs
basename: unrecognized option '--ddump-core'
Try 'basename --help' for more information.
source file: --ddump-core
dst dir: /
doCompile ... done.
implicitPrelude ... done.
doCompile ...
```

```
---- ddumpCore ----
(Main.main :: (Prelude.IO ())) =
    let

    in
      ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ())))
        (Main.l0.l0.s :: [Prelude.Char]))


bunnyc: Variable not found: Main.l0.l0.s
, fromList []
, [fromList []]
CallStack (from HasCallStack):
  error, called at src/CodeGen.hs:282:33 in main:CodeGen
```

一方、sigdoc1b.hs （これはコンパイル通る）の場合は以下：

```
$ ./test-compile.sh --ddump-core testcases/sigdoc1b.hs
basename: unrecognized option '--ddump-core'
Try 'basename --help' for more information.
source file: --ddump-core
dst dir: /
doCompile ... done.
implicitPrelude ... done.
doCompile ...
---- ddumpCore ----
(Main.s :: [Prelude.Char]) =
    "Hello, world!"

(Main.main :: (Prelude.IO ())) =
    ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ())))
      (Main.s :: [Prelude.Char]))


done.
```

renDecl の動作を、Level 管理周りや、そもそも、Absyn の仕様も含めて、きちんと確認しないといけない。

2020-04-10 (Fri)

前日の –ddump-core は比べるものがちがっていた。sigdoc1.hs は以下の通り：

```
$ cat testcases/sigdoc1.hs
main = putStrLn s
  where s :: [Char]
        s = "Hello, world!"
```

これをコンパイルすると、昨日示したとおり、core の時点で s の定義がぬけおちている。

```
(Main.main :: (Prelude.IO ())) =
    let

    in
      ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ())))
        (Main.l0.l0.s :: [Prelude.Char]))
```

では、sigdoc1.hs から s の型宣言だけ取り除いたもの：

```
main = putStrLn s
  where s = "Hello, world!"
```

これをコンパイルしたときの core は次のとおり、

```
(Main.main :: (Prelude.IO ())) =
    let
        (Main.l0.l0.s :: [Prelude.Char]) =
            "Hello, world!"


    in
      ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ())))
        (Main.l0.l0.s :: [Prelude.Char]))
```

どこで抜け落ちたのかと思い、renExp (A.LetExp ds e) に以下の通り、trace を仕掛けて比較：

```
renExp (A.LetExp ds e) = do
  enterNewLevel
  (ds', _, _) <- scanDecls ds
  tbs <- renDecls ds'
  e' <- renExp e
  exitLevel
  let bgs = toBg tbs
  trace (show bgs) $ return (Let (head bgs) e') -- TODO: (head bgs) is temporary
```

- 型シグネチャなし： `[([],[[("Main.l0.l0.s",[([],Lit (LitStr "Hello, world!"))])]])]`
- 型シグネチャあり: `[([("Main.l0.l0.s",Forall [] ([] :=> TAp (TCon (Tycon "[]" (Kfun Star Star))) (T`

どうも、この時点では正しそう。trCore が被疑個所なのでは。
trCore の該当箇所。Explicit Binding を捨てている！

```
trExpr2 (Ty.Let bg e) = do
  trace (show (Ty.Let bg e)) $ return ()
  pushBind
  ci <- getCi
  let (_, iss) = bg
      is = concat iss
      vdefs = dsgIs [] is ci
  transVdefs vdefs
  b' <- popBind
  e' <- trExpr2 e
  return $ Let b' e'
```

参考まで、上のように仕掛けた trace による表示を以下に示す：

- シグネチャなし: `Let ([],[[("Main.l0.l0.s",[([],Lit (LitStr "Hello, world!"))])]]) (Ap (Var "Prim.p`
- シグネチャあり: `Let ([("Main.l0.l0.s",Forall [] ([] :=> TAp (TCon (Tycon "[]" (Kfun Star Star)))`

解決

Explicit Binding を捨てないようにし、また、dsgIs は Implicit 以外の Bindings も扱うのだからと、名前を dsgBs に変更…

```
trExpr2 (Ty.Let bg e) = do
```

```
  pushBind
  ci <- getCi
  let (es, iss) = bg
      es' = map (\(n, _, alts) -> (n, alts)) es
      is = concat iss
      vdefs = dsgBs [] (es' ++ is) ci
  transVdefs vdefs
  b' <- popBind
  e' <- trExpr2 e
  return $ Let b' e'
```

したら、Desuger.hs では、すでに同様の問題に対処済だったことが判明。

```
dsgModule :: Id -> Ty.Program -> [Ty.Assump] -> ConstructorInfo -> Module
dsgModule modident bgs as ci =
  let
    [(es, iss)] = bgs
    is = concat iss
    es' = map (\(n, _, alts) -> (n, alts)) es
    vdefs = dsgBs [] (es' ++ is) ci
    b = translateVdefs as vdefs ci
  in
   Core.Module modident [b]
```

なぜ、そのとき一緒に直さなかったのか…。