017: Show クラスでないものでも show できてしまう

↑ up

- issued: 2020-04-12
- 分類： B 機能追加
- status: Closed (2020-04-18)

概要

クラスの継承について未実装だったときの仮実装として、`Prim.show` の型が `(a -> [Prelude.Char])` になっているため、Show のサブクラスでないものまで show できてしまう。

調査ログ

2020-04-16 (Thu)

Show Bool

まず、いちばん単純な Bool から。

Prim.show が `a -> [Char]` なこと自体は、それでよかった。ソース中に show があると、問答無用で Prim.show にリネームされていたのがよくなかった。

そこで、lib/Prelude.hs に以下を追加、それにあうよう Predefined を変更。

```
class Show a where
  show :: a -> [Char]

data Bool = False | True

instance Show Bool where
  show = Prim.show
```

Bool や Ordering など kind が * なものはうまくいったが、Show `[a]` や Show (,), Eq `[a]` など難航。

make check では 6 つ fail している状況:

- samples/sample116.hs: dictionary not found: Prelude.show, (Tyvar "t11" Star,"Prelude.Show",TAp (TCon (Tycon "Prelude.`[]`" (Kfun Star Star))) (TCon (Tycon "Prelude.Integer" Star)))
- samples/sample121.hs: context reduction, IsIn "Prelude.Show" (TAp (TAp (TCon (Tycon "(,)" (Kfun Star (Kfun Star Star)))) (TVar (Tyvar "v420" Star))) (TVar (Tyvar "v421" Star)))
- samples/sample122.hs: context reduction, IsIn "Prelude.Show" (TAp (TAp (TCon (Tycon "(,)" (Kfun Star (Kfun Star Star)))) (TVar (Tyvar "v434" Star))) (TVar (Tyvar "v433" Star)))

- samples/sample136.hs: dictionary not found: Prelude.show, (Tyvar ”t46” Star,”Prelude.Show”,TAp (TCon (Tycon ”Prelude.[]” (Kfun Star Star))) (TCon (Tycon ”Prelude.Integer” Star)))
- samples/sample138.hs: context reduction, IsIn ”Prelude.Show” (TCon (Tycon ”Main.Hoge” Star))
- samples/sample139.hs: context reduction, IsIn ”Prelude.Show” (TCon (Tycon ”Main.Hoge” Star))

kind が * -> *（またはそれ以上）のケースに対応する必要がある。少なくとも、以下の考慮が必要。

- instance 定義における扱い
- DictPass における扱い

138 と 139 は、deriving Show に未対応であるのが原因と思われる。

なお、deriving Show なコンテナに、show できないオブジェクトを「いれて」も、その時点ではエラーしない。show しようとすると（実行時ではなく静的にだが)、エラー。

たとえば、↓のコードはエラーしないが、print c しようとするとコンパイルエラーになる。

```
f x = x * x

a = [f]

data B a = B a
         deriving Show

b = B f

data C a = C a

c = C f

instance (Show a) => Show (C a) where
  show (C a) = "C " ++ show a

main = do print $ (head a) 5
          let B g = b
          print $ g 4
          let C h = c
          print $ h 3

          -- print c
```

2020-04-18 (Sat)

いくつか問題があるが、小さいものからひとつずつ片付けていこう。

1. インスタンスメソッドがまた多相になるケースの辞書渡し： いまは IO.>> がうまく動かない問題として露見
2. [Integer] や (Integer, Integer) に対する dictionary not found
3. Prim.(,) を Prelude.(,) に移し、かつ、lib/Prelude.hs 中で Show のインスタンス定義

(1) インスタンスメソッドがまた多相になるケースの辞書渡し

昨日までの改造による状況を整理：

- IO.>> の変換後の名前と型:`Prelude.IO%I.>> :: Forall [Kfun Star Star,Star,Star] ([Prelude.Monad a] :`
- TcState に Assump は追加済（↑この型が引ける）

いまの（動いていない）変換例は以下（sample100 の –ddump-core):

```
(Main.main :: (Prelude.IO ())) =
    ((((Prelude.>> :: ([Prelude.Monad TVar (Tyvar "t0" (Kfun Star Star))] :=> ((TVar (Tyvar "t0" (Kf
          ${Prelude.IO Prelude.Monad})
        ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ())))
          "Hello!"))
      ((Prim.putStrLn :: ([Prelude.Char] -> (Prelude.IO ())))
        "World!"))
```

改造しようとして、まずリファクタして、trace を仕込みつつみていて気付いたのだけど、辞書を適用した結果がまた辞書わたし形式になるような再帰は、静的に解決できるとは限らない。ランタイムにまた辞書探してくるなんてことはできないので、これは変だ。

この型がまちがっている ＞ `Prelude.IO%I.>> :: Forall [Kfun Star Star,Star,Star] ([Prelude.Monad a] :=> (`

そちらを直そう。そうすると、tcBind に [Assump] を渡すのは不要だったな、忘れないうちに戻そう。⇒ done.

trCDecl を変更して、DictDef に型シグネチャも格納するようにした。

今度は、renInstDecls でこの情報をもちいて、インスタンスメソッドの型シグネチャを出力するようにする。

DictDef に格納される型情報の例いくつか：

```
TypeSigDecl [Name {origName = ">>", namePos = (51,4), isConName = False}] (Just (AppTy (Tycon (Name
```

```
 TypeSigDecl [Name {origName = "show", namePos = (7,3), isConName = False}] (Just (AppTy (Tycon (Nam
```

各インスタンスメソッドに型宣言がつくようにすることで、IO.>> の型が多相になってしまっていた問題は解決。

以前、テストのうち 6 つ通らないが、それらは別の issue として、本件はクローズする。

sample149 を追加