# 041: tqd.hs で context reduction, Show [a] 問題

↑ up

- issued: 2020-04-18
- 分類： A サンプルコードが fail
- status: Closed (2020-05-03)

## 概要

Show クラスの扱いまわりをまともにしていく過程（参照：017）で、sample116, 136 が動かなくなってしまった。instance (Show a) => Show [a] にきちんと対応できていないため。

そこで、sample116, 136 はともに testcases に格下げ、tqd.hs, tqd2.hs とした。（両者の違いは、型シグネチャを明示しているか、していないか）

一方、tqd の show 以外の部分は動いているため、Show [a] に依存しないように書きかえたものを test/sample/sample150 として追加した。

## 調査ログ

### 2020-04-23 (Thu)

本件と 042 は同様の問題であるため、両方みながら対処することにしたい。

対処には、コンパイラの修正と lib/Prelude.hs への記述追加の両方が必要だが、lib/Prelude.hs をいきなり変えると多くの test が fail するので、そうせずに調査をすすめるためのサンプルケースを作成：

- testcases/myshowlist.hs
- testcases/myshowpair.hs
- testcases/myshowtriple.hs

Rename に trace を追加して確認したところ、instance メソッド定義に付加している型シグネチャが不適切であることが原因と思われる。

myshowlist の場合：

```
TypeSigDecl [Name {origName = "Main.[]%I.myshow", namePos = (2,3), isConName = False}] (Nothing,FunT
```

本来、`Main.[]%I.myshow :: [Show a] :=> [a] -> [Char]` であるべきところ、`[] :=> [a] -> [Char]` になっている。(Pred が空なのがおかしい)

myshowpair, myshowtriple は Rename で未実装のパターンにあたってしまうので、そこまでたどり着いていない。

インスタンスメソッドの型シグネチャを生成するときに、Pred を無条件で空にするのではなく、型に含まれる型変数が残った場合には、それに対応する Pred も残さなければならないようだ。

2020-04-24 (Fri)

型シグネチャの生成。`renInstDecl (A.InstDecl ctx t ds)` の ctx をそのまま sigvar につけるように変更。

この部分は、ひとまずこれでいいはずなのだが、myshowlist.hs はエラー。これは、Int, Char の instance 宣言において、Int, Char が Main.Int, Main.Char と解釈されてしまったせい。myshowlist2.hs のように修飾すると、別のエラーになった。

なので、instance (Show a) => Show [a] where ... の定義は直接 lib/Prelude.hs い追加して調査続行。こうなった：

```
$ ./tcheck testcases/showils.hs
# 1. test-compile
source file: testcases/showils.hs
dst dir: /showils
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: Error: dictionary not found: Prelude.print, (Tyvar "t0" Star,"Prelude.Show",TA
CallStack (from HasCallStack):
  error, called at src/DictPass.hs:232:43 in main:DictPass
```

```
$ ./tcheck testcases/printstring.hs
# 1. test-compile
source file: testcases/printstring.hs
dst dir: /printstring
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: Error: dictionary not found: Prelude.print, (Tyvar "t0" Star,"Prelude.Show",TA
CallStack (from HasCallStack):
  error, called at src/DictPass.hs:232:43 in main:DictPass
```

ここでは、[$Prelude.[] Prelude.Show, $Prelude.Int Prelude.Show] のように、辞書のリストを渡すのが正しいように思われる。

これを、((show $Prelude.[] Prelude.Show) $Prelude.Int Prelude.Show) のように処理すれば、期待されたような動作になるのではないか。

- Core の辞書型に、辞書のリストに対応するものを追加する
- DictPass (findApplyDict) を、上述のような辞書のリストを渡すように改造する
- Runtime で、辞書のリストは、複数回の辞書適用を実装するように改造する

これでうまくいくかどうかは、リストだけでなく、tuple, triple のケースも考慮したほうがよさそう。辞書のリストのリストの方がいいのかもしれない。

たとえば、Main.(,)%I.show は辞書を 2 つ引数にとるので、show に渡すのは、[[$Prelude.(,) Prelude.Show], [$Prelude.Int Prelude.Show, $Prelude.Int Prelude.Show]] のようにすべきなのではないか。

2020-04-26 (Sun)

複合辞書は、[[Dictionary]] ではなく、第一要素は常に 1 つ、第二要素がリスト（第三以降はないはず）でいいと思われるため、CompositDict Expr [Expr] のような形にした。

必要と思われる修正は以下：

1. Core に CompositDict 定義を追加
2. PPCore を CompositDict に対応させる
3. DictPass で、[a] に出会ったときに CompositDict を生成するようにする
4. STG にも CompositDict を追加、trSTG をこれに対応させる
5. CodeGen の対応

4 項まで実施したところで、CodeGen 未対応による以下のエラーが出た：

```
bunnyc: Non-exhaustive pattern in genAtomExpr: AtomExpr (VarAtom (CompositDict (AtomExpr (VarAtom (D
```

CodeGen にて、genAtomExpr は対処、最後は、OLshow にて CompositDict をハンドリングできるようにする。

現状の OLshow:

```
public static class OLshow implements LambdaForm {
 public int arity(){ return 1; }
 public Expr call(AtomExpr[] args){
  Dict_36_Prelude_46_Show d = (Dict_36_Prelude_46_Show) RTLib.extrDict(args[0]);
  Expr t0 = d.mkshow();
  return t0;
 }
}
```

CodeGen を改造し、次のような OLshow を吐くようにした：

```
public static class OLshow implements LambdaForm {
 public int arity(){ return 1; }
 public Expr call(AtomExpr[] args){
```

```
      Expr t0;
      if (args[0].a instanceof CompositDict){
        Dict_36_Prelude_46_Show d = (Dict_36_Prelude_46_Show) RTLib.extrDict(((CompositDict) args[0]
        AtomExpr[] ds = ((CompositDict) args[0].a).ds;
        Expr t1 = d.mkshow();
        t0 = RTLib.mkApp(t1, ds);
      } else {
        Dict_36_Prelude_46_Show d = (Dict_36_Prelude_46_Show) RTLib.extrDict(args[0]);
        t0 = d.mkshow();
      }
      return t0;
     }
    }

    public static Expr mkshow(){
      Expr t0 = RTLib.mkFun(new OLshow());
      return t0;
    }
```

これで（いまのところ）期待どおりの動作をするようになった。
ただ、tqd.hs や printstring.hs では次のようなエラーがでる。こころあたりはあり。

```
$ ./tcheck testcases/printstring.hs
# 1. test-compile
source file: testcases/printstring.hs
dst dir: /printstring
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: src/DictPass.hs:(220,18)-(227,65): Non-exhaustive patterns in case
```

⇒ かなりその場限りの書き方だが対処。printstring.hs、tqd.hs, tqd2.hs ともにエラーはしなくなった。
つぎに、lib/Prelude.hs における Show [a] の定義をまともにしよう。
ひとまず、[Char] 以外に対応、showils.hs は通した ⇒ sample163.hs
shoswPrec や showList をもちいたトリックは、まだ理解できていないのと、そのままやってみたらエラーしたので要継続。

2020-05-03 (Sun)

045 の 2020-05-01 の対処、および、052 により、Show [a] 定義を Haskell 2010 report 方式に書き直すことができるようになった。

まだ、いくらか問題を回避しつつだが、printstring.hs、tqd.hs, tqd2.hs が通るようになったので、本件はクローズとする。

保留、または、回避中の問題：

- Show Char において、showLitChar などエスケープ未対応 ⇒ 053
- showl "" など、String Literal パターンがエラーする ⇒ 054
- Int, Integer ともに showList のデフォルト実装ではエラーするため、各々記述 ⇒ 055