073: ptype[1—2] で dictionary not found

↑ up

- issued: 2020-05-14
- 分類： A サンプルコードが fail
- status: Closed (2020-05-23)

概要

068 の調査から派生。As-pattern `x@(p, q) = (1, 2)` を、

```
x = (1, 2)
p = case x of
  (a, b) -> a
```

のように変換すると型エラーする。(`p = case (1, 2) of ...` なら OK)

これが、Rename された結果に近い、以下のような書き換えでもだめ

```
x = (1, 2)
p = snd x
```

p の型が `Num t => t` のようになるべきところが、–ddump-assump の時点ですでにそうなっていないので、型推論以前の段階に問題があるように見えるが…。

Thih のコードに問題があるわけではないだろうし、まだ見当がついていない。

調査ログ

2020-05-14 (Thu)

ひとまず 068 の仮対処で、つい、as-pattern でないものを、as-pattern に変換するときの一時変数を "_" にしてさぼってしまったが、本件対処するためには、ここに fresh な変数を用意しないといけない。

2020-05-22 (Fri)

この件はちょっと長引きそうな気がしている。ひとつひとつ、確認していこう。

まず、本件に関する testcases をひとつディレクトリを掘ってまとめておく。(testcases/073)

ptype1 と ptype2 に本質的な違いはない。ptype2 をつかったほうが、生成される core がシンプルになって調査が楽かもしれないので、以降こちらを用いる。

ptype2.hs:

```
main = do print p
           where
             x = (1, 2)
             p = fst x
```

これをコンパイルすると、`bunnyc: Error: dictionary not found: ("Prelude.Show",Tyvar "v2321" Star,[])`
となる。

以下が、その core0 だが、Main.l1.l0.p の型が Num t => t ではなく t になっているのが、辞書渡しが解決できない理由にも見える。(一方で、p のなかで Defaulting してしまって、v2321 = Integer にしてしまうという手もありそうだが、それでは、別件（gcd など）に対応できないかも）

```
(Main.main :: (Prelude.IO ())) =
    let
        (Main.l1.l0.p :: v2321) =
            ((Prelude.fst :: (((Prelude.(,) t2) t3) -> t2))
              (Main.l1.l0.x :: ([Prelude.Num v2314,Prelude.Num v2315] :=> ((Prelude.(,) v2314) v2315

        (Main.l1.l0.x :: ([Prelude.Num v2314,Prelude.Num v2315] :=> ((Prelude.(,) v2314) v2315))) =
            (((Prelude.(,) :: (t9 -> (t10 -> ((Prelude.(,) t9) t10))))
                ((Prelude.fromInteger :: ([Prelude.Num t11] :=> (Prelude.Integer -> t11)))
                  (1 :: Prelude.Integer)))
              ((Prelude.fromInteger :: ([Prelude.Num t12] :=> (Prelude.Integer -> t12)))
                (2 :: Prelude.Integer)))


    in
      ((Prelude.print :: ([Prelude.Show t15] :=> (t15 -> (Prelude.IO ()))))
        (Main.l1.l0.p :: v2321))
```

Typing モジュールによる型推論がまちがっているとも考えづらいが（Thih そのままなので)、自前でつくっている toBg に問題がある可能性は考えられる（sa. 088)

ptype2.hs を少し変形して、次のようにすると通る (ptype2m.hs)。

```
main = do print p
           where
```

```
            p = fst x
              where x = (1, 2)
```

この場合には、p の型は Num t => t と推論されており、それに基づいて適切に辞書渡しが行われてる：

```
(Main.main :: (Prelude.IO ())) =
    let
        (Main.l1.l0.p :: ([Prelude.Num v2321] :=> v2321)) =
            \(Main.l1.l0.p.DARG0 :: ^^c3^^84) ->
                let
                    (Main.l1.l0.l0.l0.x :: ([Prelude.Num v2315,Prelude.Num v2316] :=> ((Prelude.(,)
                        \(Main.l1.l0.l0.l0.x.DARG0 :: ^^c3^^84) (Main.l1.l0.l0.l0.x.DARG1 :: ^^c3^^8
                            (((Prelude.(,) :: (t6 -> (t7 -> ((Prelude.(,) t6) t7))))
                                (((Prelude.fromInteger :: ([Prelude.Num t8] :=> (Prelude.Integer ->
                                    (Main.l1.l0.l0.l0.x.DARG0 :: ^^c3^^85))
                                  (1 :: Prelude.Integer)))
                              (((Prelude.fromInteger :: ([Prelude.Num t9] :=> (Prelude.Integer -> t9
                                  (Main.l1.l0.l0.l0.x.DARG1 :: ^^c3^^85))
                                (2 :: Prelude.Integer)))


                in
                  ((Prelude.fst :: (((Prelude.(,) t12) t13) -> t12))
                    (((Main.l1.l0.l0.l0.x :: ([Prelude.Num v2315,Prelude.Num v2316] :=> ((Prelude.(,
                        (Main.l1.l0.p.DARG0 :: ^^c3^^85))
                      ${Prelude.Integer Prelude.Num}))


    in
      (((Prelude.print :: ([Prelude.Show t17] :=> (t17 -> (Prelude.IO ()))))
          ${Prelude.Integer Prelude.Show})
        ((Main.l1.l0.p :: ([Prelude.Num v2321] :=> v2321))
          ${Prelude.Integer Prelude.Num}))
```

なので、ひとつは、型変換まえの Rename でこのように式変形してしまうかとも考えたのだが、p の binding が複雑だった場合（pattern binding など）にこのように変形するのは複雑になってしまうのと、以下のケースにやはりエラーするので、苦労して変形したところで、解決になっていない。

ptype2n.hs:

```
main =
  let
    x = (1, 2)
  in
    let
      p = fst x
    in
      print p
```

なので、型推論以前（toBg 含め）の問題かとは思われるが、仮対処として type-checking (DictPass) にて p の型に述語 (Num t) を補うことを考えてみたい。

2020-05-23 (Sat)

ptype[1—2].hs は解決

DictPass にて Pred を補うようにしたことで、ptype1, 2 ともに期待通り変換されるようになった。以下は、ptype2.hs のコア。Main.l1.l0.p の型に Num b9 という Pred が補われ、辞書渡しが行われている：

```
(Main.main :: (Prelude.IO ())) =
    let
        (Main.l1.l0.x :: ([Prelude.Num v2240,Prelude.Num v2241] :=> ((Prelude.(,) v2240) v2241))) =
            \(Main.l1.l0.x.DARG0 :: ^^c3^^84) (Main.l1.l0.x.DARG1 :: ^^c3^^84) ->
                (((Prelude.(,) :: (t4 -> (t5 -> ((Prelude.(,) t4) t5))))
                    (((Prelude.fromInteger :: ([Prelude.Num t6] :=> (Prelude.Integer -> t6)))
                        (Main.l1.l0.x.DARG0 :: ^^c3^^85))
                      (1 :: Prelude.Integer)))
                  (((Prelude.fromInteger :: ([Prelude.Num t7] :=> (Prelude.Integer -> t7)))
                      (Main.l1.l0.x.DARG1 :: ^^c3^^85))
                    (2 :: Prelude.Integer)))


    in
      let
          (Main.l1.l0.p :: ([Prelude.Num b9] :=> b9)) =
              \(Main.l1.l0.p.DARG0 :: ^^c3^^84) ->
                  ((Prelude.fst :: (((Prelude.(,) t11) t12) -> t11))
                    (((Main.l1.l0.x :: ([Prelude.Num v2240,Prelude.Num v2241] :=> ((Prelude.(,) v224
```

```
                    (Main.l1.l0.p.DARG0 :: ^^c3^^85))
                  ${Prelude.Integer Prelude.Num}))


in
  (((Prelude.print :: ([Prelude.Show t15] :=> (t15 -> (Prelude.IO ()))))
      ${Prelude.Integer Prelude.Show})
    ((Main.l1.l0.p :: ([Prelude.Num b9] :=> b9))
      ${Prelude.Integer Prelude.Num}))
```

今回の修正により、エラーするようになってしまった sample プログラムもある。testcases に降格する。これらは、それぞれ新規 issue をたてる。

- sample170 -> strlenx.hs 090
- sample193 -> divquot2.hs 091
- sample219 -> gcdlcm.hs 092

また、本件対処によって test/samples に加えられたもの：

- ptype2b.hs：sample222.hs
- ptype2mb.hs：sample223.hs
- gcd2.hs：sample224.hs
- gcd.hs：sample225.hs
- gcd2.hs：sample226.hs