

## 119: List comprehension における Fail がランタイムエラーになる

↑ up

- issued: 2020-12-13
- 分類: A サンプルコードが fail
- status: Closed (2020-12-13)

### 概要

Prelude.hs を書いているなかで発見した件。List comprehension においてパターンマッチが fail したときに、ランタイムエラーとなってしまう。

例えば、次のようなプログラム:

```
xs = [Just 3, Just 2, Nothing, Just 9, Nothing, Nothing, Just 8]
main = print $ sum $ [y | Just x <- xs, let y = x * 2]
```

これをコンパイル、実行しようとするときランタイムエラーとなる。

```
$ bunny testrun listcomprehension.hs
/home/unno/bunny/0.9.0/bin/bunnyc -d ./jout/listcomprehension --xno-implicit-prelude /home/unno/bunny/0.9.0/bin/bunnyc -d ./jout/listcomprehension --xlibrary-path /home/unno/bunny/0.9.0/bin/bunnyc
Error: Non-exhaustive patterns.
```

### 調査ログ

2020-12-13 (Sun)

そういえば、ころあたりがあった。The Haskell 2010 language report の 3.11 List Comprehensions<sup>\*1</sup> では、 $[e \mid p \leftarrow l, Q]$  の翻訳は次のように書かれている:

```
[e | p <- l, Q] = let ok p = [e | Q]
                  ok _ = []
                  in concatMap ok l
```

---

\*1 <https://www.haskell.org/onlinereport/haskell2010/haskellch3.html#x8-420003.11>

だが、私は当時、`ok p` が常にマッチするようにみえてしまい (`p` に任意のパターンが来うるので `fail` する可能性があるのだが、`p` が普通の引数に見えて勘違いした)、以下のように実装してしまっていたのだった。

```
[e | p <- l, Q] = let ok p = [e | Q]
                  in concatMap ok l
```

というわけで、理由は簡単だった。仕様通りに直して解決。

概要のところ挙げたプログラムを `test/sample323.hs` とする。