あなたの知らない Monoid の世界

2012.11.18

山本和彦

数学の話

- 半群
  - 結合則　(a・b)・ c = a・(b・c)
- Monoid
  - 結合則　(a・b)・c = a・(b・c)
  - 単位元　e・a = a・e = a
- 群
  - 結合則　(a・b)・c = a・(b・c)
  - 単位元　e・a = a・e = a
  - 逆元　　$a \cdot a^{-1} = e$
- 半環、環、体
  - 半環は計算量を低減するために大活躍する
  - Semi ring fusion と呼ばれる

Q) いつ Monoid を使うのか？

A) あるデータ型に演算子が1個あれば
それは多分 Monoid

Monoid の例

- doctest のエラー集計

```
data Summary = Summary {
  sExamples :: Int
, sTried    :: Int
, sErrors   :: Int
, sFailures :: Int
} deriving Eq

instance Monoid Summary where
  mempty = Summary 0 0 0 0
  (Summary x1 x2 x3 x4) `mappend`
     (Summary y1 y2 y3 y4) = Summary (x1 + y1)
                                     (x2 + y2)
                                     (x3 + y3)
                                     (x4 + y4)
```

Q) Num だとダメなの？

A) だって (*) も (-) も不要だもん！

突然ですが Fizzbuzz です

```
data FizzBuzz = N | I Int | S String
(<+>) :: FizzBuzz -> FizzBuzz -> FizzBuzz
N    <+> N    = N
N    <+> I i  = S $ show i
N    <+> S x  = S x
I i  <+> N    = S $ show i
I _  <+> S x  = S x
S x  <+> I _  = S x
S x  <+> S y  = S $ x <> y
S x  <+> N    = S x
I _  <+> I _  = error "FizzBuzz"

fizzbuzz :: [String]
fizzbuzz = map (\(S x) -> x) fbs
  where
    fizz = cycle [N, N, S "Fizz"]
    buzz = cycle [N, N, N, N, S "Buzz"]
    is = map I [1..]
    fbs = zipWith (<+>) is $ zipWith (<+>) fizz buzz
```

つまりこういう計算

```
I  1 <+> N         <+> N          → S "1"
I  3 <+> S "Fizz" <+>             → S "Fizz"
I  5 <+> N         <+> S "Buzz"   → S "Buzz"
I 15 <+> S "Fizz" <+> S "Buzz"   → S "FizzBuzz"
```

Monoid な Fizzbuzz

```
data FizzBuzz = N | I Int | S String
instance Monoid FizzBuzz where
    mempty                = N
    N    `mappend` N      = N
    N    `mappend` I i    = S $ show i
    N    `mappend` S x    = S x
    I i  `mappend` N      = S $ show i
    I _  `mappend` S x    = S x
    S x  `mappend` I _    = S x
    S x  `mappend` S y    = S $ x <> y
    S x  `mappend` N      = S x
    I _  `mappend` I _    = error "FizzBuzz"
fizzbuzz :: [String]
fizzbuzz = map (\(S x) -> x) fbs
  where
    fizz = cycle [N, N, S "Fizz"]
    buzz = cycle [N, N, N, N, S "Buzz"]
    is = map I [1..]
    fbs = zipWith (<>) is $ zipWith (<>) fizz buzz
```

プログラムだって Monoid

- 単位元

  {}

- 結合則

```
{                   foo;
    foo;            {
    bar;                bar;
}                       baz;
baz;                }
```

連結か選択か

- 連結している感じ

```
[1,2] <> [3,4]
→ [1,2,3,4]

Sum 1 <> Sum 2
→ Sum 3
```

- 選択している感じ

```
Any False <> Any True <> Any False
→ Any True

EQ <> GT <> LT
→ GT
```

コンテナと Monoid

- Monoid

  (<>) :: m -> m -> m

- Alternative

  (<|>) :: f a -> f a -> f a

  - Applicative レベルの Monoid
  - 名前から選択している感じがしてしまう

- MonadPlus

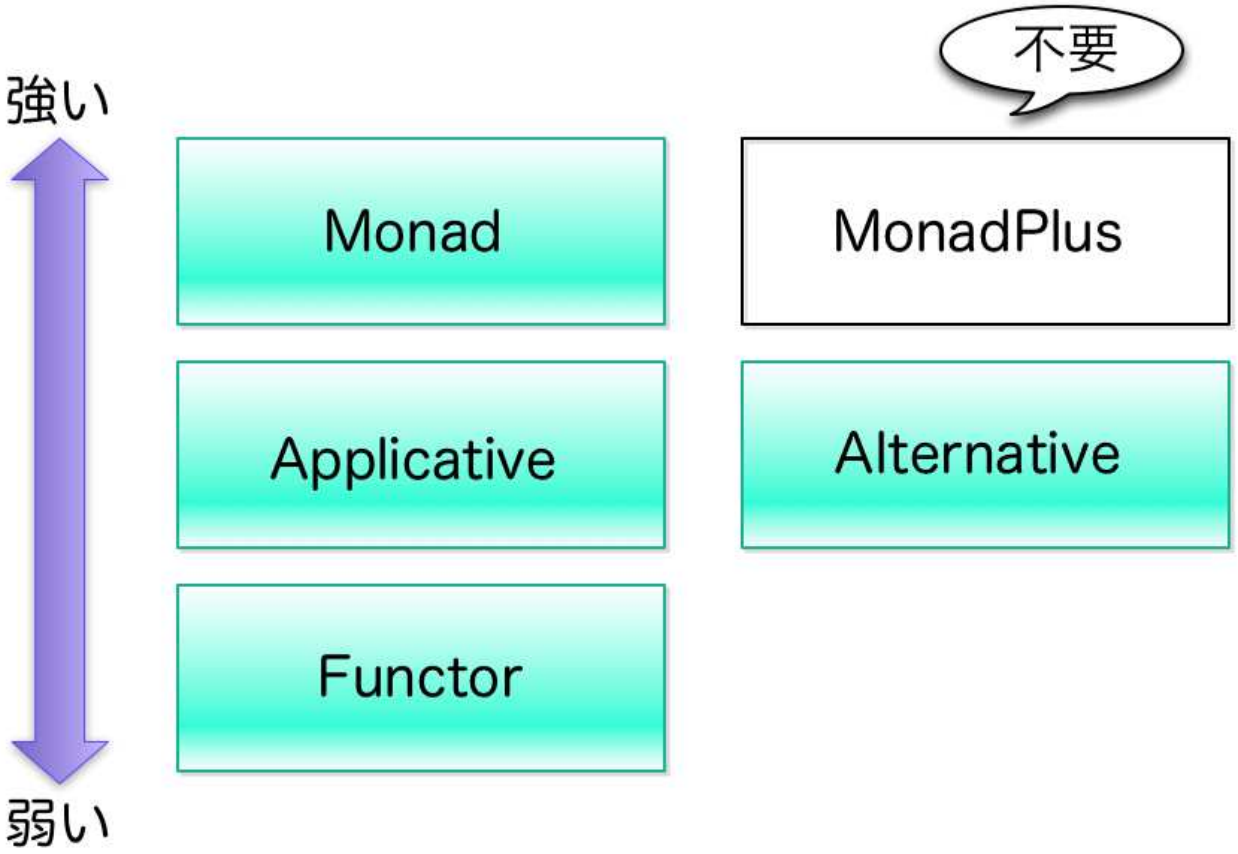
  mplus :: m a -> m a -> m a

  - Monad レベルの Monoid
  - 名前から連結している感じがしてしまう

- Alternative と MonadPlus は同じもの

MonadPlus は忘れよう

- 「力の弱いものを使え」の法則より
  MonadPlus は不要



お父さんかお母さんを探せ

- 古き良き MonadPlus

```
lookup myid fdb ‘mplus‘ lookup myid mdb
```

- Alternative

```
lookup myid fdb <|> lookup myid mdb
```

- クイズ
  - 父方のおじいちゃんか、父方のおばあちゃんを探してみよう

連結か選択か (再び)

- 連結している感じ
  - リスト

```
[1,2] <|> [3,4]
→ [1,2,3,4]
```

  - 連結する感じなのはリストのみ
  - リストを Monoid する定義と MonadPlus にする定義は一緒
  - ますます MonadPlus は不要な感じ

- 選択している感じ
  - Maybe

```
Nothing <|> Just "Alice"
→ Just "Alice"
```

  - Parsec

```
char '+' <|> char '-'
```