Emacs Lisp 脳と Haskell 脳

～関数型言語ってなぁに？～

IIJ-II
山本和彦

山本和彦のジレンマ

山本和彦が Emacs Lisp は知っていたが、
Haskell は知らなかったときの話：

Lisp は関数型言語だと言われている

Emacs Lisp も Lisp である

山本和彦は関数型プログラマーだと
思われていた

しかし、本人は何が関数プログラミング
なのか分かっていなかった

関数プログラミングを
議論する前に

オブジェクト指向の例で考えてみる

共通定義のないオブジェクト指向

- それぞれの人が適当なサブセットを選んで、オブジェクト指向と呼ぶ
  http://practical-scheme.net/trans/reesoo-j.html

Lisp と Java

- Java だってオブジェクト指向言語
- Lisp だってオブジェクト指向言語

共通定義のない関数型言語

- それぞれの人が適当なサブセットを選んで、関数型と呼ぶ

関数が第一級の値	無名関数	クロージャ
カリー化	関数の再帰的定義	末尾再帰の最適化
破壊的代入がない	参照透明性	遅延評価

Emacs Lisp と Haskell

- Emacs Lisp は命令型言語
- Haskell は関数型言語

パラダイムの違い

命令プログラミング	関数プログラミング

命令を列挙する
`A; B; C;`
`(progn ...)`
	関数を引数に適用する

状態がある	状態はない
破壊的代入を使う	(値を破壊したくなったら) 新たな値を作る

例題

- 入力として整数のリスト 10, 20, 30, 40, 50 がある
- 0 から数えて n 番目の要素には n を掛ける
- それらをすべて足し合わせる

- つまり、以下のような計算をする

10 * 0 + 20 * 1 + 30 * 2 + 40 * 3 + 50 *4 = 400

10 * 0 + 20 * 1 + 30 * 2 + 40 * 3 + 50 *4 = 400

命令型プログラマーなら
for 文か類似のループで
問題を解く

不格好な for 文

- Douglas Crockford のエッセイ
  「JavaScript: 世界で最も誤解されているプログラミング言語」
  http://www.crockford.com/javascript/javascript.html

波括弧や不格好な for 文がある
JavaScript の C ライクな文法を見ると、
通常の命令型言語のように思える。

これは誤解を与えやすい。
なぜなら、JavaScript は、
C や Java とよりも
関数型言語 Lisp や Scheme との方が
共通点が多いからだ。

for 文って不格好なの？

for の秘密

- 山本和彦は、for について授業で熱く語っていた
- for の意味
  - for は期間の for だ！
  - 例) for two days
- 非対称範囲を使え！
  - `for (i = 0; i < N; i++) { }`
  - 左の境界は入るが、右の境界は入らない
  - 0 から始めると配列と相性がよい
  - N をそのまま使える
  - 個数 = 最後 - 最初 + 1
    - (N - 1) - 0 + 1 = N
  - 不等号を使うと安全性
    - コンピュータでは、1/3 + 1/3 + 1/3 ≠ 1

Emacs Lisp で不格好な for

```
(let ((lst '(10 20 30 40 50))
      (i 0)
      (ret 0))
  (while lst
    (setq ret (+ ret (* (car lst) i)))
    (setq lst (cdr lst))
    (setq i (1+ i)))
  ret)
```

→ 400

- これが不格好と思わない人は、
  かっこいい解決方法を知らないのだろう。。。

Emacs Lisp でもう少し抽象化

```
(let ((lst '(10 20 30 40 50))
      (i 0)
      (ret 0))
  (dolist (x lst ret)
    (setq ret (+ ret (* x i)))
    (setq i (1+ i))))
```

→ 400

- 命令的な臭い
  - 一度にたくさんの仕事をする
  - 破壊的代入を使っている
- for の内側で考えている限り、
  for の呪縛からは逃れられない

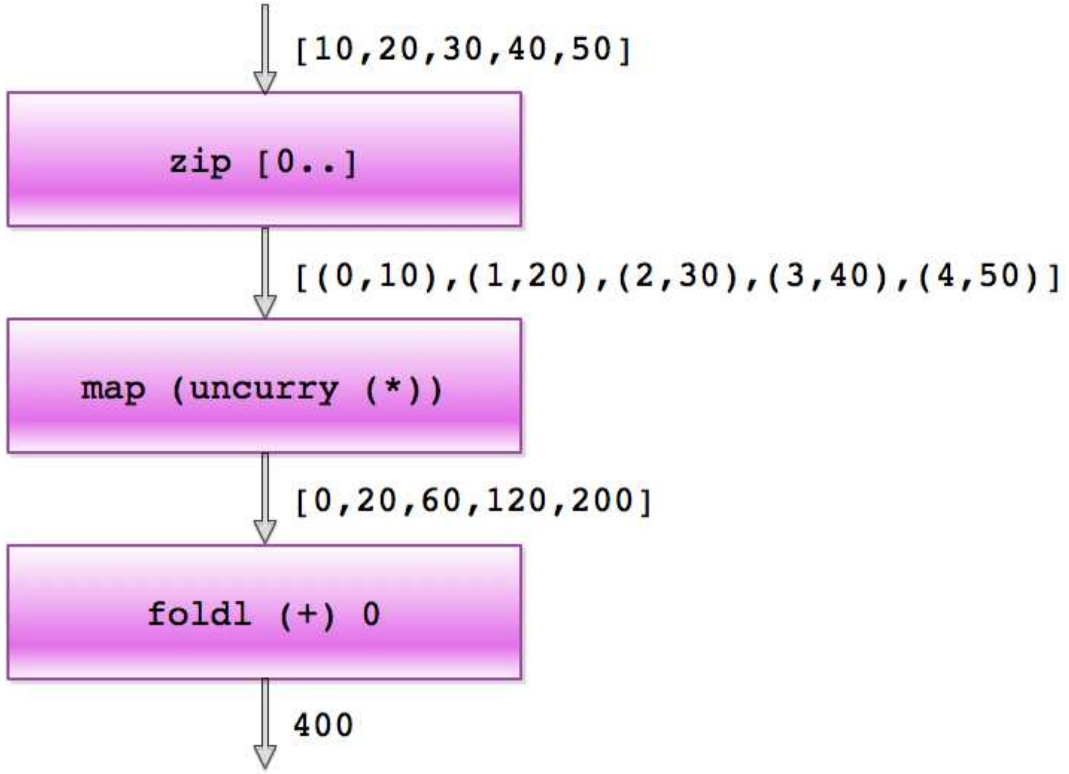
Haskell で map & fold

```
  zip [0..] [10,20,30,40,50]
→ [(0,10),(1,20),(2,30),(3,40),(4,50)]

  map (\(i,x) -> x*i) (上記の式)
→ [0,20,60,120,200]

  foldl (+) 0 (上記の式)
→ ((((0 + 0) + 20) + 60) + 120) + 200
→ 400
```

- 関数を合成する

```
  func = foldl (+) 0
       . map (\(i,x) -> x*i)
       . zip [0..]

  func [10,20,30,40,50]
→ 400
```

関数プログラミングと信号回路



関数プログラミングとは
「関数を引数に適用すること」だと言う
プログラミング手法だとみなせる。

そして、関数型言語とは、
関数型の手法を提供し奨励している
プログラミング言語である。

まとめ

- Emacs Lisp は命令型言語
  - 関数型言語の要素は持っているが、
    プログラマーは命令プログラミングしかしていない
- 他のパラダイムを知ろう
  - 問題の見方が変わる
- Haskell に興味がある Lisper の方へ
  - 「Hello Haskell, Goodbye Lisp」
    - http://www.newartisans.com/2009/03/hello-haskell-goodbye-lisp.html