関数型言語Ruby(2)

2007-07-14

Ruby勉強会@関西 17

氏久 達博

自己紹介

- 氏久達博
- 大阪大学大学院基礎工学研究科
- ラフ集合理論を研究
- RubyとHaskellを使っている

関数型言語
Ruby

- Rubyは純粋な関数型言語
- オブジェクト指向スクリプト言語Rubyと偶然名前が同じ

- 関数型言語Rubyの
  ふつうの使い方を学ぶ
- ライブラリcohiの
  紹介
- 発展的話題と演習

Hello, World

```
main = lambda { puts "Hello, World!" }
main[]
```

- プログラムはmain関数から実行される
- 関数定義の方法は、
  関数名 = lambda { 処理 }
- 末尾のmain[]はおまじない

```
main = lambda { puts "Hello, World!" }
main[]
```

- ここに出てくるputsはIOモナド
- 他にp, printなどがある
- 引数指定の[]を省略できる
  Syntax Sugar
- mainの戻り値などで使える

```
aaa = lambda { puts "buggy" }

main = lambda { aaa }
main[]
```

- こんな風に、IOモナドを返す関数も作れる
- 関数aaaの型は
  Rubyが自動で型推論してくれる
- (実際の型を確認することはできない)

足し算

```
add = lambda {|n| lambda {|m|
  lambda { n + m }
}}
main = lambda { puts add[3][5][] }
main[]
```

- 引数のある関数の定義は、
  関数名 = lambda {|第一引数|
    lambda {|第二引数| ...
      lambda { 処理 }
  }} ←引数の数だけ}を書く

関数呼び出しの
最後の[]が気に食わない

このライブラリを使えばOK

```
# f.rb
class Proc
  def to_ary
    [call(nil)]
  end

  def inspect
    call nil
  end
end
```

```
require 'f'
require 'add'
main = lambda { puts add[3][8] }
main[]
```

パターンマッチ

```
require 'f'
sum = lambda {|list|
  if list == [] then 0
  else i, is = list[0], list[1..-1]; i + sum[is]
  end
}

main = lambda { print sum[ [1,2,3,4,5] ] }
main[]
```

- あまりうれしくない…
- 条件分の中で代入ができない

  (multiple assignment in conditionalでSyntaxErrorになる)
- そろそろ限界か

cohi

- パターンマッチ、カリー化、関数合成などをサポート
- 限界突破
  - HaskellのPreludeモジュールのほとんどが既に移植されている

- 2007-07-01にcohi-0.0.1リリース
- 作者はha-tan (ここに居るかも?)
- http://rubyforge.org/projects/cohi/
- インストール↓

```
gem install cohi
```

Hello, World

```
require 'rubygems'
require 'cohi'
require 'cohi/prelude'
include Cohi

define(:main) {
  put_str_ln["Hello, World!"]
}
main[]
```

- 関数定義のための~~メソッド~~関数がある

sum
パターンマッチ
と再帰

```
require 'rubygems'
require 'cohi'
require 'cohi/prelude'
include Cohi

define(:mysum, [[]  ]) {         0 }
define(:mysum, [X_XS]) {|x, xs| x + mysum[xs] }
define(:main) {
  list = [1,2,3,4,5]
  put_str_ln[mysum[list]]
}
main[]
```

- [], X_XSなどがキモ。Haskell流。

- foldlを使えばもっとラクに書ける
  (Rubyでいうところのinject)
- 実際cohiのpreludeのsumは
  foldlを使って定義している
- cohiのpreludeのソースを読んでみよう!
  cohiでcohiが構成されている美しさ

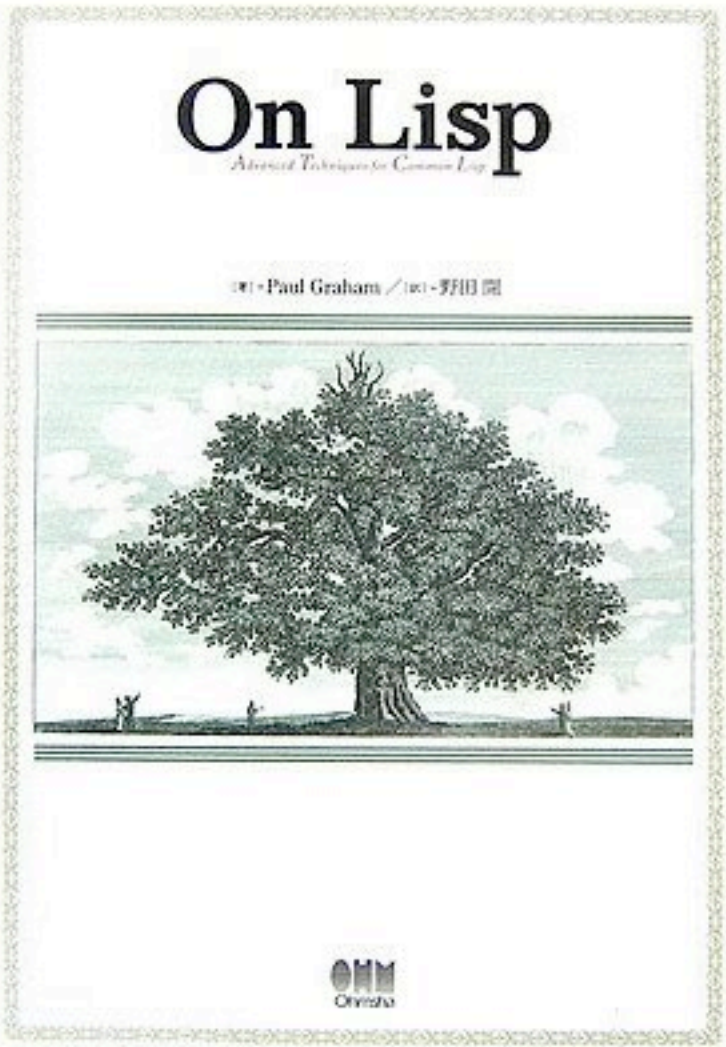
まとめ

- 関数型言語Rubyの
  ふつうの使い方を学んだ
- cohiを使うと、
  よりラクになる

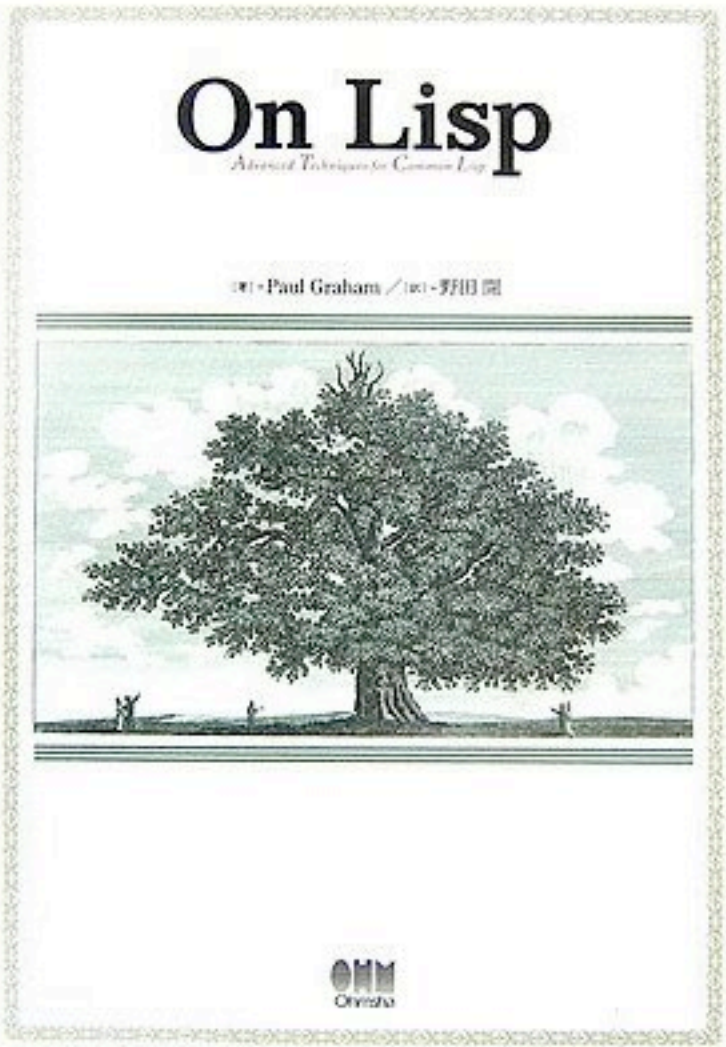
次回予告

- リスト内包表記など
- let, whereなど
- いつどこで話するか不明

ちなみに…



Lispにも手を出しはじめました



On Lisp読書会開催予定

話はここでおしまい

演習

1. cohiを用いずにmapを実装
2. cohiを用いてmapを実装
3. cohiを用いてprodを実装