

算法言語 Functional Programming

胡 振江
平成14年冬学期

目的

- プログラミングを数学的(代数的)な活動としての考え方を伝授すること。
- 代数(Algebra)とは
(The Shorter Oxford English Dictionary):
 - the reunion of broken parts
 - a calculus of symbols combined according to defined laws

内容

- 関数プログラミング言語Haskell
 - プログラム:関数の定義
 - プログラムの実行:式の簡約
 - 関数プログラミングの特徴
 - 抽象的
 - 構成的
 - 操作しやすい
 - 推論

教科書

- 武市正人訳、「関数プログラミング」, 近代科学社, 1994年. ISBN4-7649-0181-1
(R. Bird and P. Wadler, Introduction to Functional Programming, Prentice Hall, 1988)

各自購入すること

参考書など

- Richard Bird. Introduction to Functional Programming in Haskell, Prentice Hall, 1998.
- 講義ページ:
<http://www.ipl.t.u-tokyo.ac.jp/~hu/fp01>

講義形式・成績分配

- 講義 20%
 - 場所:
 - 6号館 または
 - 情報基盤センター5階:11月11日
 - レポート:2回
- 中間テスト 20%
 - 場所:情報基盤センター5階
 - 日時:11月25日
- 試験 60%
 - 期末試験

関数プログラミングの基本概念

- セッション(session): 利用者と計算機の間の一連のやりとり

```
Prelude> sin 1
0.841471
(21 reductions, 101 cells)
```

```
Prelude> 2+3*4
14
(22 reductions, 55 cells)
```

基本概念

- スクリプト(Script): 関数定義の並び

Test.hs

```
square x = x * x
side = 12
area = square side

min' x y | x <= y    = x
        | otherwise = y
```

式と値

- 式: 値を表す
- 式の評価: 計算機は式を「最も単純な等価な形」に簡約し、結果を表示すること。
 - 標準形(正規形)
 - 底要素_⊥: 定義されない値(1/0)
- 簡約の系列

```
square (3+4) => square 7
              => 7*7
              => 49
```

型

- 型: 値全体の集まり
 - 基本型
 - 整数型 Int
 - 論理型 Bool
 - 文字型 Char
 - 派生型
 - 組型 (T1,T2)
 - リスト型 [T]
 - 関数型 T1→T2
 - 強い型決め(Strong Typing)
 - 式の型が構成要素である式の型によって決まる原理

関数

$f :: A \rightarrow B$

$\text{square} :: \text{Int} \rightarrow \text{Int}$
 $\text{square } x = x * x$

$\text{ay} :: a \rightarrow \text{Char}$
 $\text{ay } x = 'A'$

$\text{id} :: a \rightarrow a$
 $\text{id } x = x$

関数の定義

$f \ x \ y \mid x > 10 \quad = \ x + a$
 $\mid \text{otherwise} = x - a$
where $a = \text{square } (y+1)$

$f :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

カリー化

構造をもつ引数を単純な引数の列に置き換える方法

<code>min x y x <= y</code>	<code>= x</code>	<code>min' (x,y) x <= y</code>	<code>= x</code>
<code> otherwise = y</code>		<code> otherwise = y</code>	

`min :: Int → (Int → Int)`

`min' :: (Int,Int) → Int`

`min 5 :: Int → Int`

`curry min' 5 :: Int → Int`

仕様と実現

- 仕様: 仕事の数学的な記述
 - プログラマーの意思の表現
 - 簡潔で分かりやすいことが望ましい
- 実現: 仕様を満たすプログラム
 - 計算機によって実行される表現
 - 効率よく実行すること

例: increase

- 仕様
すべての $x \geq 0$ に対して、
`increase x > square x`
- 実現1
`increase x = square (x+1)`
- 実現2
`increase x = square x + 1`

合成
↓

宿題

- Hugs をインストールする。
- Hugs を使ってみる。

```
:load <filenames>  load modules from specified files
:load              clear all files except prelude
:also <filenames>  read additional modules
:reload            repeat last load command
:project <filename> use project file
:edit <filename>    edit file
:edit              edit last module
:module <module>    set module for evaluating expressions
:<expr>             evaluate expression
:type <expr>        print type of expression
:?                 display this list of commands
:set <options>      set command line options
:set               help on command line options
:names [pat]       list names currently in scope
:info <names>       describe named objects
:browse <modules>  browse names defined in <modules>
:find <names>       edit module containing definition of name
:command           shell escape
:cd dir             change directory
:gc                 force garbage collection
:version            print Hugs version
:quit              exit Hugs interpreter
```