プログラムの数理
Mathematical Structures in Programs

胡　振江
平成17年度冬学期

講義の概要

目的

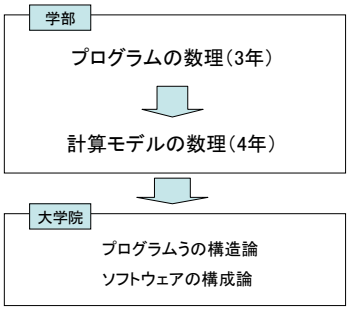
本講義では算法**言語の基本概念**を関数プログラミングを通して修得する.

関数プログラミングは算法設計・プログラミングを**数学的な活動**としてとらえる考え方であり，本講義ではそれをプログラミング言語Haskellを用いて具体的に示すとともに、厳密な**科学・工学としてのプログラミングのあり方**を学ぶ.

内容

- 関数プログラミング言語Haskellの学習
  - 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の理解
  - 問題の記述：抽象的
  - プログラム構造：構成的，再帰的
  - プログラム間の関係：推論，操作しやすい
  - プログラム性質：証明しやすい

他講義との関係



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)

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/pub/teach/pm05/

日程

- 10月: 17, 14, 31
- 11月: 7, 14, 21, 28
- 12月: 5, 12, 19
- 1月: 16, 23, 31(復習)

評価・成績

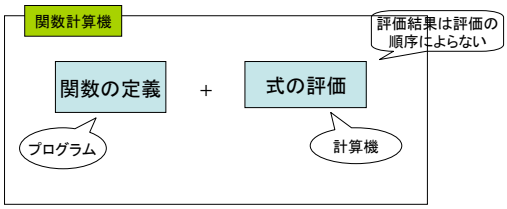
• レポート	30%
• 期末試験	70%

学習方法

- 講義中で内容を理解すること。
- 練習問題をやること。
- プログラムを書くこと。

関数プログラミングの基本的な概念

関数プログラミング



セッション (session)

```
Prelude> 17
17
Prelude> 2+5*3
17
Prelude> sin(1) + cos(1)
1.3817732906760363
Prelude> pi
3.141592653589793
Prelude> 7/2
3.5
```

スクリプト (Script): 関数定義

プログラム名

test.hs

定義の並び

```
square x = x * x
min x y | x<=y      = x
        | otherwise = y
```

Prelude.hs: よく使われている関数定義の集まり

新しい関数を使ってみる

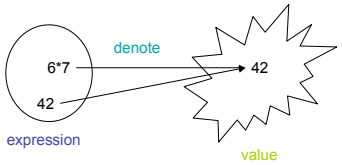
```
Prelude> :load test.hs
Reading file "test.hs"
Hugs session for
:/sw/share/hugs/lib/Prelude.hs
Test.hs
Prelude> square (3+4)
49
Prelude> min 3 4
3
Prelude> square (min 3 4)
9
```

練習問題

1. 関数 square を用いて引数を4乗する quadを設計せよ.
2. 2つの引数のうち大きい方を返す関数 maxを定義せよ.
3. 与えられた半径rの円の面積を計算す関数を定義せよ.

式と値

「式」は「値」を「表す」



簡約 (reduction)

```
square (3+4)
⇒ square 7
⇒ 7*7
⇒ 49
```

```
square (3+4)
⇒ (3+4) * (3+4)
⇒ 7*(3+4)
⇒ 7*7
⇒ 49
```

- 標準形 (canonical form), 正規形 (normal form)
  -- それ以上簡約できない式のこと.
- 底要素 (bottom value): ⊥
  -- 定義されない「値」(例: 1/0)

練習問題

1. 定義 three x = 3 について，three (3+4) を標準形に簡約する方法はいくつあるか．
2. 次のような構文規則で定義される整数を表す式を考えよう．
   (1) zero は式である．
   (2) eが式であれば，(succ e)及び(pred e)も式である．
   評価形は以下に述べた規則を可能な限り繰り返し適してこの言語の式を簡約する．
   (succ (pred e)) => e
   (pred (succ e)) => e
   式(succ (pred (succ (pred (pred zero))))を簡単にせよ．
   この式に対して，何通りに簡約規則を適用することができるか．それらは，最終的に同じ結果を導くか．与えられたどんな式に対しても簡約処理が終了することを証明せよ．

型 (type)

- 基本型: Int, Bool, Char, String
- 合成型
  - リスト型: [t]
  - 組型: (t1, t2, …, tn)
  - 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．
- 型を適切に割り当てることのできない式は正しく定義されていると認めない．

型エラーの例

```
ay x = 'A'
bee x = x + ay x
```

型解析

関数と定義

f :: A -> B

- 関数定義

  自動導出可能

  ```
  double :: Int -> Int
  double x = x + x
  ```

- 関数適用 (function application)

  ```
  double 5        ==> 10
  ```

- 多様型関数 (polymorphic function)

  ```
  id :: a -> a
  id x = x
  ```

定義の形式

場合分け (case analysis)

guard

```
min x y | x <= y  = x
        | x > y   = y
```

局所的な定義 (local definition)

local definition

```
f x y | x >= 0  = x + a
      | otherwise = x - a
  where a = square (y+1)
```

カリー化 (currying)

```
min' (x, y) | x <= y  = x
            | x > y   = y
```

1引数関数

カリー化

```
min x y | x <= y  = x
        | x > y   = y
```

2引数関数

min :: Int -> (Int -> Int)

構造をもつ値の引数を単純な引数の列に置き換える方法をカリー化という

カリー化の例

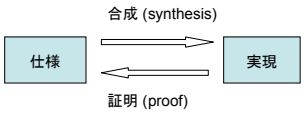
- add x y = x + y
  - add :: Int -> Int -> Int
    「Int -> (Int -> Int)」
  - add x y = (add x) y
  - (add x): xを出し込む関数
  - (add 0): 恒等関数

練習問題

1. 次のような型の関数の例をあげよ.
   1. (Int -> Int) -> Int
   2. Int -> (Int -> Int)
   3. (Int -> Int) -> (Int -> Int)
2. 次のような関数 sign :: Int -> Intを定義せよ. 関数 sign は引数が正であれば1を返し, 引数が負であれば-1を返し, それ以外の場合は0を返すものである.

仕様と実現

- 仕様 (specification)
  - プログラムが成し遂げるべき仕事の数学的記述
- 実現 (implementation)
  - 仕様を満たすプログラム



仕様と実現:例

- 関数 increase :: Int -> Int の仕様:
  すべての x>=0 に対して,
  increase x > square x.
- 実装
  - increase x = square (x+1)
  - increase x = square x + 1

Running Haskell Programs

Hugs をインストールする。
http://www.haskell.org/hugs/
(ECC has Hugs installed)

Primitive Library: Prelude.hs

Extended Library: Char.hs, List.hs, System.hs, …

Your Program: Test.hs, …

宿題

- 教科書を購入し, 第一章を読む.
- Hugs をインストールする.
- Hugsを使ってみる.

http://cvs.haskell.org/Hugs/pages/hugsman/basics.html