関数プログラミングの基本概念

胡 振江
東京大学 計数工学科
2006 年度

内容

- 関数
- 関数プログラミング言語
- 関数プログラムの構成
  - 関数の定義
  - 型
- 関数プログラムの評価
- 関数プログラムの処理系

関数（数学）

入力 x に対して，出力 y のただ一つの値を決定する規則が与えられているとき、 y を x の関数という．対応規則を明示するときは，

$$y = f(x)$$

のように対応規則に名前を付与する．

C の関数 $\neq$ 関数 (数学)

例：プログラミング言語 C の関数定義

```
1: int y = 10;
2: int f (x: int) {
3:     printf (‘‘Hello!’’);
4:     return y * x;
5: }
```

次の点で数学の関数とは異なる.

- 関数の処理の実行によってシステムに変化が発生する（副作用を持つ）
- 引数が同じでも状況に応じて戻り値が異なる（状態を持つ）

本講義中の関数 = 関数 (数学)

関数

$$f :: A \to B$$

は型 A の引数 (argument) をとり，型 B の結果 (result) を返すという．また，x が A の元を表しているとき，x に関数 f を適用した結果を $f(x)$ または $f\ x$ で表す.

関数プログラミング言語

関数プログラミング言語は，**ラムダ計算** *(lambda calculus)* の概念をプログラミング言語として体現したもの．すべての計算は関数の適用 (簡約) によって行われるというもので，C などの手続き型言語と違い，状態という概念をもたない．

代表的な関数プログラミング言語

- Lisp 1958- (Scheme 1970 年代後半-)
- ISWIM 1966-
- ML 1970 年代後半-
- Miranda 1985-
- **Haskell** 1987- (純粋関数プログラミング言語，Haskell の名前は論理学者 Haskell B. Curry に由来, 本講義で使う言語)

関数プログラムの構成

関数プログラムは関数の定義の集まりである.

```
square x = x * x
min x y = if x <= y then x else y
side = 12
area = square side
fac n = if n==0 then 1 else n * fac (n-1)
```

関数プログラムの実行は関数適用である.

```
square (3+4) => 49
area => 144
fac 5 => 120
```

関数の定義（Core の部分）

```
関数定義 ::= 関数名 変数 変数 ... 変数 = 式

式 ::= 定数
     | 変数
     | 関数名 式 式 ... 式
     | if 式 then 式 else 式
     | let 変数 = 式 in 式
```

例：

```
f x1 y1 x2 y2 = let a = x2-x1 in
                let b = y2-y1 in
                if a>b then square a else 0
```

関数 until:

```
until p f x = if p x then x else until p f (f x)
```

型

型 (type) はある種類の値の集まりである.

- 基本型
  - 整数型 Int: $\{\ldots, -3, 2, 0, 10, \cdots\}$
  - 浮動小数点数型 Float: $\{\ldots, -3.14, 3, 0, 123.45, \ldots\}$
  - 論理型 Bool: $\{True, False\}$
  - 文字型 Char: $\{'a', 'b', \ldots, 'A', 'B', \ldots\}$
  - 文字列型 String: $\{\text{“}HelloWorld\text{”}, \ldots\}$
- 合成型 (派生型)
  - リスト型 $[t]$: $\{[], [2, 5, 4], \ldots\}$
  - 組型 (t_1, t_2): $\{(1, 2), ('a', 3), \ldots\}$
  - 関数型 $t_1 \to t_2$: $\{square, add1, \ldots\}$

関数の型

関数プログラミングにおいて，関数は値であり，他の値と対等のものである．**関数を引数として関数に渡したり，結果として返したりする．**

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

addSquare :: Int -> Int -> Int
addSquare x y = square x + square y

app ::  (Int -> Int) -> Int -> Int
app f x = f x
```

関数を定義するときに，型をつけましょう．

*注 $a \to b \to c$ は $a \to (b \to c)$ の略形．

関数プログラムの評価

関数プログラムの評価は，式を「最も単純な等価な形」に簡約し結果を表す過程である．

評価例：

$$\begin{array}{lll} square\ (3+4) & \Rightarrow & square\ 7 \quad + \\ & \Rightarrow & 7*7 \qquad square \\ & \Rightarrow & 49 \qquad\ * \end{array}$$

注：式 $square\ (3+4)$ を簡単にする方法はこのほかにもある．

$$\begin{array}{lll} square\ (3+4) & \Rightarrow & (3+4)*(3+4) \quad square \\ & \Rightarrow & 7*(3+4) \qquad\quad + \\ & \Rightarrow & 7*7 \qquad\qquad + \\ & \Rightarrow & 49 \qquad\qquad * \end{array}$$

関数プログラムの処理系

本講義で関数プログラミング言語 Haskell (コア部分)
http://www.haskell.org/haskellwiki/Haskell
を使う.

- Haskell のコンパイラ: ghc
  http://www.haskell.org/ghc/
- Haskell のインタプリタ: hugs
  http://haskell.org/hugs/

Haskell のインタプリタ: Hugs

- Mark P. Jones らによって開発された Haskell のインタプリタで，現在の最新版は Hugs 98 である．
- Hugs は，多くの Unix や Windows 上で動くことが確認されており，コンパイル済のバイナリコードが Win32, Linux, Machintosh の各プラットフォームに用意されている．
- Windows 用の Hugs (winhugs) は GUI を備えており，初心者でも扱いやすくなっている．
- Hugs のインストルは簡単である．

Hugs システムをインストルしてください．