プログラム言語論 第5週

亀山幸義, 2009/05/18

1 今日の目標

関数型プログラム言語 (主として ML 系言語) の基本的な特徴を**理解**すること。

- (C より簡単) 「式」だけからなる.
- (C と同じ) ブロック構造を持つ. 静的束縛である.
- (C より簡単) 単一代入である.
- (C より高級) 関数を普通のデータとして扱うことができる.
- (C より便利) 複雑なデータ型を自由自在に定義できる.

2 MinML の例題

・let 式を使った簡単な例:

```
let x = 1 in
  let y = 2 in
    x + y

let f = (fun x -> x + 1) in
  f (f 3)

(let x = 1 in
   x + x)
+ 5

(fun x -> if x = 1 then 2 else 3) 1

(fun x -> if x = 1 then 2 else 3) 10
```

・ブロック構造, 変数のスコープ

```
let x=5 in
  (let f=(fun y->x+y) in
    let x=10 in
     (f 20) + x * 2)
  + x * 3

let rec f = (fun x->
              if x=0 then 1
```

```
            else x * f (x-1)) in
  (f 10)
```

・関数をデータとして扱う

```
let f = (fun x -> fun y -> x + y) in
  f 3 5

let f = (fun x -> fun y -> x + y) in
  (f 3) 5

let f = (fun x -> fun y -> x + y) in
  let g = f 3 in
    (g 5) + (g 7)

(fun f -> f (f 3)) (fun x -> x + 1)

(fun f -> fun x -> f (f x)) (fun y -> y + 1) 3
```

・値呼び，名前呼び

```
let f = fun x -> (print x; 2) in
  (fun y -> y + y + y) (f 1)

let f = fun x -> (print x; 2) in
  (fun y -> (y 1) + (y 1) + (y 1)) f

let f = fun x -> (print 1; 5) in
  let g = fun x -> (print 2; 7) in
    f g
```

・対 (ペア) を表すデータ型

```
(1,2)

fst (1,true)

fst (snd (1,(true,3)))

(fun x -> fun y -> ((x,y),(y,x))) 10 true

(fun x -> (fst x) ((snd x) 10))
   ((fun z -> z * 3), (fun y -> y + 2))
```

参考: OCaml の使い方について: インターネット上に解説テキストがいくつかあるので検索してみるとよい。本家 (フランス INRIA 研究所) の解説は、http://caml.inria.fr/pub/docs/manual-ocaml/ である。また、日本語での教科書も 3 冊 (それぞれの著者は、京都大学五十嵐淳氏、お茶の水女子大学浅井健一氏、OCaml-Nagoya グループ) 出版されている。