ML 演習 第 1 回

Yutaka Oiwa
大岩 寛 (米澤研)
April 10, 2001

実験1 言語コースの形式

- 前半: ML (ocaml) 言語の演習
- 後半: Prolog 言語の演習

(合計 15回)

各演習の担当者

- ML 演習: 大岩 寛 (米澤研)
  - ml-query@yl.is.s.u-tokyo.ac.jp
  - oiwa@yl.is.s.u-tokyo.ac.jp
- Prolog 演習: 吉田 稔 (辻井研)
  - mino@is.s.u-tokyo.ac.jp

ＭＬ演習の形式

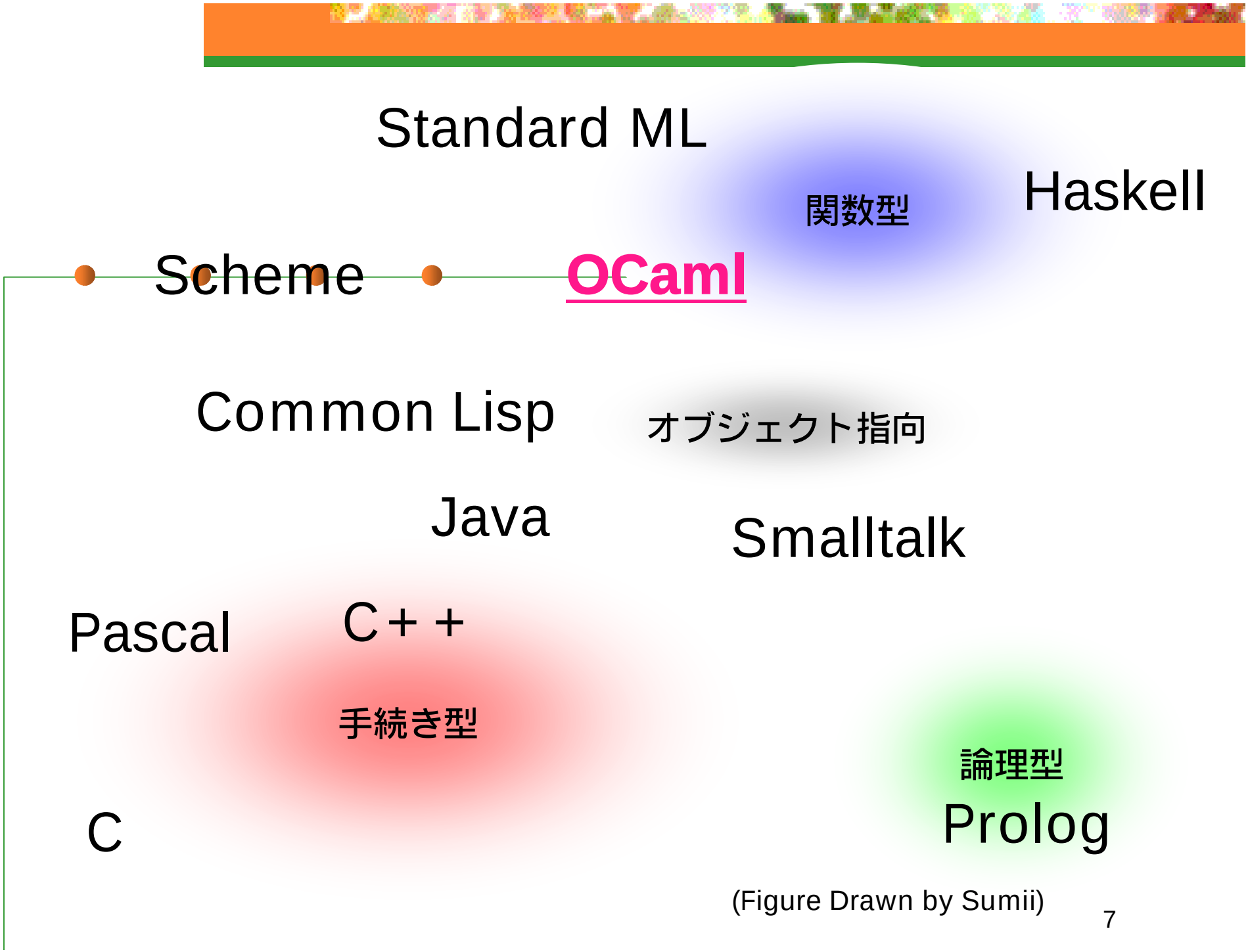
- 採点は基本的にレポート
  - 期限: 基本的に出題後2週間
  - 提出状況に大きな重点
    - 途中までででも1回は締め切りまでに進捗を報告してください
      - 何がわかったのか、何がわからないのか…
- 質問歓迎
  - ml-query@yl.is.s.u-tokyo.ac.jp

判定基準

- 実験1全体は3コースの成績の and
- 言語コースは ML と Prolog の and
- ML 演習の判定条件は次の 3 つの or
  - 相当数のレポートをきちんと提出すること
  - 全ての課題を完成させること
  - 最終課題でとてもとても素晴らしいレポートを提出すること

参考資料

- 演習資料
  - http://www.yl/~oiwa/lecture/ocaml/ 以下
- OCaml の本家サイト
  - マニュアル・紹介など
  - http://caml.inria.fr/
- The Functional Approach to Programming (Cambridge Univ. Press)



(Figure Drawn by Sumii)

OCaml の特徴

- 型システム
- データ型定義
- パターンマッチング
- モジュールシステム
- 例外処理
- オブジェクト指向のサポート

MLの型システム

- 強い静的な型付け (Strong Static Typing)
  ←→動的な型付け (e.g. Scheme, Perl),
  弱い型付け (e.g. C, C++)
- 型推論
  ←→ 型の明示的な指定 (e.g. STL (C++))
- Parametric Polymorphism (型多相性)

OCaml 処理系 (1)

```
[oiwa@harp] ~> ocaml
 Objective Caml version 3.00

# 1 + 2;;
- : int = 3
# #use “test.ml”;;
- : int = 3;
- : string = “Test”
# ^D
[oiwa@harp]~>
```

OCaml 処理系 (2)

- エラー処理

```
# 1 + 2.0;;
This expression has type float but
  is here used with type int
```

- **Emacs との連携**
  - **ocaml-mode**
  - **ocamldebug**
    - 詳しくはマニュアル参照

値の定義と利用 (1)

- let 文

```
# let a = 3;;
val a : int = 3
# let f x = x + 1;;
val f : int -> int = <fun>
# f a;;
- : int = 4
```

値の定義と利用 (2)

- let ... in 文

```
# let x = 3 in x + x;;
- : int = 6
# let f x = x + x in f 2;;
- : int = 4
```

組み込み型 (1)

- 整数 (int)

```
# (3 + 5) * 8 / -4;;
- : int = -16
# 5 / 4;;
- : int = 1
# 5 mod 4;;
- : int = 1
# 3 < 2;;
- : bool = false
```

組み込み型 (2)

- 実数 (float)

```
# (3.0 +. 5.0) *. 8.0 /. -3.0;;
- : float = -21.333333
# 1.41421356 ** 2.0;;
- : float = 2.000000;;
# 3.0 < 2.0;;
- : bool = false
```

組み込み型 (3)

- 真偽値 (bool)

```
# 2 < 3 && 2.0 >= 3.0;;
- : bool = false
# 2 < 3 || 2.0 = 3.0;;
- : bool = true
# not (3 < 2);;
- : bool = true
```

組み込み型 (4)

- 文字列 (string)

```
# “Str” ^ “ing”;;
- : string = “String”
# print_string “Hello¥nWorld¥n”;;
Hello
World
- : unit = ()
```

組み込み型 (5)

- Tuple

```
# (3 + 5, 5.0 -. 1.0);;
- : int * float = 8, 4.000000
# fst (3, 2);;
- : int = 3
# snd (3, 2);;
- : int = 2
# (3, true, “A”);;
- : int * bool * string = 3,true,“A”
```

関数型 (1)

- 関数 (function)

```
# let f x = x + 2;;
val f : int -> int = <fun>
# f 2;;
- : int = 4
# fun x -> x + 2;;
- : int -> int = <fun>
# (fun x -> x + 2) 2;;
- : int = 4
```

関数型 (2)

- 多引数関数

```
# let f x y = x * (x + y);;
val f : int -> int -> int = <fun>
# f 2 4;;
- : int = 12
```

関数型 (3)

- [参考] 多引数関数の型
  - カリー化 (Curried) 表現

```
# let f x y = x + y;;
val f : int -> int -> int = <fun>
# f 2;;
- : int -> int = <fun>
# (f 2) 4;;
- : int = 6
```

組み込みの構文 (1)

- 局所定義 (let ... in ...)
- 条件分岐

```
# let f x = if x < 2
            then “smaller than 2”
            else “not smaller than 2”;;
val f : int -> string = <fun>
# f 1;;
- : string = “smaller than 2”
```

組み込みの構文 (2)

- 再帰関数

```
# let rec fib x =
    if x < 2 then 1
    else fib(x-1) + fib(x-2)
val fib : int -> int = <fun>
# fib 10;;
- : int = 89
```

組み込みの構文 (3)

- 再帰関数 (続)

```
# let rec pow x n = if n = 0 then 1
                    else x * pow x (n-1);;
val pow : int -> int -> int = <fun>
# pow 3 10;;
- : int = 59049
```

組み込みの構文 (4)

- 相互再帰関数の同時定義

```
# let rec even x = if x=0 then true
                   else odd(x-1)
  and odd x = if x=0 then false
              else even(x-1);;
val even : int -> bool = <fun>
val odd : int -> bool = <fun>
# odd 5423;;
- : bool = true
```

課題1

1. 非負整数2つの最大公約数を求める関数 gcm : int → int → int を定義せよ。
   - ヒント: ユークリッドの互除法。

課題2

- pow を改良してより高速にせよ。
  - 計算量を引数 n の log オーダにする。
  - ヒント:

$$x^0 = 1,$$
$$x^{2n} = (x^2)^n \qquad (n > 0),$$
$$x^{2n+1} = x \cdot x^{2n}$$

課題3

- fib を改良してより高速にせよ。
  - 計算量を引数 n の1次のオーダにする。
    - ヒント: 3引数の補助関数を作って...。

課題の提出方法

- 締め切り: 4月30日 (月) 24:00
  - 今回は計算機の準備状況を見て考えます。
- 提出方法: 電子メール
  - ml-report@yl.is.s.u-tokyo.ac.jp
  - Subject を “Report 1 10xxx” (学生証番号) とすること。
  - 地下計算機以外から提出する際は地下計算機のアカウント名を明示すること。