ML 演習 第 1 回

担当: 大岩 寛, 水上 達夫

April 9, 2002

実験1 火曜日の形式

- 火曜日: 言語コース
  - 前半: ML (OCaml) 言語コース (50点)
  - 後半: Prolog 言語コース (50点)

各演習の担当者

- ML 演習: 大岩 寛, 水上 達夫 (米澤研)
  - ml-query@yl.is.s.u-tokyo.ac.jp
  - oiwa@yl.is.s.u-tokyo.ac.jp
  - tatsuo@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4コースの成績の and
- ML コースの判定条件は次の 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 (例外)

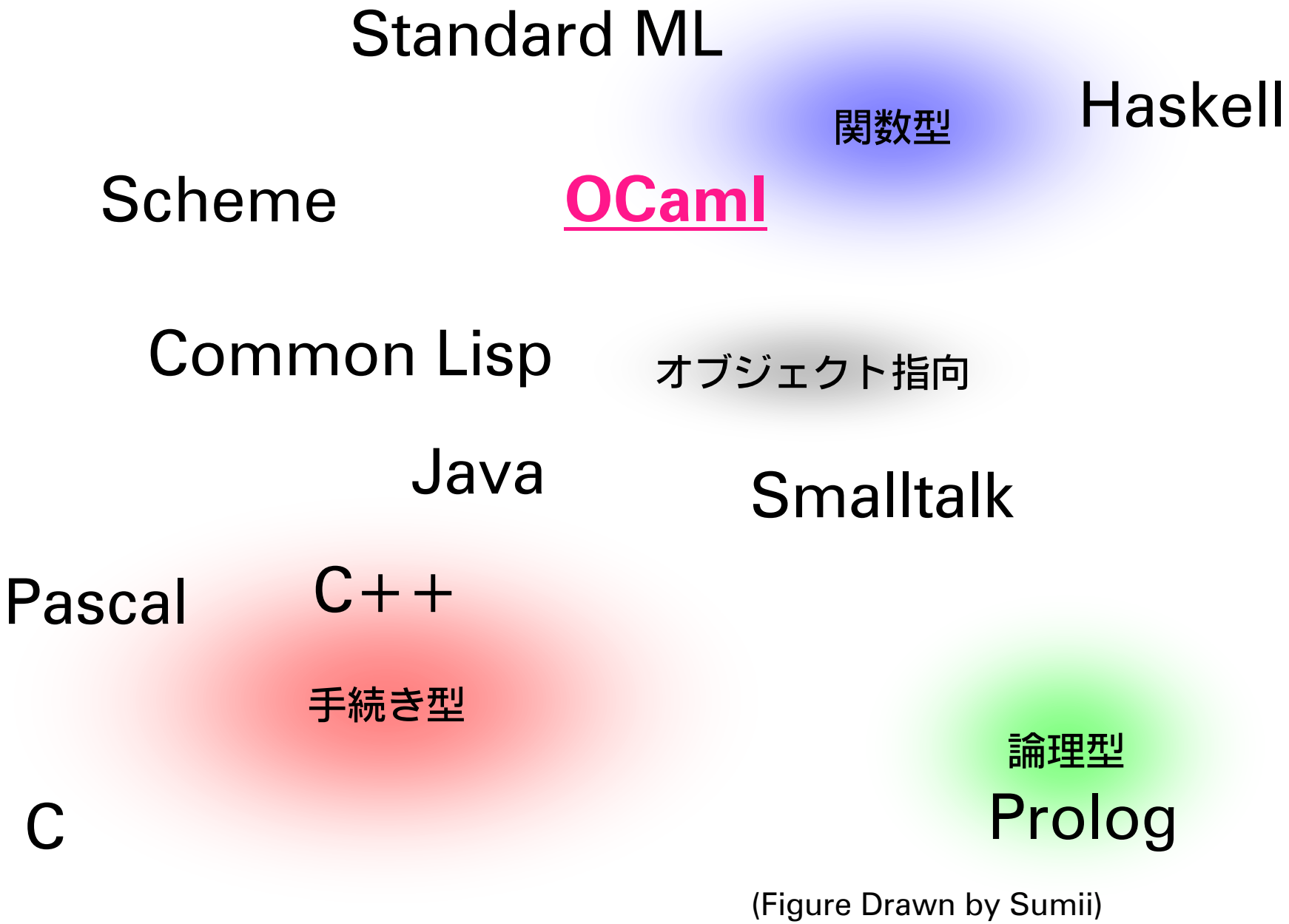
参考資料

- 演習資料
  - http://www.yl.is.s.u-tokyo.ac.jp/~oiwa/lecture/ocaml/
- OCaml の本家サイト
  http://caml.inria.fr/
  - マニュアル・紹介など
    - マニュアルの tutorial (1章) はよくできてます
  - 処理系のダウンロード (Windows 版など)

コースガイド

- 最初の4回: OCaml 言語の基礎演習
- そのあと3回: 言語処理系の基礎
  - 簡単な言語の処理系を作ってみることで
    処理系の内部をちょっとだけ覗いてみます
- 最終回: 最終課題の出題

- 冬学期: コンパイラ演習


Standard ML
関数型
Haskell
Scheme
OCaml
Common Lisp
オブジェクト指向
Java
Smalltalk
Pascal
C++
手続き型
論理型
C
Prolog
(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 インタプリタ (1)

```
[oiwa@harp] ~> ocaml
        Objective Caml version 3.04

# 1 + 2;;
- : int = 3
# #use “test.ml”;;
- : int = 3;
- : string = “Test”
# ^D
```

OCaml インタプリタ (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: トップレベルの値の定義

```
# 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: ローカルな宣言

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

- 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:

$$a^0 = 1$$
$$a^{2n} = (a \times a)^n \qquad [n > 0]$$
$$a^{2n+1} = a \times (a^{2n})$$

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22日 (火) 13:00
- 提出方法: 電子メール
  - ml-report@yl.is.s.u-tokyo.ac.jp
    - 受領通知が届くと思うので確認してください。
  - Subject を “Report 1 210xx” (学生証番号) などとすること。
  - 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書くこと。