ML 演習 第 1 回

担当: 大山, 大岩, 永田

April 8, 2003

情報科学実験1 火曜日の形式

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

各演習の担当者

- ML 演習
  - 大山 恵弘, 大岩 寛, 永田 章人 (米澤研)
    - ml-query@yl.is.s.u-tokyo.ac.jp
    - {oyama,oiwa,ganat}@yl.is.s.u-tokyo.ac.jp
- Prolog 演習
  - 宮尾 祐介, 吉永 直樹, 薬師寺あかね (辻井研)

ML演習の形式

- 採点は基本的に課題レポート
  - 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

講義予定 (1)

- 第1回～第4回
  - ML言語の基礎
    - 文法、意味、動作
- 第5回～第7回
  - 言語処理系の基礎
    - 式の評価、評価順序、型
- 第8回
  - 最終課題

講義予定 (2)

- 第8回: 最終課題（夏休みの自由研究）
  - 予定課題の例 (選択式)
    - なにか使えるプログラムを OCaml で作る。
    - LOGO など簡単な言語の処理系を作ってみる。
    - 関数型言語に関する調査、論文読み。
    - ICFP Programming Contest に挑戦する。

判定基準 (1)

- 実験1全体の判定: 4コースの AND
  - どれか1つでも未達成の場合、
    そのコースだけ再履修になります。
- ML コースの判定基本方針: 努力 or 実力
  - こつこつ地道に努力するか、
  - 文句の出ない課題を出すか、
  - 「不可」のつけようのない最終課題を出せば
    単位はでます。

判定基準 (2)

- (例) 努力コース:
  - 課題をこつこつ出す。（締め切り厳守）
  - 目安:
    - いい成績 → 1~6 辺りまでの基本課題を
      きちんとやって、最終課題も出す。
    - 50点 → 1~3 辺りはきちんとやって、
      4~6 は完成しなくても中途レポートを期日に
      提出して、やれる範囲で追加レポートも
      出して、最終課題もなにか小さいものを作ればOK?

判定基準 (3)

- (例) 実力コース:
  - とにかく学期末までに課題を全部出す。
  - Optional 課題も基本的にはやる。
  - 最終課題ももちろんやる。
  - 出席とかはうるさく言う気はありません。

判定基準 (4)

- (例) 一発逆転コース:
  - どんなに僕が怒り心頭でも「不可」を
    つけられないような最終課題を出す。
    - 基準: 世の人から拍手喝采で迎えられるもの。
    - 例:
      - Perl より速い Perl インタプリタ
      - gcc より速い C コンパイラ
      - Mozilla より便利な Web ブラウザ
      - ICFP コンテストで入賞（2位以内）

課題の解き方

- 地下端末室には処理系が既に
  インストールしてあります。
- 自宅のマシンでやる場合は各自
  入手・インストールしてください。
  - Unix: ソースファイルからコンパイル
    - Debian GNU/Linux 等はバイナリもある
  - Windows 版: バイナリ配布がある
    - または自前でコンパイル (VC6, mingw32, cygwin)

演習資料について

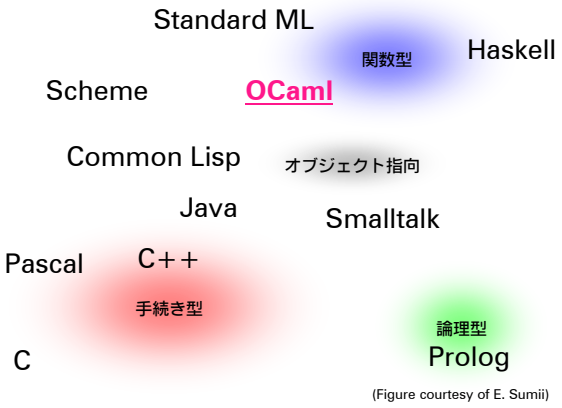
- ML演習ホームページ
  - http://www.yl.is.s.u-tokyo.ac.jp/~oiwa/lecture/ocaml/
  - 講義資料 (PDF形式)
  - 演習で使用するソースファイル
  - そのほか参考文書
  - OCaml のソース
  - ML演習サポート掲示板 (永田君運営)

参考資料

- OCaml の本家サイト http://caml.inria.fr/
  - マニュアル・紹介など
    - 第1章のチュートリアルは簡潔でよい
  - 処理系のダウンロード (Windows 版など)
- Developing Applications With Objective Caml
  - Online pre-release
  - http://caml.inria.fr/oreilly-book/
- その他フランス語書籍はやまほど…

Objective Caml とは?

- 関数型言語 ML の1流派
  - 強い型システム
  - 柔軟なデータ型定義とパターンマッチ
  - 強力なモジュールシステム
  - オブジェクト指向プログラミングのサポート



(Figure courtesy of E. Sumii)

MLの型システム

- 強い静的な型付け (Strong Static Typing)
  - コンパイル時に型の整合性をチェックする
    - ↔ 動的な型付け (e.g. Scheme, Perl)
  - 型整合を強制する
    - ↔ 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@tuba] ~> ocaml
        Objective Caml version 3.06

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

test.ml の中身:

```
1 + 2;;
"Te" ^ "st";;
```


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: トップレベルの定義

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

- 関数適用

```
# 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: if

```
# 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)

- 再帰関数の定義: let rec

```
# 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
```


組み込みの構文 (5)

- let と let rec の違い…

```
# let f x = x + 3;;

# let f x = f (x-1);;
# f 2;;
- : int = 4

# let rec f x = f (x-1);;
# f 2;; (* 止まらない *)
```


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: Euclid 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: n mod 2 で分岐。

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

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 310xx”
    (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。