OCaml Golf

中野 圭介 / Keisuke NAKANO
（電気通信大学）

平成廿一年東都大駱駝会 (OCaml Meeting Tokyo), 2009年8月30日

OCamlとGolfにまつわる自己紹介

- 19xx年 Tiger Woods より 3 日早く生まれる
- 1996年末 Objective Label (OLabl) に触れる

  - OLabl … OCamlから派生して，後に OCamlに合流
- 2006年夏 Code Golfに興味を持つ

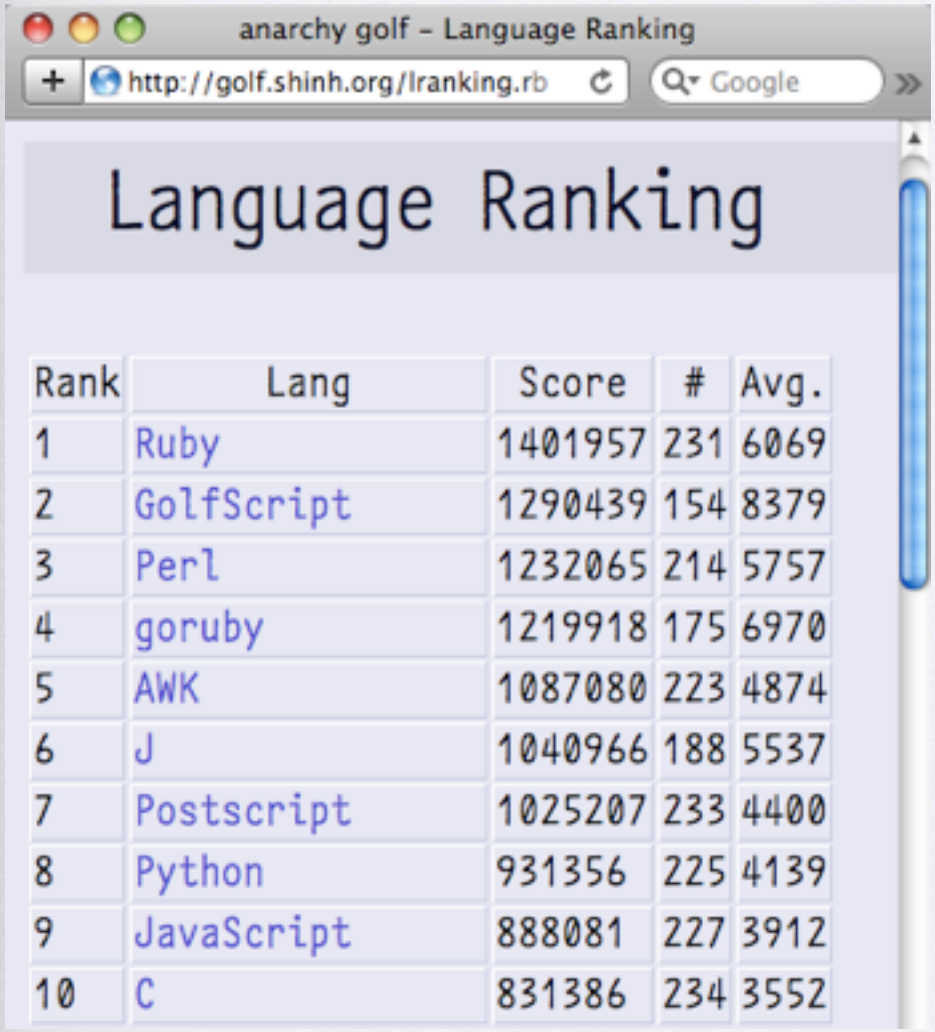
  - Perl,Ruby,Python,PHP が対象だが OCaml で解く
- 2007年初 Anarchy Golf 開設され OCaml で参戦

Code Golfとは

- 少ないキーストローク数で問題を解く競技
  - スポーツのゴルフは少ないストローク数でカップにボールを入れる競技
  - 実際にカウントするのはバイト数
    - キーストロークだと　キーボードの配列に依存
  - 要するに，短いコードで問題を解く競技　(遊び)
  - Golf と最初に名付けらたのは Perl Golf (1999)
    - 同様のことは Knuth が 1950 年代にマシン語でしていた?

Golf における OCaml の地位

- 69 言語中 20 位 [8月30日現在, anarchy golf 調べ]
  - バイト数の少なさで比較



Language Ranking

Rank	Lang	Score	#	Avg.
1	Ruby	1401957	231	6069
2	GolfScript	1290439	154	8379
3	Perl	1232065	214	5757
4	goruby	1219918	175	6970
5	AWK	1087080	223	4874
6	J	1040966	188	5537
7	Postscript	1025207	233	4400
8	Python	931356	225	4139
9	JavaScript	888081	227	3912
10	C	831386	234	3552
11	z80	816064	131	6229
12	Groovy	762137	203	3754
13	Common LISP	716298	216	3316
14	PHP	661532	173	3823
15	Haskell	654622	209	3132
16	sed	630109	147	4286
17	Bash	611374	127	4813
18	BASIC	597828	181	3302
19	Lua	529748	176	3009
20	OCaml	520998	185	2816
21	Io	491868	163	3017
22	Smalltalk	482565	159	3035
23	Scheme	479835	163	2943
24	D	454572	180	2525

http://golf.shinh.org/lranking.rb を参照

なぜ OCaml は Golf に弱いか
— 文字列処理の場合 —

- 文字列の出力だけで 12 バイト
  - print_string, print_char
  - print_endline, print_newline
  - Printf.printf
    - printf くらいは Pervasive に入れてほしい
    - あるいはデフォルトで open Printf か open Format

Rubyなら
p とか puts とか
putc とか $><< とか

- 文字列の処理ライブラリが貧困
  - Ruby の s[i..j]="abc" をどう書く?
  - 正規表現を使うには #load"str.cma"
    - /a+(b)/ が Str.regexp"a+\(b\)"
    - しかも Str.match_string とか長い

なぜ OCaml は Golf に弱いか
— 数値処理の場合 —

- int は符号を除いて 30 ビットまで
  - それ以上は #load"nums.cma"
    - open Num しないと中置演算子 (+/) などが使えない
  - Int64 でもよいが 12+34 が Int64.add 12L 34L
    - なぜか中置演算子が用意されていない
- float の中置演算子 (+.),(-.),(*.),(/.)
  - よく槍玉にあがる OCaml の弱み
  - Standard ML は real の足し算も (+) なのに…
- 他にもいろいろ
  - intに冪関数がない，剰余関数が(%)ではない…

本発表の目的
— OCamlで如何に短く書くか —

- Golf が苦手な OCaml における Golf の技術を紹介
  - コードを縮める基本的なテクニック
  - OCamlの特徴を利用した縮め方
    - 中置演算子の(再)定義
    - オプショナル引数の利用
- 無駄な(?)技術を学ぶことで言語の問題点を発見
  - OCaml 4 になる頃には改善されていてほしい！

OCaml Golf テクニック
── 初級編 ──

基本中の基本

- 「;;」は高々1回，直後は改行不要
  - そもそも「;;」を書く派？ 書かない派？

```
open Printf

let apply_empty f = f "";;

let rec repeat i s tmp =
    if i<1 then tmp else repeat (i-1) (s^tmp);;

let repeat i s = apply_empty (repeat i s);;

for i=1 to 10 do
    printf "%s
"  (repeat i "ocaml")
done
```

基本中の基本

- 「;;」は高々1回，直後は改行不要
  - そもそも「;;」を書く派？ 書かない派？

```
open Printf

let apply_empty f = f "";;

let rec repeat i s tmp =
    if i<1 then tmp else repeat (i-1) (s^tmp);;

let repeat i s = apply_empty (repeat i s);;

for i=1 to 10 do
    printf "%s
"  (repeat i "ocaml")
done
```

不要

不要

必要

基本中の基本

- 「;;」は高々1回，直後は改行不要
  - そもそも「;;」を書く派？ 書かない派？
- 余分な空白は除去
  - `if x > 100 then x - 10 else f (f (x + 11))`
  - `if x>100 then x-10 else f(f(x+11))`
  - `if x>100then x-10else f(f(x+11))`
    - keywordの前が数字なら空白は不要
- 余分な括弧は除去
  - LISPじゃないんだから
- 文字列内の改行は "\n" でなく生で

基本中の基本

- 「;;」は高々1回，直後は改行不要
  - そもそも「;;」を書く派？ 書かない派？
- 余分な空白は除去
  - `if x > 100 then x - 10 else f (f (x + 11))`
  - `if x>100 then x-10 else f(f(x+11))`
  - `if x>100then x-10else f(f(x+11))`
    - keywordの前が数字なら空白は不要
-

```
open Printf;;for i=1to 100do printf"%d
"i  done
```

- 文字列内の改行は "\n" でなく生で

原則的には if 式禁止 (1)

- if A then B else C は長い
  - 「A & B || C」で代用する
    - 型が変わるので注意が必要 (B や C は bool)
    - ()=print_string s などでごまかす
    - 「(&) は deprecated」とか気にしない

```
let rec fact i=if i<1 then 1 else i*fact(i-1);;print_int(fact 10)

let rec f i=if i<1then 1else i*f(i-1);;print_int(fact 10)

let rec f i n=if i<1 then print_int n else f(i-1)(n*i);;f 10 1

let rec f i n=i<1&()=print_int n||f(i-1)(n*i);;f 10 1
```

原則的には if 式禁止 (2)

- A & B ; C で十分なときもある
  - 再帰を続けると B や C がエラーになるとき
  - A & B || C より 1byte 少ない！

```
let rec f i n=i<1&()=print_int n;f(i-1)(n+100/i);;f 10
```

$$\sum_{i=1}^{10} \left\lfloor \frac{100}{i} \right\rfloor$$

- 配列を使って分岐する方法もある

```
if n mod 2=0 then f(n/2) else f(n*3+1)
f[|n/2;n*3+1|].(n mod 2)
```

  - 片方だけ再帰のときは注意 (OCamlは値呼び)
    - let rec f i=[|i*f(i-1);1|].(1/i) はダメ
    - let rec f i x=i*[|f(i-1);(+)1|].(1/i)0 は OK

再帰 vs. ループ

- 記憶する変数が多いときは再帰
  - `let rec f x y = ... f(x-y)(x+y) ...`
- 各処理が独立しているなら while 式
  - `while 1=1do ... done`
- 回数が決まっているなら for 式
  - 時には `Array.init` が役に立つことも
- これらはあくまで原則
  - とりあえず全部試してチェックした方がよい

OCaml Golf テクニック
── 中級編 ──

中置演算子の利用

- 汎用的な有効手段の一つ

```
let rec f i n=i<1&()=print_int n||f(i-1)(n*i);;f 10 1
let rec($)i n=i<1&()=print_int n||i-1$n*i;;10$1
```

- 括弧を節約するために優先順位に注意
  - ** < *,/,% < +,- < :: < @,^ < =,<,>,$ < & < ||
    - 返り値を別の演算子に渡すなら (%)
    - 返り値を使わないなら ($)
  - 結合にも注意 @,^,:: は右，その他は左
- ときには 3 引数関数にも有効
  - f x y z ⇒ (x$y)z

前置演算子の利用

- ！ か ~ で始まる記号列は前置
  - ~の場合は 2文字以上なので有効なのは(!)のみ

```
let f=Char.code;;Printf.printf"(%d,%d,%d)"(f c)(f d)(f e)

let(!)=Char.code;;Printf.printf"(%d,%d,%d)"!c!d!e
```

- ときには 2 引数以上の関数にも有効

```
open List;;map((land)3)x@map((land)7)x@map((land)15)x

let(!)=(land)open List;;map!3x@map!7x@map!15x
```

let の数を減らす

- 複数の定義は 1 つにまとめる
  - let a=1 let b=2 より let a,b=1,2
  - 関数のときは値多相に注意
    - let(%),a=Array.init 3,1 とすると (%) の型は (int -> '_a) -> '_a array になってしまう
  - 複数の let rec があるときは and を
    - let rec f x= ... and g x=... and h x=...
- letの代わりにプショナル引数を使う

```
let f x y=let z=x*10000in Printf.printf"%d %d"(x+z)(y+z)

let f x?(z=x*10000)y=Printf.printf"%d %d"(x+z)(y+z)
```

エラーでループを止める

- 終了条件を省略できるときがある
  - 零除算

```
let rec f x=print_int x;x>0&f(x-1)
let rec f x=print_int x;f(x-x/x)
```

  - 文字列・配列への不正アクセス

```
let rec f i=print_char"hoge".[i];x>0&f(x-1)
let rec f i=print_char"hoge".[i];f(x-1)
```

  - 標準入力の読み込み

```
let rec f()=f(print_endline(read_line()))
```

禁断の魔法　Obj.magic

- 型を強制的に変える魔法
  - Obj.magic 'a'+3 ⇒ 100
  - Obj.magic(1>0)*10 ⇒ 10
- 関数の型も変換可能

```
String.iter(fun c->print_int(Char.code c))"test"
String.iter(Obj.magic print_int)"test"
Obj.magic String.iter print_int"test"
```

- 整数の配列を文字列で圧縮可能
  - [|1234567;8901234;5678901|] (27byte) を
    Obj.magic"\015\173%\000\229\164\015\001kN\173" (20byte) で

その他の細かいテクニック

- match ... with の代わりに

```
let rec f(a::l)=...;l>[]&f l
```

- 入力を 1 文字 string 型で読み込むときは

```
String.make 1(input_char stdin)
Scanf.scanf"%1s"(^)""
```

- A，Bを両方実行して，両方偽なら偽
  - A || B ではダメ.

```
(i=i/3*3&()=p"Fizz")||(i=i/5*5&()=p"Buzz")← ダメ
(i>i/5*5||()>p"Buzz")>(i=i/3*3&()=p"Fizz")
```

※評価順序に注意

残り 1byte を縮める

- 引数の順序を変えてみる
  - f x(i-1)
  - f(i-1)x
- 式の位置を変えてみる
  - ...&(print_int i;f(i-1)x)
  - ...&f(print_int i;i-1)x
- 宣言の順序を変えてみる
  - #load"nums.cma"open Num let n= ...
  - open Num#load"nums.cma"let n= ...

おまけ

絶対に解けない問題

- 記号ゴルフ（記号のみで解く問題）
  - $>><<[*$<]（Rubyのechoプログラム）
  - OCaml では何も出力できない
- Helloworld-less Helloworld
  - 'h','e','l','o',',',' ','w','r','d',を使わずに "Hello, world!" を print する問題
  - print_XXX も output_XXX も printf も ダメ

[追記（2009.9.7）]
xsd さんが Obj.magic を使って解くことに成功しました.
それを縮めれば 73 バイトで書けます.

```
Obj.magic(+)242(^^)"\072\101\108\108\111\044\032\119\111\114\108\100\033"
```

運任せ

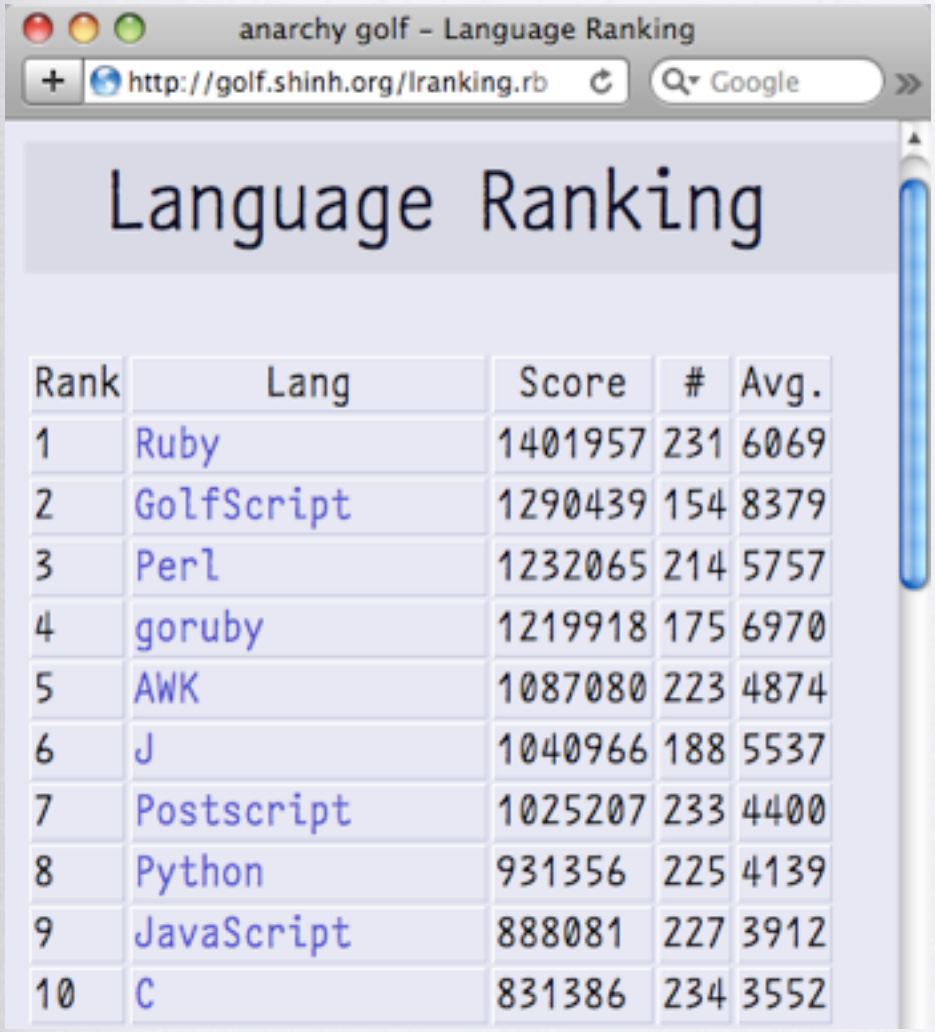
- 出力が少ないときに有効
  - [|答え1;答え2;答え3|].(-Obj.magic(@)mod 3)
  - 例: http://golf.shinh.org/p.rb?Greatest+Common+Divisor
    - 最大公約数を求める問題
    - 問題は 3 問，各入力は 2 行，各出力は 1 行
  - 十分な入出力を用意しない出題者が悪い

まとめ

- OCaml Golf のテクニックを紹介
  - 初級から中級まで
  - 他の人の技を盗むのもよいかも
    - http://golf.shinh.org/l.rb?ml が参考になるはず
- OCaml Golf で OCaml の改善点を発見
  - Golfのためでなく実用的な問題も
    - ファイルの処理でも OCaml を使いたくなるような
  - OCaml 4 へのフィードバックを希望

最後に

- まだ解かれていない問題も解いて
  OCamlの順位を上げましょう！



Language Ranking

Rank	Lang	Score	#	Avg.
1	Ruby	1401957	231	6069
2	GolfScript	1290439	154	8379
3	Perl	1232065	214	5757
4	goruby	1219918	175	6970
5	AWK	1087080	223	4874
6	J	1040966	188	5537
7	Postscript	1025207	233	4400
8	Python	931356	225	4139
9	JavaScript	888081	227	3912
10	C	831386	234	3552
11	z80	816064	131	6229
12	Groovy	762137	203	3754
13	Common LISP	716298	216	3316
14	PHP	661532	173	3823
15	Haskell	654622	209	3132
16	sed	630109	147	4286
17	Bash	611374	127	4813
18	BASIC	597828	181	3302
19	Lua	529748	176	3009
20	OCaml	520998	185	2816
21	Io	491868	163	3017
22	Smalltalk	482565	159	3035
23	Scheme	479835	163	2943
24	D	454572	180	2525