

A pink and yellow plumeria flower is the central focus, resting on a sandy beach. The petals are a mix of light pink and white, with a bright yellow center. The background is a soft-focus view of the sand and other flowers in the distance.

OCamlでらくらく関数型プログラミング

ocaml-nagoya

今井 宜洋

2009年8月22日

<http://www.itpl.co.jp/ocaml-nagoya/>

A pink and yellow flower with five petals is centered on a sandy background. The petals are a light pink color, and the center is a bright yellow. The sand is a light beige color with some darker spots.

OCamlとは:

- Objective Camlというプログラミング言語
- フランスの国立研究所INRIAで研究、開発
- 数学的な理論に基づいて設計された言語

A pink and yellow flower with five petals is centered on a sandy background. The petals are a light pink color, and the center is a bright yellow. The sand is a light beige color with some darker spots.

OCamlと名古屋:

- 名大の数学の研究室に開発者の一人がいる
- 勉強コミュニティ `ocaml-nagoya` がある
- 名古屋の会社(有)ITプランニング はOCamlで仕事をしている

A pink and yellow flower, possibly a plumeria, is shown in bloom on a sandy beach. The flower has five petals, with a yellow center and pink outer petals. The background is a soft-focus view of the sand and the ocean waves.

発表の流れ

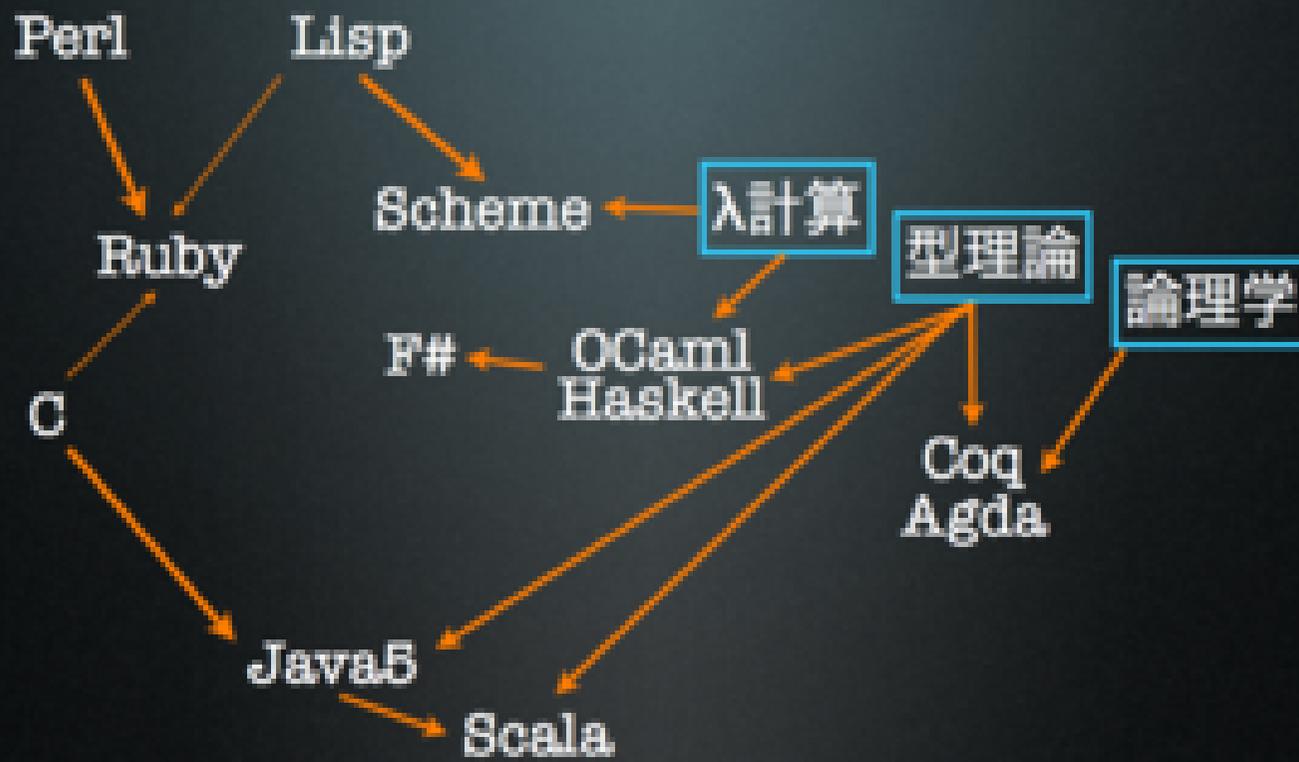
- OCamlと数学理論
- OCamlの5つの長所
- OCamlはこんなところで使われている
- OCamlの欠点
- まとめ

A close-up photograph of a pink and yellow flower, possibly a frangipani, resting on a sandy beach. The flower has five petals, with a bright yellow center and pinkish-purple outer petals. The background is a soft-focus view of the sand and the ocean waves.

OCamlと数学の理論

OCamlはプログラミング言語を研究している人が作った。

- λ 計算
- 型理論



A close-up photograph of a pink and yellow flower, possibly a plumeria, resting on a sandy beach. The flower has five petals, with a bright yellow center and pinkish-purple outer petals. The background is a soft-focus view of the sand and some distant foliage.

OCamlと数学の理論

一言でいうとCやJavaより安全で、
Perl,Rubyみたいにシンプル。

A close-up photograph of a pink and yellow plumeria flower resting on a sandy beach. The flower has five petals, with a bright yellow center and pink outer petals. The background is a soft-focus view of the sand and other flowers in the distance.

OCamlの5つの長所

- 安全である
- 柔軟
- シンプル
- 実行効率がよい
- 定理証明器と仲良し

A close-up photograph of a pink and yellow flower, possibly a frangipani, resting on a sandy beach. The flower has five petals, with a yellow center and pink outer edges. The background is a soft-focus view of the sand and some distant water.

OCamlは安全

静的で強い型システム

→ コンパイル時に型チェック

OCamlは安全

例:

```
let register name =  
  ...  
  
register "Yoshihiro";  
register "Ayumi"
```

OCamlは安全

例:

```
let register age name =  
  if age >= 20 then  
    ...  
  
register "Yoshihiro";  
register "Ayumi"
```

OCamlは安全

例:

```
let register age name =  
  if age >= 20 then  
    ...  
  
  register "Yoshihiro"; (* 型エラー *)  
  register "Ayumi"      (* 型エラー *)
```

OCamlは安全

例:

```
let register age name =  
  if age >= 20 then  
    ...  
  
  register "Yoshihiro"; (* 型エラー *)  
  register "Ayumi"      (* 型エラー *)
```

型チェックがないと、関係する全ての部分を目で探して修正しなければならない。

OCamlは安全

代数的データ型とパターンマッチング → null参照エラーは起き得ない。

OCaml:

```
match search "imai" with
| Some imai ->
  ...
| None ->
  ...
```

Java:

```
Person imai = search "imai";  
if (imai != null) {  
    ...  
} else {  
    ...  
}
```

A close-up photograph of a pink and yellow flower, possibly a frangipani, resting on a sandy beach. The flower has five petals, with a yellow center and pink outer edges. The background is a soft-focus view of the sand and some distant foliage.

OCamlは安全

問題: リスト (配列) の 1 番目と 2 番目を入れ換える

Java:

```
public <T> void swap(ArrayList<T> xs) {  
    if (xs.size() >= 2) {  
        T tmp = xs.get(0);  
        xs.set(0, xs.get(1));  
        xs.set(1, tmp);  
    }  
}
```

OCaml:

```
let swap = function
  | x1 :: x2 :: tail -> x2 :: x1 :: tail
  | 1 -> 1
```

A close-up photograph of a pink and yellow flower, possibly a frangipani, resting on a sandy beach. The flower has five petals, with a bright yellow center and pinkish-purple outer petals. The background is a soft-focus view of the sand and some distant waves.

OCamlは柔軟

高階関数

関数が第一級。関数が関数を返したり、クロージャーを渡すことができる。

OCamlは柔軟

高階関数 例: $\text{map } [x_1, x_2, \dots, x_n] \rightarrow [f(x_1), f(x_2), \dots, f(x_n)]$

OCaml:

```
List.map (fun x -> x * x) [1; 2; 3]
```

ruby:

```
[1, 2, 3].map { |x| x * x }
```

OCamlは柔軟

高階関数 例: iterator

OCaml:

```
List.iter print_string [ "It "; "is "; "cool!" ]
```

ruby:

```
[ "It ", "is ", "cool!" ].each { |x| puts x }
```

OCamlは柔軟

高階関数 例: `db proc`

```
let exec proc =  
  let conn = DB.open () in  
  let result =  
    try proc conn with  
    | err ->  
      DB.close conn;  
      raise err  
  in  
  DB.close conn;  
  result
```

OCamlは柔軟

OCamlではループじゃなくて再帰を使う。
再帰関数 例: GCD 最大公約数を求める関数

OCaml:

```
let rec gcd x y =  
  if y=0 then x  
  else gcd y (x mod y)
```

ruby:

```
def gcd( a, b )  
  i = a  
  while i > 0  
    break if (a % i == 0 && b % i == 0)  
    i -= 1  
  end  
  return i  
end
```

A close-up photograph of a pink and yellow plumeria flower resting on a sandy beach. The flower has five petals, with a yellow center and pink outer petals. The background is a soft-focus view of the sand and some distant foliage.

OCamlはシンプル

完全な型推論

型を書かなくてもよい。すべての型を省略することができる。

OCamlはシンプル

完全な型推論 例: succ

Java:

```
public static int succ(int x) {  
    return x + 1;  
}
```

OCaml:

```
let succ x = x + 1
```

OCamlはシンプル

完全な型推論 例: Hash

Java:

```
HashMap<String,ArrayList<Integer>> hash =  
    new HashMap<String, ArrayList<Integer>>();  
hash.put("Imai",  
        new ArrayList(new Integer[]{1,2,3}));
```

OCaml:

```
let hash = Hashtbl.create() in  
Hashtbl.set hash "Imai" [1;2;3]
```

A close-up photograph of a pink and yellow plumeria flower resting on a sandy beach. The flower has five petals, with a yellow center and pink outer edges. The background is a soft-focus view of the sand.

Ruby:

```
hash = Hash.new  
hash["Imai"] = [1,2,3]
```

OCamlはシンプル

完全な型推論 例: id Java:

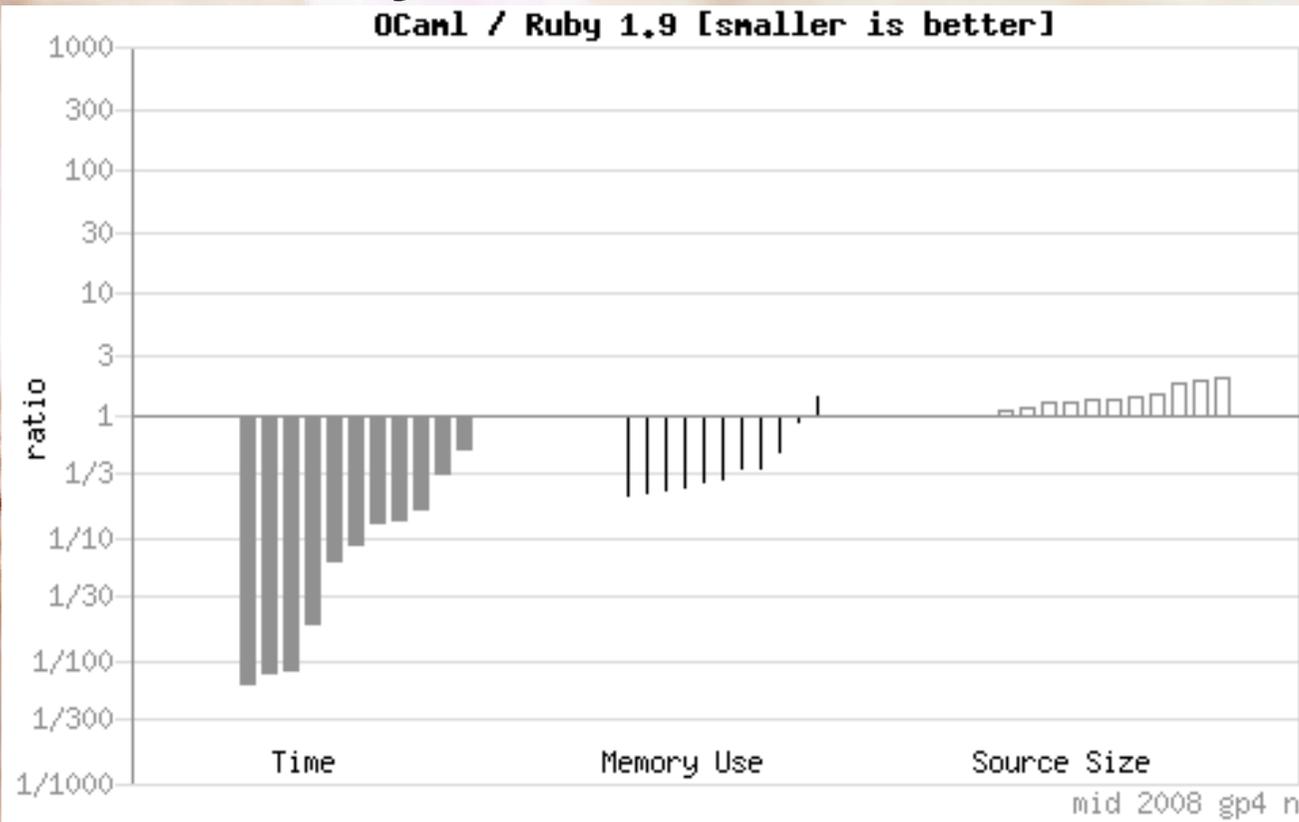
```
public static <T> T id(T x) {  
    return x;  
}
```

OCaml:

```
let id x = x
```

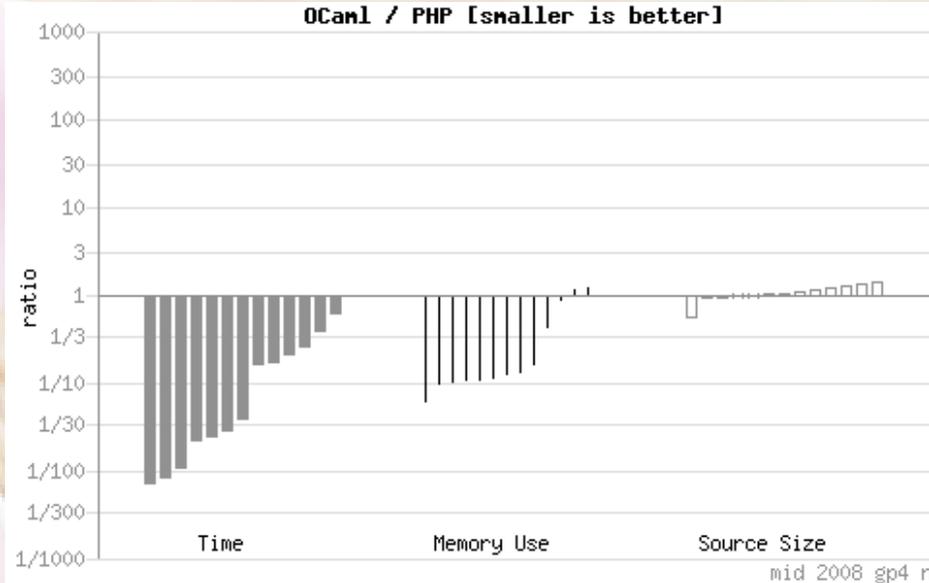
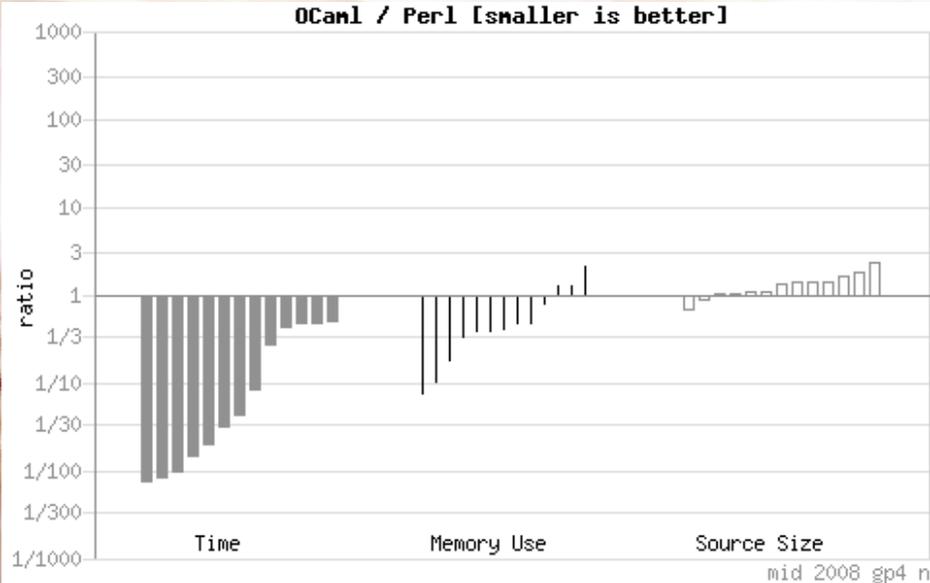
効率が良い

OCamlとRuby 1.9:



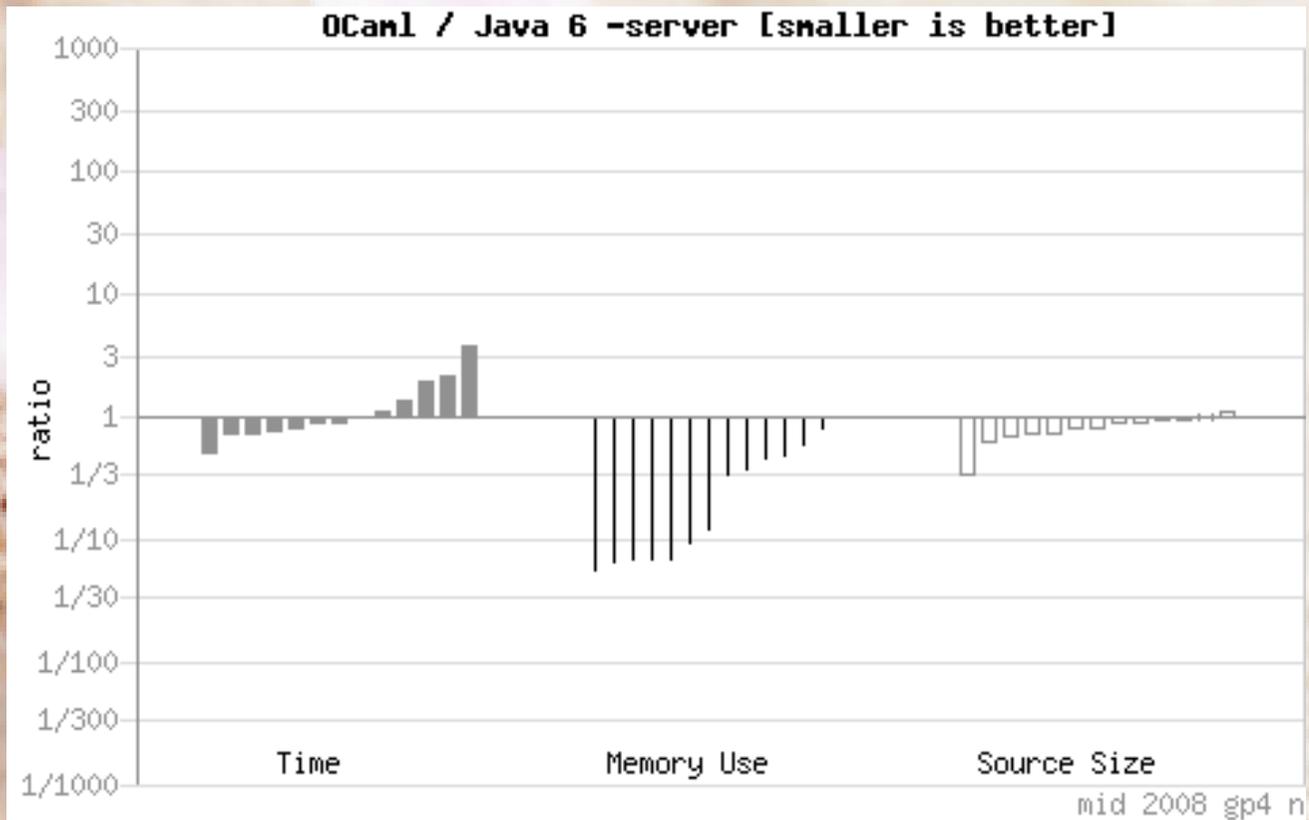
効率がよい

OCamlとPerl,PHP:



効率が良い

OCamlとJava6:



A close-up photograph of a pink and yellow flower, possibly a frangipani, resting on a sandy beach. The flower has five petals, with a yellow center and pink outer edges. The background is a soft-focus view of the sand and some distant water.

OCamlは定理証明器と仲良し

Coqと仲良し

OCamlは定理証明器と仲良し

定義した関数の性質を述べて保証する事ができる。

- 入力が0より大きいなら戻り値も0より大きい
- エンコード → デコードで必ず元に戻る
- cgiの結果は必ずHTMLの規格に準拠している
- 最大公約数



OCamlは定理証明器と仲良し

OCamlなどのソースコードに変換できる。

- 既存のOCamlのライブラリや自分のプログラムと容易に連携できる
- 逆に誰かの作ったCoqライブラリをOCamlから使える

A close-up photograph of a pink and yellow plumeria flower resting on a sandy beach. The flower has five petals, with a yellow center and pink outer edges. The background is a soft-focus view of the sand and some distant foliage.

OCamlの欠点

- ドキュメントが少ない
- 標準ライブラリが少ない

OCamlはこんなところで使われている

- オープンソースプログラム
 - ・ ファイル同期システム **unison**
 - ・ 世界最速のフーリエ変換
 - ・ **tennis game**
- 産業
 - ・ 有限会社 IT プランニング
 - ・ **Jane Street Capital**

OCamlのインストール

Debian, Ubuntuの場合:

```
# apt-get install ocaml
```

その他のLinux:

<http://caml.inria.fr/download.html>からソースコードをダウンロード

```
$ ./configure  
$ make world.opt  
# make install
```

A close-up photograph of a pink and yellow flower, possibly a frangipani, resting on a bed of light-colored sand. The flower has five petals, with a yellow center and pink outer edges. The background is a soft-focus view of the sand.

Mac OS X:

```
# ports install ocaml
```

おわりに

私たちと一緒に型推論しませんか？

ocaml-nagoya はメンバーを大募集しています。

参加希望者は `ocaml-nagoya @ freeml.com` までメールください。

URL:

- ocaml-nagoya: <http://www.itpl.co.jp/ocaml-nagoya/>
- OCaml-JP: <http://ocaml.jp/>

東都大駱駝會

會日己丑歲八月卅日
處東京大學山上會館

○此會之由來
○此會之由來

駱駝之為物，其性最馴，其力最健，其行最速，其食最廉，其用最廣，其功最偉，其德最厚，其壽最長，其性最剛，其力最柔，其行最穩，其食最潔，其用最勤，其功最著，其德最顯，其壽最久，其性最烈，其力最猛，其行最疾，其食最雜，其用最廣，其功最偉，其德最厚，其壽最長，其性最剛，其力最柔，其行最穩，其食最潔，其用最勤，其功最著，其德最顯，其壽最久。

駱駝之為物，其性最馴，其力最健，其行最速，其食最廉，其用最廣，其功最偉，其德最厚，其壽最長，其性最剛，其力最柔，其行最穩，其食最潔，其用最勤，其功最著，其德最顯，其壽最久。駱駝之為物，其性最馴，其力最健，其行最速，其食最廉，其用最廣，其功最偉，其德最厚，其壽最長，其性最剛，其力最柔，其行最穩，其食最潔，其用最勤，其功最著，其德最顯，其壽最久。

○此會之目的
○此會之目的

○此會之經過
○此會之經過



駱駝
駱駝

○此會之結果
○此會之結果

○此會之感想
○此會之感想