The Algorithm $\mathcal{M}'_{annot}$ for Coq Extraction to Statically Typed Languages without Type Inference

Yoshihiro Imai @yoshihiro503

IT Planning, Inc

ML-Nagoya 2012

Yoshihiro Imai

- Twitter: @yoshihiro503
- Home Page: http://proofcafe.org/ yoshihiro503/
- Hatena: id:yoshihiro503

Coqすごい！

Extractionすごい！

- 2007 LL **魂 @東京 一ツ橋ホール**
- 2008 LL Future **@東京 なかの** ZERO
- 2009 OSC **名古屋 @名古屋市立大学**
- 2010 Coq **庵 @名古屋市青少年文化センター**
- 2011 QCon Tokyo **@東京ファッションタウン**
- 2011 TopSE **講義 @国立情報学研究所**
- 2011 **特殊研究 1 後期講演 @京都大学**
- 2012 **特別講義 @名古屋大学**
- 2012 TopSE **講義 @国立情報学研究所**

- Coqでアルゴリズムを定義し、証明できる
- Extraction機能でアルゴリズムからコードを自動生成
- 既存のコードと混ぜて使える！

問題点

Extraction **の対象となる言語が限られている。**

現状では次の 3 **種類**:

- OCaml
- Haskell
- Scheme

問題点

実は大丈夫。

- 内部データでは小さな仮想言語 mini-ML に一旦変換している。
- つまり対象言語の構文のためのプリンタを書けば簡単に拡張できる。

対象言語を増やす拡張の例

- coq-ruby: https://github.com/mzp/coq-ruby
- coq-clojure: http://patch-tag.com/r/leque/coq-clojure-ext/home
- coq2javascript: (*開発中*) https://bitbucket.org/yoshihiro503/coq2javascript
- coq2sml: (*開発中*) https://bitbucket.org/yoshihiro503/coq2sml

問題点再び

mini-ML の特徴から、対象言語は次の機能を持たなければならない:

問題点再び

mini-ML の特徴から、対象言語は次の機能を持たなければならない:

- GC

問題点再び

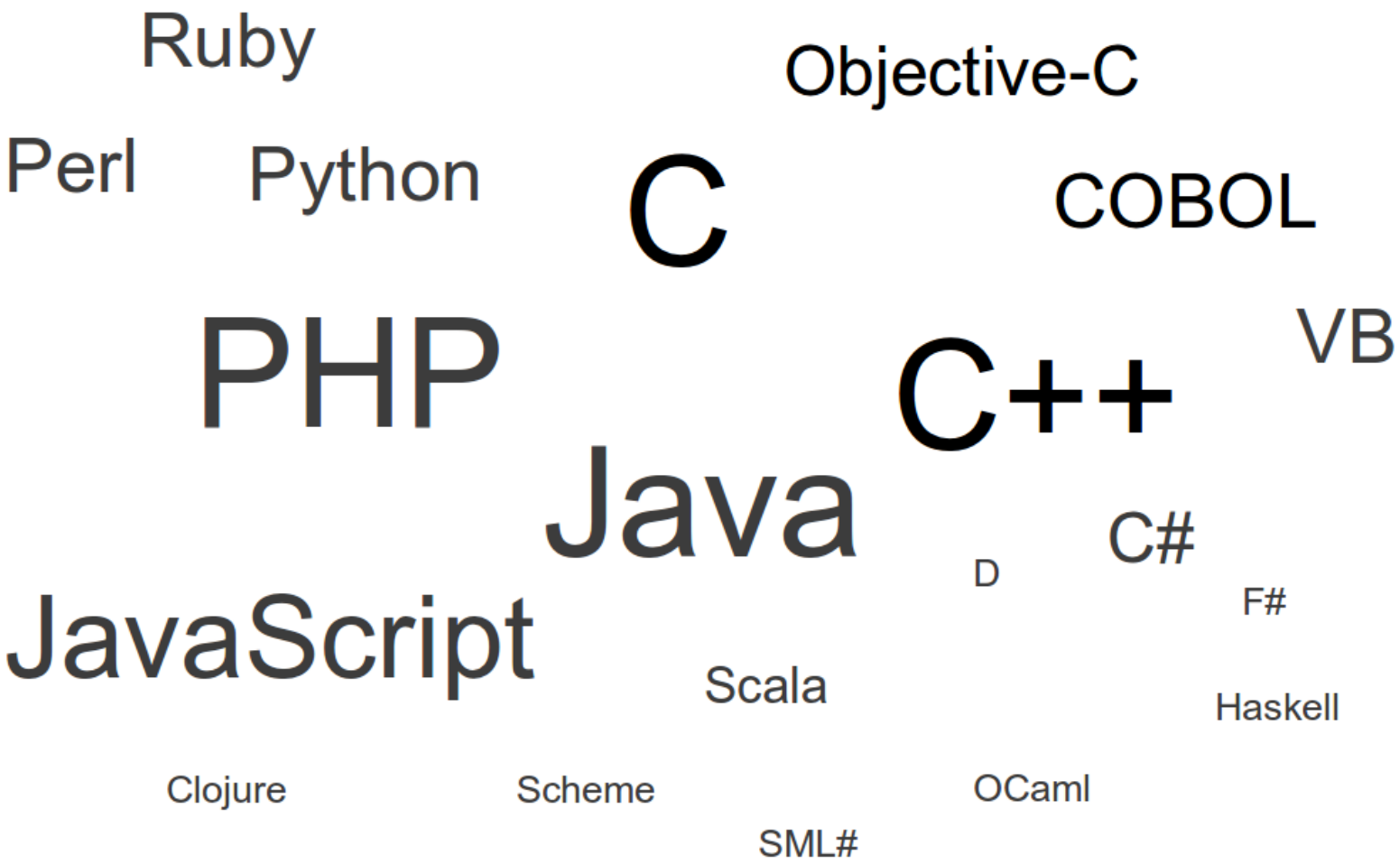
mini-ML の特徴から、対象言語は次の機能を持たなければならない:

- GC
- ラムダ式(無名関数)

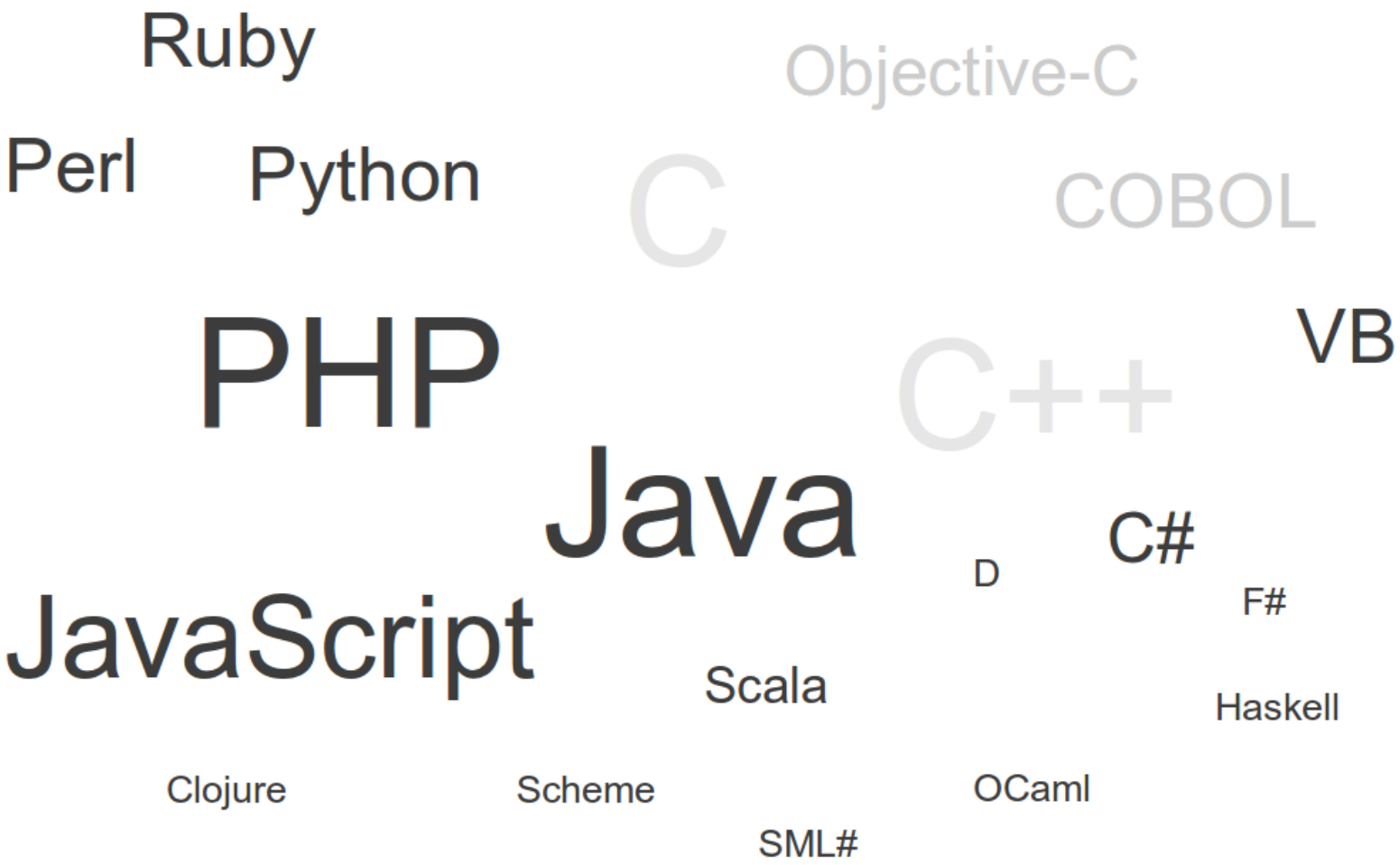
問題点再び

mini-ML の特徴から、対象言語は次の機能を持たなければならない:

- GC
- ラムダ式(無名関数)
- 型推論


Ruby
Objective-C
Perl
Python
C
COBOL
PHP
C++
VB
Java
C#
D
F#
JavaScript
Scala
Haskell
Clojure
Scheme
OCaml
SML#

GC + Lambda


Ruby
Objective-C
Perl
Python
C
COBOL
PHP
VB
C++
Java
C#
D
F#
JavaScript
Scala
Haskell
Clojure
Scheme
OCaml
SML#

GC + Lambda + Type Inference

Ruby

Perl Python

PHP

JavaScript

F#

Haskell

Clojure Scheme OCaml

SML#

結論

- GC
- **ラムダ式(無名関数)**
- **~~型~~推論**

結論

- GC
- **ラムダ式(無名関数)**
- **~~型推論~~ ← ここを消せた！**

ここから細かい話

coq-8.3pl3/plugins/extraction/extraction.ml

Coq **の式** (Gallina) **を** mini-ML **の式に変換する関数**

```
extract_term :
  env -> Mlenv.t -> ml_type ->
  constr -> constr list -> ml_ast
```

constr **が** Gallina, ml_ast **は** mini-ML **の式**,ml_type **は** mini-ML **の型式。**$\mathcal{M}'$ **と呼ばれるアルゴリズムを使っている。**

アルゴリズム $\mathcal{M}'$

$$
\begin{array}{rcl}
t & := & x \\
& | & c \\
& | & (\texttt{fun}\ x \rightarrow t) \\
& | & t\ t \\
& | & \texttt{let}\ x = t\ \texttt{in}\ t \\
& | & \texttt{magic}\ t
\end{array}
$$

アルゴリズム $\mathcal{M}'_{annot}$

$$
\begin{array}{rcl}
t & := & x[\tau, \ldots, \tau] \\
& | & c \\
& | & (\texttt{fun}\ (x\texttt{:}\ \tau) \rightarrow t) \\
& | & t\ t \\
& | & \texttt{let}\ (x\texttt{:}\ \tau) = t\ \texttt{in}\ t \\
& | & \texttt{magic}[\tau]\ t
\end{array}
$$

アルゴリズム $\mathcal{M}'$

$\mathcal{M}'(\Gamma,\ x,\ \tau) =$
  let $\sigma = \mathtt{mgu}(\tau,\ \mathtt{Inst}(\Gamma(x)))$ in
  if $\sigma = \mathtt{error}$ then $(\mathtt{id},\ \mathtt{Obj.magic}\ x)$ else $(\sigma,\ x)$

$\mathcal{M}'(\Gamma,\ c,\ \tau) =$
  let $\sigma = \mathtt{mgu}(\tau,\ \mathtt{Inst}(E(c)))$ in
  if $\sigma = \mathtt{error}$ then $(\mathtt{id},\ \mathtt{Obj.magic}\ c)$ else $(\sigma,\ c)$

$\mathcal{M}'(\Gamma,\ \mathtt{fun}\ x \to a,\ \tau) =$
  let $\sigma = \mathtt{mgu}(\tau,\ \alpha \to \beta)$ in with α and β fresh
  if $\sigma = \mathtt{error}$ then
    let $(\phi,\ \tilde{a}) = \mathcal{M}'(\Gamma + \{x : \alpha\},\ a,\ \beta)$ in
    $(\phi,\ \mathtt{Obj.magic}\ (\mathtt{fun}\ x \to \tilde{a}))$
  else
    let $(\phi,\ \tilde{a}) = \mathcal{M}'(\Gamma + \{x : \sigma(\alpha)\},\ a,\ \sigma(\beta))$ in
    $(\phi \cdot \sigma,\ \mathtt{fun}\ x \to \tilde{a})$

$\mathcal{M}'(\Gamma,\ a_1\ a_2) =$
  let $(\phi_1,\ \tilde{a_1}) = \mathcal{M}'(\Gamma,\ a_1,\ \alpha \to \tau)$ in with α fresh
  let $(\phi_2,\ \tilde{a_2}) = \mathcal{M}'(\phi_1(\Gamma),\ a_2,\ \phi_1(\alpha))$ in
  $(\phi_2 \cdot \phi_1,\ \tilde{a_1}\ \tilde{a_2})$

$\mathcal{M}'(\Gamma,\ \mathtt{let}\ x = a_1\ \mathtt{in}\ a_2) =$
  let $(\phi_1,\ \tilde{a_1}) = \mathcal{M}'(\Gamma,\ a_1,\ \alpha)$ in with α fresh
  let $(\phi_2,\ \tilde{a_2}) = \mathcal{M}'(\phi_1(\Gamma) + \{x : \mathtt{Gen}(\phi_1(\alpha),\ \phi_1(\Gamma))\},\ a_2,\ \phi_1(\tau))$ in
  $(\phi_2 \cdot \phi_1,\ \mathtt{let}\ x = \tilde{a_1}\ \mathtt{in}\ \tilde{a_2})$

FIG. 3.2: Definition of $\mathcal{M}$'

[1] P. Letouzey, Certified functional programming Program

アルゴリズム $\mathcal{M}'_{annot}$

$\mathcal{M}'(\Gamma,\ x,\ \tau) =$
  let $\sigma = \mathtt{mgu}(\tau,\ \mathtt{Inst}(\Gamma(x)))$ in
  if $\sigma = \mathtt{error}$ then $(\mathtt{id},\ \mathtt{Obj.magic}\ x)$ else $(\sigma,\ x)$

$\mathcal{M}'(\Gamma,\ c,\ \tau) =$
  let $\sigma = \mathtt{mgu}(\tau,\ \mathtt{Inst}(E(c)))$ in
  if $\sigma = \mathtt{error}$ then $(\mathtt{id},\ \mathtt{Obj.magic}\ c)$ else $(\sigma,\ c)$

$\mathcal{M}'(\Gamma,\ \mathtt{fun}\ x \to a,\ \tau) =$
  let $\sigma = \mathtt{mgu}(\tau,\ \alpha \to \beta)$ in with α and β fresh
  if $\sigma = \mathtt{error}$ then
    let $(\phi,\ \tilde{a}) = \mathcal{M}'(\Gamma + \{x : \alpha\},\ a,\ \beta)$ in
    $(\phi,\ \mathtt{Obj.magic}\ (\mathtt{fun}\ \_ \to \tilde{a}))$
  else
    let $(\phi,\ \tilde{a}) = \mathcal{M}'(\Gamma + \{x : \sigma(\alpha)\},\ a,\ \sigma(\beta))$ in
    $(\phi \cdot \sigma,\ \mathtt{fun}\ \_ \to \tilde{a})$

$\mathcal{M}'(\Gamma,\ a_1\ a_2) =$
  let $(\phi_1,\ \tilde{a_1}) = \mathcal{M}'(\Gamma,\ a_1,\ \alpha \to \tau)$ in with α fresh
  let $(\phi_2,\ \tilde{a_2}) = \mathcal{M}'(\phi_1(\Gamma),\ a_2,\ \phi_1(\alpha))$ in
  $(\phi_2 \cdot \phi_1,\ \tilde{a_1}\ \tilde{a_2})$

$\mathcal{M}'(\Gamma,\ \mathtt{let}\ x = a_1\ \mathtt{in}\ a_2) =$
  let $(\phi_1,\ \tilde{a_1}) = \mathcal{M}'(\Gamma,\ a_1,\ \alpha)$ in with α fresh
  let $(\phi_2,\ \tilde{a_2}) = \mathcal{M}'(\phi_1(\Gamma) + \{x : \mathtt{Gen}(\phi_1(\alpha),\ \phi_1(\Gamma))\},\ a_2,\ \phi_1(\tau))$ in
  $(\phi_2 \cdot \phi_1,\ \mathtt{let}\ \_ = \tilde{a_1}\ \mathtt{in}\ \tilde{a_2})$

FIG. 3.2: Definition of $\mathcal{M}$'

結論

- GC
- **ラムダ式(無名関数)**
- **~~型~~推論**

結論

- GC
- **ラムダ式(無名関数)**
- **~~型推論~~ ← ここを消せた！**

系

Coq2Scala: Scala への Extraction

- http://proofcafe.org/wiki/Coq2Scala
- **応用例**: **証明された定理証明器の** Scala **実装** https://bitbucket.org/yoshihiro503/coqincoq_scala/
- **応用例**: [2] **姜 帆**, **田辺良則**, **本位田真一**, Coq **を使用した** MapReduce **アプリケーションの検証と** Scala **コードの抽出**, PPL 2012

Coqを学ぼう

- **ソフトウェアの基礎(日本語)**:
  http://proofcafe.org/sf/
- ProofCafe **毎月第4土曜日**:
  http://proofcafe.org/wiki/