プログラム言語論

亀山幸義

筑波大学 情報科学類

No. 7: 型システムその 2

目次

先週のまとめ

先週のまとめ

	C/C++	Lisp	ML,Haskell	Java	Ruby,JavaScript
静的/動的	静的	動的	静的	静的	動的
検査/推論	型検査	—	型推論	型検査	—

- 静的型付けと動的型付け
  - 静的: 実行前に検査/推論
  - 動的: 実行時に検査/推論
- 型検査と型推論
  - 型検査: 変数などの型は宣言済み⇒型の整合性を検査
  - 型推論: 変数などの型は未知⇒型を推論しつつ型の整合性を検査

ML 言語の多相型 (polymorphic type)

例題 map 関数

```
let inc x = x + 1;;
map inc [1; 2; 3];;
=> [2; 3; 4]
```

(map の型は、(int -> int) -> (int list -> int list))

```
let add1 x = x ^ "1";;
map add1 ["kameyama"; "yukiyoshi"];;
=> ["kameyama1"; "yukiyoshi1"]
```

(map の型は、
(string -> string) -> (string list -> string list))

map 関数は多相型を持つ

$(\alpha \rightarrow \beta) \rightarrow (\alpha\ \texttt{list} \rightarrow \beta\ \texttt{list})$

ML 言語の多相型 (polymorphic type)

ML では let で多相型を持つ式を導入

```
let id x = x ;;
==> id : α → α
let swap (x,y) = (y,x) ;;
==> swap : α * β → β * α
```

多相型をもつ再帰関数

```
let rec append l1 l2 =
  match l1 with
  | [ ] → l2
  | h::t → h :: (append t l2)
==> append : α list → α list → α list

let rec iterate n f x =
  if n = 0 then x
  else iterate (n−1) f (f x)
==> iterate : int → (α * α) → α → α
```

多相型の利点

もし、map 関数を (多相型がない)C 言語で書くとしたら‥

- 方法 1. int 型に対する map, string 型に対する map などを別々に定義。
- 方法 2. 「void 型に対する map」を定義して、使うときに各型に cast (型変換)
- 方法 3. C++ の template を使う。

方法 1 は、コード量が多くなる、同じコードを何度も書くため保守性が悪い.
方法 2 は、型に関する間違いのチェックができない.
方法 3 は、型エラーが起きたときに原因を発見しにくい.

多相型

型推論アルゴリズム

ML 言語 (多相型を含む) に対する 型推論問題を解くアルゴリズム (あるいは計算機プログラム) が存在する。

3 種類の多相型

- パラメータ多相: この資料での多相
- サブタイプ多相: オブジェクト指向言語
- アドホック多相: Haskell の type class

まとめ

型が整合していたからといって、プログラムが「正しい」わけではない。しかし、型が整合しているかどうかの検査をやるだけでも、(人間がよくやる) 多くの間違いを早期に発見できることが多い。[経験的事実]

- 型システム
- 静的型付け vs 動的型付け
- 多相型