

ML 演習 第 7 回

おおいわ
May 22, 2001

今回の内容

- MiniML 第 3 回: 型推論
 - ML の型規則
 - 型推論の例
 - Unification
 - Parametric polymorphism

2

MiniML その3

- 静的型付きの ML-like 言語
 - 型推論の理論と実装
- ML 処理系の内部ではどうなっているのか?

3

MiniML の型

- 今回扱う型

$\tau ::=$	int	整数
	bool	論理値
	$\tau * \tau$	ペア
	τ list	リスト
	$\tau \rightarrow \tau$	関数 (定義域 \rightarrow 値域)

4

型付けの例 (1)

- `fun x -> if x = [] then true else hd x`
 $(hd : \tau_{hd} := \tau \text{ list} \rightarrow \tau)$
 - $\{hd : \tau_{hd}\}$ の元で `fun ...` の型を考える。
 - 関数なので $(fun x \rightarrow \dots) : \alpha \rightarrow \beta$ と置く。
 - 引数の型と見比べると $x : \alpha$ 。
 - $\{hd : \tau_{hd}, x : \alpha\}$ の元で $(if \dots) : \beta$ 。
 - $\{hd : \tau_{hd}, x : \alpha\}$ の元で `if ...` の型を調べる。

5

型付けの例 (2)

- `fun x -> if x = [] then true else hd x`
 $(hd : \tau_{hd} := \tau \text{ list} \rightarrow \tau)$
 - $\{hd : \tau_{hd}, x : \alpha\}$ の元で `if ...` の型を調べる。
 - `if` の条件節 `x = []` より $\alpha = \tau \text{ list}$ 。
 - `then` 節 `true` より $(if \dots) : \text{bool}$ 。
 - $hd : \tau \text{ list} \rightarrow \tau$ と $x : \tau \text{ list}$ より
 $\tau \text{ list} = \tau \text{ list}, (hd x) : \tau$ 。
 - $(hd x) : \tau$ と $(if \dots) : \text{bool}$ より $\tau = \text{bool}$ 。

6

型付けの例 (3)

- $\text{fun } x \rightarrow \text{if } x = [] \text{ then true else hd } x$
($\text{hd} : \tau_{\text{hd}} := \tau \text{ list} \rightarrow \tau$)
 - $(\text{fun } x \rightarrow \dots) : \alpha \rightarrow \beta$ 。
 - $(\text{if } \dots) : \text{bool}$ より $\beta = \text{bool}$ 。
 - $\alpha = \gamma \text{ list}$, $\gamma = \text{bool}$ 。
- よって $(\text{fun } x \rightarrow \dots) : \text{bool list} \rightarrow \text{bool}$ 。

7

型推論の実装方針

- 実際の処理: unification
 - 構文の各要素について、部分式と式全体の型に関する条件を match させていく。
 - 矛盾による unification 失敗 \rightarrow ill-typed

8

Unification

- 2つのパターンを一致させる代入を探す
 - 例1: $X, \text{int} \Rightarrow \{X = \text{int}\}$
 - 例2: $\text{bool} * X, Y * \text{int} \Rightarrow \{X = \text{int}, Y = \text{bool}\}$
 - 例3: $A \rightarrow B, \text{bool} \rightarrow C \Rightarrow \{A = \text{bool}, B = C\}$
 - 例3では $\{A = B = C = \text{bool}\}$ なども条件を満たす: 上のようにもっとも一般的なものを Most General Unifier (mgu) という
 - 例4: $A \rightarrow B, \text{bool} \Rightarrow$ 失敗

9

Unification による型推論

- 例: $\text{fun } f \rightarrow \text{fun } x \rightarrow f x + f 1$
 - 部分式 e の型を $\tau(e)$ と書くと
 - $\tau(\text{fun } f \dots) = \alpha \rightarrow \beta$
 - $\tau(\text{fun } x \dots) = \gamma \rightarrow \delta = \beta$ [fun f... の返値]
 - $\tau(f x + f 1) = \text{int} = \delta$ [fun x... の返値]
 - $\tau(f x) = \text{int}, \tau(f 1) = \text{int}$
 - $\tau(f) = \alpha = \gamma \rightarrow \text{int} = \text{int} \rightarrow \text{int}$
 - 結論: $\alpha = \beta = (\text{int} \rightarrow \text{int}), \delta = \gamma = \text{int}$
 $\tau(\text{fun } f \dots) = (\text{int} \rightarrow \text{int}) \rightarrow \text{int} \rightarrow \text{int}$

10

型環境

- 自由変数の型に関する情報を保持
 - let 文や関数適用で出現
 - 値環境と対応
- 例: $\text{let } x = 5 \text{ in } x + 3$
 - $x + 3$ における型環境: $\{x : \text{int}\}$

11

型判定

- 各部分式に関する条件
 - 型判定 $\Gamma \vdash e : t$
 - 型環境 Γ の元で式 e は型 t に型付け可能
 - 具体的なルールはプリント参照
- 実装
 - miniMLTyping.ml の type_expr

12

型判定の実装 (1)

- 型変数の表現: type mtypes
 - TVar: 型変数
 - フィールド v は変更可能
 - TVar { id = n; v = TUnknown } : 未定型変数
 - TVar { id = _; v = (他の型) } : v の型と同じ型

13

型判定の実装 (2)

- unification の実装
 - 今回は破壊的代入に基づく unification
 - TVar { id = n; v = TUnknown } とその他の値を unification する時に、v のフィールドを直接もう1つの型で書き換える
 - この TVar が別の TVar から参照されていれば、自然に参照元の示す型も置換 → unification の結果の伝播
 - 実装: unify, (shorten: TVar 連鎖の短縮)

14

polymorphic type (1)

- 多相型の処理
 - 多相型の発生: unification の結果値の決まらない項が残ることがある
 - (例: fun x -> x からは例えば
 - TArrow (
 - TVar { id = 0; v = TUnknown },
 - TVar { id = 1;
 - v = TVar { id = 0; v = TUnknown } })
 - といった型が出る ['a → 'a に相当]

15

polymorphic type (2)

- 多相型の処理 (続)
 - 多相型の利用:
 - ML の多相型は限定的: let で束縛した値は複数回の利用で別の型として使える
 - ex. let f = (fun x -> x) in (f 5, f true) (OK)
(fun f -> (f 5, f true)) (fun x -> x) (NG)

16

polymorphic type (3)

- 多相型の処理: 実装方針
 - let 束縛を処理するときに、多相的な型変数を記録しておく
 - polymorphic に使える型変数 = (形に含まれる未束縛の形変数) - (形環境に含まれる形変数)
 - 形環境から形を取り出すときに、記録された一般化可能形変数を新しい未束縛の形に置換する

17

polymorphic type (4)

- 例: hd の「型」: 'a list → 'a
 - これは使用時に 'a をどのような型に置き換えてもいいことを意味している
 - $\forall \alpha. \alpha \text{ list} \rightarrow \alpha$ と表現 型スキーマと呼ぶ
 - 実装での表現: schema 型
 - \forall 節の中の型変数の id のリスト * mtypes
 - mtypes → 型スキーマ : generalize
 - 型スキーマ → 個別化した型 : instantiate
 - 型環境: (識別子 * 型スキーマ) のリスト

18

課題1

- = (Equal) と :: (ConsExp) に対する型チェック処理を実装せよ。
 - Equal の条件: (左辺型) = (右辺型)
結果の型 = bool
 - Cons: (結果型) = (右辺型) = list (左辺型)

19

課題2

1. 関数適用の型チェックを実装せよ。
 - (e1 e2) で、結果とe1とe2の型の関係は？
2. let rec 式の型チェックを実装せよ。
 - 少し複雑。先に束縛の式を型検査してから、in 節を型検査する。しかし、let rec f = e1 in e2 で、f を多相型として使うことができるのは e2 の中だけである事に注意。

20

課題3 (optional)

- match 式の型チェックを実装せよ。
 - match e with p1 -> e1 | p2 -> e2 の形の式で、何と何がマッチすればいいのかを考える。
 - pattern_type を補助に使ってもよい。
- function 式の型チェックを実装せよ。
 - match ができればあと1歩。

21

課題4 (optional)

- 一般の let (rec) 式の型付けを実装せよ。
 - 基本は1引数・パターン無しの場合と同じ。
 - match 文とは趣が違うので注意。
 - 手間がかかるので let だけでもいいです。

22

課題5 (おまけ)

- 第5回の eval 関数の実装と今回の型推論の実装を組み合わせ、型付き MiniML のインタプリタを作成してみよ。
 - 入出力関数などは適当に調べてください。
 - 型の表示には print_mtypes が使えます。

23

提出方法

- 締め切: 2001年6月5日 (月) 24:00
- 提出先: ml-report@yl.is.s.u-tokyo.ac.jp
- 題名: Report 7 学生証番号

24

次回からの予定

- 次回から7回は Prolog の演習です。
- 最後 (7/10?) に最終課題を出します。
 - 1問選択: 現在の予定:
 - Prolog インタプリタの実装 [default]
 - ocaml を用いた実用アプリの実装
 - Mini-ML インタプリタの更なる拡張
 - 新たな言語の具体的な設計と実装 など...

25