ML 演習 第 7 回

おおいわ

May 27, 2003

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

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いうことが行われているのか?

MiniML の型

- 今回扱う型

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

型付けの例 (1)

$$\underbrace{\texttt{fun}\ \underbrace{\texttt{x}}_{\alpha}\ \texttt{->}\ \underbrace{\texttt{if x = [] then true else hd x}}_{\beta}}_{\alpha \to \beta}$$

- fun x -> if x = [] then true else hd x

(hd : $\tau_{hd} := \tau$ list $\to \tau$)

- { hd: τ_{hd} } の元で fun ... の型を考える。
- 関数なので (fun x -> ...): $\alpha \to \beta$ と置く。
- 引数の型と見比べると x : α。
- { hd: τ_{hd}, x : α } の元で (if ...) : β。
- { hd: τ_{hd}, x : α } の元で if ... の型を調べる。

型付けの例 (2)

- $\text{fun } \underbrace{x}_{\alpha} \text{ -> } \underbrace{\text{if } \overset{\alpha}{x} = \overset{\gamma \text{ list}}{[]} \text{ then true else hd x}}_{\beta}$

(hd : τ_{hd} := τ list → τ)

- { hd: τ_{hd} , x : α } の元で if ... の型を調べる。
- if の条件節 x = [] より $\alpha = \gamma$ list。

型付けの例 (2)

- $\text{fun } \underset{\gamma\ \text{list}}{x} \text{ -> } \underbrace{\text{if } \overset{\gamma\ \text{list}}{x} = \overset{\gamma\ \text{list}}{[]} \text{ then } \overset{\text{bool}}{\text{true}} \text{ else hd } x}_{\beta}$

(hd : τ_{hd} := τ list → τ)

- { hd: τ_{hd} , x : α } の元で if ... の型を調べる。
- if の条件節 x = [] より $\alpha = \gamma$ list。
- then 節 true より (if ...) : bool, hd x = bool。

型付けの例 (2)

bool

γ list γ list bool γ list

- fun x -> if x = [] then true else hd x

γ list

bool

(hd : τ_{hd} := τ list → τ)

- { hd: τ_{hd}, x : α } の元で if ... の型を調べる。
- if の条件節 x = [] より α = γ list。
- then 節 true より (if ...) : bool。
- hd: τ list → τ と x : γ list, hd x: bool より
  τ list = γ list, τ = bool。故に γ = bool。

型付けの例 (2)

bool

bool list bool list bool bool list

- fun x -> if x = [] then true else hd x

bool list

bool

(hd : τ_{hd} := τ list → τ)

- { hd: τ_{hd} , x : α } の元で if ... の型を調べる。
- if の条件節 x = [] より $\alpha = \gamma$ list。
- then 節 true より (if ...) : bool。
- hd: τ list → τ と x : γ list, hd x: bool より τ list = γ list, τ = bool。故に γ = bool。

型付けの例 (3)

bool list → bool

- $\underbrace{\text{fun x -> } \underbrace{\text{if x = [] then true else hd x}}_{\text{bool}}}_{}$

fun x -> if x = [] then true else hd x

bool list

bool

(hd : τ_{hd} := τ list → τ)

- (fun x -> ...): $\alpha \rightarrow \beta$。
- (if ...) : bool より β = bool。
- x = γ list = τ list = bool。

- よって (fun x -> ...): bool list → bool 。

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 → ill-typed

Unification

- 2つのパターンを一致させる代入を探す
  - 例1: X, int ⇒ { X = int }
  - 例2: bool * X, Y * int ⇒ { X = int, Y = bool }
  - 例3: A → B, bool → C ⇒ { A = bool, B = C }
    - 例3では { A = B = C = 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 → B, bool ⇒ 失敗

Unification による型推論

- 例: fun f → fun x → f x + f 1
  - 部分式 e の型を τ(e) と書くと
    - τ(fun f…) = $\alpha \to \beta$
    - τ(fun x…) = $\gamma \to \delta = \beta$ [fun f… の返値]
    - τ(f x + f 1) = int = δ [fun x… の返値]
    - τ(f x) = int, τ (f 1) = int
    - τ(f) = $\alpha = \gamma \to$ int = int → int

    - 結論: $\alpha = \beta$ = (int → int), $\delta = \gamma$ = int
      τ(fun f…) = (int → int) → int → int

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: let x = 5 in <u>x + 3</u>
    - <u>x + 3</u> における型環境: { x : int }
    - <u>x + 3</u> における値環境: { x = 5 }

型判定

- 各部分式に関する条件
  - 型判定 Γ ├ e : τ
    - 型環境 Γ の元で式 e は型 τ 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

型判定の実装 (1)

- 型変数の表現: type mltypes
  - 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

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)

型判定の実装 (3)

- 実際の型判定
  - 実装: type_of_expr
    - 部分式の型を制約と unify して、全体の型を返す
    - 例1: Plus, PairExp
    - 例2: IfExp に対する実装
      - new_type_variable () の使い方
    - 例3: LambdaExp
      - 型環境の拡張 (generalize は後述)

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]

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 (rec) で束縛した値は in 以下の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)

✗ (fun f -> (f 5, f true)) (fun x -> x)

✗ let rec f x = f [x] in f 0

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 = (型に含まれる未束縛の型変数) − (型環境に含まれる型変数)
  - 型環境から型を取り出すときに、記録された一般化可能型変数を新しい未束縛の型に置換する

Polymorphic type (4)

- 例: hd の「型」: α list → α
  - 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
  - ∀α. α list → α と表現 型スキーマと呼ぶ
  - 実装での表現: schema 型
    - (∀節の中の型変数の id のリスト) * mltypes
    - mltypes → 型スキーマ : generalize
    - 型スキーマ→個別化した型 : instanciate (see Var)
    - 型環境: (識別子 * 型スキーマ) のリスト

Polymorphic type (5)

- 実際の型推論の例

  $\forall \alpha . \alpha \to \alpha$

  - let id = (fun x -> x) in (id 5, id true)

    $\beta \to \beta$ $\quad\quad$ $\gamma \to \gamma$

    - 2つの id の出現が別の型変数に展開される

Polymorphic type (5)

- 実際の型推論の例

  int * bool

  $\forall\alpha.\alpha\rightarrow\alpha$ int bool

  - let id = (fun x -> x) in (id 5, id true)

  int→int bool→bool

  int * bool

    - id を多相的に使えている

課題1

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) ＝ (左辺型) list
2. 関数適用の型チェックを実装せよ。
   - (e_1 e_2) で、結果とe_1とe_2の型の関係は？

課題2

- let rec 式 “let rec f = 〈body〉 in 〈exp〉” の型チェックを実装せよ。
  1. 最初に 関数全体の型を α と置く。
  2. [f : α] を環境に付け加えて body を型検査。
     - この段階では一般化しない。(Forall([], alpha))
  3. α を一般化して α′ にして、
[f : α′] を付け加えて exp を型検査。

課題3 (optional)

- match 式の型チェックを実装せよ。
  - match e_0 with p_1 -> e_1 | p_2 -> e_2 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。

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。

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て、
  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_mltypes が使えます。
  - 第6回 おまけ課題との組み合わせも
    面白いかもしれません。

提出方法

- 〆切: 2003年8月31日 (日) 24:00
- 提出先: ml-report@yl.is.s.u-tokyo.ac.jp
- 題名: Report 7 *学生証番号*

次回からの予定

- 次回からは Prolog の演習です。
  - 担当TAが代わります。