5Y-5

LISP/MCコンパイラの型推論

松野年宏（ファコム・ハイタック㈱）　井田哲雄（筑波大・電子情報工学系）

1. はじめに

MCは大規模科学技術計算のためのマルチパラダイム計算環境であり，現在，大型計算機FACOM M-780/20上で稼働している．LISP/MCはその計算環境の一部であり，COMMON LISP の仕様に準拠している．大規模科学技術計算には計算の高速化が特に要求されるが，LISP/MC では高速化を

(1)LISPに適したシステム全体の設計

(2)最適化コンパイラの構築

によって実現している．(1)については〔3〕で報告しており，今回は，特に現在作成中の型推論を応用した最適化コンパイラについて報告する．

従来のLISPコンパイラの最適化法には，

(1)関数のインライン展開

(2)再帰呼び出しの繰り返し型への変換

(3) (LISPに限らないが) peephole optimization

が採用されているが，我々の最適化コンパイラは上の方法に加え，型推論による

(a)型チェックコードの除去

(b)型情報を活用した再帰呼び出しの繰り返し型への変換

を行う．

2. LISPにおける型推論

型を指定していないプログラムに型を与えることを型推論という．型を推論することによりプログラムの領域(domain)のより詳細な構造が得られるならば，

(1)プログラムの（部分的な）正しさを検証すること

(2)型の情報を用いて効率の良いコードをコンパイラが生成すること

に役立たせることができる．

型推論の手法は上記２つの観点から有効であるが，LISPは基本的には型の指定を必要としない(free type) 言語であり，その言語仕様が型推論を前提として作られていないので，完全な型推論が困難である．

しかし，型推論が完全に行えないとしても，LISPのプログラムの意味が型推論が失敗することによって失われるものではないこと，上記(1), (2)の点に関して有益な情報が得られることから，型推論のアルゴリズムをLISPの処理系にインプリメントすることは有効であると考えられる．

Type inference of LISP/MC compiler

Toshihiro MATSUNO * , Tetsuo IDA **

* Facom Hitac Ltd., ** University of Tsukuba

3. 型推論の方法

我々の型推論の方法は次の考え方に基づいている．

(1)型の集合Tを集合の包含関係で順序付けられた半順序集合と考える．Tは型に関する演算∧，∨を各々最小上界，最大下界とすることによって，束〔T, ∧, ∨〕をなす．

(2)Milnerの型推論規則を部分型に関する型推論規則に加えることによって拡張する．ここで，型$\tau \in T$に対し，$\tau' \subseteq \tau$となる型$\tau' \in T$をτの部分型という．

(3)(2)で得られる型推論規則を用いて型推論を行い，互いに半順序関係にない可能な型をすべて列挙する．

3. 1 用語の定義

$T_0 = \{\tau_1, \tau_2, \ldots, \tau_n\}$ を基本型$\tau_1, \tau_2, \ldots, \tau_n$ からなる型の集合とする．〔$T_0, \subseteq$〕は集合の包含関係$\subseteq$で順序付けられた半順序集合である．本稿では，

T_0 = {T, LIST, CONS, NULL, SYMBOL, FUNCTION, NUMBER, FLOAT, INTEGER, FIXNUM, NIL}

に限定して考える．〔$T_0, \subseteq$〕は図１．のように型の順序付けを行うことによって，束〔$T_0, \wedge, \vee$〕になる．なお，この定義はCOMMON LISP の仕様を満たしている．

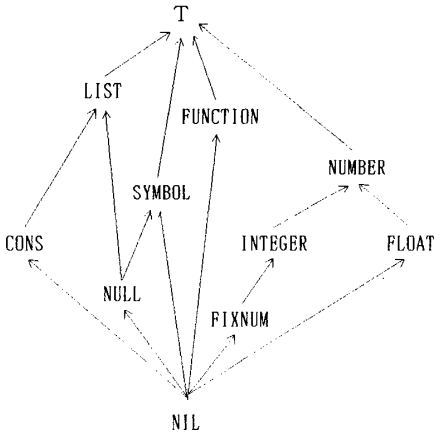

図 1. 型により構成される束

さらに，T_0 上の演算として，×（直積を作る）と→（関数を作る）を導入する．

Tを次のように定義する．

$\sigma \in T_0 \Rightarrow \sigma \in T$,

$\sigma, \tau \in T \Rightarrow \sigma \rightarrow \tau \in T$,

$\sigma, \tau \in T \Rightarrow \sigma \times \tau \in T$

また，$\sigma \subseteq \sigma'$ かつ $\tau \subseteq \tau' \Leftrightarrow \sigma \times \tau \subseteq \sigma' \times \tau'$，

$\sigma \to \tau \subseteq \sigma' \to \tau' \Leftrightarrow \sigma' \subseteq \sigma$ かつ $\tau \subseteq \tau'$

とする．このとき，〔T, ⊆〕は半順序集合，〔T, ∧, ∨〕は束となる．

図1．の型FUNCTIONは→によって作られる型全体を表している．

3．2 型推論規則

型の環境 $A = \{x_1:\tau_1, x_2:\tau_2, \ldots, x_n:\tau_n\}$ のもとで，表現eの型τを推論することを

$A \vdash e:\tau$

と書く．また，$A_{x_1x_2\ldots x_n}$ は

$A - \{x_1:\sigma_1, x_2:\sigma_2, \ldots, x_n:\sigma_n\}$

を示す．

〔I〕 eがアトムの場合

$A_e \cup \{e:\sigma\} \vdash e:\sigma$

〔II〕 関数の型付け

$$\frac{A_{x_1\ldots x_n} \cup \{x_1:\tau_1, x_2:\tau_2, \ldots, x_n:\tau_n\} \vdash e:\sigma}{A \vdash (\text{lambda } (x_1\ x_2\ \ldots x_n)\ e):\tau_1 \times \tau_2 \times \ldots \times \tau_n \to \tau}$$

〔III〕 条件表現の型付け

$$\frac{A \vdash e_0:T,\ A \vdash e_1:\tau_1,\ A \vdash e_2:\tau_2}{A \vdash (\text{if } e_0\ e_1\ e_2):\tau_1 \vee \tau_2}$$

〔IV〕 関数呼び出し

$$\frac{A \vdash e_0:\sigma_1 \times \ldots \times \sigma_n \to \tau,\ A \vdash e_1:\sigma_1,\ \ldots,\ A \vdash e_n:\sigma_n}{A \vdash (e_0\ e_1\ \ldots e_n):\tau}$$

〔V〕 LET

$$\frac{A \vdash e_1:\sigma_1,\ \ldots,\ A \vdash e_n:\sigma_n,\quad A_{x_1\ldots x_n} \cup \{x_1:\sigma_1,\ \ldots, x_n:\sigma_n\} \vdash e_0:\sigma}{A \vdash (\text{let } ((x_1\ e_1)\ \ldots (x_n\ e_n))\ e_0):\sigma}$$

〔VI〕 総称化

$$\frac{A \vdash e:\tau}{A \vdash e:\forall\alpha.\tau}$$

〔VII〕 具体化

$$\frac{A \vdash e:\forall\alpha.\tau}{A \vdash e:\tau\ [\alpha:=\sigma]}$$

〔VIII〕 一般化

$$\frac{A \vdash e:\tau}{A \vdash e:\tau',\ \tau \subseteq \tau'\ (\in T)}$$

この他に，LISPの特殊形式ごとに推論規則が必要となるが，ここでは省略する．

4．例

```
(defun length (x)
  (if (null x) 0 (1+ (length (cdr x)))))        (*)
```

で定義される関数lengthの型を推論する．

null, cdrの型を次の公理で与える．

A ⊢ null : T → T ①

A ⊢ cdr : LIST → LIST ②

A ⊢ 1+ : INTEGER → INTEGER ③

A ⊢ 1+ : FLOAT → FLOAT ④

また，型環境として，$A_0 = \{0:\text{FIXNUM}\}$ を仮定する．

length: $\tau_1 \to \tau_2$ とする．つまり，

(lambda (x) (if (null x) 0 (1+ (length (cdr x))))): $\tau_1 \to \tau_2$

ここでは，ボトムアップに(*)の表現の構成要素から型を推論する．

$A_1 = \{0:\text{FIXNUM}, \text{length}:\tau_1 \to \tau_2, x:\tau_1\}$

〔VIII〕と①より，null:LIST→T (T→T ⊆LIST→T に注意)

〔IV〕と①より，$A_2 \vdash$ (null x):T,

$A_2 = \{0:\text{FIXNUM}, \text{length}:\text{LIST} \to \tau_2, x:\text{LIST}\}$

〔I〕より，$A_2 \vdash 0:\text{FIXNUM}$

〔IV〕と②より，$A_2 \vdash$ (cdr x):LIST

〔IV〕と③より，$A_3 \vdash$ (length (cdr x)):INTEGER,

$A_3 \vdash$ (1+ (length (cdr x))):INTEGER,

$A_3 = \{0:\text{FIXNUM}, \text{length}:\text{LIST} \to \text{INTEGER}, x:\text{LIST}\}$ を得る．

〔III〕とFIXNUM∨INTEGER=INTEGER より，

$A_3 \vdash$ (if (null x) 0 (1+ (length (cdr x)))):INTEGER

したがって，

```
(lambda (x) (if (null x) 0 (1+ (length (cdr x)))))
    : LIST→INTEGER
```

これはA_3から推論されるlength:LIST→INTEGER と一致するので，全体として型推論は成功する．

5．インプリメンテーション

推論規則の適用手順を定めることにより型推論のアルゴリズムを得るが，我々は以下の洞察に基づいて型推論のアルゴリズムを得た．

(1)3．2の推論規則はMilnerの推論規則の拡張であり，Milnerの推論規則から得られたアルゴリズムW〔1〕は，基本的にユニフィケーションに基づいていることから，我々のアルゴリズムもユニフィケーションの拡張によって得られること

(2)LISP/MC は，S表現を統一表現とし，ユニフィケーションを計算の基礎機構とするPROLOG/MC を呼び出すことができること

から，PROLOG/MC によって，PROLOGのプログラムの形で推論規則を表現すればよい．推論規則の適用を伴う非決定的要因を完全には取り除く必要はなく，同一の表現に対しても，PROLOG処理系では自動的に実現されるバックトラックを併用することにより，（公理系が正しければ）互いに矛盾の無い複数個の型を得ることができる．

参考文献

〔1〕 L. Damas and R. Milner, Principal type-schemes for functional programs, Conference Record of the ninth annual ACM symposium on principles of programming languages, 1982

〔2〕 Guy L. Steele Jr., Common Lisp: The Language, Digital Press, 1984

〔3〕 松野・井田，「マルチパラダイム環境MC上のLispコンパイラについて」，情報処理学会第36回全国大会論文集, 1987