宣言的プログラムの理論に基づく型付き論理型言語

3U－1

川口 雄一†　赤間 清††　宮本 衛市††

† 苫小牧工業高等専門学校　†† 北海道大学工学部

1 はじめに

従来から行われてきた型付きの計算は、以下のよう特徴付けられる方針で行われて来た。

- 型はデータとは区別された存在である。
- 型の計算とデータの計算は区別される。

このため、型とデータの情報が混在したような計算を自然に記述することが難しく、また、それぞれの計算に対する理論をそれぞれ用意する必要があるので相互の関係をつかみにくくなり、理論的な明解さに欠けていた。

そこで本研究では従来と異なる新しい立場から型付きの計算体系を構築することを試みる。それは以下のような全てを統合化し、単純で明解な計算体系を構築するという方針として示すことができる。

- 型とデータは対等で、区別されないものである。両者を計算対象として一つのものに統合する。
- 統合した計算対象の単一化 (unify) のみを用いて計算を行なう。計算機構も一つに統合する。

このように、計算の機構としては計算対象同士の単一化のみを用いて計算を行うが、それを色々な側面から見ることにより、型の計算とかデータの計算などの意味付けがなされることになる。計算対象としても、その構造は単一のものが用いられるが、計算中の文脈によって型としての扱いを受けたりデータとしての扱いを受けたりする。

[0]An Typed Logical Language based on The Theory of Declarative Programinng

[0]Yuuichi KAWAGUCHI†, Kiyoshi AKAMA††, Eiichi MIYAMOTO††

[0]†Tomakomai National College of Technology, ††Dept. of Engineering, Hokkaidou University

[0]†yuuichi@jo.tomakomai-ct.ac.jp

本論文では、この統合された計算の機構を言語という形で示す。なお、利用者界面に関する議論は行わない。

2 型とデータの統合

型とデータを統合化し、「計算対象」という単一の構造で表現する。さらに、述語もこの計算対象に統合し、全てを計算対象として統合する。

2.1 計算対象

まず、計算対象を構成する「定数」を定義する。定数は木構造の階層構造を持つ。木の根には ALL という名前を割り当てる。全ての定数は唯一の親を持つこととする。階層構造の例を図 1に示す。

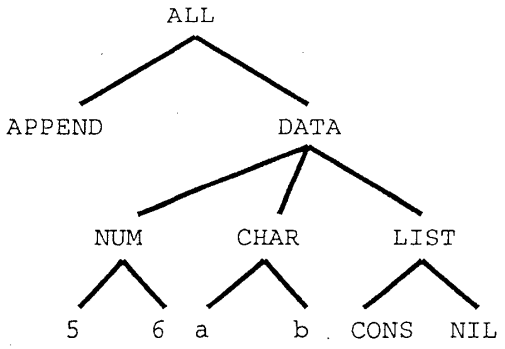

図 1: 定数の階層構造

定数の親子関係を用いて、定数の間には子孫の関係が自然に導かれる。

定数から「型」を定義する。型は以下のように定数の列として定義される。

<型> ::= （定数$^+$）

従来は「型」といえば(NUM)や(LIST)などのみを指し、(a)や(5)は「データ」、(APPEND CHAR)は「述語」などと呼び、それぞれ区別された対象であったが、本言語ではこれら全てを単一の枠組の中で扱い、全てを「型」と呼ぶ。

型を構成する定数の子孫関係から、型の半順序関係が次のように定義される。

定義 型S=($S_1 \cdots S_m$)とT=($T_1 \cdots T_n$)がS≤Tの関係にあることを以下のように定義する。

(1) $n \leq m$

(2) $1 \leq \forall i \leq n,\ S_i \leq T_i$

S≤Tの場合、SをTの子孫の型と呼ぶ。

型を用いて「計算対象」を以下のように定義する。

<計算対象> ::= [<型>]
| [<型> <計算対象>+]

計算対象の内側に含まれる計算対象を「子の計算対象」とよぶ。以下に計算対象の例とその直観的な意味を示す。

[(5)]	数値の5
[(CONS) [(a)] [(NIL)]]	carが[(a)]でcdrが[(NIL)]のcons
[(APPEND NUM) [(LIST)] [(NIL)] [(LIST)]]	引数を三つ持つappend

このように、本言語では全てを計算対象という単一の構造でとらえることに成功している。

2.2 単一化

はじめに述べたとおり、計算対象に対する計算機構は計算対象同士の単一化のみを用いる。

単一化のために「最汎子孫」(MGD)を定義する。これは二つの型の共通の子孫のうち、最も一般的な型のことである。

定義 型SとTに対して、以下の条件を満たす型UをSとTの最汎子孫と呼ぶ。

(1) U ≤ S , U ≤ T

(2) $\forall U' \in \{A \mid A \leq S, A \leq T\},\ U' \leq U$

二つの計算対象の単一化は、計算対象の型を最汎子孫に変化させることで行われる。なお型の長さが異なる場合は、長い方の型の定数を短い方の型に追加する。子の計算対象の長さに関しても同様とする。

単一化の例を以下に示す。

```
[(CONS) [(5)] [(NIL)]]
[(LIST NUM)]
          単一化
           ⇓
[(CONS NUM) [(5)] [(NIL NUM)]]
[(CONS NUM) [(5)] [(NIL NUM)]]
```

2.3 プログラムと計算の例

プログラムとして次のような問い合わせを実行する。

```
[(APPEND)
    [(LIST NUM)]
    [(LIST CHAR)]
    [(CONS) [(5)]
        [(CONS) [(6)]
            [(CONS) [(a)]
                [(CONS) [(b)]
                    [(NIL)]]]]]]
```

これは Prolog でいえば append(X, Y, [5,6,a,b]) に相当するが、変数Xには(NUM)型、YにはCHAR型が付いているため、答えとしては

X = [5, 6] , Y = [a, b]

という組合せのみが考えられる。

このように本言語では、型とデータの情報が混在したような問題も、自然に記述することができる。

3 おわりに

はじめに述べたとおり、本言語の特徴は従来から行われてきた型付きの計算体系における方針とはまるで異なっている。しかし、本論文で示したようにこのような発想の転換を行うことにより、従来よりも単純で明解な計算体系を構築することができた。

なお本言語は、宣言型言語 UL/α[1] 上に実装されている。

今後はこの計算体系を用いた型推論の計算を行うことを考えている。

参考文献

[1] 出葉 義治, 赤間 清, 宮本 衛市: 一般化論理プログラミング言語 UL/αのコンパイラ, 日本ソフトウェア科学会第9回大会論文集, A5-4, 1992.