実験用LISP言語処理系の製作

鬼頭繁治・稲垣正章*

An Experimental LISP Interpreting System

Shigeharu KITO・Masa-aki INAGAKI*

An experimental LISP interpreting system is presented based on a new interpreter construction methodology that permits the user to change or augment the implementation schema. The fundamental model of the resulting interpreter has been implemented on a microcomputer-based personal computer by using Pascal language. In the current implementation the utility library contains structure editor and pretty printer as supporting tools for LISP program development. It has been found that the LISP interpreter is easily modifiable mainly due to the use of Pascal language.

緒　言

LISPは人工知能，数式処理などの研究分野において長い間使用されているプログラミング言語で，FORTRANにつぐ歴史を有している。LISPはラムダ計算に基礎をおいたリストの生成・参照・書き換えおよび不要なリストの再生といったいわゆるリスト処理用の言語であり，ユーザー・サイドからは記号処理言語と見なされるものである。この点から最近注目されだしたProlog[1]と共に人工知能用言語と言われている。

一般にLISPによって有用なプログラムを作成するためには大きなメモリー容量を必要とし，これまでは主に大型汎用計算機上に実現された処理系が使われてきた。代表的なものにINTERLISP[2]，MACLISP[3]がある。一方，近年のLSI技術の進歩に支えられた計算機ハードウエアの低廉化によりパーソナル型のLISP専用マシンが商用に供されると共に汎用MPUに基づいたパーソナルコンピュータ上にも実現され，LISPの使用できる環境が著しく拡がり，LISPによるソフトウエアの蓄積が増大しつつある。

さらに上述した技術進歩とあいまって，手続き型アルゴリズムに基づくプログラミングでは複雑すぎて実現不可能な問題解決に人工知能研究の成果を利用する知識工学の発達が，これまで一部の研究者のツールとしての用途から実用言語へとLISPを展開させつつある。このような理由からこれまであまり努力が払われなかった実行速度の向上などの種々の必要性が現れ，処理系実現のアルゴリズムの改良といった研究が必要になる。

ここではLISP処理系の設計のための"実験装置"，すなわちその内部構造の変更が容易であるようなLISP処理系が手近に使用できることを目標として16bit MPUを使用したパーソナルコンピュータ用LISPを作製した。

1．設計の基本方針

LISP処理系実現のための"実験装置"を目標とするものであり，実用性は二義的なものとした。そのため処理系はプログラムの構造化とアルゴリズムの見通し易さを徹底し製作者以外でも容易に改造・変更ができることを第一とした。したがって記述には上記を満足する高級言語を使用し，さらにプログラム記述に冗長性を導入し，一方メモリー効率や絶対的実行効率については殆ど考慮していない。

2．処理系記述言語

高級言語によるLISPの処理系としてはこれまでにFORTRANによるもの[4,5]，BCPL[6]によるもの[7]などが報告されているがこれらは設計の主目的が移植性を高めることにあり実用的効率化も指向したものである。ここでは処理系内部の改造の容易さを目的としているのでFORTRANのような非構造化言語は不適当である。当然ながらもっとも好ましい言語はLISP自身である。"LISP in LISP"の場合はプログラムサイズも小さくなる。しかし現状ではパーソナルコンピュータ上のLISPで実用的なコンパイラを有するものは存在せず[8]，この方法は不可能である。

他に考えられる言語としてはPascal，Cがあげられるが，ここでは構造化言語でオブジェクト効率が良く特に

＊　現在，中部日本電気ソフトウエア（株）

教育用に開発されたことからもreadabilityがすぐれているPascalを採用した。

3．言語仕様

ユーザー・インターフェースとしての言語仕様は広く使用されているINTERLISP[2]のサブセット版であるLISP F3[4]を参考にして決定した。組み込み関数は67種であり，後述する使用可能メモリー上の制約から本来は処理系に組み込む方が望ましい一部のものはEXPR関数としている。なお本処理系の目的上，以下の特殊関数を組み込み関数とした。

CELL　：セルの内容を出力する。
FULLW　：フルワードの内容を出力する。
GETMATCH：属性リスト中の該当するものを出力する。
LISTF　：ファイル中に存在する全てのEXPR関数名を表示する。

TRACE関係の関数は全て組み込み関数とし，ファイル入出力はできるだけ自動化した。

ユーザー定義関数のタイプは引数の個数が一定で関数適用前に引数評価をするEXPR関数一種のみである。

4．ユーティリティ・プログラム

ユーティリティ・プログラムとしてはS式のカッコレベルに応じて清書印字するプリティプリンタ（関数名：PP）および構造エディタを有する。この構造エディタはLISP F3[4]に準ずる非常に使い勝手の良好なものとなっている。これらのプログラムは本処理系の動作を確認する目的も含めて作成されEXPR関数として用意されている。なおLISPコンパイラは実装されていない。

5．処理系の実現

本処理系で使用したコンピュータはCPUとしてIntel 8088を使用しメモリー容量256Kバイトのパーソナルコンピュータ（WAVE Corp.製 TALOS-86）である。なお外部記憶装置はフロッピーディスク装置で容量は2Mバイトのものである。

PascalコンパイラはCP/M-86上で動作するPascal/MT＋86[9]を使用した。このコンパイラと付属のリンカーの採用しているCPUのメモリーモデルの関係でコードエリアが64Kバイト，変数エリアが64Kバイト，スタック・ヒープエリアが64Kバイトの制限がある。したがって文法的な言語仕様は問題はないが書いたLISP処理系の実行段階ではやや不都合が感じられる。

前述したように本処理系はその内部構造を可変とすることを原則とするが，以下に第一段階での実現法について述べる。変数の束縛は属性リストの概念にもとずいたshallow bindingによっている。セル構造についてはCAR部，CDR部は各々1ワード（16ビット）よりなる。内部の見通しを良くする目的からメモリー消費増大の犠牲を払ってガーベッジコレクションの際のマーキング用にCAR部，CDR部とは別にchar型の変数を割りあてた。したがってセル1個あたり5バイトを要する結果となった。ガーベッジコレクション用の1バイトを節約するためには例えばCDR部の正負を使用する方法[5]などが適当である。さらに同じ理由でLISPのスタックについてもPascalコンパイラの設定するスタックは用いず，別にスタック領域を設けることとした。これらのほかにも処理系の実行効率，メモリー効率を犠牲にしてあえて記述の冗長性を追ったケースが多々ある。しかしその結果，ソースプログラムのreadabilityは良好となった。

さらに，現状では不可能であるが内部の変更によってコードエリアに余裕ができた場合には組み込み関数の追加・削除がhashテーブルを利用して半ば機械的に実施可能とするような工夫もなされている。LISPプログラムに対するTRACE関係の関数は本処理系の場合使用頻度が多くなることを考えて組み込み関数とした。

以上のような形でLISP処理系を実現した結果，本LISPインタプリタは外部から見た機能に比較するとオブジェクトコードのサイズが非常に大きくなり（本処理系よりかなり機能が大きい8ビットCPU用のLISPインタプリタでオブジェクトコードが約7Kバイトのものもある[8]），かつ許容メモリー量から予想されるセル数などと比較してLISP用データエリアが小さくなっている。

本LISP処理系の仕様をまとめて表1に示す。

表1　LISPインタプリタの仕様

トップレベル	EVAL-LISP
セル数	9330
フルワード数	321
スタックサイズ	1800
数値アトム	整数型のみ
Pascalソース	
プログラムサイズ	62Kバイト
オブジェクト	
プログラムサイズ	58Kバイト

6．LISPプログラムの実行例

図1に本処理系のユーザーインターフェースの様子を示すためにLISPプログラムの実行例を示した。例題プログラムはよく知られたn-queenゲームを解くもので

```
1$ (PP 'QUEEN)

<LAMBDA (Y)
        (QUEEN1 Y Y NIL)>

QUEEN

1$ (PP 'QUEEN1)

<LAMBDA (Y N B)
        <COND < (ZEROP N)
                 NIL >
              < <OR (MEMBER N B)
                    (QP 1 B)>
                <QUEEN1 Y
                        (SUB1 N)
                        B > >
              <T <NCONC <COND < <EQUAL (LENGTH B)
                                       (SUB1 Y)>
                                <PRINT <LIST (CONS N B)> > >
                              <T <QUEEN1 Y
                                         Y
                                         (CONS N B)> > >
                        <QUEEN1 Y
                                (SUB1 N)
                                B > > > > >

QUEEN1

1$ (PP 'QP)

<LAMBDA (K M)
        <COND < (NULL M)
                 NIL >
              < <EQUAL <ABS <DIFF N
                                  (CAR M)> >
                       K >
                T >
              <T <QP (ADD1 K)
                     (CDR M)> > > >

QP


1$ (QUEEN 4)

((2 4 1 3))
((3 1 4 2))
((2 4 1 3)(3 1 4 2))
```

図1　LISP プログラムの実行例

n＝4の場合を実行した結果である。図中で"1$"はシステムよりのプロンプトメッセージである。

7．考　察

本研究においてはLISP処理系設計のための"実験装置"としてのシステムを作成したが，これに類するものは殆ど報告されていない。基本的目的が似ているものとしてはプログラミング言語仕様およびコード生成についての実験を行うためAllenら[11]が試みた実験用コンパイラが報告されている。(Allenらのものはいわゆるコンパイラ・コンパイラとは異なる。) インタプリタにせよコンパイラにせよ言語仕様が多機能になったりその抽象化度が増加したりする場合に効率のよい言語処理系を設計するためにはここで述べたような"実験装置"を用いることは非常に便利である。

本処理系はその設計目的から考えて当然ではあるが，LISPのプログラムを作製するという立場から見ると種々の問題点を含んでいる。これは処理系の第一段階での実現に関するものが殆どで処理系の改造によって除去できるものであるが，ここではそれらの問題点のうち本研究の目的上からも問題となる点について以下に検討する。

まず第一にユーザー定義関数のタイプが一種類のみである。一般のLISP処理系ではユーザー定義関数の種類は，引数の数が一定か不定かおよび関数の適用前に引数を評価するか否かによる最低4種類のタイプが使用可能である。本処理系はEXPR関数一種のみであるのは最も不都合な点である。特に今後LISPコンパイラを実装するような場合には重大な問題である。

第二に入出力関係の機能が簡略化されている。ここではLISPによるプログラミングを実行することが主目的ではないので簡略自動化して実現したが実際種々のテストの結果機能を増す必要性が明かとなった。

第三に使用したMPUとコンパイラの問題がある。これらは取りあえず使用可能なものとして採用したが特にIntel 8088のメモリー管理方式が主原因となっている。最も不都合な点はPascal/MT＋86コンパイラがコードエリアとデータエリアを別々のセグメントに設定している[9]ことからプログラミング上種々の制限を受けることである。特に実用的なLISP処理系の記述はこのコンパイラでは不都合であると思われる。また，これはコンパイラのバージョンの問題であるがファイルのサンダムアクセス手続き部分にbugがありここではシーケンシャルアクセス法の採用を余儀なくされた。しかし結果としては記述言語仕様が殆ど標準Pascalと同じになり移植性は良好となっている。この点は現在新しいバージョンのコンパイラを使用して書き換え中である。この第三の問題点はMPUとコンパイラを変更することによって解決が可能である。現在本処理系によるテストの結果を取り入れてMC68000MPUのパーソナルコンピュータ上にPascal 2.1コンパイラ[12]を使用して知識表現が容易となる機能を含めた実用LISP処理系を製作中であり，この問題は解決される見込みである。

第四の問題として処理系の機能があげられる。"実験装置"ではあっても種々のテストを行うためには機能の拡張が必要であることがわかった。つまり任意精度整数（bignum）や実数型数値，配列といったいわゆる"データ型"およびアトムの種類の増加等は今後要求される多目的LISP処理系の実験の際には必要となる。

以上述べたような問題点はあるが，本研究の主目的とした処理系の改造を容易にするという点は記述言語にPascalを選択したことから非常に満足できるものとなった。このことは作製者以外の者によるdebug，小改造および前述したMC68000MPU使用のパーソナルコンピュータ上へのLISP処理系の実現過程によって実証されている。

結　言

本研究でパーソナルコンピュータ上へ実現した実現アルゴリズム実験用のLISP処理系はいくつかの問題点はあるものの目的通りの透明さを示している。これはその

大部分がPascalを記述言語として採用したことに帰因する。一方，実験用処理系であっても有する機能は実用処理系に準ずるものが必要であることがわかった。

謝　辞

各種LISPプログラムの作成及び処理系の改良については近藤久晴氏（現ソフトウエア・マネージメント㈱）に負うところが大である。ここに謝意を表します。

参考文献

1）Clocksin, W. F. and C. S. Mellish : Programming in Prolog, Springer, Berlin (1981)
2）Teitelman, W.: Interlisp Reference Manual. Xerox PARC, Palo Alto, CA. (1974)
3）Winston, P. H. and B. K. P. Horn : LISP, Addison-Wesley, Reading, Mass. (1981)
4）Nordstrom, M.: LISP F3 User's Guide, Uppsala Univ., Uppsala (1978)
5）太田義勝ら：情報処理学会論文誌，**23**，341 (1982)
6）Richards, M. and C. Whitby-Strevens : BCPL-the language and its compiler, Cambridge Univ, Pr., Cambridge (1980)
7）Fitch, J. P. and A. C. Norman : Software-Practice and Experience, **7**, 713 (1977)
8）松永均：情報処理学会第26回全国大会請演論文集，**1**，19 (1983)
9）Pascal/MT+86 Language Reference Manual, Digital Research, Pacific Grove, CA. (1982)
10）Allen, J.: Anatomy of LISP, McGraw-Hill, New York (1978)
11）Allen, F. E. et al.: IBM J. Res. Develop., **24**, 695 (1980)
12）Pascal 2.1 User's Manual for HP Series 200 computer, Hewllet-Packard, Fort Collins, Colo. (1983)

（受理　昭和59年1月17日）