パーソナルコンピュータ用数式処理システム (UMATH) の拡張

大久保　明　伸*

EXTENSION OF SYMBOLIC MATHEMATICAL OPERATION SYSTEM USING PERSONAL COMPUTER

Akinobu Ohkubo

Abstract

In this paper, an extension of symbolic mathematical operation system useing personal computer is described. This system (UMATH) has faculties of plotting and resolution in addition to symbolic mathematical operation. These faculties are used in muSIMP/muMATH and in the other language system using hybrid method.

This resolution system is mini-PROLOG system and these function are program function of muSIMP.

1．はじめに

先に，パーソナルコンピュータ上で動作する数式処理システムの改良と工学教育への利用を提案した．[1][2][3] 本報告は改良したシステム（UMATH）の具体的なプログラムの一部，muSIMP によるプログラミング技法と今回試作した数式処理の可能な小型 PROLOG システムについて述べる．また数式処理システムの拡張についてのべる．

数式処理システムを構成するには次のハードとソフトウエアが最低限必要である．

(1) ハードウエア
　CP/M が使用可能なパソコン
　カーブプロタ
(2) ソフトウエア
　CP/M, muSIMP/muMATH, BASIC （または数値計算に適した言語）

＊宇部工業高等専門学校電気工学科

2．数式処理システムで使用する数式のデータ構造

数式処理システムで数式を処理するためには，その構造を定義しておく必要がある．記述言語 muSIMP は LISP 系の言語であるために数式もＳ式で取り扱う．ただし，入力と出力は muSIMP の組み込み関数 PARSE と PRTMATH を利用すれば通常の数式の表現でよい．しかし，数式処理のプログラムを製作する場合は内部のデータ構造（Ｓ式）を定義しておく必要がある．Ｓ式を用いた数式の表現法には色々あるが，ここでは次のような構造を採用する．

［数式の内部構造の定義］

〈数式〉：：＝〈項〉｜（〈演算記号〉 △ 〈項列〉）
〈項列〉：：＝〈項〉｜〈項〉 △ 〈項列〉
〈項〉　：：＝〈アトム〉｜〈数式〉
〈アトム〉：：＝〈数値〉｜〈文字列〉
〈演算記号〉：：＝ ＋ ｜ ＊ ｜ ＾ ｜〈文字列〉
〈数値〉：：＝ 600桁までの正の整数 ｜ －１
〈文字列〉：：＝ アルファベットの列 ｜ ＃ 〈文字列〉
　　｜ ＃PI ｜ ＃I

ただし，△はスペース記号，#PIは円周率，#Iは虚数単位ををあらわす．

この定義にもとずいた数式の例を示しておく．

(例1) A+B+C は (+ A B C)
(例2) A*B*C は (* A B C)
(例3) A ^ B は (^ A B)
(例4) (A+B)／(C+D) は
(* (+ A B) (^ (+ C D) −1))
(例5) SIN (X) は (SIN X)

この様に，S式の特別な集合が数式である．このS式を評価 (EVAL) することによりシステムは処理を行う．入力や出力ではS式で取り扱うより自然な表現(例の左の形式)が望ましいので，関数 PARSE や PRTMATH が用意されている．また，これらの入力，出力関数を作り替えて新しい数式の外部表現を採用することも可能である．例えば，A TASU B HA の様な外部表現を入力出来る様にするためには TASU と HA を演算記号として定義し，処理関数を作ればよい．

数式をこのようなS式で記憶すると次のような問題が発生する．X*Y+X*Z と X*(Y+Z) すなはち(+(* X Y) (* X Z))と(* X (+ Y Z))は式の意味としては同じであるがS式としては同じであるとは判定できない．これについては別の機会に議論することにする．本システムでは別の式として判定する．

muSIMP や移植したミニ PROLOG のプログラムも同様にS式で処理される．さらに LISP の処理系もS式を処理する．従って，数式の構造もS式にしておけば各処理系の間でデータの互換性があり便利である．

3．数式システムの構成法

まえがきで述べたように，数式処理システムを構成するためには次のソフトが最低限必要である．

CP/M, muSIMP,muMATH である．まず，CP/M で muSIMP を起動する．muMATH は muSIMP で記述された数式処理プログラム群である．muMATH はあるまとまった機能の複数のファイルに分割されている．各ファイルの中にある関数は相互に呼び出されるので出来るだけまとまった機能を持った関数を同一のファイルに格納することが望ましい．

すべての muMATH ファイルを取り込んだシステムを作るとユーザ用のメモリ領域が不足するので必要最小限に止めておくべきである．muSIMP は必要に応じて機能を増強するコマンド (RDS) を備えているが構成にかなりの時間を必要とする．そこで，RDS コマンドにより構成したシステムを muSIMP の SYS ファイルとして保存しておくと便利である．

RDS コマンドによりシステムを構成する方法は文献(7)(8)を参考にしてほしい．ここでは muSIMP 入出力関数の使用法をかねてちいさなサンプルプログラムを示しておく．

関数 READ ( ) は，S式を指定されたファイルから読み取る関数である．アトムもS式なので文字列もこの関数で読み取ることができる．PRINTLINE ( ) も同様で指定されたファイルにS式を出力する関数である．初期状態ではキイボードとディスプレイが指定されている．この指定を変える関数が RDS と WRS である．SAVE はメモリの内容を指定されたディスク装置に格納する．

以上の数式処理システムの構成手続きを muSIMP の関数 (MKMATH) として**プログラム1**に示しておく．ユーザは MKMATH の指示に従って必要なファイル名を入力すればよい．ただし，この関数の引数は空である．この手法は数式処理プログラムを作成するとき，既にファイル保存されている公式を正確に挿入する場合に利用できる．詳しくは以降の節で述べる．

4．公式の生成とその利用法

数式処理システムを用いて公式を生成し，その公式を直接利用するプログラム技法についてのべる．この技法は複雑な数式をプログラム内に記述したり，既に求めた数式解を利用したり，また公式データベースを構築してより高度なシステムを作成するのに重要である．また，システムの出力結果を BASIC や FORTRAN に引き渡すときにも用いる．

手順は次のとうりである．

[公式を作り出す手順]
(1) WRS (fname, atr, dev)
ファイルのオープン
(2) 公式を作る．
(3) PRINTLINE (〈式〉)
S式を出力

```
FUNCTION MKMATH(%LOCAL:% W,A,FLAG,FNAME,ATR,DEV),
 W:FALSE,
 LOOP,
  PRINTLINE("INPUT FILE-NAME ?"),
  FNAME:READ(),
  PRINTLINE("INPUT ATRI"),
  ATR:READ(),
  PRINTLINE("DEVICE NAME"),
  DEV:READ(),
  PRINTLINE("GO OR END"),
  FLAG:READ(),
  WHEN FLAG='END,EXIT,
  RDS(FNAME,ATR,DEV),
  LOOP
   EVAL(PARSE(SCAN(),0)),
   WHEN RDS=FALSE,EXIT,
  ENDLOOP,
  RDS(),
  PRINTLINE("DO YO WANT TO SAVE ?.... YES OR NO"),
  FLAG:READ(),
  WHEN FLAG='YES,PRINTLINE("FILENAME FOR SAVE ?"),SAVE(READ(),A),EXIT,
 ENDLOOP,
ENDFUN&
RDS();
```

プログラム1　数式処理システムの構成手続き (muSIMP)

(4) PRINTLINE (▼(RDS))
　ファイルの終わりを示す記号
(5) WRS ( )
　ファイルのクローズ

ただし，fname はファイル名，atr はその属性，そして dev はドライブ名である．

[例]　SIN (X) の15項までのテーラ展開をS式で TAYLOR. SUB に保存する．

```
FUNCTION MKFILE ( % LOCAL : % W),
  W : TAYLOR (SIN (X), X, 0, 15),
  WRS (TAYLOR, SUB, B),
  PRINTLINE (W),
  WRS ( ),
ENDFUN &
```

[公式を利用する手順]

(1) RDS (fname, atr, dev)
　ファイルのオープン
(2) EVAL (READ ( ))
　公式を読む
(3) RDS ( )
　ファイルのクローズ

[例]

```
FUNCTION GETFUN (%LOCAL : % W),
  RDS (MKFILE, SUB, B),
  W : READ ( ),
  RDS ( ),
  W,
ENDFUN &
```

5．ハイブリッド処理

数式処理システムで得られた結果を BASIC や FORTRAN の数値計算用の言語に引き渡すことが，時には必要である．又逆も必要になることもある．ハイブリッド処理とはこのように，異種の言語間でデータのやりとりを行う処理のことをいう．CP/M の言語は，ほとんどが TEXT ファイルを処理するので，ハイブリッド処理をおこなうには各言語に文法に適した TEXT ファイルを経

由すればよい．各言語に万能なトランスレタを作るより問題に応じた専用のトランスレタを作れば十分である．時には数ページに及ぶ数式を誤りなく，人手により入力することは不可能に近いことである．ここでは，数式処理の結果（行列式）をBASICへ引き渡すためのファイル生成関数の例を**プログラム2**に示しておく．**プログラム2**の関数DIMBASE（Z）は行列Zの要素をBASICの配列要素の値として設定する関数である．通常はこの出力をBASIC用のファイルにコピーする．

UMATHはmuSIMPで記述されたPROLOGの処理系を持っている．数式処理の一部をPROLOG表現で記述したほうが便利なこともある．これもここでいうハイブリッド処理である．特にmuSIMPもPROLOGもデータやプログラムはずべてS式であらわすために，お互いに整合性がよい．それぞれの処理系に対して，S式の意味さえ確定しておけば相互にその特徴を用いた処理が可能である．数式処理を工学の研究やCAIシステムに利用する場合特に重要な機能である．ただし，すべての機能をとりこんだ一つの処理系では，このクラスのパソコンには荷が重すぎる．従ってパソコンではこのような手法に頼らざるをえない．

```
FUNCTION DIMBAS(Z,%LOCAL:%W,ELM,I,J),
 WHEN NOT FIRST(Z)='{ OR NOT FIRST('[)  EXIT,
 PRINT(1),SPACES(2),
 PRINTLINE(COMPRESS(LIST('DIM," ",'A,LPAR,M,COMMA,N,RPAR))),
 PRINT(2),SPACES(2),
 POP(Z),I:1,
 LOOP,
  W:POP(Z),
  WHEN EMPTY(W),PRINTLINE("REM .....BY MUSIMP"),  EXIT,
  J:1,POP(W),
   LOOP,
     ELM:POP(W),
     WHEN EMPTY(ELM),EXIT,
     PRINT(COMPRESS(LIST('A,LPAR,I,COMMA,J,RPAR,'=,ELM,':))),
    J:J+1,
   ENDLOOP,
  I:I+1,
ENDLOOP,
ENDFUN&


Z;
@: {[A, B, C, D, E, F],
    [G, H, I, J, K, L],
    [M, N, O, P, Q, R],
    [S, T, U, V, W, X],
    [Y, Z, A, B, C, D],
    [E, F, G, H, I, J]}



? DIMBAS(Z);
@:1  DIM A(M,N)
2  A(1,1)=A:A(1,2)=B:A(1,3)=C:A(1,4)=D:A(1,5)=E:A(1,6)=F:A(2,1)=G:A(2,2)=H:
A(2,3)=I:A(2,4)=J:A(2,5)=K:A(2,6)=L:A(3,1)=M:A(3,2)=N:A(3,3)=O:A(3,4)=P:
A(3,5)=Q:A(3,6)=R:A(4,1)=S:A(4,2)=T:A(4,3)=U:A(4,4)=V:A(4,5)=W:A(4,6)=X:
A(5,1)=Y:A(5,2)=Z:A(5,3)=A:A(5,4)=B:A(5,5)=C:A(5,6)=D:A(6,1)=E:A(6,2)=F:
A(6,3)=G:A(6,4)=H:A(6,5)=I:A(6,6)=J:REM .....BY MUSIMP
 REM .....BY MUSIMP
```

プログラム2　配列（Z）をBASICへ変換する関数と実行例（muSIMP）

6．muSIMPによるミニPROLOGシステムの移植

現在，PROLOGは数多くのパソコンで使用可能であるが，数式処理能力を持ったシステム(文献6)は数少ないし，市販されていない．PROLOGの移植の目的は数式処理システムと推論処理系との結合，公式ファイルのインテリジェント化等の実験と，muSIMPの記述性を確かめるためである．インタプリタでインタプリタを製作するため処理速度はかなり遅くなるのはやむおえない．しかし，プログラムの記述性はmuSIMPのみの場合よりははるかに良くなる．

そこで文献(5)LISPで記述されているミニPROLOGをmuSIMPで記述して数式処理システムに移植した．PROLOGシステムで数式処理システムを構成した例もあるが，muSIMPで構成することによりmuMATHやUMATHの数式処理関数を直接利用する事ができる．このPROLOGシステムは後で示すように，やはりS式で記述するためmuSIMPとのプログラムやデータのやりとりがなじみやすいという利点がある．データベースの検索や推論形式によるアルゴリズムの表現はこのPROLOGを用いることができるし，数式処理における逐次型のアルゴリズムはmuSIMPで記述しほうが有利である．

ミニPROLOGによるプログラムは次の形式である．

(0) 変数は文字列の前に＊を付けたシンボルである．

(1) 述語や関数f (a b c … x) はS式 (f a b c … x) の形式で入力する．
特に主張は
(ASSERT S式) または
(ASSERT S式 S式…S式)

(2) 数式処理やmuSIMPの基本関数を利用する場合は次のEVALかCALLを用いる．

[副作用が必要なとき]
(CALL PRINT ABC)
ABCをPRINTする．
(CALL RDS PRIM PRO A)
RDS (PRIM, PRO, A); と同じで，ファイルをオープンする．

[値が必要なとき]

(EVAL (INT (SIN X) X) ＊X)
SIN (X) をXで積分して (muSIMPで実行される.) その値を＊Xに得る．
(ASSERT (FIRST ＊P ＊Q) (EVAL (FIRST ＊P) ＊Q))

FIRST (＊P) の値を＊Qに得る主張である．

プログラム3にmuSIMPの基本的な関数にたいする主張を示しておく．
次にPROLOGシステムの構成法を示しておく．

(1) muSIMPと数式処理システムの必要なファイルをロードする．
(2) RDSコマンドでPROLOGシステムを構成する．
(3) PROLOG ( ); でPROLOGシステムを起動する．
(4) (CALL RDS PRIM PRO) を実行してPRIM. PROファイルに記述されているmuSIMPの基本関数を直接利用可能にする．
(5) (CALL SAVE PROLOG A) を実行してPROLOG. SYSファイルをドライブAに保存する．

システムを使用した実行結果を**図1**に示しておく．%で囲まれた文字列はコメントとみなされる．＊S＊は主張が正当であったことを示す．正当でない場合はFALSEを表示する．％1％と％2％は事実に関する主張であり，％3％と％4％はANDのいわゆる定義である．％5％はANDの実行でその結果(＊Xと＊YがAなる関係にあり，同時に＊Yと＊ZがAなる関係になるような＊X ＊Y ＊Zをリストとして出力した)である．％6％は関数＊Fを0の近傍でのテイラー展開を6項まで出力する主張である．％7％は＊FをEXP (X) として実行したものである．％8％は積分の定義である．％6％と％8％はEVALによりmuSIMP数式処理が実行される例でもある．

7．プロッタ用関数

前に述べたハイブリッド処理の技法を用いればグラフィク処理は可能であるが，直接muSIMPでも簡単な図形の処理が出来るようにした．プロッタはグラフテック社のMIPLOT IIでインターフェイスはセントロニク

```
A>TYPE PRIM.PRO
%THESE ARE PRIMITIVE FUNCTION OF MUSIMP FOR PROLOG.
THESE ARE READ BY PROLOG INPUTCOMMAND(CALL RDS PRIM.PRO).%
(ASSERT (ATOM *P *Q)(EVAL (ATOM *P) *Q))
(ASSERT (FIRST *P *Q)(EVAL (FIRST *P) *Q))
(ASSERT (REST *P *Q)(EVAL (REST *P) *Q))
(ASSERT (SECOND *P *Q)(EVAL (SECOND *P) *Q))
(ASSERT (RREST *P *Q)(EVAL (RREST *P) *Q))
(ASSERT (THIRD *P *Q)(EVAL (THIRD *P) *Q))
(ASSERT (RRREST *P *Q)(EVAL (RRREST *P) *Q))
(ASSERT (ADJOIN *P *Q *R)(EVAL (ADJOIN *P *Q) *R))
(ASSERT (REVERSE *P *Q *R) (EVAL (REVERSE *P *Q) *R))
(ASSERT (OBLIST *P *Q)(EVAL (OBLIST)*Q))
(ASSERT (REPLACEF *P *Q *R)(EVAL (REPLACEF *P *Q) *R))
(ASSERT (REPLACER *P *Q *R)(EVAL (REPLACER *P *Q) *R))
(ASSERT (CONCATEN *P *Q *R)(EVAL (CONCATEN *P *Q) *R))
(ASSERT (NAME     *P *Q   )(EVAL (NAME     *P   ) *Q))
(ASSERT (INTEGER  *P *Q   )(EVAL (INTEGER  *P   ) *Q))
(ASSERT (EMPTY    *P *Q   )(EVAL (EMPTY    *P   ) *Q))
(ASSERT (POSITIVE *P *Q   )(EVAL (POSITIVE *P   ) *Q))
(ASSERT (NEGATIVE *P *Q   )(EVAL (NEGATIVE *P   ) *Q))
(ASSERT (ZERO     *P *Q   )(EVAL (ZERO     *P   ) *Q))
(ASSERT (EQ       *P *Q   )(EVAL (=        *P   ) *Q))
(ASSERT (<        *P *Q *R)(EVAL (<        *P *Q) *R))
(ASSERT (>        *P *Q *R)(EVAL (>        *P *Q) *R))
(ASSERT (NOT      *P *Q   )(EVAL (NOT      *P   ) *Q))
(ASSERT (ASSIGN   *P *Q *R)(EVAL (ASSIGN   *P *Q) *R))
(ASSERT (:        *P *Q *R)(EVAL (:        *P *Q) *R))
(ASSERT (ASSOC    *P *Q *R)(EVAL (ASSOC    *P *Q) *R))
(ASSERT (GET      *P *Q *R)(EVAL (GET      *P *Q) *R))
(ASSERT (PUT   *P *Q *R *S)(EVAL (PUT   *P *Q *R) *S))
(ASSERT (REMPROP  *P *Q *R)(EVAL (REMPROP  *P *Q) *R))
(ASSERT (COMPRESS *P *Q   )(EVAL (COMPRESS *P   ) *Q))
(ASSERT (EXPLODE  *P *Q   )(EVAL (EXPLODE  *P   ) *Q))
(ASSERT (LENGTH   *P *Q   )(EVAL (LENGTH   *P   ) *Q))
(ASSERT (MINUS    *P *Q   )(EVAL (MINUS    *P   ) *Q))
(ASSERT (PLUS     *P *Q *R  )(EVAL (PLUS     *P  *Q ) *R))
(ASSERT (DIFFERENCE *P *Q *R)(EVAL (DIFFERENCE *P *Q) *R))
(ASSERT (TIMES    *P *Q *R)(EVAL (TIMES    *P *Q) *R))
(ASSERT (QUOTIENT *P *Q *R)(EVAL (QUOTIENT *P *Q) *R))
(ASSERT (MOD      *P *Q  *R)(EVAL (MOD      *P *Q) *R))
(ASSERT (DIVIDE   *P *Q  *R)(EVAL (DIVIDE   *P *Q) *R))
(ASSERT (MULTIPLY *P *Q  *R)(EVAL (*        *P *Q) *R))
(ASSERT (READCHAR    *P    )(EVAL (READCHAR       ) *P))
(ASSERT (SCAN        *P    )(EVAL (SCAN           ) *P))
(ASSERT (READ        *P    ) (EVAL (READ          ) *P))
(ASSERT (PARSE    *P *Q  *R)(EVAL (PARSE    *P *Q) *R))
(ASSERT (SYNTAX   *P       )(CALL  SYNTAX  *P       ))
(ASSERT (MATCH    *P *Q    )(EVAL (MATCH    *P   ) *Q))
(ASSERT (TERMINATOR  *P    )(EVAL (TERMINATOR    ) *P))
(ASSERT (ECHO        *P    )(EVAL (ECHO          ) *P))
(ASSERT (RDS       *P *Q *R)(CALL RDS        *P *Q *R))
(ASSERT (PRINT     *P       )(CALL PRINT      *P       ))
(ASSERT (NEWLINE   *P       )(CALL NEWLINE    *P       ))
(ASSERT (PRINTLINE *P       )(CALL PRINTLINE  *P       ))
(ASSERT (SPACES    *P       )(CALL SPACES     *P       ))
(ASSERT (PRTMATH   *P       )(CALL PRTMATH *P 0 0 TRUE))
(ASSERT (WRS       *P *Q *R)(CALL WRS        *P *Q *R))
```

プログラム3　ミニPROLOGによるmuSIMPの基本関数の定義（muSIMP）

```
@: TESTPRO

@: TRUE

?
PROLOG();@:(PORTABLEPROLOG)

%1% (ASSERT (A B C))*S*

%2% (ASSERT (A C D))*S*

%3% (ASSERT (AND FALSE))*S*

%4% (ASSERT (AND *U)(EVAL (FIRST *U) *V)(EVAL (REST *U) *W) *V (AND *W))*S*

%5% (AND ((A *X *Y)(A *Y *Z)(PRINT (*X *Y *Z))))(B C D)*S*

%6% (ASSERT (TENKAI *F)(EVAL (TAYLOR *F X 0 6) *ANS)(PRTMATH *ANS))*S*

%7% (TENKAI (^ #E X))1 + X + X^2/2 + X^3/6 + X^4/24 + X^5/120 + X^6/720*S*

%8% (ASSERT (INT *F)(EVAL (INT *F X) *ANS)(PRTMATH *ANS))*S*

%9% (INT (SIN X))- COS(X)*S*

(CALL RDS FALSE FALSE)*S*
```

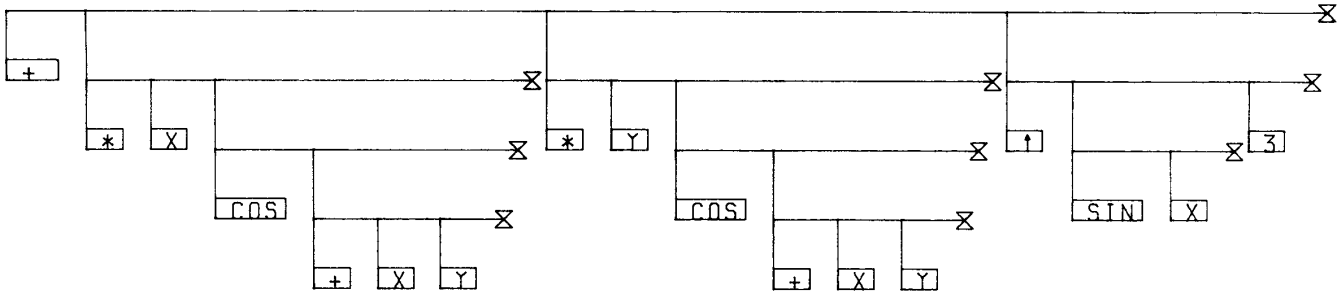
図－1　ミニPROLOGによる実行例

スタイプのものである．これにより処理結果のグラフ表示や数式の清書(数式を数式らしく表示すること)等に便利である．また，再帰処理が必要な図形処理はこの関数を利用して muSIMP で記述したほうが便利である．現在，タートルグラフィクスの関数を製作中である．関数の基本的なものを**プログラム4**に示し，これを用いて出力した結果を**図2**に示しておく．

プログラム4はS式を木構造でプロットする関数(TREE)である．プログラム中にあるD,S,N等の関数はプロタのマニュアルを参照して欲しい．PLOT MATH は数式をプロッタに描く関数である．TREE を起動して数式の内部構造を木構造として表示したものが**図2**である．

8．まとめ

以上，数式処理システムの特殊な技法や数式処理可能な PROLOG の仕様について述べた．これによりかなり UMATH は使いやすいものとなった．しかし，8 BIT パソコンではメモリが少ないため，これらの機能を同時に



X*COS(X+Y)+Y*COS(X+Y)+SIN(X)↑3

図－2　関数TREEの実行例

```
FUNCTION TREE(F,X,Y,AS,%LOCAL:% H,XMAX),
XMAX:0,
BLOCK
  WHEN X=FALSE OR Y=FALSE OR AS=FALSE,
   X:0,Y:2700,AS:3,H:AS*7,XMAX:0,S(AS),
  EXIT,
ENDBLOCK,
S(AS),H:AS*7,XMAX:X, MKTREE(F,X,Y,H,AS),
ENDFUN&

FUNCTION MKTREE(OB,X,Y,H,AS),
M(X,Y),
WHEN EMPTY(OB),S(AS*2),N(5),S(AS),EXIT,
WHEN ATOM(OB),
 XMAX:XMAX+(LENGTH(OB)+2)*7*(AS+1),D(XMAX,Y),
 D(XMAX,Y+7*(AS+1)+10),D(X,Y+7*(AS+1)+10),D(X,Y),
 P(" "),PLOTMATH(OB),
EXIT,
D(X,Y-5*H),MKTREE(FIRST(OB),X,Y-5*H,H,AS),
BLOCK WHEN XMAX<X,XMAX:X,EXIT,ENDBLOCK,
M(X,Y),
BLOCK WHEN XMAX>X OR XMAX=X,
        D(XMAX+H,Y),X:XMAX+H,XMAX:XMAX+H,
      EXIT,
ENDBLOCK,
D(X+H,Y),MKTREE(REST(OB),X+H,Y,H,AS),
BLOCK WHEN XMAX<X,XMAX:X,EXIT,ENDBLOCK,
ENDFUN&

FUNCTION D(X,Y),
LPRINTER:TRUE,
 PRINT('D),
 PRINT(X),
 PRINT(COMMA),
 PRINT(Y),
 NEWLINE(),
 LPRINTER:FALSE,
ENDFUN&

FUNCTION S(X),
 LPRINTER:TRUE,
 PRINT('S),
 PRINTLINE(X),
 LPRINTER:FALSE,
ENDFUN&

FUNCTION N(X),
 LPRINTER:TRUE,
  PRINT('N),
  PRINTLINE(X),
 LPRINTER:FALSE,
ENDFUN&

FUNCTION PLOTMATH(X),
 LPRINTER:TRUE,
 PRINT('P),
 PRTMATH(X,0),
 NEWLINE(),
 LPRINTER:FALSE,
ENDFUN&
```

プログラム4　S式を木構造で出力する関数（TREE）と補助関数

メモリに常駐させることができない．このために，使用に際してはかなりの熟練を必要とする．現在安価な16 BITパソコンが普及しつつあるのでmuSIMPをこの上に移植し，より本格的なシステムの構築が必要である．また，PROLOGと数式処理の結合は興味ある問題であるので研究を続けていく予定である．

参考文献

(1) 大久保：パーソナルコンピュータによる数式処理，宇部高専研究報告30号，P. 21（昭和59年）

(2) 大久保：パーソナルコンピュータ用数式処理システムの改善について，情報処理学会（59年度）全国大会論文集，1B-6，（昭和59年）

(3) 大久保：数式処理システムの電卓的な利用法につい

て，高等専門学校情報処理研究協議会（昭和59年）
(4) CP/M はデジタルリサーチ社の，muSIMP/muMATH はザソフトウエアハウス社の商標登録である．
(5) 中島秀之：Prolog，産業図書（58年）
(6) 徳山高専の LISP-PROLOG（58年）
(7) 大久保：パーソナルコンピュータによる数式処理（その1），数式処理通信（Vol. 1-No. 3）（58年）
(8) 大久保：同上（その2），数式処理通信（Vol. 1-No. 4）（59年）

謝　辞

日頃，お世話になる電子計算機室の関係のかたがたに謝意を表します．

（昭和59年9月17日受理）