集合演算の機能をもつプログラム言語とその処理系について

山口　忠*

(昭和46年8月30日受理)

On the Programming Language with Set Operations and its Processor

Tadashi YAMAGUCHI

Abstract

The author has designed the language (SETL-1) with set operations, based on the next three view points:

(a) to treat a set (or a family) as one unit of operation or I/O,

(b) to write the usual Boolean expression on sets (or families),

(c) to treat functions on sets in a natural way.

This is a report on designing SETL-1 language and programming its processor, which is written in PL/I language.

要　旨　本報告書は，集合演算を主体にしたプログラム言語の設計および処理システムの作成に関して述べるものである。

1. 緒　　言

順序機械，オートマトン等の解析に伴うアルゴリズムでは有限集合（以下単に集合という）が多く取扱われる。この種のアルゴリズムを計算機で行なうために，現在のコンパイラー言語(FORTRAN, ALGOL, PL/I 等）を使うとすれば，演算に本質的でない情報（例えば集合の要素数）や集合のための領域の確保等に対する処置までプログラムに組まなければならない。一方，FORTRAN 等の言語の開発の動機は数式の自然な記述が出来る点にあった。これ等の点から，集合を取扱うアルゴリズムのために，集合の演算が自然に記述出来る言語が開発されて当然であろうと考える。

筆者の知る限りこの種の言語は作られていないようである。唯，LISP 等のリスト処理言語では，例えば集合 (1, 2, 3) を１つの記号例とみて扱うことは出来るが，集合演算式等の自然な記述は出来ない[1]。

そこで，

(a) 集合，集合族を演算，入出力の１単位に出来ること。

(b) 集合（族）に関する通常の演算式が書けること。

* 電気工学科　系統工学講座

(c) 集合から集合への関数が扱えること。

の3点に視点を絞って1つのプログラム言語を設計した。またこの言語の基本演算は従来の言語のそれよりもマクロであるため，処理システムは翻訳 phase と解読実行 phase からなるインタープリターと類似の形で作成した。なおこの処理システムの作成には，汎用言語といわれている PL/I を用いた[2]。以下言語系を SETL-1，処理システムを SETLPROC-1 と呼ぶことにする。

2. SETL-1 の文法概要

ここで，1. で述べた基本的な考え方に従って設計した言語の文法について簡簡に説明しておく。言語要素となる記号，文字，またカード等とくに述べない点については FORTRAN と同じである。

2-1 定数と変数

定数には数定数，集合定数，集合族定数の3つのタイプ（以下，数，集合，集合族の属性をタイプという）のものがある。数定数は FORTRAN と同じである。集合定数とは，「(数定数，数定数，…)」で「( )」は空集合を表わす。記号「と」で囲まれたものが言語要素である。また集合族定数は「(集合定数，集合定数，…)」の型をしている。

変数も上の3つのタイプに対応して，数変数，集合変数，集合族変数がある。変数名の作り方は，FORTRAN と同じ英文字ではじまる6文字以内の英数字で作る。

例. (1, 2, 3), (−1.2, 3.4, 5.0) は集合定数であり，((1, 2), (3, 4, 5)), ((1.2, 3.4), (−1, 2)) などは集合族定数である。

2-2 ステートメントの種類

ステートメントは大きく分けて宣言文，制御文，算術代入文，入出力文，END 文それに注釈文の7種類である。END 文と注釈文は FORTRAN と同じである。

(1) 宣 言 文

プログラム中に現われるすべての変数名，関数名は宣言文で宣言されなければならない。数変数，集合変数，集合族変数に対してはそれぞれ「NUMERICAL」，「SET」，「FAMILY」，なる key word により宣言する。また関数に対する宣言は「FUNCTION fname $(x)=y$」なる定型で行なう。ただし，fname は定義しようとする関数の名前（作り方は変数名と同じ）で，x, y は定義域，値域を表わす集合定数または集合変数である。

(2) 制 御 文

これには，IF 文，JUMP 文，STOP 文の3種類がある。IF 文の定型は「IF$(v_1 . \mathrm{lop} . v_2)n$」で，$v_1, v_2$ は同タイプの定数または変数で，n は文番号である。また lop は「=」，「>」，「<」なる論理演算子で，v_1, v_2 が数タイプのときは，大小関係を意味するが，集合（族）タイプのときは，包含関係 $=, \supseteq, \subseteq$ を意味する。IF 文全体の意味としては，FORTRAN の論理 IF 文と同じである。

(3) 入 出 力 文

入出力文は，FORMAT なしの標準の REDA 文，WRITE 文の2種類がある。READ 文の定型は「READ 変数, 変数, …, 変数」でカード上のデータは数定数，集合（族）定数と同じ形で与えデータ間の区切りはカンマまたはスペースで行なう。

WRITE 文は，WRITE のあとに出力したい変数，文字 '…'，改ページ指示の「0」，改行指示の「/」をカンマで区切って並べたものである。出力動作はリストに並んだ順に行なわれる。

例えば，「WRITE(0), 'A=', A, 'B=', B」ではじめに改ページをして，文字 A= を印

字し，そのあとにAの内容を印字する。Bについても同じである。

(4) 標準関数

集合（族）を扱う問題で比較的よく使われると思われる機能を標準関数として登録してある。表-1での関係 R_1, R_2 について説明する。S を引数の集合，f を S を含むある集合からある集合へのユーザー関数名とするとき，R_1 を次のように定義する。

(i) $s_1, s_2 \in S$ に対して

$$s_1 R_1 s_2 \Longleftrightarrow f(s_1) = f(s_2)$$

R_2 は R_1 の拡張で次のように定義される。ここで T は1つの集合族とし f の値域集合をCoverする。

(ii) $s_1, s_2 \in S$ に対して

$s_1 R_2 s_2 \Longleftrightarrow f(s_1)$ と $f(s_2)$ が T の同じ要素集合に入る。

表—1　標準関数名とその内容

関数名	引数の個数とタイプ		結果のタイプ	内容
SHARP	1	集合（族）	数	集合（族）の要素数を求める。
CHOOSE	2	集合，数	数または集合	集合の指定された要素を選択する。
		集合族，数	集合	集合族の指定された要素を選択する。
REDUCE	1	集合（族）	集合（族）	重複している要素があれば重複分を消去する。
REORDER 1	1	集合（族）	集合（族）	集合の場合は要素の大きい順に並べかえる。集合族の場合は要素数の大きい順に並べかえる。
REORDER 2	1	集合（族）	集合（族）	REORDER 1 の逆順
PARTITION 1	2	集合，ユーザー関数	集合族	集合を関係 R_1(注) で分割する。
PARTITION 2	3	集合，ユーザー関数および集合族	集合族	集合を関係 R_2(注) で分割する。

(注) R_1, R_2 については本文中で説明する。

(5) 算術代入文

算術代入文には，単純文，2項演算文，関数代入文の3種類がある。単純文は「$v_1 = v_2$」なる型で右辺を左辺の変数に代入するもので右辺と左辺は同じタイプでなければならない。2項演算文の定型は「$v_1 = v_2$.op.v_3」で，v_1 は変数，v_2, v_3 は定数または変数である。ここで，.op. とは v_2 と v_3 が数タイプのときは +，−，*，/ のいずれかであり，v_2 と v_3 が集合（族）タイプのときは，和集合をとる「+」，差集合をとる「−」，共通集合をとる「*」のいずれかである。

関数代入文は fname がユーザーの宣言した関数名であれば「v_1 = fname(v_2)」の型で，v_2 の fname による像集合を v_1 代入へすることを意味する。fname が標準関数名のときは，引数の個数およびタイプ，左辺のタイプは表-1にあげた約束に従わねばならない。

2-3 プログラム

この言語でのプログラムは，必要な宣言文とステートメント群それに END 文からなる。注釈文は END 文の前なら何処にあってもよい。

以上文法の概略を述べたが，ここでわかる通り SETL-1 では，1. で述べた (a), (b), (c) の機能の最小限度は持っていると思われる。次にこの言語の内部仕様すなわち処理システム作成について述べる。

3. 処理システムの PL/I 言語による作成

本節では，2. で述べた言語 (SETL-1) で書かれたプログラムを処理し実行するシステム (SETLPROC-1) について説明する。SETLPROC-1 は，プログラムを仮想のオブジェクトコードに翻訳する phase と，そのオブジェクトコードを解読実行する phase にわかれる。

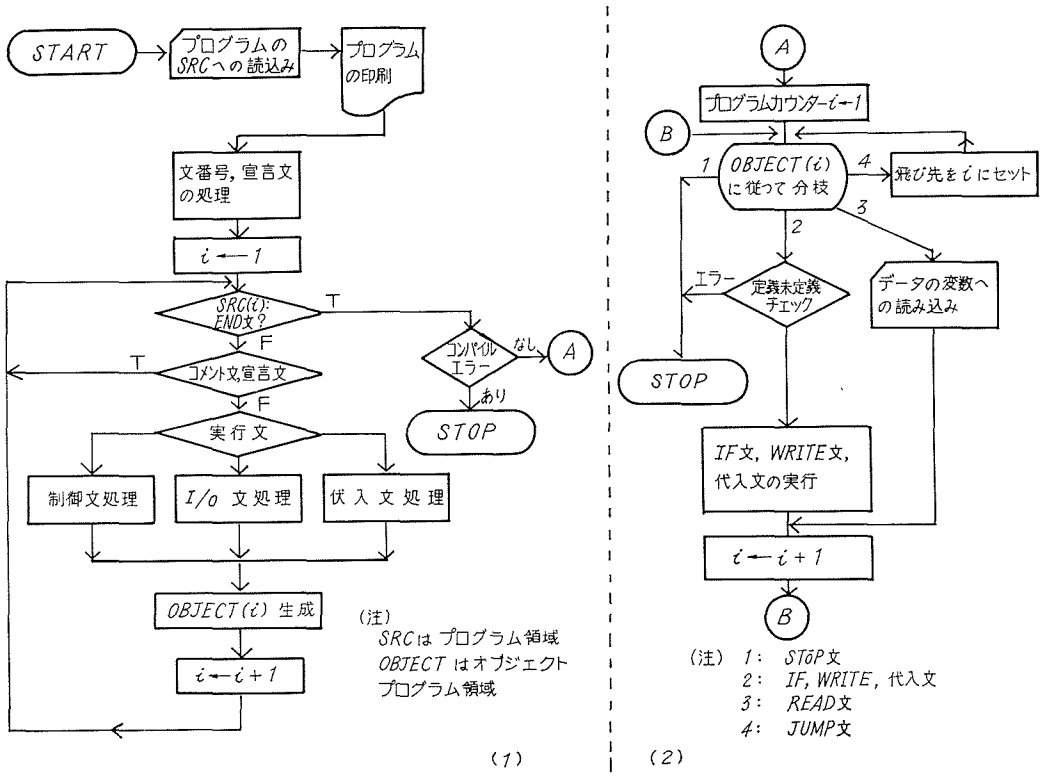
翻訳 phase の大半は記号処理であり，解読実行 phase は一種のリスト処理専用の仮想計算機である。この両 phase を作成する言語としては，記号処理，リスト処理の機能を持っている PL/I 言語が最も適していると考え，これを用いた[3),4)]。

3-1 翻訳 phase

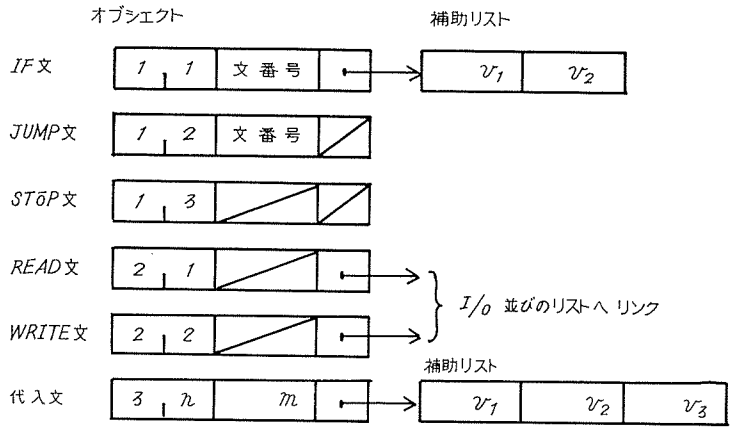
この phase での処理は従来のコンパイラーと大体同じである。これは 16 個の PL/I の手続き (procedure) から構成されていて，その主なものを挙げると，

(1) 文番号の処理
(2) 宣言された変数の名前表への登録
(3) 実行ステートメントの翻訳
(4) 記号の PACK, UNPACK, カッコ整合のチエック
(5) 変数名，関数名等のテーブル探索

等がある。翻訳 phase 全体の流れ図を 図-1-(1) に挙げておく。(3) について説明すると各実行ステートメントは，図-2 に示した様なオブジェクトに翻訳される。その際，従来の機械語のアドレス部とはちがって，文の種類によって補助リストを持つ形に翻訳される。



図—1 プロセッサーの流れ図

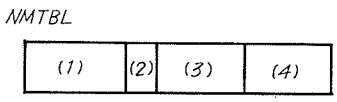


注.　n と m は代入文のヴアリエション.　v_1, v_2, v_3 は本文 2－1の(5)と同じ意味。

図—2　オブジェクトコード

3-2　名前表の構成

処理の一様さを計るために，変数はもとより関数名，定数も一応名前表 (NMTBL) と呼ばれるテーブルに登録しておく。名前表の各項は 図-3 の形をした PL/I での構造体で構成されている。名前欄は登録しようとする名前を文字で入れておき，定義未定義欄は関数の定義時，あるいは実行時に値が入ったとき1になり，未定義の間0となっている。値リストへのポインターは 表-2 に示すようにタイプによって内容が異なる。また名前表に登録される項目は，同じタイプのもの同志リンクされており，そのリンクされたリストには，先頭と終端の位置を示す2つのポインター H, T が対応している。これは項目の追加，探索のために使われる。図-4 は次の4個の宣言文



(1) 変数名あるいは関数名
　　定数のときはブランク

(2) 定義　未定義のサイン

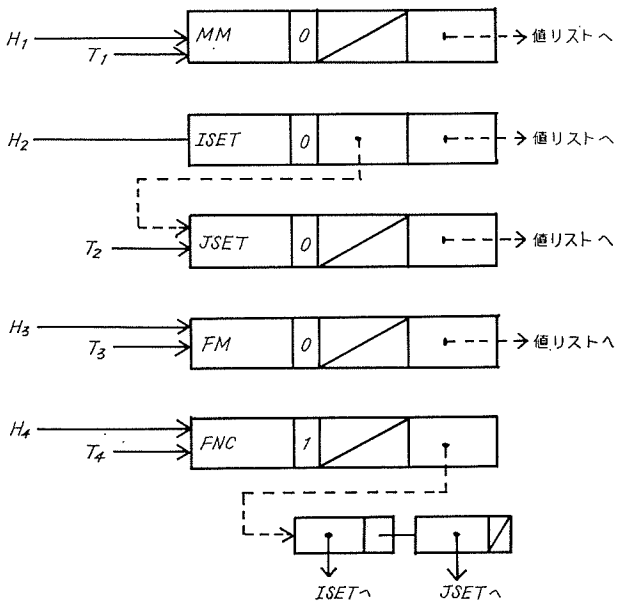
(3) 次の同タイプの項へのポインター

(4) 値リストのポインター

図—3　名前表の項目の構成

表—2　値リストポインターの内容

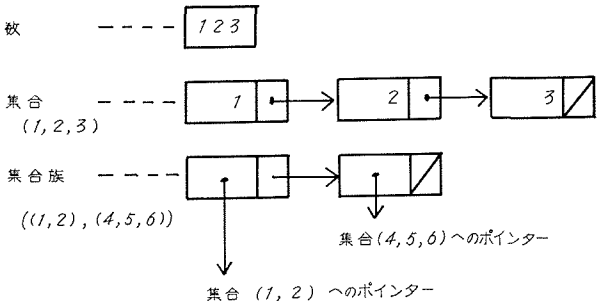
タイプ	内容
数	具体的な値の入っている位置
集合	集合データの先頭要素の位置
集合族	要素集合の名前表における位置
ユーザー関数	定義域である集合の名前表における位置



図—4　名前表の一例

NUMERICAL MM
SET ISET, JSET
FAMILT FM
FUNCTIŌN FNC (ISET)=JSET

の処理後における名前表の例である。



図—5 データの内部表現

3-3 データの内部表現

値リストはプログラム中に現われる定数の登録の外は，実行 phase に入ってデータとして入力されるときに作られる。各タイプのデータは図-5に示すような構造で作られる。特に集合族データの場合は，その要素集合へのポインターが値となる。

3-4 解読実行 phase

これは翻訳 phase で生成されたオブジェクトを解読し，対応するルーチンへ分枝して実際の計算処理を行なう phase である。phase 全体はソフト的に作られた一種の仮想計算機になっている。今の場合この仮想計算機は 20 個の PL/I の手続き (procedure) から作られている。その主なものの手続き名とその内容を表-3 に挙げておく。また全体の流れ図を示したのが図-1-(2)である。その主な点について説明する。

表—3 解読実行 phase の手続

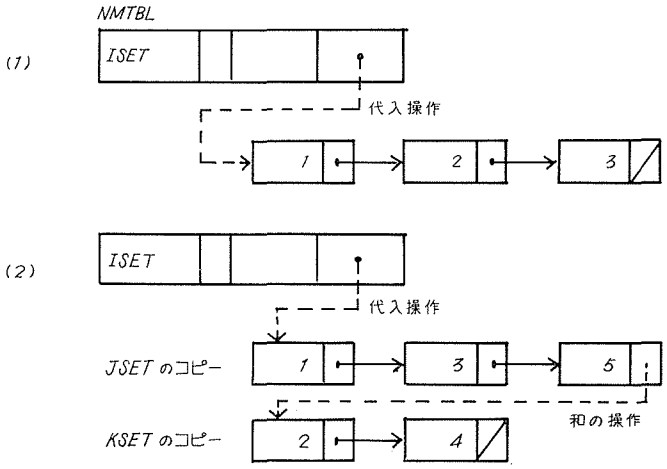
手続名	内容
NMCOMP	数値タイプの比較演算
SETCOMP	集合タイプの比較演算
FMCOMP	集合族タイプの比較演算
IOEXCT	入出力処理
EQTRANS	データ転送
ARITH	数タイプ四則演算
SETBOOL	集合の和，差，共通計算
FMBOOL	集合族の和，差，共通計算
EQBLDFT	標準関数パッケージ
EQUSRFT	ユーザー関数の実行
SETELM	ある要素が指示された集合に入るか否かのチェック
SETCOPY	集合データの複写
RTRSEEK	各種リスト中のある項目の探索
SETOUT	集合データの出力
SETRSV	集合データの登録
FMRSV	集合族データの登録

(1) 定義未定義のチェック： これは IF 文，代入文，WRITE 文の実行に先立って行なうものである。このチェックは名前表の値リストポインターに PL/I のポインター変数を用いているために必要になる。このポインター変数はハード的な番地 (memory addres) と密接に関連しているため，若しこの変数の値が未定義のまま使用するとプログラムを破壊したり，読み出しあるいは書き込み侵害を起す。

(2) この演算処理の大半はリスト (今の場合は linked list[4]) の処理である。演算の種類によって，リストを破壊するものとそうでないものに分かれる。前者の場合は必ずリストを複写してから処理に入る。場合によってはこの複写処理のために不要になるリストも出てくるが，今の場合この不要リストの再生即ちゴミ集め (garbage collection) は行なっていない。

(3) 代入文の処理について集合タイプを例にとり簡単に説明する。JSET, KSET は定義済みとする。

(i) 実行文「ISET=(1, 2, 3)」に対しては，名前表における ISET の値リストポインターへ集合定数 (1, 2, 3) の位置を示す値を入れ，同時に定義未定義サインを 1 にする (この動作を代入と言う。図-6-(1))。



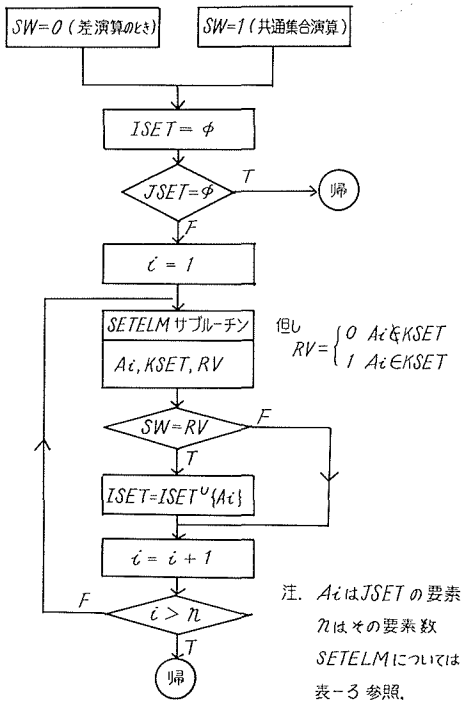
図—6　集合の代入と和の操作

(ii) 実行文「ISET＝JSET＋KSET」に対しては，イ．JSET, KSET の値すなわち集合データの複写，ロ．2つの複写のリンク，ハ．このリンクされたデータを ISET へ代入する (図-6-(2))。この和の計算では重複している要素があっても消去はしない。この簡約化のためには標準関数 REDUCE を使えばよい。

(iii) 実行文「ISET＝JSET－KSET」と「ISET＝JSTE＊KSET」はスウィッチを設けて一つの手続きで実現している。図-7 はその流れ図である。

集合族演算はこの集合演算を使って行なっているが説明は省略する。

(4) 集合(族)等の各種比較演算は今の場合要素間の総当りの比較を行なっている（データ数が大きくなると SORT/MERGING などで使われている手法が必要になるであろう)。



図—7　集合の差，共通のフローチャート

4. 計　算　例

SETL-1 で書かれたプログラム例とその計算結果をあげておく。なおこのプログラムの翻訳および実行には約 12 秒かかっている。

5. 今後の問題

直積集合については触れて来なかったが，今回作成した SETLPROC-1 でもある程度扱えるが，他のタイプのデータと有機的に継がっていない。

今後次の様な点について機能拡張を計りたい。

(1) タイプの拡大——例えば集合族から更に集合を作る等。

```
      ---- SET PROCESSOR SOURCE PROGRAM LIST ----
NO.          STATEMENT
  1          C   EXAMPLE
  2              NUMERICAL MM,ICNT
  3              SET IST
  4              SET JST,KST,IST1,IST2,IST3
  5              FAMILY JFM
  6              WRITE /,'  *** RESULT ***  ',/
  7              WRITE /,/,'   ------   ',/
  8              JST=(1,2,3,4,5,6)
  9              READ KST
 10              IST1=JST+KST
 11              IST2=JST-KST
 12              IST3=JST*KST
 13              WRITE  'JST=',JST,'KST=',KST
 14              WRITE  '+...',IST1,'-...',IST2,'*...',IST3
 15              WRITE /,/,'   ------',/
 16              READ JFM
 17              WRITE 'JFM=',JFM
 18              MM=SHARP(JFM)
 19              ICNT=1
 20           10 IST=CHOOSE(JFM,ICNT)
 21              WRITE IST
 22              ICNT=ICNT+1
 23              IF(ICNT,>,MM) 20
 24              JUMP 10
 25           20 STOP
 26              END

  *** RESULT ***


   ------
JST=(    1 ,    2 ,    3 ,    4 ,    5 ,    6 )
KST=(    3 ,    4 ,    7 ,    8 ,    9 )
+...(    1 ,    2 ,    3 ,    4 ,    5 ,    6 ,    3 ,    4 ,    7 ,    8 ,    9 )
-...(    1 ,    2 ,    5 ,    6 )
*...(    3 ,    4 )


   ------
JFM=( (   10 ,   20 ,   30 ,   40 ) , (   15 ,   25 ,   35 ) , (  100 ,  200 ,  300 ,  400 ) )
(   10 ,   20 ,   30 ,   40 )
(   15 ,   25 ,   35 )
(  100 ,  200 ,  300 ,  400 )
```

図—8

(2) 配列名が使えること。

(3) 算術代入文を多項の演算にすること。

(4) 入出力に自由度を持たせること。

(5) 論理変数，論理式が扱えること。

(6) サブルーチンが使えること。

(7) 標準関数の追加。

(8) FORTRAN, PL/I 等の言語に埋め込むかあるいは結合が可能であるようにすること。

(9) 処理系としては，(1)～(8) までに対する処理と，前にも一寸触れた不要領域の再生をすること。

6. 結　　語

本報告書では，今回作成した言語とその処理について述べたが，これは筆者の意図するものの第1版で，5. で述べた点が解決してはじめて応用の広い実用的言語になると思う。今回の作成には北大大型計算機センター FACOM 230-60 の PL/I 言語を用いた。筆者の PL/I 言語の経験不足のため使いきれなかった機能もあるが，両 phase でカード約 2,300 枚になりそのコンパイルおよび結合編集に約5分を要する。

最後に，日頃御指導していただいている電気工学科加地教授ならびに精密工学科小山助教授に感謝します。また討論に参加してくれた研究室の方々と，プログラム作成で相談にのってくれた計算センターの長田君に感謝します。なお本システム作成は北大大型計算機センター FACŌM 230-60 を用いた。センターの職員の方にお礼申しあげます。

参 考 文 献

1) J. McCarthy: LISP 1.5 Programmer's Manual, MIT, Press (1962).
2) FACŌM 230-60 PL/I 文法編, 使用手引書.
3) H. W. Lawson Jr: PL/I List Processing, CACM Vol. 10, No. 6 (1967).
4) D. E. Knuth: The Art of Computer Programming, Vol. 1, Addison-Wesley (1969).
5) Mark B. Wells: Elements of Combinatorial Computing. Pergamon Press (1971).

** MADCAP という言語に集合演算機能が入っていることを知った (5) (校正追記).