FORTRAN に基づく記号処理システム (COSMOS-2)*

吉田雄二** 福村晃夫**

Abstract

We have developed the symbol manipulation system (COSMOS-2) based on FORTRAN. This system differs from most existing systems (e.g. SNOBOL) in the sense that the language is the extension of FORTRAN and few additional knowledges is needed beyond FORTRAN. The whole system is also implemented by FORTRAN. Symbol strings are represented in the storage by one directional lists, and, for their internal manipulation, some special techniques are employed. By these techniques STRING type functions defined by programmers and subprograms with STRING type arguments are enabled to be used, and also it becomes unnecessary to provide the routine of the garbage collection.

1. まえがき

記号処理システムは，電子計算機のための基礎的なソフトウエアとして，早くから注目され研究開発が進められてきた．これまでに開発された多くのシステムは，種々の興味から，専用の言語によるプログラムの記述を必要としたり，リスト処理システムを作製してその機能の一部を用いたりするようになっている．

一方，電子計算機の応用分野においては，記号処理の技法を用いることによってプログラムの記述や処理が容易になると思われる問題が少なくない．このような問題に対して上に述べたような処理システムを利用することは，単に処理系の製作にかなりの労力を要するということのみならず，利用者に種々の新しい知識を要求したり，記号処理以外の部分の手続きの記述や処理のうえでの能率低下をしいることが多い．

このような理由から，既存のよく知られた言語に，その言語の構成を乱さない範囲で記号処理能力をもたせるようにすることは，問題の一つの有効な解決方法であると思われる．

本論文では，このような観点から構成された，FORTRAN に基づく記号処理システム COSMOS-2 (COmmon String Manipulation Oriented System-2) について論ずる．このシステムは，(1) FORTRAN に，記号処理のための種々の文，演算，標準関数を追加して得られた言語，(2) この言語で書かれたプログラムを FORTRAN のプログラムに翻訳するための翻訳プログラム，(3) 翻訳されてできた FORTRAN プログラムの実行に必要な種々のユーティリティ・ルーチン，の三つから構成されている．

以下，**2.** において COSMOS-2 の言語について，**3.** においてその翻訳について述べる．**4.** では，実行時における記号列の処理について述べ，最後に，**5.** で COSMOS-2 を使用した簡単な例について述べる．

2. 言語の構成

COSMOS-2 の記述言語は，**1.** で述べた理由から，FORTRAN を拡張した言語になっている．したがって，構文は FORTRAN の構文と同じである．従来の FORTRAN に，新たにデータの型 STRING が追加され，この型のデータの処理のために，数種の文，演算子，標準関数が追加されている．以下では，FORTRAN に新たに追加された部分を中心に述べる．

2.1 COSMOS-2 における記号列と仮想文字

COSMOS-2 により処理される記号列は，外部表現をもつ記号と，外部表現をもたない記号の 2 種類の記号より構成される．前者は，FORTRAN 記述用の 48 文字セット（通貨記号，引用符を含む）からなる．これらの文字（引用符を除く）のみからなる記号列は，プログラム上では，記号列を引用符で囲った形で表わされる．

後者の，外部表現をもたない記号は，COSMOS-2 の大きな特長の一つである．この種の記号をここでは

* Symbol Manipulation System based on FORTRAN (COSMOS-2), by Yuuji Yoshida and Teruo Fukumura (Nagoya University)

** 名古屋大学工学部

仮想文字と呼ぶ.

仮想文字は，単に文字番号と呼ばれる番号のみをもち，その表わす意味はプログラマーにより自由に定められる．仮想文字は外部表現をもたないので，プログラマーがそれをプログラム上で記述する場合には，やや間接的な形式になる.

外部表現をもつ記号列についてはその有用性はいうまでもない．仮想文字は，プログラマーが特定の記号列を1個の文字と同等に扱いたい場合，あるいは，非常に数の多い文字セットを用いた記号列を扱いたい場合に有用である．図 5.1 (a) のステートメント,

/ALPH=NTOCH(101)

などはその使用例である (**2.4** の (5) 参照).

2.2 演算子の追加

COSMOS-2 では，記号列のデータを対象とする種々の演算のうち，しばしば用いられて，しかも，その内容が比較的単純な演算については，演算記号が定められている．これらの演算記号を用いることにより，記号処理のための演算を式の形で記述することができる．これらは，並記 ($ または¥) と比較 (.SEQ., .SNE., .SLT., .SLE., .SGT., .SGE. の6種) である．比較の演算子 (ストリング関係演算子) は，二つの記号列の辞書的な比較を行ない，その結果を論理値で与える.

これらの演算子を用いて表わされる式は，FORTRAN における式とほぼ同じ扱いを受ける．また，それらを，他の式と組み合わせて新しい式を構成してもよい.

これらの演算子の優先順位は，比較については，FORTRAN における関係演算子と同じ順位とし，並記は，算術演算子より低く，関係演算子よりは高い順位とする．並記，比較を用いた例が図 5.1 (a) のステートメント,

/ASEQ=ALPH¥ALPH

などに示されている.

2.3 ステートメントの追加

(1) 型宣言文

プログラム上で，記号列を値とする変数名，配列名(以下では記号型変数名，配列名と呼ぶ)を用いる場合に，それらをあらかじめ宣言するために次のステートメントが設けられている.

STRING list

list は名前，または，寸法つき名前の並びである．この文の機能は，FORTRAN におけるいくつかの型宣言文と同じで，プログラム単位内において用いられる記号型の名前は，すべてこの型宣言文によってあらかじめ宣言されなければならない.

(2) パターン・マッチングのためのステートメント

パターン・マッチングは記号処理における最も重要な処理の一つである．したがって，記号処理言語では，それがどの程度直接的に表現できるかがその言語の使いやすさの一つの尺度となりうる．COSMOS-2 では，パターン・マッチングのための機能を直接的に与えるものとして，以下にあげる MATCH ステートメントがある．このステートメントは SNOBOL[1] のステートメントの記法を参考にして構成されている.

パターン・マッチングの機能については，現システムにもたせる能力の程度と言語がFORTRAN の拡張であることなどを考慮して，最も単純な，内容の固定した記号列同志のマッチングのみを扱うこととした.

MATCH ステートメントの形は次のようである.

MATCH (S_1, S_2), n (1)

MATCH (S_1, $S_2=S_3$), n (2)

ここに，S_1, S_2, S_3 は記号列を値とする式，n は文番号である．この文を実行すると，S_1 が S_2 を部分列として含むか否かを調べ，含む場合には，((2) のステートメントでは，その部分列を S_3 で置き替えたのち) 次のステートメントに進む．S_1 が S_2 を含まない場合には，S_1 の内容はそのままにして，文番号 n の文へ飛びこす．MATCH 文の使用例は図 5.1 (a) に示されている．たとえば，同図文番号 40 のステートメントは，記号列 STR1 が記号列 DIGIT を含めば，STR1 は記号系列，'.DIGIT.' に置き替えられることを示す.

(3) 記号型データの入出力文

記号型のデータは，特殊な形で内部表現されるため，その入出力は，特殊なステートメントにより，他の型のデータの入出力とは区別されなければならない．このため，COSMOS-2 では，記号型データの入出力のためのステートメントとして，次の二つのステートメントを設けている.

SREAD list

SWRITE list

list は記号型の変数名，配列要素，または配列名の並びである.

記号列データの入出力の例は図 5.3 に示されている.

(4) 記号列データの管理に関するステートメント

記号データは，**4.**で述べるように，定められた領域のなかに，リストの形で内部表現される．この領域，および記号列を値とする変数に対しては，特別な管理が必要となる．COSMOS-2 トランスレータは記号処理に関連するステートメントを対象として翻訳作業を行ない，FORTRAN ステートメントについては，なにも処理を行なわずにそのまま出力する．このため，これらの管理については，特別なステートメントをいくつか設けて，それらを通じて，実行時に，プログラマーの指示をあおぐ形になっている．しかしながら，COSMOS-2 トランスレータが，FORTRAN ステートメントを含めてソース・プログラム全体を処理するようにすれば，これらのステートメントはいずれも不要となる．

以下に，これらのステートメントを列記する．

(1) INITIAL

実行時における記号処理のための初期設定を行なう．COSMOS-2 を用いるプログラマーは，すべての記号処理の実行以前にこの文が実行されるようにプログラムしなければならない．

(2) STRSET list

記号型変数の初期設定を行なう．list には，そのプログラム単位内で用いられる記号型変数名，または配列名を書く．ただし，記号型の COMMON 変数は，実際に用いられるよりまえに，どれか一つのプログラム単位でこのステートメントにより初期設定する．

(3) ERASE list

list に書かれた記号型変数，配列を初期設定以前の状態にする．

副プログラム内で使用される仮引数を除く局所的な記号型変数は，副プログラムから主プログラムにもどるまえに，必ずこの文により消去されなければならない．

(4) KEEP list

副プログラムの仮引数が記号型である場合に，その値が副プログラムの実行終了まで保存されるための処置を指示する．list には記号型の仮引数となっている変数名，配列名を書く．なお，配列については，STRING ステートメントにおいて指定された寸法に従って KEEP 動作を行なうので，仮引数である配列の寸法の指定には注意を要する（実際に使われている寸法と一致しなければならない）．

(5) RELEASE list

KEEP ステートメントにより値の保護を指定された仮引数について，その保護を解消する（このとき，実引数が記号型の式の値であるときにはその値が消去される）．KEEP により保護された変数がある場合には副プログラムより主プログラムにもどるまえに，必ず RELEASE 指定をしなければならない．

(6) WORKSET list

この文は，おもに関数値に対する記号型の処理を指示するのに用いられ，list で指定された記号型変数の値が一度演算に使われると，自動的に消去されるよう指示をする．関数副プログラムを書く場合，関数の値が計算されて，副プログラムより脱出する直前に，この文により関数値に対して 'WORK' のセットをする (**4.1** 参照)．記号型関数副プログラムの形式は，たとえば，図 5.1(b) のようになる．

2.4 標準関数の追加

COSMOS-2 では，上記の記号処理のための演算子やステートメントのほかに，記号型データを対象とするいくつかの関数が標準関数の形でプログラマーに提供される．これらの関数を利用することにより，記号処理をより能率的に記述することが可能となる．これらには次のような関数がある．

(1) LENGTH (S)

記号列 S の長さを与える整数型関数．

(2) IELEM (S, N)

記号列 S の第 N 番目の文字1字よりなる記号列を与える記号型関数．

(3) IPART (S, M, N)

記号列 S の第 M 文字目より第 N 文字目までの部分記号列を与える記号型関数．

(4) ICHTON (S)

1文字よりなる記号列 S について，その文字の文字番号を与える整数型関数．

(5) NTOCH (N)

文字番号 N の文字1字からなる文字列を与える記号型関数．

3. 翻訳プログラム

COSMOS-2 翻訳プログラムは，COSMOS-2 の言語で書かれたプログラムを FORTRAN プログラムに

翻訳する．全体が FORTRAN で作成されていて，約1200ステートメントよりなっている．今回作成されたシステムでは，翻訳作業を簡単にするため，翻訳を必要とする文には，その先頭に / (スラッシュ) をつけることになっている．しかし，これは本質的には不要である．

COSMOS-2 トランスレータは，一つのソース・プログラムをプログラム単位ごとに翻訳する．各プログラム単位に対応するオブジェクト・プログラムは，そのつど出力されるが，定数記号列の設定を含む初期設定ルーチンは，すべてのプログラム単位が処理されたのち出力される (図 5.2 (c) 参照)．したがって，COSMOS-2 トランスレータは，一つのソース・プログラムの始まり，各プログラム単位の区切り，一つのソース・プログラムの終了を識別する必要がある．今回作成されたシステムでは，このために簡単な形式のコントロール・カードを使用することにより，トランスレータが上に述べた識別を容易に行なえるようになっている．コントロール・カードの形式についてはここでは省略する．

翻訳の例が図 5.1，図 5.2 に示されている．

4. 記号列の内部表現とその処理

記号処理システムを構成する場合，記号列がどのように内部表現されるかはきわめて重要な問題となる．特に，COSMOS-2 のように既存の言語に記号処理機能を埋め込む場合には，その言語の機能より受ける制約があり，それに対してどのような方策を用いるかは，記号処理システムの能力に大きな影響を及ぼす．この節では，はじめに記号列の内部表現について述べ，それに基づいて，記憶の動的管理，記号型関数の処理，翻訳プログラムとの関連等の問題について述べる．

4.1 記号列の内部表現

COSMOS-2 では，実行時に FORTRAN の整数型配列として比較的大きな領域を用い，このなかに記号列を一次元一方向リストの形で記憶する．以下ではこの領域のことをリスト領域と呼ぶ．

内部表現では記号列は，ポインタ，ヘッダ，リスト要素の三つの部分に分かれる．これらはいずれも1語を用いて表現される．このうち，ポインタは COSMOS-2 ソース・プログラムにおいて記号型として定義された名前 (オブジェクト・プログラムでは同じ寸法の整数型の名前となる) に対応する値としてリスト領域外に記憶されるが，残りの二つはリスト領域のなかに記憶される．これらの三つについて，それぞれの構造を図 4.1 に示す．M, L, P_h, P_s, P_e, S_i, P_i 等のもつ意味は以下のようである．

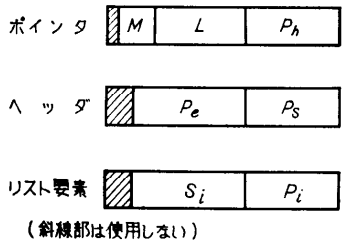

図 4.1 Forms of pointer, header and list element

M: ポインタにより与えられる記号列が，
定数記号列のとき 1，
記号型変数の値のとき 2，
一時的な値のとき3以上という値をとる

L: ポインタが与える記号列の長さ．
$L=0$ のときは空記号列を表わし，対応するヘッダは存在しない

P_h: $L>0$ である記号列について，リスト領域内におけるそのヘッダの位置

P_s: 記号列を表わしているリスト要素の並びのうち，最初の要素の位置

P_e: 記号列を表わしているリスト要素の並びのうち，最後の要素の位置

S_i: 記号列を構成する文字の文字番号

P_i: 次のリスト要素の位置．最後のリスト要素については $P_i=0$

文字の内部表現に文字コードそのものでなく文字番号 S_i を用いた理由は次のようである．一般に FORTRAN において文字を表現するには文字型定数によることになるが，その内部表現は，使用する機種のハードウエアに強く依存することは衆知の事実である．COSMOS-2 のように，FORTRAN を基礎として機種間の互換性を重要視するシステムではこのような事態はできるかぎり避けることが望ましい．また，記号の概念を一般化して仮想文字までを扱う場合には，全体を文字番号という形で統一することが，プログラマーにとってもシステムにとっても有用である．

たとえば，

STR="ABC"

というステートメントの実行により，STR の値は，図 4.2 に示されるような形で内部表現される．

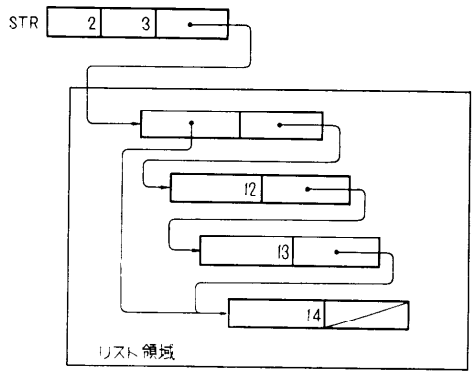

図 4.2 Representation of string 'ABC'

M の値は，次節で述べるように，記号列を実引数とするいくつかのサブ・プログラムの連鎖的な呼び出しの深さに関連する．

4.2 リスト領域の管理

COSMOS-2 では，記号列データの管理についてプログラマーが関与する必要はほとんどなく，システムが自動的に処理する．種々の演算により生ずる不要の記号列をどのように処理するかは，きわめて重要な問題である．ここでは，この問題に対して COSMOS-2 ではどのように解決がなされたかについて述べる．

COSMOS-2 では，次のような二つの原則に基づいて，不要な記号列の検出とその処理を行なう．

（1） 記号型変数に新しい値が代入されるとき，その変数のまえの値は不要となる．

（2） 記号列を値とする演算子，あるいは，関数の値は他の演算子の被演算項，サブ・プログラムの実引数としてただ一度使用されるのみで，使用終了と同時にその値は不要となる．

(2)によれば，記号列を引数にもつサブ・プログラムでは，記号型の引数の値が記号型関数の値である場合には，主プログラムへもどると同時に，その値は不要となるので，消去しなくてはいけない．これが正しく行なえるためには，ある記号型の値が記号型関数の値であるか否かが識別できることが必要となる．このために記号列のポインタに M 部が設けられているのであって，$M=3$ によってその値が記号型関数の値であることが識別される．COSMOS-2 が実行時に自動的に組み込む記号型のデータを対象とするサブ・プログラムには，すべてこの機能が組み込まれている．また，プログラマーが定義するサブ・プログラムについてもこのような処理が正しく行なわれるために，プログラマーは，記号型データに関連するサブ・プログラム内において，KEEP, RELEASE, WORKSET 等のステートメントを適宜用いることが要求されているのである．

KEEP 文は指定された記号型変数のポインタの M 部の値が3以上のとき，その値を1だけ増すことを行なう．これにより，記号型の引数の値は，上に述べた自動消去の対象からはずされることになる．一方，RELEASE 文は，この逆の操作を行なう．すなわち，指定された記号型変数のポインタの M 部の値が4以上のとき，その値を1だけ減じ，その結果が3となったときには，対応するデータを消去する．これらのステートメントをサブ・プログラムの入口と出口で一度ずつ用いることにより，記号型データを引数とするサブ・プログラムの連鎖的な呼び出しが可能となる．

WORKSET 文は，指定された変数のポインタの M 部の値を3にする．すなわち，記号型関数の定義を行なう．

以上の操作により，プログラマーは，記号列を値とする関数を定義し，かつ，サブ・プログラムの実引数として記号型の式を書くことが自由にできるようになる．

一方，このようにして検出された不要の記号列は，そのヘッダに書かれている情報に基づいてシステムが管理する自由リストに組み込まれる．自由リストは，他の記号列と同じく，一次元一方向リストの形になっている．COSMOS-2 では，ある記号型変数の値を他の記号型変数に代入する場合には，前者の値を自由リストより得た新しい領域に書き写すことにより，同一の値を複数個のポインタがさすことはないようにしてある．

以上のように，COSMOS-2 においては，いわゆるゴミ集め (Garbage Collection) は一つ一つの演算のつど比較的簡単な操作により行なわれるようになっていて，リスト処理システムでしばしば問題になる特別なゴミ集めルーチンは存在しない．

4.3 各種ユーティリティ・ルーチン

COSMOS-2 のオブジェクト・プログラムを実行するために，多くのルーチンが実行時に付加される．これらには，上に述べたような基本的な処理を行なうためのシステム・ルーチンと，COSMOS-2 の標準関数とがある．これらは，いずれも，多かれ少なかれ語内のビット処理を必要とするため，アセンブラ語によるのが能率の点で望ましいが，今回作成されたシステム

では，すべて，FORTRAN ステートメントにより行なっている．

ユーティリティ・ルーチンは，全体で約650ステートメントである．システム・ルーチンには，すべてZで始まる名前がつけられているので，プログラマーがソース・プログラムにおいてZで始まる名前を用いることは禁止されている．

5. 応 用 例

この節では，COSMOS-2 の簡単な例題をあげ，入出力の形式，翻訳などの例とする．この例題では読み

```
COSMOS-2 SOURCE PROGRAM LIST

      COMMON ALPH,DIGIT
          /STRING STR1,ALPH,DIGIT,ASEQ,DSEQ
          /INITIAL
          /STMSET STR1,ALPH,DIGIT,ASEQ,DSEQ
          /ALPHMNTOCH(103)
          /DIGIT,TOCH(102)
          /ASEQ,ALPH,ALPH
          /DSEQ,DIGIT,DIGIT
          /SREAD STR1
      WRITE(6,100)
  100 FORMAT(2H1//1H ,12HINPUT STRING/)
          /SWRITE STR1
          /STR1=STRF(STR1)
   10     /MATCH(STR1,DSEQ,DIGIT),20
          GO TO 10
   20     /MATCH(STR1,ASEQ,ALPH),30
          GO TO 20
   30 INDIC=1
          /IF(STR1,SEQ,DIGIT,OR,STR1,SEQ,ALPH) INDIC=0
   40     /MATCH(STR1,DIGIT,*),50
          GO TO 40
   50     /MATCH(STR1,ALPH,*),60
          GO TO 50
   60 WRITE(6,200)
  200 FORMAT(/1H ,14HREDUCED STRING/)
          /SWRITE STR1
          IF(INDIC.EQ.1) GO TO 70
          WRITE(6,300)
  300 FORMAT(1H ,13HSIMPLE STRING)
          STOP
   70 WRITE(6,400)
  400 FORMAT(1H ,15HCOMPOUND STRING)
          STOP
          END
```

(a)

```
COSMOS-2 SOURCE PROGRAM LIST

          FUNCTION STRF(STR)
          COMMON ALPH,DIGIT
              /STRING STRF,STR,ALPH,DIGIT
              /STMSET STRF
              /KEEP STR
          L=LENGTH(STR)
          DO 10 I=2,L
          NS=ICHTONC(ELEM(STR,I))
          IF(NS.LE.12) GO TO 20
              /STMP=STRF,DIGIT
          GO TO 10
       20     /STRF=STRF,ALPH
       10 CONTINUE
              /RELEASE STR
              /WORKSET STRF
          RETURN
          END
```

(b)

図 5.1 Source programs

```
COSMOS-2 OBJECT PROGRAM LIST

      COMMON ALPH,DIGIT
      LOGICAL*1ZMLOG,ZSNE,ZSEQ,ZSGT,ZSGE,ZSLT,ZSLE
      INTEGER*2CONC
      INTEGER STR1,ALPH,DIGIT,ASEQ,DSEQ
      CALL ZINIT
      CALL ZSET(STR1  ,1)
      CALL ZSET(ALPH  ,1)
      CALL ZSET(DIGIT ,1)
      CALL ZSET(ASEQ  ,1)
      CALL ZSET(DSEQ  ,1)
      CALL ZTSF(ALPHMNTOCH(103))
      CALL ZTSF(DIGIT,TOCH(102))
      CALL ZTSF(ASEQ,ZCONC(ALPH,ALPH))
      CALL ZTSF(DSEQ,ZCONC(DIGIT,DIGIT))
      CALL ZSREAD(STR1  ,1)
      WRITE(6,100)
  100 FORMAT(2H1//1H ,12HINPUT STRING/)
      CALL ZWRITE(STR1  ,1)
      CALL ZTSF(STR1,STRF(STR1))
   10 CALL ZMATCH(ZMLOG,STR1,DSEQ,DIGIT)
      IF(.NOT.ZMLOG)GOTO20
      GO TO 10
   20 CALL ZMATCH(ZMLOG,STR1,ASEQ,ALPH)
      IF(.NOT.ZMLOG)GOTO30
      GO TO 20
   30 INDIC=1
      IF(ZSEQ(STR1,DIGIT),OR,ZSEQ(STR1,ALPH))INDIC=0
   40 CALL ZMATCH(ZMLOG,STR1,DIGIT,1074200577)
      IF(.NOT.ZMLOG)GOTO50
      GO TO 40
```

(a) 一部

```
COSMOS-2 OBJECT PROGRAM LIST

          FUNCTION STRF(STR)
          COMMON ALPH,DIGIT
          LOGICAL*1ZMLOG,ZSNE,ZSEQ,ZSGT,ZSGE,ZSLT,ZSLE
          INTEGER*2CONC
          CALL ZINIT
          CALL ZSET(STR  ,1)
          CALL ZSET(ALPH  ,1)
          CALL ZSET(DIGIT ,1)
          CALL ZSET(STMP  ,1)
          L=LENGTH(STR)
          DO 10 I=2,L
          NS=ICHTONC(ELEM(STR,I))
          IF(NS.LE.12) GO TO 20
          CALL ZTSF(STMP,ZCONC(STMP,DIGIT))
          INT=10
       20 CALL ZTSF(STMP,ZCONC(STMP,ALPH))
       10 CONTINUE
          CALL ZREL(STR  ,1)
          CALL ZSET(STMP  ,1)
          RETURN
          END
```

(b)

```
COSMOS-2 OBJECT PROGRAM LIST

      SUBROUTINE ZINIT
      COMMON/ZSYS/LIST(10000),MASK0,MASK1,MASK2,BIT16,IBIT30,
     1LSTAT,LEND,LLENG,MOJI(48),LLSIZE
      DO 40 I=3,LLSIZE-1
      LIST(I-1)=I+1
   40 CONTINUE
      LIST(I-1)=0
      LSTAT=1
      LEND=LLSIZE
      LLENG=LLSIZE
      CALL ZSET(1,'MARK',*,'DIGIT',*,   7)
      CALL ZSET(1,'MARK',*,'ALPH',*,   6)
      RETURN
      END
```

(c)

図 5.2 Object programs

```
012ABC345DEF
 ■■■  ■■■
■0000000000000000000000000000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
1■1■111111111111111111111111111
22■2■22222222222222222222222222
33333■■33333333333333333333333
4444444■4■44444444444444444444
5555555■5■55555555555555555555
6666666666■6666666666666666666
7777777777777777777777777777777
8888888888888888888888888888888
9999999999999999999999999999999
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

(a)

```
INPUT STRING
*012ABC345DEF*
REDUCED STRING
*,DIGIT,,ALPH,,DIGIT,,ALPH,*
COMPOUND STRING
```

(b)

図 5.3 Input/Output Format

込んだ記号列を，数字の並び，英字の並びに区分するとともに，それが一種類のみの文字からなるか否かをチェックしている．図 5.1 には，ソース・プログラムが示されており，これに対するオブジェクト・プログラムが図 5.2 に示されている．実行時における，入力データ，出力結果が図 5.3 に示されている．出力

において，記号列の両端の＊印は記号列の区切りを示す記号で，それ自身は記号列に含まれない．

6. むすび

本論文では，既存のプログラム言語を拡張することにより，あまり特殊な知識を必要としないという条件のもとに作成された記号処理システムについて述べた．このシステムは全体が FORTRAN に基づいて作成されており，他の機種への移行も比較的容易である．

従来のやや専門的な記号処理システムに比べれば，機能の点ではやや欠けるところがないとはいえない．これらのうちで，特に重要な点をいくつかあげて，それぞれについて簡単に検討する．

（1） パターン・マッチングの機能

COSMOS-2 において，パターン・マッチングは MATCH ステートメントにより処理されるが，このステートメントにより処理可能なパターンはきわめてかぎられている．たとえば SNOBOL にみられるような，かなり高級なパターン・マッチング機能を COSMOS-2 にもたせることは不可能ではないが，オブジェクト・プログラム (FORTRAN による) の能率に問題があること，および，現システムのままでも，他のステートメントを合わせ用いることにより，十分能率のよい処理が期待されることなどを考えると，MATCH 文の機能を高めることは必ずしも得策とはいえない．

（2） 再帰的手続きの記述

COSMOS-2 は FORTRAN を基礎とするため，この点についてはまったく実現の余地がないといってよい．記号処理の分野では，再帰的手続きが重要な役割を果たすことはよく知られているので，この点は明らかに欠点といえる．なお，FORTRAN による再帰的手続きを実質的に実現することについては牛島[4] による例があり，そのような技法を用いることは COSMOS-2 でも可能である．

（3） 文字セットと仮想文字

COSMOS-2 で使用可能な文字セットは，初めに述べたように，FORTRAN の 48 文字セットのみであり，他は仮想文字となる．しかし，FORTRAN における H フィールドのように，任意の文字を用いることができるようにするのが望ましい．これについては容易に実現可能である．

以上，COSMOS-2 のいくつかの問題点を指摘した．これらは今後のシステム改善の方向ともくされるものである．

なお，本システムの開発作成にあたっては，京都大学大型計算機センターを利用した (課題番号 4001SB 012, 4001IC048)．また，日本学術振興会計算機学術利用委員会 (ユニコン) を通じ，IBM360/75 の PL/I を利用し，その文字処理機能を参考にした (課題番号 M-019) のでここに明記する．

終わりに，熱心なご指導をいただいた東北大学本多波雄教授に深謝する．また，日ごろ有益な討論を重ねていただいた福村研究室の皆様に謝意を表する．

参考文献

1) R. E. Griswold, et al.: "The SNOBOL 4 Programming Language", Prentice-Hall (1971).
2) 浅井他: "記号処理言語", 総合図書 (1971).
3) IBM System/360 PL/I Reference Manual.
4) 牛島: "FORTRANによる再帰的呼び出し", 情報処理, Vol. 11, No. 2, pp. 101～103 (1970).
5) 吉田, 福村: "FORTRAN を拡張した記号処理システム", 昭45情報処理学会大会, 講演番号 101.

(昭和46年12月1日受 付)