論　文

新しい記号処理概念による SNOBOL インタプリタ*

三上和敬** 豊田順一** 田中幸吉**

Abstract

This paper describes a new string processing method for SNOBOL and its implementation on PDP-8 family computers. Main features of this method are as follow; (1) string operations are completly performed only on "string processing registers", and (2) resulted new strings on above registers can be stored in arbitrary format into "the data area". As a result, the method offers significant advantages over the simplification of string processings and the garbage collection. Based on the method, a SNOBOL interpreter is developed. And some SNOBOL program and its results are shown.

1. まえがき

最近，人工知能問題向きに開発された PLANNER[1] や個々の問題向き記号処理言語の出現など，記号処理に対する関心と要請が高まりつつある．これらの記号処理思想の母体となる代表的言語は，リスト処理の LISP[2]であり，ストリング処理の SNOBOL[3]である．このなかで，各方面において適用余地が広いと考えられる SNOBOL は，大型計算機の1部，たとえば IBM 360/65, CDC 6000 等で実現されているが，ミニコンでは YHP 2100 A 機種を除きほとんど実現されていないのが現状である．現在のミニコンの普及度と今後の発展方向を考えると，ミニコン用記号処理ソフトウェアの開発は，機械翻訳や文献検索等の記号処理分野への応用のほかに，大型計算機への記号処理インタフェースとしても有望視される．

一方，記号処理言語の普及につれて生じる問題は，計算機の記号処理速度が数値演算に比べてはるかに遅く，早晩，記号処理速度の高速化に対する根本的解決が必至なことである．

このような背景にもとづき，筆者らは高速記号処理計算機***に対する基礎研究として，ハードウェア化に適する記号処理構造に注目し，以下の特徴を備えた新記号処理法による SNOBOL インタプリタをミニコン上で開発した．

本記号処理法の特徴は記号処理過程を『記号処理演算』と『データの格納操作』に明確に分離したことである．分離による利点は次の2つである．

1) 記号処理過程で生じる中間結果の表現と，中間結果の廃棄で生じるゴミの処理が容易となる．

2) データの格納法は記号処理演算と独立に設定できるため，使用メモリ最小で能率のよい方式が選べる．

2. 適用計算機と周辺装置

本処理システムの実現に必要な最小機器構成は，ミニコンシステムの経済性の視点から，ミニコンと小容量の補助記憶装置とタイプライタ程度と考え，その他の入出力機器はオプションとして利用する．適用計算機は PDP 8 ファミリミニコン**** で1語 (12 ビット) 構成で 12k 語の主記憶装置を実装している．小容量補助記憶装置はカセットコーダ***** でダイレクトアクセス機能を備えている．その他の入出力機器は **Fig. 1** (次頁参照) に示す．

3. ミニ SNOBOL の言語仕様

ミニ SNOBOL は SNOBOL 4 のミニコンレベル

---

* SNOBOL interpreter based on the new concept of string processings by kazuyoshi MIKAMI, Junichi TOYODA and Kohkichi TANAKA (Faculty of Engineering Science, Osaka University).

** 大阪大学基礎工学部情報工学科

*** 記号処理命令をもつ計算機の意味

**** PDP 8/E 相当の CEC 555 H

***** SYKES COMPU/CORDER 1φφ MAGNETIC TAPE UNIT

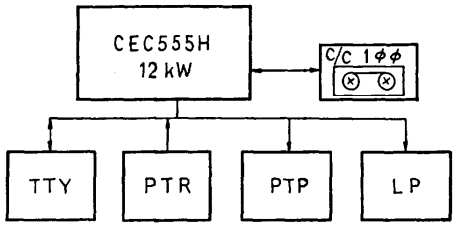


Fig. 1 Hardware organization

のサブセットとする．言語仕様の水準は約 6k 語程度内のインタプリタプログラムサイズ[5]で実現できる記号処理機能を中心に定めた．

（1） ミニ SNOBOL の記号処理機能

i） 記号列の連結操作が取扱える．

ii） 記号列の照合操作が取扱える．

iii） 記号列の選択照合操作が取扱える．

iv） 記号列の置換操作が取扱える．

v） 文字列変数が取扱える．

vi） 固定長文字列変数が取扱える．

vii） バランス文字列変数が取扱える．

viii） 予約語 SYS を接頭語としてもつ文は入出力の機能をもつ．

ix） 予約語 $ を接頭語としてもつ文はインタプリタの照合に関する制御機能をもつ．

また，ミニ SNOBOL で除外した主な事項は次の通り．

i） 数値計算．

ii） 組込み関数．

iii） ユーザが作る関数副プログラム．

（2） ミニ SNOBOL の書式

ミニ SNOBOL の書式は主にプログラムの記述の容易性から，簡潔な表現*の SNOBOL 3[2]の書式に従っている．その他の書式における変更は次の通り，

i） ラベルの終端記号をスペースから，コンマスペースとしたこと．

ii） END ラベルを END 文として取扱うこと．

ミニ SNOBOL のプログラム例を 6 章に示す．

4. 記号処理構想

4.1 ハードウェア化に適する記号処理構造の考察

記号処理が数値計算に比べて複雑になる原因は，数値計算の取扱うデータの 1 単位が 4 バイト程度の『点』とみなすと，記号処理ではこの 1～100 倍程度の『構造をもつデータ』を取扱うことにある．したがって記号処理操作はデータ構造の変換過程といえる．ハードウェア化に適した処理形態は，データ構造の逐次変換方式**ではなく，一括変換方式である．すなわち，**Fig. 2** に示すような，必要なデータのコピーを演算領域に転送し，処理結果をデータ領域に戻す方式である．一括変換の処理形態を実現させるための条件は次の 3 である．

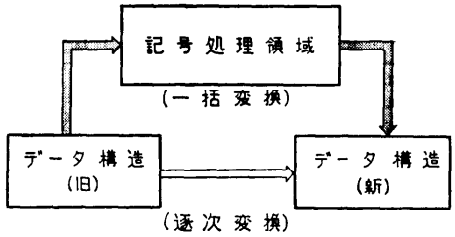


Fig. 2 Transformation of data structure

1) 使用メモリ空間を記号処理演算領域とデータ領域とに明確に分離，または独立させること．

2) 記号処理過程を概念的に記号処理演算過程とデータの更新過程に分離すること．

3) ハードウェア化の対象を演算領域と演算過程とし，具体的な記号処理演算の内容を求めること．

一方，記号処理の一括変換方式はゴミ処理[4]問題を簡単化する利点をもつ．

1) 演算領域で発生するゴミ：中間結果の廃棄で発生するゴミは，発生領域が限定されるため，ゴミ集めよりも簡単なゴミの消去（演算領域をクリアすること）で処理できる．

2) データ領域で発生するゴミ：データの更新で生じるゴミは通常のゴミ集め処理でされる．しかし，1)で大部分のゴミ処理がなされる関係上，2)の処理プログラムがコンパクト化される．

4.2 記号処理レジスタ

記号処理演算領域は『幅』をもつ 5 個の記号処理レジスタで構成される．ただし，ハードウェアレジスタが用意されていないので，ソフトウェアレジスタを代用する．レジスタの個数は SNOBOL ステートメントの一般形の名部の意味にもとづき定めた．レジスタの機能とサイズを **Table 1** (次頁参照)にまとめる．レジスタ内の中間結果や最終結果は，部分記号列表示用

---

* SNOBOL 4 では高度の記号処理をおこなうため個有の関数名を多数定義しており，プログラムの記述面でこれらの関数名の意味に精通する必要がある．

** たとえば，ポインタをたどりながら変換する方式をさす．

Table 1 String processing registers

名 称	使 用 目 的	標準サイズ*
レジスタ I	新記号列生成	192 語
レジスタ II	対象記号列格納	192 語
レジスタ III	パターン生成（比較）	128 語
レジスタ IV	パターン生成（選択）	192 語
レジスタ V	置換記号列生成	192 語

* 本システムではユーザがレジスタサイズを変更できる.

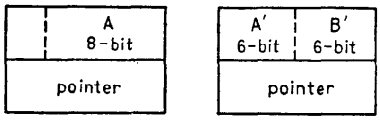


Fig. 3 Examples of List cell on 12-bit word size

の指示子で示される．指示子の値はレジスタ内の絶対番地である．

レジスタ間の記号処理演算は記号処理過程を基本操作に分解し，かつ統合可能な操作をまとめた結果，照合演算と転送演算とする．詳細は **5.2** で述べる．

4.3 データ領域の記号列の格納法と管理法

4.3.1 記号列の格納法

記号列の格納法は演算処理と独立であることから，使用計算機に最適な構造を用いることができる．リスト構造は本システムのように単に記号列を更新する際においても非常に有力な手段となるが，以下の検討事項の結果，本システムでは **Fig. 4** に示すようにデータ領域の連続した番地に記号列を格納する方法を用いた．

（1） 格納データの性質

本システムはデータ領域で演算処理をおこなわないため，記号列を分割して格納する必要がない．

（2） ハードウェアのメモリ構成

PDP 8 系のミニコン上でリスト構造のセルを仮定すると，**Fig. 3** の(a)(b)形式などが考えられる．(a)は1セルが2語構成で格納密度 0.5 文字/1語となり，連続格納方式の1文字/1語に比べ明らかに不利となる．(b)は1セルが2語構成でパック* により平均1文字/1語で連続格納方式と同等であるが，処理プログラムのサイズを問題にすると，処理が単純な連続格納方式が有利となる．

（3） リスト処理のゴミ処理問題

計算機上でリスト処理を実現する場合，ゴミ集め機構は不可欠の機能である．ゴミ集めの種々の技法は[5]，

* 文字は ASCII コード（8ビット）表現であるが，1語（12 ビット）に2文字を格納するため下位6ビット表現を用いる.

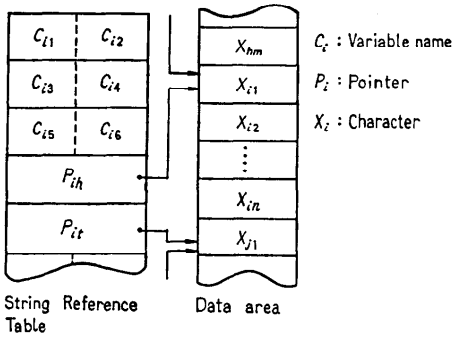


Fig. 4 String reference Structure

リスト構造をたどるための探索用フラグや識別フラグをセルに付加する必要がある．結局，1セルは3語構成となり，リスト処理はメモリ的に不利である．

4.3.2 記号列の管理法

データ領域の記号列は変数名の値として，**Fig. 4** に示す変数名表で管理される．変数名表の1単位は5語で構成される．C_{i1}～C_{i6} は6文字以内の変数名のパック表現形を格納する．P_{ih} はデータ領域の記号列の先頭番地を示し，P_{it} は後尾番地の次の番地を示す．これは記号列長の計算 $(L(X_i)=P_{it}-P_{ih})$ のためである．また空系列 $(L(X_i)=\phi)$ の表示は $P_{ih}=P_{it}\neq\phi$ で，P_{ih}, P_{it} の示す番地は意味のないものとする．

5. SNOBOL インタプリタの構成

5.1 ミニコン上で実現するための留意点

4章の記号処理法を実現する際，適用計算機がミニコンであるため，つぎのような側面からの工夫が必要となる．

（1） システムの融通性

システムはユーザの多様な記号処理要求に答えねばならない．記憶容量の大きい計算機システムでは，ユーザプログラム領域，データ領域，その他の表領域を十分な余裕をもって設定することができるが，ミニコンでは領域の固定がシステム使用上の制約事項となる．この対策として，本システムは各領域の和が一定である条件下で，記号処理の種類や規模に応じて個々の領域の大きさを変更できる柔軟な内部構造をもつ．

（2） オーバレイ構造の使用

オーバレイを使用する際，プログラムの分割は補助記憶装置との入出力時間に最小にする方向で検討すべきである．本インタプリタをオーバレイ構造にすれば個々のサブルーチンが階層構造を取るため，セグメント単位が大になるほか，オーバレイの回数にともなう

時間的損失が大である．また，主記憶上に十分なユーザプログラム領域を占有できないことから，オーバレイの対象はユーザプログラムとする．オーバレイ領域を広く，かつ確実に確保する方策としてインタプリタの処理内容をつぎの2段階に分ける．

i） ユーザプログラムを入力装置からカセットコーダに格納する段階．

ii） カセットコーダからステートメントをオーバレイ領域にスワップインして解釈実行する段階．

ii）の段階ではi）の処理プログラムを必要としないので，i）の処理プログラム領域をオーバレイ領域として利用する．

5.2 SNOBOL インタプリタ

4章の記号処理構造と，5.1 の留意点を実現するため，本インタプリタは **Fig. 5** の処理過程をとる．Fig. 5 において，フェーズ I は初期化 I から初期化 II の直前までとし，フェーズ II は初期化 II から初期化 III の直前までとし，フェーズ III は初期化 III 以後とする．

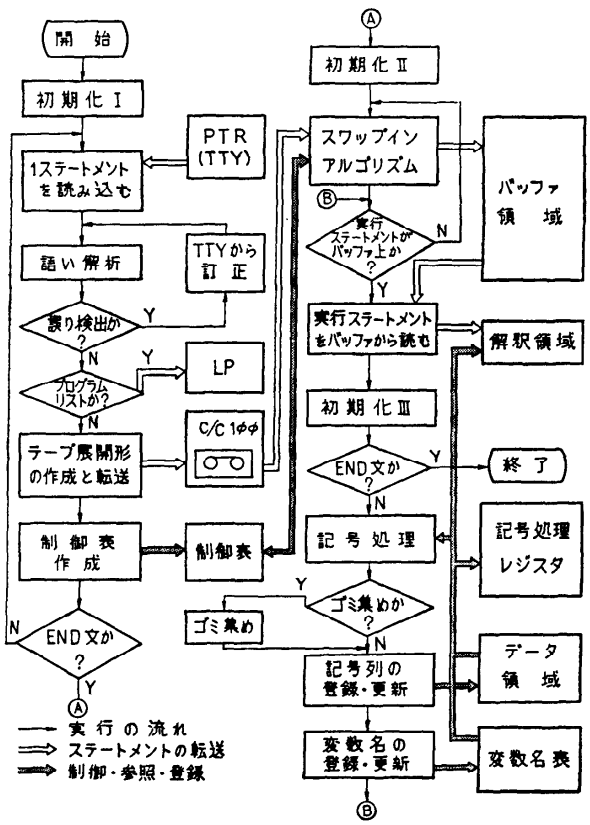


Fig. 5 Simplified diagram of SNOBOL interpreter

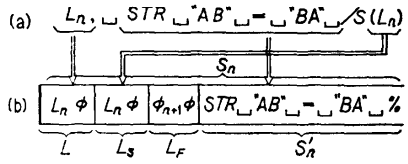


Fig. 6 A cassette-tape format

5.2.1 フェーズ I

フェーズIの主目的はフェーズIIのスワップインアルゴリズムに必要なステートメントのテープ展開形をテープ上に作成することであり，ステートメントのテープアドレスを制御表に登録することである．

テープ展開形は実行時においてつぎに実行すべきステートメントとの完全なリンクのためのラベル部を持つ．リンクはラベル，または文番号*を用いる．**Fig. 6** の例で，S-go-to（以後，略記号 Ls）は指定のラベルを書き込み，F-go-to(略記号 LF) は文番号を書き込む．Lの役割は Fig. 6(a) のような SNOBOL 独特の自己ループをもつステートメントのリンクに用いる．

制御表はフェーズIIにおいてつぎの手続きのために作成される．

i） ステートメントがオンコアであるかの判定．

ii） スワップインアルゴリズムにおいて，レベル2（**5.2.2** 参照）のステートメントのテープアドレス参照．

制御表の1単位はステートメントごとに作成され，**Fig. 7**（次頁参照）の要素をもつ．また，制御表の1単位は主記憶を有効に使用する観点から，ラベル数によって Fig. 7 の3語構成から9語構成となる．Fig. 6(a) の例はラベル数2で Fig. 7(c) を構成し，コアアドレス部以外すべて書き込まれ，F=6 ($11\phi_{(2)}$) の値をもつ．

5.2.2 フェーズ II

スワップインアルゴリズムはスワップインの回数を減少させるため，プログラムの自然な流れ**と，レベル2すなわち，つぎに実行

* 本インタプリタはリンクのため，すべてのステートメントに自動的に文番号を付け，ラベルの無い場合には文番号を代用する．

** プログラム上のつぎのステートメントを実行すること．

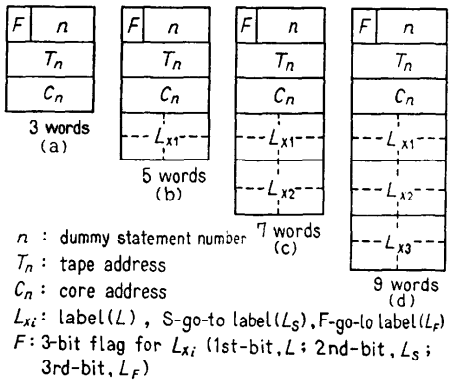


Fig. 7 4 tags associated with one statement

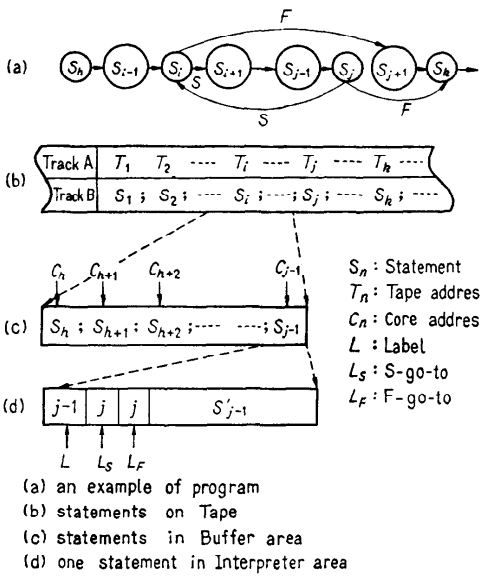


Fig. 8 Transferring Statements during the second phase

すべきステートメントとそのつぎに実行される可能性を持つステートメントまでを調べ，より多くの実行に必要なステートメント群をカセットコーダからバッファ領域へ転送する．

アルゴリズムの概略を **Fig. 8**, **Fig. 9** を用いて説明する．Fig. 8(d)は j−1 番目のステートメント(略記号 S_{j-1}) の実行後，つぎに実行すべきステートメント（略記号 S_E) が S_j であることを示す．制御表の S_j の C_j を調べると Fig. 8(c) により $C_j=\phi$ で，S_j はオンコアでないので，$T_E=T_j$ とおく．Fig. 8(a)は S_j の実行後，S_i と S_k へ飛ぶ可能性を示す．

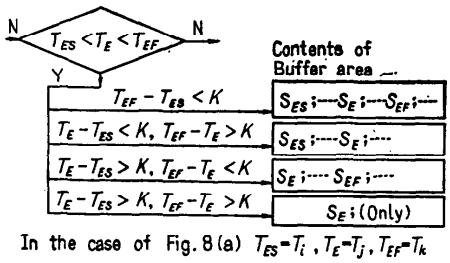


Fig. 9 An example of swap-in flow

ジャンプに関するラベルは制御表の S_j 単位のラベル部に書き込まれているから，制御表を用いて S_i と S_k のテープアドレスを求めることができる．ジャンプに関するラベルをもたない場合は制御表の S_j 単位のつぎの S_{j+1} 単位を調べてテープアドレスを求めることができる．以上により $T_E=T_j$, $T_{ES}=T_i$, $T_{EF}=T_k$ の値が定まる．スワップインアルゴリズムは T_E, T_{ES}, T_{EF} の値の大小関係を用いて8分類しさらに Fig. 9 のように小分類し，適用すべき格納サブルーチンを求める．

ステートメントはバッファ領域に順次格納され，制御表のそれぞれのコアアドレス部にバッファにおけるステートメントの先頭番地が書き込まれる．つぎにバッファ領域から解釈実行領域へ S_j を転送し，S_j の解釈実行準備が完了する．

5.2.3 フェーズ III

解釈実行段階で，中心となる記号処理演算について詳しく述べ，記号列の格納，データ領域のゴミ処理について簡単に述べる．

（1） 記号処理レジスタの照合演算

SNOBOL の記号列照合を便宜的に，3章の ⅱ)を単純照合，ⅲ)を選択照合，ⅴ)を複合照合，ⅵ)を固定長照合，ⅶ)をバランス照合と呼ぶことにする．

個々の照合形式を統一的に取扱うことが，ハードウェア，ソフトウェアの両面から必要であり，以下にすべての照合形式はバランス照合で解決できることを示す．**Fig. 10**, **Fig. 11** (10, 11 は次頁参照) において，レジスタⅡは対象記号列をロードし，レジスタⅢは比較要素をロードする．レジスタⅡの指示子は記号列の幅を示すため，2個の成分をもち，第1成分(H)は記号列の先頭番地を示し，第2成分(T)は後尾番地のつぎの番地を示す．また，レジスタⅢに2個の比較要素がロードされるとき，分離記号を ϕ で，終端記号を $\phi\phi$ で示す．

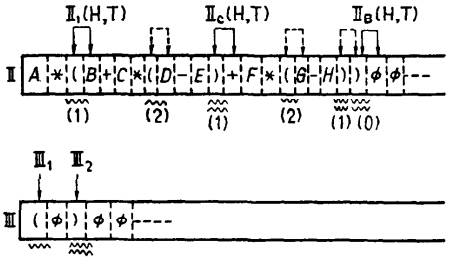


Fig. 10 Two examples of pattern matchings

(a) バランス照合と複合照合

複合照合はバランス照合を用いておこなわれ，そのために意味のある部分記号列を表示するための指示子と1個のカウンタが使用される．Fig. 10 はパターンが "("*S*")" の複合照合と，パターンが "("*(S)*")" のバランス照合の例を示す．

カウンタは初期値が ϕ で，指示子 III_1 の比較要素と照合すれば +1 を加え，指示子 III_2 と照合すれば −1 を加える．カウンタが最初に +1 になるとき指示子 II_1 を付け，最初に −1 を加えられたとき複合照合指示子 II_C を付け，カウンタが ϕ になるときバランス照合指示子 II_B を付ける．Fig. 10 においてカッコ内の数値はカウンタ値である．したがって複合照合はバランス照合をおこない，指示子 $\mathrm{II}_{1(T)}$ と指示子 $\mathrm{II}_{C(H)}$ より，部分記号列の先頭番地 $\mathrm{II}_{1(T)}$ と，部分記号列長 $L=\mathrm{II}_{C(H)}-\mathrm{II}_{1(T)}$ を得ることができる．

(b) 固定長照合

固定長照合は複合照合における部分記号列長Lに指定を加えたものである．(a)の処理によってLを求め，指定された長さと比較することで処理できる．

(c) 単純照合

単純照合は比較要素1個である．(a)の特殊形(指示子 $\mathrm{III}_2=\phi$) として処理され，照合が成功すれば指示子 $\mathrm{II}_1\neq\phi$ で，失敗すれば $\mathrm{II}_1=\phi$ で示される．

(d) 選択照合

選択照合は単純照合の反復と考え，レジスタⅣからⅢへの比較要素の転送を含む．選択照合には，選択記号列中の最左記号列から照合をおこない成功した記号列を合致記号列とする方法と，対象記号列中の最左合致記号列を求める方法がある．方法の選択はインタプリタに対する照合制御文の値でおこなわれる．Fig. 11 は選択照合の例である．前方法では II_1 の示す RED が，後方法では II_1 の値の最小のものを選び READ が合致記号列となる．したがって，選択照合は(a)処理で得た指示子 II_1 の値を保持することで処理できる．

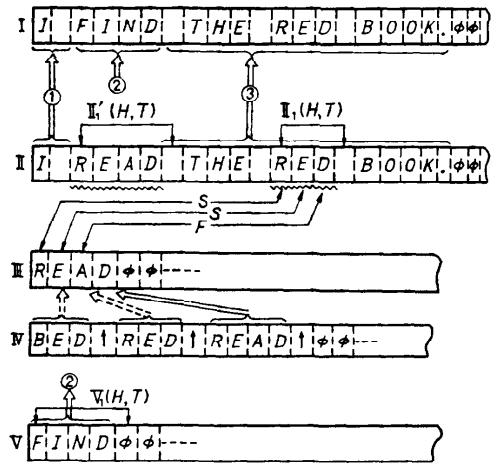


Fig. 11 An example of a new string generation on the string processing registers

(2) 記号処理レジスタの転送演算

転送演算は照合演算の結果として得られた指示子を用いておこなわれる．Fig. 11 はレジスタI内に新記号列が図中の番号順の転送演算によって生成される過程を示す．

(3) データ領域における記号列の登録と更新

記号列の登録更新は **Fig. 12**(次頁参照) を用いて概説する．図中の(a-1)は登録状況を示す．記号列 (X_j) の更新は旧記号列 (X_j) の消去後，レジスタIから新記号列 (X_j') を格納することであり，図中の(a-2)となる．同様に記号列 (X_k) の更新状況を(a-3)に示す．

記号列の登録更新アルゴリズムの特徴は，自然なゴミ処理機能を備えていることである．すなわち，図中で発生したゴミ（空番地）は永久的なゴミではなく，(a-2)のように再使用される可能性をもつゴミである．そのためゴミ集めルーチンを働かせる回数が減る．

(4) データ領域のゴミ処理

データ領域のゴミ集めは，Fig. 12(b-1)，(b-2)に示すように後続の記号列をつめることでおこなわれ，変数名表のポインタを新番地に更新する．

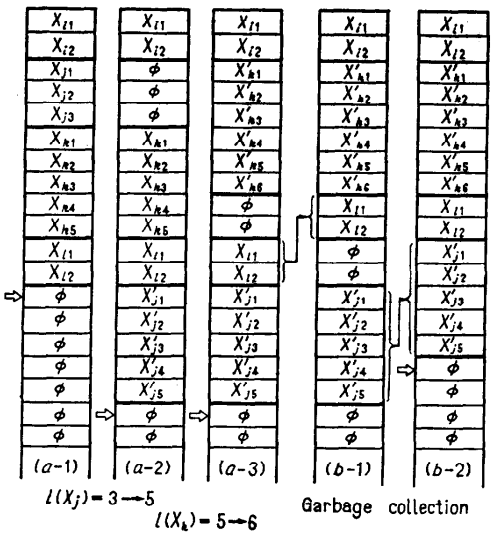


Fig. 12 Storage format and Garbage Collection in Data area

ゴミ集めルーチンの特徴はゴミを見つけるためデータ領域のすべてを調べる必要がないことである．すなわち，データ領域内で使用された最大番地(図中の⇨)を記録し，最大番地の範囲内でゴミを調べる．ゴミ集めの時期はデータ領域にセットされた番地を使用最大番地が越えた時におこなわれる．

6. システムの評価と応用例

(1) 本システムの静的評価はプログラムサイズ，領域サイズを用いておこない，**Table 2** に示す．

(2) 本システムの動的評価は実行時間を用いておこない，SNOBOL プログラム例の処理結果を **Table 3** に示す．プログラムAはループを主体とする**例 2** (次頁参照)のプログラムである．プログラムBは文献

Table 2 Area sizes and sub-program sizes

名　　称	サ　イ　ズ
記号処理レジスタ (合計)	約 0.9k語*
解釈領域 (1ステートメント用)	約 0.1k語*
変 数 名 表	約 0.3k語*(64 個)
個制御表+データ領域	約 2.6k語*
オーバレイ領域 I	約 0.5k語*
インタプリタ (フェーズI) (オーバレイ領域)	約 1.2k語*
インタプリタ (フェーズII+フェーズIII)	約 5.5k語*
リンキングローダ	約 0.9k語*
合　　計	12k語

* 標準システムのサイズ．これらのサイズはユーザによって変更可能である．

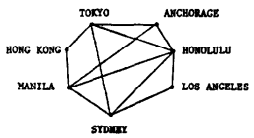
Table 3 Results of the example programs

項　　目	プログラムA	プログラムB
プログラムステップ数 ①**	61	352
フェーズI平均処理時間 (1ステートメント当り) ②	約 1.1 sec (プログラムリスト有り)	約 0.9 sec (プログラムリスト無し)
全実行ステップ数 ③**	1633	333 (5個のデータ処理の場合)
バッファ領域の大きさ (格納ステートメント数) ④	60～70*	12～16
スワップインアルゴリズムの作動回数 ⑤**	1	36
スワップインされた転送ステートメントの総数 ⑥**	61	599
ゴミ処理ルーチンの作動回数 ⑦**	18	1
フェーズII，III平均処理時間 (ステートメント当り) (⑤，⑦時間と出力時間を含む) ⑧	約 310 msec (オンコア処理)	約 1100 msec (カセット稼動時間を含む)

* 標準システムのサイズ．これらのサイズはユーザによって変更可能である．

** これらの項目の数値は，システムアウトプットに明示される．

```
PASS-PROBLEM. K.MIKAMI
GRAPH= TOKYO-ANCHORAGE,TOKYO-HONOLULU,TOKYO-SYDNEY,TOKYO-HONG KO
NG,ANCHORAGE-HONOLULU,ANCHORAGE-MANILA,HONOLULU-LOS ANGELES,HONO
LULU-SYDNEY,HONOLULU-MANILA,LOS ANGELES-SYDNEY,SYDNEY-MANILA,MAN
ILA-HONG KONG,
BEGIN= TOKYO END= MANILA
S-PASS=TOKYO-ANCHORAGE-HONOLULU-LOS ANGELES-SYDNEY-MANILA
S-PASS=TOKYO-ANCHORAGE-HONOLULU-MANILA
S-PASS=TOKYO-ANCHORAGE-HONOLULU-SYDNEY-MANILA
S-PASS=TOKYO-ANCHORAGE-MANILA
S-PASS=TOKYO-HONG KONG-MANILA
S-PASS=TOKYO-SYDNEY-MANILA
S-PASS=TOKYO-SYDNEY-HONOLULU-MANILA
S-PASS=TOKYO-SYDNEY-HONOLULU-ANCHORAGE-MANILA
S-PASS=TOKYO-SYDNEY-LOS ANGELES-HONOLULU-MANILA
S-PASS=TOKYO-SYDNEY-LOS ANGELES-HONOLULU-ANCHORAGE-MANILA
S-PASS=TOKYO-HONOLULU-MANILA
S-PASS=TOKYO-HONOLULU-LOS ANGELES-SYDNEY-MANILA
S-PASS=TOKYO-HONOLULU-ANCHORAGE-MANILA
S-PASS=TOKYO-HONOLULU-SYDNEY-MANILA
I HAVE FOUNDED ALL PASSES.
```



例 1 入力例と結果例

検索システムの主題分析プログラム[6]であり，ほとんどループをもたず，分枝の多いプログラムである．ただし，プログラムBの処理結果は5個の標題を主題分析したものである．

(3) 本システムの簡単な記号処理への応用を**例 1** に示す．例2はグラフ上の始点から終点に至るすべてのパスを求めるプログラムで，記号処理的に解いたものである．例1は例2の入力と出力を示す．実行時間は，始点と終点の入力後，印刷時間も含めて約8分10秒である．

7. むすび

筆者らはハードウェア化に適する記号処理構造に注目し，つぎの特徴を備えた新記号処理法による SNOBOL インタプリンタを開発した．

i) 記号処理過程は記号処理演算とデータの格納

```
CASSETTE SNOBOL INTERPRETER V-747-12
SNOBOL SOURCE PROGRAM LIST          PAGE 0001

0001          SYSLPT "PASS-PROBLEM. K.MIKAMI "
0002    L1,   SYSIN *GRAPH*
0003    L2,   SYSIN *BE* "-" *DE* /F(L2)
0004          SYSLPT "GRAPH= " GRAPH
0005          SYSLPT "BEGIN= " BE " END= " DE
0006          W = GRAPH
0007          PASS = BE
0010          S0 = BE
0011          PDS = "#"
0012          PDA = "#"
0013    A,    W S0 /S(A0)
0014          SYSOUT S0 "??" /(L2)
0015    A0,   W *P1* "," *PL1* = PL1
0016          P1 S0 /S(A1)
0017          W = PL1 P1 "," /(A0)
0020    A1,   PL1 S0 /F(A3)
0021    A2,   PL1 *X1* "," *PL1*
0022          X1 S0 /F(A2)
0023          X2 = X1
0024          X2 *V1* "-" *V2*
0025          V1 S0 /S(A8)
0026          X2 = V2 "-" V1
0027    A8,   PDA = X2 "," PDA
0030          W X1 "," = X2 "," /(A1)
0031    A3,   P1 *V1* "-" *V2*
0032          V1 S0 /F(A4)
0033          G0 = V2 /(A5)
0034    A4,   G0 = V1
0035    A5,   PDS = P1 "," PDS
0036          W P1 "," =
0037          P1 *TV1* "-" *TV2*
0040          P1T = TV2 "-" TV1
0041          W P1T "," =
0042          PASS BE "-" =
0043          PASS G0 /S(W0)
0044          PASS = G0 "-" PASS
0045          DE G0 /S(W3)
0046          S0 = G0
0047          W S0 /S(A0)F(B0)
0050    W0,   PASS = G0 "-" PASS /(B0)
0051    W3,   ANS =
0052          PDW = PASS "-"
0053    W4,   PDW *X1* "-" *PDW* /F(W5)
0054          ANS = "-" X1 ANS /(W4)
0055    W5,   ANS "-" =
0056          SYSLPT "S-PASS=" ANS
0057    B0,   PASS *S0* "-" *PASS*
0060    B,    PASS *S0* "-" *PASS* /S(B3)
0061          S0 = BE
0062    B3,   PDS *X1* "," *PDS*
0063          W X1 /S(B4)
0064          W = W X1 ","
0065    B4,   PDA *PA* "," *PDA* /F(EN)
0066          PA *V1* "-" *V2*
0067          V1 S0 /S(C)
0070          PDA = PA "," PDA /(B)
0071    C,    P1 = PA
0072          PASS = S0 "-" PASS
0073          G0 = V2 /(A5)
0074    EN,   SYSLPT "I HAVE FOUNDED ALL PASSES."
0075          END
```

例 2　プログラム例

に明確に分離されていること.

ii) 記号処理演算は複雑な記号処理操作を整理統合した結果，記号処理レジスタのバランス照合と転送の2操作の組合せでおこなわれる.

iii) データの格納法は使用計算機に適合した方法を選択できる.

本システムは PDP-8 ファミリミニコン上で実現されているが，単にミニコンの適用分野を拡大する以上の意義をもっており，高速記号処理計算機への手がかりとなる点で，一応の目的を達せられた．また，本システムは文献検索システムの開発[6]等に利用されている.

今後の重要な課題はハードウェアの記号処理レジスタの実現である．また，本システムは経済性の観点からカセットコーダを使用した点で，実行時間に改善の余地がある.

最後に，有益な助言や討論をいただいた研究室の水本雅晴氏，山本順人氏に感謝します．また，プログラム作成に御協力いただいた当時本学学生（現，三和銀行）の神田晴郷氏に感謝します.

参 考 文 献

1) Hewitt, C. Description and theoretical analysis (using schemas) of PLNNER: A language for proving theorems and manipulating models in a robot. MIT, 1971.
2) R. W. Hamming, et al.: "Programming Systems and Languages", pp. 419～ 511. Mc Graw-Hill (1968)
3) R. E. Griswold, et al.: "The SNOBOL 4 Programming Language",Prentice-Hall(1971)
4) H. Schorr, et al.: "An Efficient Machine-Independent procedure for Garbage Collection in Various List Stractures", C. ACM vol., 10, No., 8, Aug. (1967)
5) 三上，豊田，田中："ミニコンにおける SNOBOL コンパイラの試作" 電子通信学会研究資 AL-40, PRL-38 (1972-07)
6) 馬野，三上，豊田，田中："SNOBOL による文献標題検索システムの作製" 電気関係学会関支連合大会 G 8-12 (1974-11)

（昭和49年10月17日受付）
（昭和49年12月16日再受付）