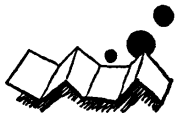
解　説



文字列処理とアーキテクチャ†

山本昌弘†† 梅村　護††
小長谷明彦†† 横田　実††

1. はじめに

記号処理の応用分野において，文字列データはリスト構造データとともに非常に重要である．文字列データとして，計算機のコンパイラが処理するような高級言語で書かれたソースプログラム，テキストエディタが取り扱う原稿テキスト，情報検索システムに蓄積されたテキスト情報，などが代表的なものといえる．

文字列データを高速に処理するハードウェアやアーキテクチャの研究は，数値データの高速処理を目的とした研究開発ほど大々的に行われてきていないが，一部の研究者により古くから熱心に進められてきた．一方，最近の OA，知識情報処理，などの新しい応用分野の出現，半導体技術の進歩に伴う VLSI アーキテクチャの可能性，などによって，文字列処理の応用が拡大し，これを高速に実行するハードウェアやアーキテクチャの研究が重要視されつつある．

このような背景から，本稿においては，文字列処理について概説し，これまでに提案されている文字列処理用のハードウェア，アーキテクチャについて解説する．第2章では，本分野の代表的高級言語である SNOBOL 4 をベースに文字列処理機能，第3章では，文字列照合用専用ハードウェア，第4章では，マシンアーキテクチャ，第5章では，文字列処理の応用，について述べる．

2. 文字列処理言語

文字列を直接扱う言語として最も完成された形式を整え，広く利用されている SNOBOL は，ベル電話研究所の Farber, Griswold, Polonsky らが 1962 年に開発に着手した言語である[1]．その後改良，拡張が行われ，1971 年に SNOBOL 4 プログラミング言語第2版[2]が出版されて開発作業を完了し，現在に至っている[3),4)]．本章では SNOBOL 4 を中心に，この言語に備えられている文字列操作機能について概観する．

2.1 SNOBOL 4 の特徴

SNOBOL 4 は文字列を基本データとし，その内容の検査や置換をパターンと呼ぶデータ型を用いた照合操作によって実現する言語である．基本となる文は代入文，パターン照合文，および置換文の3種で，1文内の出現順と記号 (=, :) によって要素が区別され，構成要素の組み合わせで文の種類が決まる．

パターンは文字列，文字セット，文字列の長さおよび基本関数 (primitive function) の合成によって構成される構造であり，非常に複雑な記述が可能である．またパターンには変数や遅延評価式 (unevaluated expression : 後述) を含むことができ，動的なパターンの構成を行うことができる．さらに使用者が自由に関数を作る機能も備えられていて，非常に柔軟で強力なパターン照合操作を実現できる．

SNOBOL 4 プログラムは実行時に翻訳を行うインタプリタ方式を採用している．このため使用者は対話的に文字列処理を行うことができ，変数に割り当てられた値のトレースや実行時の関数再定義等を自由に指示できる．入出力は特定の基本関数に値が割り当てられた時に動作する連想方式が採られている．

2.2 パターン照合

SNOBOL 4 のパターン照合は次の二種類の文型によって発生する．

label subject pattern : goto　　(1)
label subject pattern=object : goto　　(2)

(1)はパターン照合文で，対象文字列 (subject) とパターン (pattern) で記述される構造との照合がスキャナと呼ぶ手続によって行われる．照合の結果は成功または失敗で，いずれかを条件とする行先指定 (goto) によって次に実行すべき文を選択できる．(2)は置換文であり，パターン照合の結果，対象文字列とパターンとの一致部分が置換文字列 (object) によって置き換

† Character String Processing and Machine Architecture by Masahiro YAMAMOTO, Mamoru UMEMURA, Akihiko KONAGAYA and Minoru YOKOTA (C & C Systems Research Laboratories Nippon Electric Co. Ltd.).

†† 日本電気(株) C & C システム研究所

えられる．

パターンはスキャナに対し，対象文字列の操作手順を指示するプログラムのように働く．スキャナは対象文字列を左から右へ走査しつつパターンとの照合を試み，一致を検出するか，または文字列を右端まで走査し尽くすと手続を終了する．パターンに選択肢(alternative) が含まれているときには，先に書かれている選択肢によるパターン照合の結果が失敗に終わると後戻りが起こって次の選択肢による照合が試みられる．

スキャナの動作は SNOBOL 4 の機能を理解する上できわめて重要であるため，以下に文献[2] に示されている具体例* を紹介する．

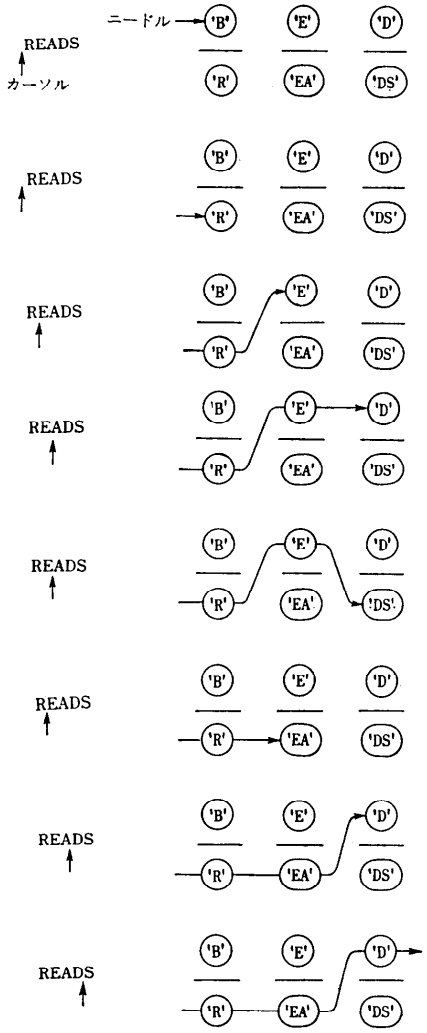


図-1　SNOBOL 4 スキャナの動作（ビーズ図）

* 文献[2] pp. 26～29

```
BR=('B'|'R') ('E'|'EA') ('D'|'DS')        (3)
'READS' BR                                 (4)
```

(3)は代入文で，( )でかこまれる3つの文字列が結合 (concatenate) されたパターン構造に BR という名前を与える文である．( )内は記号 | で区切られる選択肢を含み，結果的に BR は，BED, BEDS, BEAD, BEADS, RED, REDS, READ, READS の8つの文字列から成る和集合を構成する．(4)はパターン照合文で，対象文字列 READS と上記8つの文字列パターンとの照合を指示している．

スキャナは対象文字列の参照位置を示すカーソルとパターン構造の参照位置を示すニードルとを用いて照合操作を行う．**図-1** はこの2つのポインタを用いたスキャナの操作手順を示すビーズ図で，○で囲まれるパターン要素をニードルによって動的にパターンを構成しながら対象文字列との一致を検出する様子を示している．

2.3　基本関数と遅延評価式

SNOBOL 4 にはパターン照合時に動的にパターンを創り出す基本関数が用意されている．SPAN および BREAK には，アーギュメントとして文字セットが与えられる．SPAN はアーギュメントに含まれる文字が連続して現れる限り一致するパターンを構成し，BREAK はアーギュメントに含まれる文字が出現する直前までの文字列すべてに一致するパターンを構成する．LEN はアーギュメントで指定される長さの任意の文字列に一致する．これらを用いて強力なパターン照合機能が実現されている．

遅延評価式は，評価を一時保留し，パターン照合時に動的にパターン構造を作り出す機能を実現している．これを用いて再帰的パターンを構成することができる．

```
P=*P 'Z'|'Y'                               (5)
```

(5)で＊はPが遅延評価式であることを示すオペレータである．パターンPは，後に定まる文字列とZとが結合した文字列，またはYと一致する．したがってPは先ずYと一致し，この一致のパターンが＊Pに代入されてZと結合して作られる YZ とも一致する．さらに YZ と Z の結合 YZZ とも一致する．すなわち，Pは，Y, YZ, YZZ, …, YZ…Z のすべてと一致するパターンを表わしている．

2.4　SNOBOL 4 の後継言語

SNOBOL 4 は機能的には非常に強力であり，パターンを動的に生成しながら文字列を処理する言語が他に

はあまりないこともあって広く利用されているが，プログラミングのしやすさ，実行効率，および融通性の面では必ずしも満足されているわけではない[3),4)]．SNOBOL 設計の頭初から研究に携わっている Griswold を中心とするグループでは，これらの欠点を改善するいくつかの提案を行っている．

SNOBOLX[5)] はプログラムの制御機構として，EXEC, DO, REPEAT, IF の4つの関数を導入し，また，特殊記号によるオペレータを定めて構文内の要素を見やすくしたシンタクスを提案している．そしてパターン照合により一致した部分とその前後をそれぞれ引き出す (extract) ことによる，より広範囲な拡張の可能性を示唆している．この提案は機能的には SNOBOL 4 を超えるものではなく，主としてプログラムの書きやすさ，見やすさとシンタクス解析の効率向上を狙ったものであった．

SNOBOL 4 のパターン照合はスキャナによって実現されるが，この操作はすべてシステムに組み込まれており，使用者が制御することができない．このためきめ細かい最適化や機能拡張が困難で融通性に欠けている．文献[6),7)] はスキャナの走査方向やカーソルの位置の制御を使用者が指示し，パターン照合操作を定義できるような拡張を提案している．

SL5[8)] はプログラミング言語の研究のための道具として設計された言語で，思想的には SNOBOL 4 の多くを継承している．シンタクスは Algol 68 流に begin-end または { } でグループ化して書くことができ，コルーチンが書ける．制御構造として if-then-else, while-do, until-do, repeat, for 等が採り入れられている．

Icon[9)] は文字列処理向汎用言語で，プロダクションシステムを意識した SNOBOL の後継言語として作られた．SNOBOL 4 および SL 5 では文字列走査と文字列処理は分離した機能であるが，Icon では発生器 (generator) と呼ぶ，式の目的地指向評価 (goal-directed evaluation) 手法を導入し，言語上でパターン照合操作を記述できるようにして上記2つの機能を統合している．目的地指向評価は，環境の要求によって後戻り処理を行い，他の選択 (alternative) を次々に導出することで目的地に到達しようとする戦略をいう．Icon プログラミングではパターン照合の手続きを記述することが基礎となるため，プログラミングの容易さやわかりやすさの点ではやや高級で専門家向きの言語である．

3. 文字列照合専用ハードウェア

文字列照合専用ハードウェアの利点の1つは並列処理および部分照合の結果を利用することにより，DC 文字*，VLDC 文字** を含むパターンとの文字列照合を後戻り制御を用いずに高速に処理できる点にある．専用ハードウェアは照合操作方式の違いにより，

(1) セルラアレイ方式
(2) 有限状態オートマトン方式
(3) 連想照合方式

に分類できる[10)]．

セルラアレイ方式は1文字単位の照合機能を持つ論理セルの組合わせでハードウェアを構成するもので，高レベルの並列処理が実現可能である．また，論理セルの規則性ならびに処理の局所性から VLSI 化に適した方式といえる．一方，有限状態オートマトン方式はパターン文字列が有限状態オートマトンで表現できることを利用した方式で，DC 文字や VLDC 文字を含むパターン文字列を容易に扱える点に特徴がある．最後の連想照合方式は連想プロセッサあるいは連想メモリを用いて与えられたパターン文字列を含むテキスト文字列を検索する方式で，複雑な文字列パターンは扱えないが，現実的なアプローチの1つといえる．

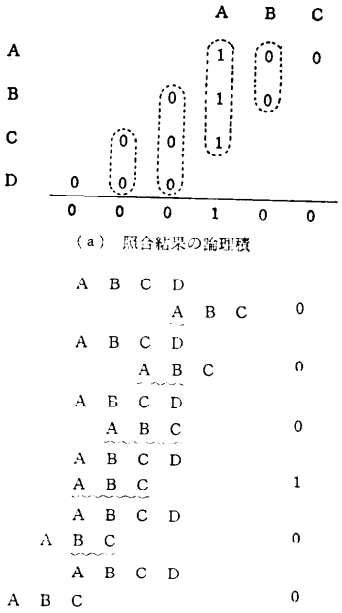
3.1 セルラアレイ方式

セルラアレイ方式によるパターン照合の基本原理はテキスト文字列の各文字とパターン文字列との照合結果を重ね合わせることにより，テキスト文字列全体との照合を実現することにある．**図-2**(a) にテキスト文字列 ABCD とパターン文字列 ABC との照合操作例を示す．テキスト文字列 ABCD の各文字とパターン文字列 ABC との照合を行い，結果を1段ずらして並べ，論理積をとる．ただし，文字が一致したときを“1”，一致しないときを“0”とする．このとき得られるビット列“000100”はテキスト文字列 ABCD とパターン文字列 ABC を1文字ずつずらして並べたときの重なった部分の照合結果に等しい (**図-2**(b))．

このような照合操作を実現する論理セルの構成は並列処理方式の違いにより，ブロードキャスト型とパイプライン型の2つに大別できる．

図-3(a), (b) にブロードキャスト型の基本論理セルの構成とその接続例を示す．各論理セルはパターン文

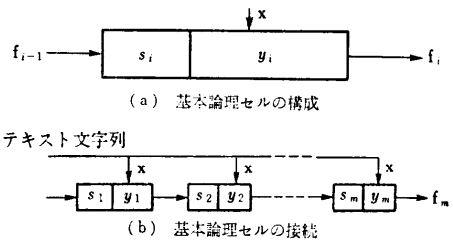
* DC (Don't Care) 文字は任意の1文字と一致するパターン文字を表わす．
** VLDC (Variable Length Don't Care) 文字は空列を含む任意長の文字列と一致するパターン文字を表わす．



(a) 照合結果の論理積

(b) 部分文字列間の照合結果

図-2 セルラアレイ方式の照合原理



(a) 基本論理セルの構成

(b) 基本論理セルの接続

図-3 ブロードキャスト型の基本論理セル構成

字 y_i を1文字貯える記憶領域と照合結果を貯える状態ビット s_i を備えている．照合操作は次のようにして行われる．

(i) 初期値として $s_1 \leftarrow 1$, $s_i \leftarrow 0$ $(i \geq 2)$ とする．

(ii) テキスト文字 x を各論理セルにブロードキャストする．

(iii) 各 i に対して，$s_i=1$ かつ $x=y_i$ ならば，$f_i \leftarrow 1$，そうでなければ $f_i \leftarrow 0$ とする．

(iv) 各 i に対して $s_{i+1} \leftarrow f_i$ とする．

(ii)から(iv)をテキスト文字列の各文字に対して繰り返し，$f_m=1$ となったときにパターン文字列 $y_1y_2\cdots y_m$ とテキスト文字列が一致したことが検出される．なお，(iv)において，常に $s_1 \leftarrow 1$ とすれば UNANCHOR モード* の照合となり，常に $s_1 \leftarrow 0$ とすれば ANCHOR モード** の照合となる．

ブロードキャスト型の論理セルの提案は多く，Mukhopadhyay[11] は SNOBOL 4 の基本パターンをこのような論理セルの組合わせで構成する手法について述べている．また，Pramanik[12] は部分文字列の照合結果から文字列全体の照合結果を合成する一般的手法を示し，複数個のテキスト文字と複数個のパターン文字を同時に照合させる論理セルを提案している．

この他にも，テキスト文字列の方に照合機能を持たせた Lee のセルラメモリ[13]，トラック・バー・ロジックの思想に立った INDY[14]，CASSS[15]，RAPID[16] においても同様な照合操作手法が用いられている．

一方，パイプライン型の論理セルは Kung らによって提案されている[17]．Kung のシストリック・アレイを用いた文字列照合機では，テキスト文字をブロードキャストする代わりにテキスト文字列とパターン文字列とを互いに逆方向に走査させながら文字列照合を行う．シストリック・アレイの動作原理を図-4 に示す．シストリック・アレイにはテキスト文字列とパターン文字列が逆向きにかつ1つ間を置いて入力され，論理セル間で同期しながら動いてゆく．各論理セルは文字単位の照合機能を有し，テキスト文字とパターン文字が交差したときに照合が行われ，結果が論理セルに貯えられる．ある特定の論理セルに着目すると，その論理セル上で引き続くパターン文字列との照合が行われ

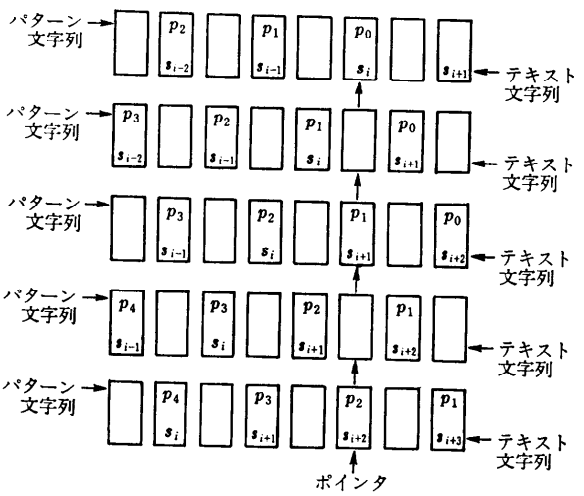


図-4 シストリック・アレイの動作原理

* UNANCHOR モードではテキスト文字列の任意の部分文字列とパターン文字列との照合を行う．

** ANCHOR モードではテキスト文字列の最初の文字列とパターン文字列との照合しか行わない．

る．全てのパターン文字列との照合が終了すると照合結果の論理積がテキスト文字列が流れる方向に出力される．

パイプライン型の論理セルは次の点で VLSI 化に適している．

（1） 規則性が高く，各論理セルの文字比較器もさらにビットレベルのシストリック・アレイで構成できる．

（2） シストリック・アレイへの入力は両端の論理セルだけでよく，入力ピン数が少なくて済む．

3.2 有限状態オートマトン方式

有限状態オートマトン方式ではパターンを有限状態オートマトン (FSA と記す) で表現し，パターン照合を FSA 上の状態遷移で実現する．すなわち，各 FSA にテキスト文字を 1 文字転送するたびに状態が 1 つ遷移し，最終状態に遷移したか否かにより照合結果の成否が判定される．

“AB” で始まり “A” で終るパターン文字列に対応する FSA を **図-5** に示す．状態 0 が初期状態であり，状態 5 が最終状態である．# 印は矢印で示された文字以外のパターン文字の遷移先を表わし，□印は空白に対する遷移を表わす．図-5 の FSA にテキスト文字列 “□ ABCABA □”を与えると，状態が 0→1→2→3→3→4→3→4→5 と遷移し，パターン文字列と一致したことが検出される．

FSA は状態とパターン文字列に対する状態遷移表で表わせるので，専用ハードウェアとしては状態遷移をエミュレートする機構を持てばよい．メモリ効率のよい状態遷移表の作り方については Roberts[18] が述べている．

図-5 では状態遷移先が高々 1 つしかない決定性 FSA を用いたが，非決定性 FSA を用いて複数個のパターン文字列との照合を行う手法が Haskin[19] によって論じられている．非決定性 FSA を用いると同一パターン文字列の共有が可能となるが，状態遷移先が複数個になるため，そのままでは実現は難しい．Haskin はこれを状態遷移先が高々 1 つしか起きないように非決定性 FSA を分割することにより解決してい

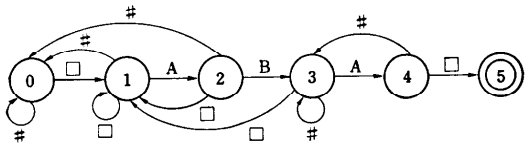


図-5 AB で始まりAで終わる語を受理する有限状態オートマトン

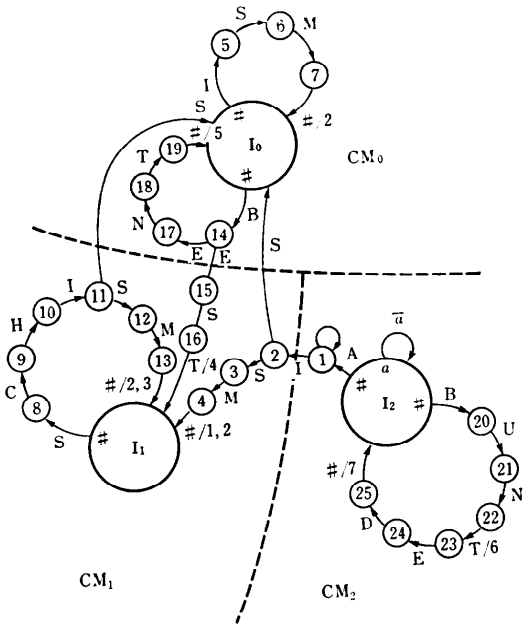


図-6 有限状態オートマトンのキャラクタ・マッチャ (CM) への割り付け例

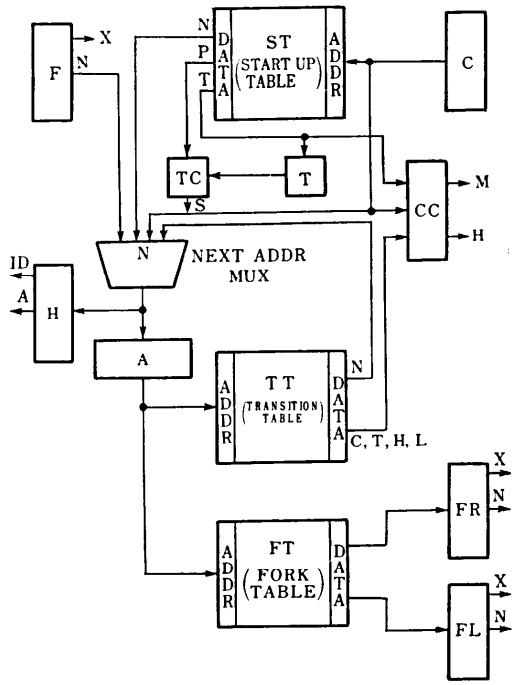


図-7 キャラクタ・マッチャ (CM) ブロック図

る.

図-6 はその分割例を示す. 分割された非決定性 FSA はそれぞれキャラクタ・マッチャ (CM) に割り当てられる. 1つの CM 内では状態遷移は高々1つしか起きず, かつ状態遷移の分岐も隣接する CM に対してのみ行われる.

CM のブロック図を 図-7 に示す. ST (Start up Table) には FSA の初期状態に対する状態遷移表が格納され, TT (Transition Table) には CM 内での状態遷移表が, FT (Fork Table) には他の CM への分岐を伴うときの状態遷移表が格納される. テキスト文字はレジスタCに格納され, 比較回路 CC により文字単位の照合が行われる. そして, 最終状態に入るとレジスタ H にそのときの状態番号が格納され, 外部の文字列照合管理ユニットに転送される.

CM の起動はレジスタ C に転送されたテキスト文字が ST に含まれているとき, および隣接する CM から状態分岐が通知されたときに行われ, レジスタ A にその状態に対応する TT および FT のアドレスが格納される. TT から次のパターン文字が読み出されると文字比較機 CC においてテキスト文字との照合が行われ, 成功すると遷移先の状態のアドレスがレジスタ A に格納され, 引き続く文字との照合が行われる.

状態分岐は CM 内での状態遷移と並行して行われ, レジスタ FR および FL に分岐先のアドレスが格納される. そして, 次の文字との照合が成功すると信号 X が ON となり隣接する CM が起動される.

3.3 連想照合方式

文字列照合の大部分は連想メモリの機能によりカバーできる. 文字列照合専用ハードウェアとして連想メモリを使用したときの問題点は次の2点である.

(1) VLDC 文字が扱えない.

(2) 大容量化が困難である.

現在の連想メモリの機能では VLDC 文字を扱うのは困難であるが, 後者に対しては小量の連想メモリを効率良く用いて連想照合を行う手法が Burkowski[20] により提案されている. この手法では, 複数個のパターン文字列を 32 文字幅の RAM に行単位に格納し, テキスト文字列を1文字ずつずらしながらパターン文字列との照合を行う. このとき, 各行の特定の位置の4文字が全て異なるようにパターン文字列を RAM に割り当て, その4文字分のみを連想メモリに格納する. そして, 連想メモリでヒットした行に対してのみパターン文字列全体との照合を行う. したがってこの場合は必要な連想メモリの量は8分の1で済む.

この他, Operating System Inc.[21] では連想プロセッサを用いたテキスト・リトリーバル・コンピュータを発表しているが, 照合メカニズムの詳細は明らかにされていない. また, 連想メモリの代わりに並列ハッシングを用いるのも1つの手法である.

4. マシンアーキテクチャ

文字列処理を高速に行う計算機のアーキテクチャについての研究は非常に古くから進められているが, 現状においては, まだ本格的なものが出現していないといえる.

これまでに提案または開発された文字列処理用マシンのアーキテクチャはその用途からみると, 汎用文字列処理を対象にしたものと, 特定の応用分野のための専用文字列処理を対象としたものに分類できる.

4.1 汎用文字列処理マシンアーキテクチャ

汎用の文字列処理用高級言語を高速に実行する構造を備えたコンピュータがこれに属し, これまでに, ADAM マシン[22], SYMBOL マシン[23], Snobol マシン[24] などが研究されている.

ADAM (A Problem-oriented Symbol Processor) マシンは, 可変長文字列から成るテキストデータを蓄積し, 高速に処理するプロセッサで, テキストデータは, Character, Symbol, Phrase, Sentence などと区別される構造を備えており, 構造を含む形で要求するテキストを見つけ出すことが可能である. この処理を高速化するために, 三次元構成の特殊メモリを提案し, 上記構造を三次元メモリの次元に対応させることにより, 並列検索を実現している.

SYMBOL マシンは可変長文字列を高速に処理するために, 強力なメモリ管理機構と文字列処理を専門に行う専用プロセッサを備えている. SYMBOL マシンのメモリ管理ハードウェアにより, 任意長の文字ストリングの蓄積, 抽出が可能である. SYMBOL マシン

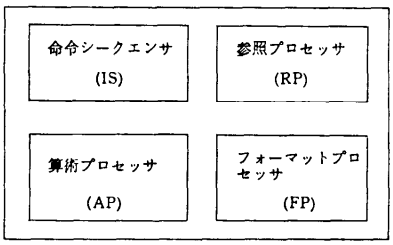


図-8 SYMBOL マシンのセントラルプロセッサの構成

は8つのプロセッサから構成されており，その中のセントラルプロセッサは，図-8 に示すように，命令シークエンサ，参照プロセッサ，演算プロセッサおよびフォーマットプロセッサ (FP) から成っている．FP は文字列処理を高速に行うもので，文字列の結合，比較，タイプ変換や編集処理などのフォーマット操作を実行する．

文字列処理用高級言語として，SNOBOL 4 の特徴について第2章ですでに述べたが，Purdue 大学の Shapiro は SNOBOL 4 を高速に処理する高級言語マシンである Snobol マシンを提案した．

SNOBOL 4 では，データタイプの宣言は行われず，かつ，実行時に動的にデータタイプが変化するので，Snobol マシンはデータ形式や長さを直接表現できるようにデータ記述子を導入している．また，文字列の蓄積と高速処理を行えるように，キャラクタストリング・レジスタを複数個持っている．このレジスタには，走査対象の文字列データ，走査パターン，などを含んでいる．そして，対象文字列データは，一時にその一部しかストリング・レジスタに蓄積できないため，走査の過程で文字列データを右から左へシフトすると，残りの文字列を右から自動的につめ込むハードウェアを備えている．さらに，複雑な文字列パターン照合処理を行う場合には，バックトラック機構が必要となるために，データ記述子や戻りアドレスを保存するためにプッシュダウン・スタックを備えている．

4.2 専用文字列処理マシンアーキテクチャ

これまでに，各種のオンライン巨大テキスト処理システムが開発されている．たとえば，アブストラクトを対象とする MEDLINE[25], ORBIT[26], DIALOG[27], テキスト全体を対象とする LEXIS, Westlaw & Juris[10], などが代表的なものである．しかし，これらはいずれも汎用コンピュータ上に実現されたソフトウェアシステムであり，記憶量と処理速度の点から巨大化するデータベースに対してオンライン処理を行うには限界がある．

データベースに対する処理を高速化するために，フォーマット化されたデータベースを対象として，RARES, RAP, CASSM, DBC, などのデータベースマシン[28] が研究されているが，これらは，テキストデータなどの非フォーマットデータに対しては効率良く処理できないと考えられる．

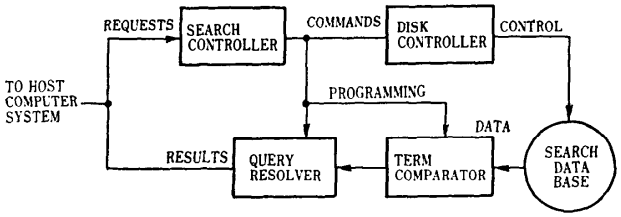
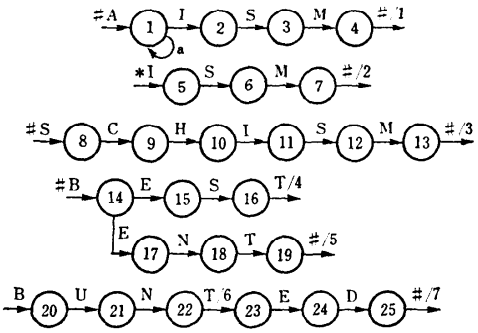


図-9　テキスト検索コンピュータ

1. ＃A2ISM＃
2. ISM＃
3. ＃SCHISM＃
4. ＃BEST
5. ＃BENT＃
6. ＃BUNT
7. ＃BUNTED＃

(a) Term のリスト例



(b) 整理された状態テーブル

図-10　照合パターンと状態遷移テーブル

このような状況から，第3章で述べたような専用ハードウェアを用いたマシンアーキテクチャがいくつか提案されている．

イリノイ大学の Haskin らの専用マシン[19] は有限オートマトンの方式を用いたテキスト検索専用コンピュータで，図-9 に構成を示す．この中で，Term Comparator が中核をなし，文字列照合を高速に行うために複数個の図-7 に示すような Character Matcher (CM) から構成される．図-10 に示すような文字列パターンを検出する場合には，照合パターンを有限オートマトン (FSA) 形式の状態遷移で表現し，各々の CM は FSA の各状態に対応づけられ，所定のパターンか否かを並列にしらべる．

Haskin らと同様な方式で，Central Intelligence Agency の Roberts も専用マシン・アーキテクチャを提案[8] しているが，詳細は明らかにされていない．

Central Florida 大学の Mukhopadhyay のパターン照合用高速ハードウェアについてはすでに述べた

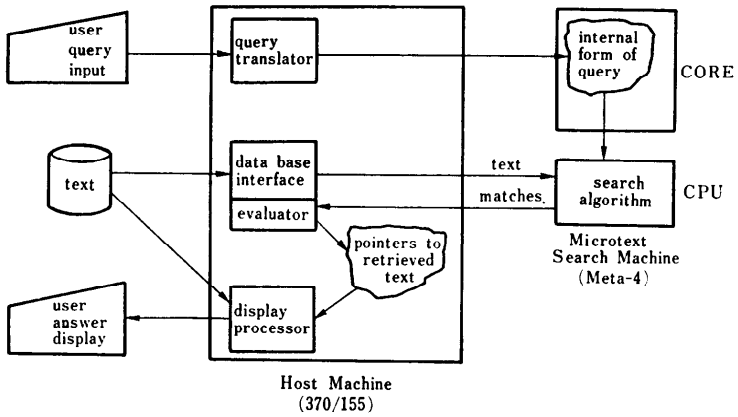


図-11 MICROTEXT のハードウェア構成

が，これを用いたハードウェアテキストプロセッサが提案されている．このプロセッサは Match, Match with Alteration, Count, Replace, Remove, Extract, などストリング処理用の19種類の命令を備えており，これらの命令は前記高速ハードウェアを用いて実現される．

MITRE 社の Bullen らはテキスト全体を検索する専用サーチマシン (MICROTEXT)[29] を実現している．**図-11** にシステム構成を示すように，IBM 370/155 をホストマシンとして，汎用エミュレータである Meta-4 を用いてサーチマシンを構成している．Meta-4 のマイクロプログラムを用いて，有限状態サーチアルゴリズムを実行することにより実現しており，ホストマシンで処理するよりも5倍高速になると報告されている．

5. 文字列処理の応用

文字列処理の応用は情報検索，テキスト処理を始め，非常に多くの分野に及んでいる．また，最近では，知識情報処理の分野で述語論理が重要視されており，そのための言語として注目を浴びている Prolog では文字列処理の基本であるパターン照合操作が処理の中核を形成している．

本章では，代表的文字列処理の応用について，特徴

```
? SELECT INFORMATION(W)RETRIEVAL
          1  3210 INFORMATION(W)RETRIEVAL
? SELECT ONLINE
          2  2028 ONLINE
? SELECT ON(W)LINE
          3  187 ON(W)LINE
? SELECT ON-LINE
          4  13 ON-LINE
? SELECT BIBLIOGRAPHIC
          5  334 BIBLIOGRAPHIC
? COMBINE 1 AND (2 OR 3 OR 4) AND 5
          6  30 1 AND (2 OR 3 OR 4) AND 5
? TYPE 6/6/1-3
6/6/1
1122432  C77027452
  ON-LINE BIBLIOGRAPHIC RETRIEVAL SYSTEMS USE
6/6/2
1096822  C77020388
UNDERGRADUATE USE OF ONLINE BIBLIOGRAPHIC RETRIEVAL SERVICES'
EXPERIENCES AT THE UNIVERSITY OF CALIFORNIA, SAN DIEGO
6/6/3
1086698  C77020069
  ON-LINE SERVICES AND OPERATIONAL COSTS
```

図-12 DIALOG での検索例

```
*000' % us 'troop' 3 'withdr' | ('pull' out)
```

(a) 検索例

```
...30,000 us troops withdrew...
...5,000 us troops will withdraw...
...46,000 troops will be pulled out...
...10,000-man troop withdrawal...
...365,000 troops could have been withdrawn...
...44,000 troops to pull out...
```

(b) 検索により抽出される文

図-13 SHOEBOX での検索例

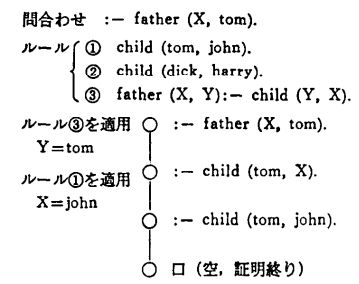


図-14 PROLOG の実行メカニズムの概念図

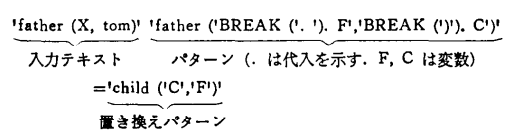


図-15 SNOBOL4 による文字列の変形例

とそこで要求される文字列処理に関して述べる.

(1) 情報検索

情報検索は文字列処理の代表的応用の1つであり, これには, 検索対象として, Bibliographyレベルのものから, アブストラクトに対するもの, さらには, テキスト全体を対象とするものと, 各種行われている.

DIALOG や MEDLINE はアブストラクトを検索の対象とするシステムで, ここで要求される文字列処理は基本的には SNOBOL 4 レベルの機能があればよい. **図-12** は DIALOG での検索例を示す.

MICROTEXT[29] や SHOEBOX[30] はテキスト全体を対象とするものである. このようなシステムでは, テキストでの柔軟な表現に対しても検索できる機能が要求される. たとえば,「南ヴェトナムに関するリポートの中から, アメリカ軍撤退を示すニュースを見つけ出す」検索例を **図-13** に示す. このように, 各語に対して, stem (単語の基本形) 表現を用いて規定することにより, 同じ意味を表わす変形文まで検索できる.

このようなシステムの発展形として, Data Point 社は OA 用の情報検索システム (AIM: Associative Index Method)[31] を発表している. AIM は一連のキーワードを入力することにより, それらを満たす特定のドキュメントを見つけ出すソフトウェアシステムで, 上級管理者向けの実用システムである. また, NDX/Nelma 社においても, オフィスのテキスト情報処理を実現するためのシステムとして, 電子ファイリングマシン (NDX-100)[32] を発表している. NDX-100 はビジネスレターやメモ, などのドキュメントを多量に蓄積し, これに対する高速検索が可能であり, 機能分散型のマイクロプロセッサシステムで構成されている.

(2) テキスト処理および英単語検査

OA は文字列処理の応用の重要な1つであり, その中でも, ワードプロセッシングとしてのテキストエディタが代表的なものである. そして, 英文章を対象とするテキストエディタはすでに実用レベルにある. テキストエディタでは, 大容量のテキストデータに対する文書清書が行われ, 特定文字列 (複数個でも可能) の検索, 特定文字列を指定文字列で置き換える文字列置換, 指定文字列の除去, 文字列挿入, などの処理が要求される. これらの処理を行うことにより, 必要なテキストを作成し, さらに, 出力イメージにフォーマットして出力する.

テキストデータの作成過程での支援ツールとして, UNIX の Writers Workbench[33] などでは, 単語検査プログラムが準備されている. これは, 単語レベルの検査を行うもので, 英単語より Prefix や Suffix を取り除き, 得られた単語の stem が正しいか否かを辞書と照合することにより実現している. したがって, 英単語検査では, 文字列の抽出と照合処理が頻繁に用いられることになる.

さらに, EPISTLE システム[34] では, 単語レベルではなく, 単語間の関係から正しくない単語を見つけ出す高度な機能を備えている. この場合には, 単なる文字処理だけでなく, LISP にみられるようなリスト構造を扱えることが要求される. すなわち, テキストから単語に分解するまでは文字列処理で行い, その後はリスト構造で表現したテキストに基づいて分析することが必要になる.

(3) PROLOG でのパターン照合処理

PROLOG の実行メカニズムは入力された問合わせからトップダウン的に推論を続け, すでに与えられている公理体系 (データベース) との一致もしくは矛盾

を導出することが基本である[35]. この場合のトップダウンの推論とは問題をより下位の問題に置き換えてゆくことであり一種のリダクションと見なすことができる. たとえば 図-14 において問合わせ father (X, tom) から child の表現に変形し, それ以上変形できなくなった時点で実行は完了する. これは文字列の変形 (string reduction) 処理である. したがって, たとえば SNOBOL 4 を用いて〈入力テキスト〉〈データまたはルール〉=〈変形されたテキスト〉の形式でプログラムすることができる (図-15). ルール中の"変数"とは任意の文字列と一致するパターンとして実現される.

しかしながら, 従来のパターン照合と異なる点は変形を受ける入力テキストとパターンの双方に変数が存在し得ることで, 照合の結果, 入力テキストからパターンへまたはパターンから入力テキストへの代入が行われる. 文字列パターン照合機能の最も強力なSNOBOLにおいてさえ操作を受ける側は変数を含まない単純な文字列しか許されていない. この意味で PROLOG におけるパターン照合機能は一般化された, より強力なものとなっているといえる. さらに, パターン照合の対象としているデータは単なる文字列というよりも意味を持った構造化データと考えるべきものであり, より抽象化されたパターン照合操作といえる. ただし, 残念ながら文字列を効率良く処理できるアーキテクチャがまだ実現されていないため実際の PROLOG 処理系ではリスト構造を用いるのが普通である.

6. むすび

文字列処理の現状について, 基礎から応用に亘って論じたが, 現段階において, 本分野の技術はまだ研究レベルにあるといえる. しかし, 新しい応用分野の拡大と VLSI 技術の進歩と相俟って, さらに発展するものと考えられる. 特に, 今後, OA や知識情報処理などにおいて, 高度な応用が進むにつれて, システムの高級化, 知能化を実現するための手段として, 文字列処理はますます重要になるものと予想される. その場合に, 本稿でもすでに若干触れたように, 文字列処理が単独に存在するのではなく, 本特集号で論じられているリスト処理と融合して用いられることにより, さらに高度なシステムを構成することが可能となろう.

参考文献

1) Farber, D. J., Griswold, R. E. and Polonsky, I. P.: SNOBOL, A String Manipulation Language, J. ACM, Vol. 11, No. 1 pp. 21-30 (Jan. 1964).
2) Griswold, R. E., Poage, J. F. and Polonsky, I. P.: The SNOBOL4 Programming Language (Second Edition), 256 p., Prentice-Hall, Inc., Englewood Cliffs, N. J. (1971).
3) 角田博保: SNOBOL, 情報処理, Vol. 22, No. 6, pp. 473-476 (1981).
4) Wexelblat, R. L. (Editor): History or Programming Languages, pp. 601-660, Academic Press (1981).
5) Griswold, R. E.: Suggested Revisions and Additions to the Syntax and Control Mechanism, SIGPLAN Notices, Vol. 9, No. 2, pp. 7-23 (1974).
6) Druseikis, F. C. and Doyle, J. N.: A Procedural Approach to Pattern Matching in SNOBOL 4, Proc. Annual Conference on the ACM '74, pp. 311-317 (1974).
7) Griswold, R. E.: Extensible Pattern Matching in SNOBOL 4, Proc. Annual Conference of the ACM '75, pp. 248-252 (1975).
8) Griswold, R. E. and Hanson, D. R.: An Overview of SL 5, SIGPLAN Notices Vol. 12, No. 4, pp. 40-50 (1977).
9) Griswold, R. E., Hanson, D. R. and Korb, J. T.: The Icon Programming Language An Overview, SGIPLAN Notices, Vol. 14, No. 4, pp. 18-31 (1979).
10) Hollaar, L. A.: Text Retrieval Computers, Computer, Vol. 12, No. 3, pp. 40-50 (Mar. 1979).
11) Mukhopadhyay, A.: Hardware Algorithms for Nonnumeric Computation, IEEE Trans. on Computer Vol. C-28, No. 6, pp. 384-394 (June 1979).
12) Pramanik, S.: Highly parallel associative search and its application to cellular database machine design, Proc. AFIPS conf., Vol. 50, pp. 521-528 (1981).
13) Lee, C. Y.: Intercommunicating Cells, Basis for a Distributed Logic Computer, Proc. FJCC Conf., pp. 130-136 (1962).
14) Copeland, G. P.: String Storage and Serching for Data Base Applications: Implementation on the INDY Backend Kernel, Proc. 4th Non-Numeric Workshop, pp. 8-17 (Aug. 1978).
15) Healy, L. D., Lipovski, G. J. and Doty, K. L.: The Architecture of a Context Addressed Segment-Sequential Storage, Proc. FJCC conf.,

Vol. 41, part II, pp. 691-701 (1972).
16) Parhami, B.,: A Highly Parallel Computing System for Information Retrieval, Proc. FJCC conf., Vol. 41, part II, pp. 681-690 (Dec. 1972).
17) Foster, M. J. and Kung, H. T.: The Design of Special-Purpose VLSI Chips, Computer, Vol. 13, No. 1, pp. 26-40 (Jan. 1980).
18) Roberts, D. C.: A Specialized Computer Architecture for Text Retrieval, Proc. 4th Non-Numeric Workshop, pp. 51-59 (Aug. 1978).
19) Haskin, R.: Hardware for Searching Very Large Text Databases, SIGIR, Vol. 15, No. 2, pp. 49-56 (Mar. 1980).
20) Burkowski, F. J.: A High Level Architecture for a Text Scanning Processor, Proc. HLLM Computer Architecture, pp. 206-211 (May 1980).
21) Bird, R. M., Tu, J. C. and Worthy, R. M.: Associative/Parallel Processors for Searching Very Large Textual Data Bases, Proc. 3rd Non-Numeric Workshop, pp. 8-16 (May 1977).
22) Mullery, A. P. et al.: ADAM-A Problem-Oriented Symbol Processor, AFIPS SJCC pp. 367-380 (1963).
23) Smith, W. R., et al.: SYMBOL-A large experimental system exploring major hardware replacement of software, AFIPS SJCC, pp. 601-616 (1971 Spring).
24) Sapiro, M. D.: A Snobol Machine: A Higher-level language processor in a conventional hardware framework, Digest of Compcon pp. 41-44 (1972).
25) McCarn, D. B.: On-line Services of the National Library of Medicine, Digest of Compcon pp. 48-53 (1978 Fall).
26) Black, D. V.: System Development Corporation's Search Service, Digest of Compcon pp. 59-64 (1978, Fall)
27) Bayer, M. P.: DIALOG-An Online Retrieval System For Bibliographic Information, Digest of Compcon, pp. 54-58 (1978 Fall).
28) Bray, O. H. and Freeman, H. A.: Data Base Computers, Lexington Books (1979).
29) Bullen, R. H., et al.: MICROTEXT — The design of a microprogrammed finite state search machine for full-text retrieval, AFIPS FJCC, pp. 479-488 (1972).
30) Glantz, R. S.: SHOEBOX—A Personal file handling system for textual data, AFIPS FJCC, pp. 535-545 (1970).
31) DATAPOINT Corporation News: Datapoints New AIM™ Provides Unique File Access Capability (Sep. 1980).
32) Slonim J., et al.: NDX-100: An Electronic Filing Machine for the Office of the Future, Computer, pp. 24-36 (May 1981).
33) Cherry, L.: A toolbox for writers and editors, Digest of Office Automation Conference, pp. 221-227 (1981).
34) Miller, L. A., et al.: Text-critiquing with EPISTLE System: an author's aid to better syntax, AFIPS NCC, pp. 649-655 (1981).
35) Kowalski, R. A.,: Logic for Problem Solving, North-Holland Publishing Co., New York (1979).

(昭和57年4月16日受付)