

サーチエンジンとソートエンジンの記号処理への応用

北海道大学 工学部

田中 譲

1. 序論

データベース処理，特に 2 つ以上の関係に跨がる関係演算の高速化を目指し，著者等は 2 種類の機能モジュールを開発した⁽¹⁾⁽²⁾。データベース処理においては，大量のデータの転送が隘路となる。この点を考慮して，2 種類の機能モジュールを設計するにあたって，これらのモジュールにおけるデータ処理が，モジュールへのデータの転送，およびモジュールからのデータの転送と重畳され，同時実行されるように工夫した。2 種類の機能モジュールは，サーチとソートというデータベース処理における 2 つの基本的な処理を行い，それぞれ，サーチエンジン，ソートエンジンと名付けられている。

サーチエンジンとソートエンジンは，各々のプロトタイプが TTL で組み立て完成している。プロトタイプ of サーチエンジンは，16 bit の長さのキーを 4095 個，ソートされた順番でテーブルに格納しておき，これに流れ込んでくる任意個の探索キーの各々をテーブル内で探索し，見つけた場合

にはそのテーブルアドレスを出力し、見つからない場合には、そのキーより大きい最小のキーが入っているテーブルアドレスを出力する。 n 個の探索キーをテーブルアドレスに変換するのに $n \mu\text{sec}$ (かかからない)。プロトタイプのソートエンジンは 16 bit のキー 4095 個が 1 個ずつ転送され、最後の 1 個の転送が終了するやいなや、これらをソートした順序で 1 個ずつ他のモジュールに転送し始めることができる。プロトタイプは、4095 個のソートを 100 msec で終了する。300 nsec 程度のメモリーを用いて 2 つのモジュールを構成した場合、両者とも 1 つのデータあたり $1 \mu\text{sec}$ 以下の時間で各々の処理を行うことができる。

各々のモジュールは、ビットスライスのアーキテクチャになっいて、 n 個の同種のモジュールを横方向に結合することにより、 n 倍のワード長のデータのサーチとソートを行う回路を構成することができる。両モジュールとも、VLSI 化した際のピン数は教 10 ピン以内に制限することができる。上述の拡張性と共に、これらのモジュールが VLSI モジュールとして通していることを示している。

サーチエンジンとソートエンジンが VLSI 化されるとすると、これらの機能であるサーチとソートが基本演算と見做し得るようになり、種々のデータ処理に新しいアルゴリズム

を提供するものと期待する。それは、このような機能モジュールにより、サーチとソートの時間複雑性が変化し、これにより、従来はサーチとソートを用いて処理することなど考えられていなかった問題を、2種類のモジュールを用いて処理することにより、今までのアルゴリズムより速く、スマートな処理方法を開発することができるとは知られていないからである。

本論文では、2つのモジュールの記号処理への応用を目指し、この分野での基本的な問題の一つであるストリングマッチングを、上述のような問題の一つとして取り上げ、サーチエンジンとソートエンジンがこの種の問題にどのように応用され得るかを示す。本論文で示すストリングマッチングモジュールを核として、PROLOG等の論理型言語の実行を行うマシンを構成することは可能であると思われるが、これについては別の機会に報告する。このストリングマッチングのアルゴリズムでは、パターンをソートエンジンで前処理したものをサーチエンジンに格納し、複数のサーチエンジンで構成されるパイプに、サブジェクトストリングを先頭から順にデータの流れるように流し込むことにより処理する。これに関連して、本論文では可変長データのソートに関して新しいアルゴリズムを与える。

以下、第2章ではサーチエンジン（以下ではSEEと略記する）とソートエンジン（以下ではSOEと略記する）の機能の概説を行い、第3章では可変長データのソートアルゴリズムを、第4章ではストリングマッチングアルゴリズムを述べる。

2. サーチエンジン(SEE)とソートエンジン(SOE)の機能

SEEは値の順にソートされた n 個の被探索キーからなるテーブル中で、別に与えられる m 個の任意の順序に並んだ探索キーを順々に探索し、値の等しい被探索キーが見つかった時は、その旨を示すフラグと、値の等しい被探索キーの内、そのテーブル内アドレスの最小のもの（の）のアドレスを出力する。値の等しい被探索キーが見つからない時は、探索キーより大きい最小の被探索キーのテーブルアドレスの内、最小のもの（の）を出力する（Fig. 2.1）。実際のSEEによる処理は、探索キーを1個ずつ逐次にテーブルアドレスに変換するのでもなければ、すべての探索キーの入力が終了してからテーブルアドレスの出力を開始するのでもない。 m 個の探索キーは長さ m のデータの流れとしてSEEに流れ込み、SEEはこの流れを通す有限長のパイプとして働く。データの流れは、パイプから出るときにはフラグとテーブルアド

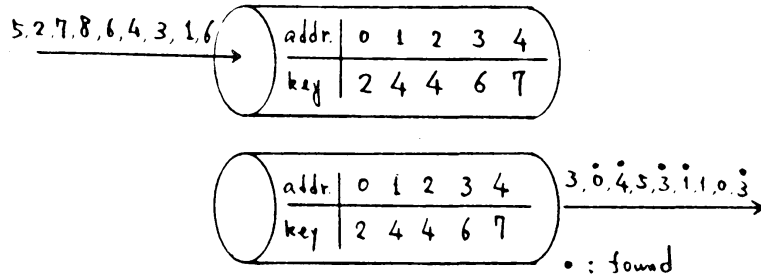


Fig. 2.1. SEEのサーチ機能

レスの流れへと変換されている (Fig. 2.2)。SEEのパイプの長さをSEEのレベル数と見做し、レベル数 L のSEEは $2^L - 1$ 個の被探索キーからなるテーブルを格納することが出来る。

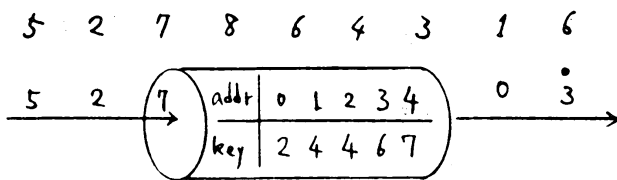


Fig. 2.2. SEEによるビットストリーム処理.

フラグが ON でテーブルアドレスが i のとき、これを i^* と表わすことにする。 Z を非負の整数の集合とし、 Z^* を

$$Z^* = \{ i, i^* \mid i \in Z \},$$

$$\forall i \in \mathbb{Z} \quad i < i^* < i+1$$

と定義し、本論文ではこの数体系 \mathbb{Z}^* を用いる。 \mathbb{Z}^* から \mathbb{Z} への変換は存在し、例えば

$$\alpha: \mathbb{Z}^* \rightarrow \mathbb{Z}$$

$$\alpha(i) = 2i$$

$$\alpha(i^*) = 2i+1$$

とすればよい。以下ではフラグ出力とテーブルアドレス出力をまとめて \mathbb{Z}^* の要素の出力と見做したものを単にテーブルアドレス出力と言うことにする。SEE はテーブルアドレスの流れだけでなく、 σ と σ の探索キーの流れも同期して出力するものとし、SEE を Fig. 2.3 のように記号化する。テーブ

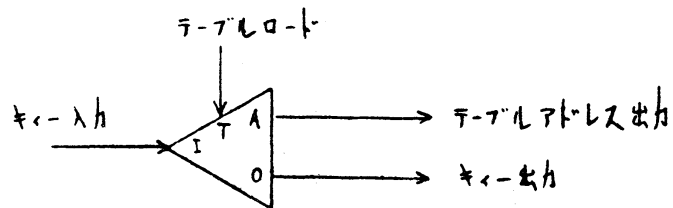


Fig. 2.3 SEEの記号

ル入力、探索に先立って被探索キーを入力するための入口がある。この入力 σ 、値の順にソートされたデータの流れが流入するのと同期して行われる。

SOEは勝手な順序で並んだ複数のキーからなるデータの流しがSOEに流入され終わると同時に、この複数のキーを値の順序にソートしたデータの流しを流出し始める (Fig. 2.4)。SOEはすべてのキーのソートを行うので、キーのすべてがSOEに流入し終わる前にSOEからの流出が始まることとなる。

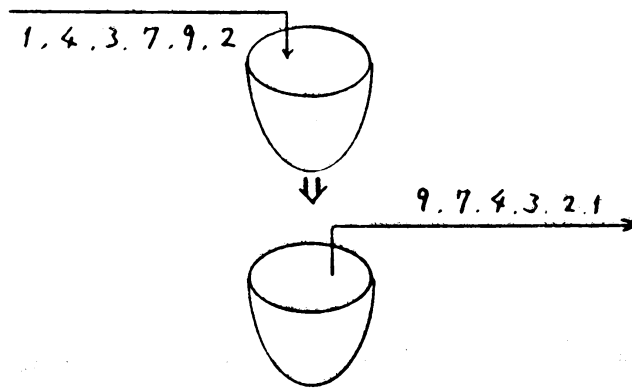


Fig. 2.4 SOEによるソート

SOEにはレベル数が定義されており、レベル数がLのSOEは、単独では最大 $2^L - 1$ 個のデータのソートを行う。SOEは Fig. 2.5 のように記号化する。

SEEのSOEのビットスライスアーキテクチャになっており、多倍長語のデータ処理するには各々を倍長だけ結合すればよい。⁽¹⁾

サーチにおいて、ターゲットが大きくなると1つのSEEには格納

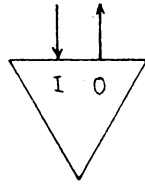


Fig. 2.5 SOE の記号化

いさなくなつたときは、被探索キーを順に $2^L - 1$ 個づつ、レベル数 L の列々の SEE に格納し、探索キーはすべての SEE に流し込めばよい。データ数が多く、1 個の SOE では引取できない場合には、複数個の SOE をトータル接続と名付けた形に接続することにより、使用した SOE の個数倍のデータの引取が可能になる⁽¹⁾。したがって、1 個の SEE、SOE のレベル数を L 、語長を ω とすると、等面的には、 h 、 h を任意の正整数として、語長が $h\omega$ で、扱われるデータ数が単体の場合の $2^L - 1$ に対して $h(2^L - 1)$ 個であるような SEE、SOE を構成することが出来る。

$2^L - 1$ 語のメモリーの各番地 i に、SEE に格納したレベルの i 番地のキーに対応するレコードを格納しておく。SEE の出力 j^* に対して i のメモリーの j 番地を読み出すことにすると、このシステムは全体として連想読み出しメモリーとなる (Fig. 2.6)。後置メモリーに、レコード部を上述のような順序でロードするためには、任意個の (キー、レコ

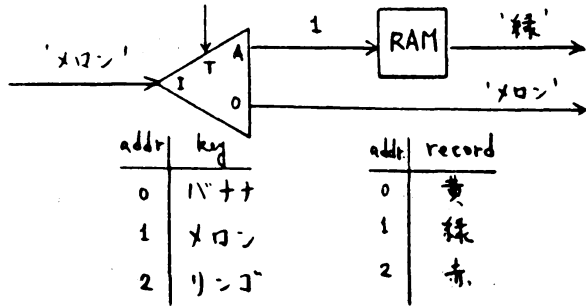


Fig. 2.6 SEEを用いた連想メモリー

ト)の対応を、キーの値の順序に並べ替えなければならぬ。このような処理は、SOEと、このSOEにレベル数+1本の信号線を結合されるSOEシミュレータ(SOES)を用い、(キー、レコード)対の流れのキー部の流れとレコード部の流れを同期させて別々に各々SOEとSOESに流し込み、全部を流し込んだ後に、SOEからの出力とSOESからの出力を1個ずつ対にして出力すればよい。SOE, SOESの結合をFig. 2.7 (a)のように記号を表わす。先述したSOE, SEEの多倍長化のための結合はFig. 2.7 (b)のように表わす。

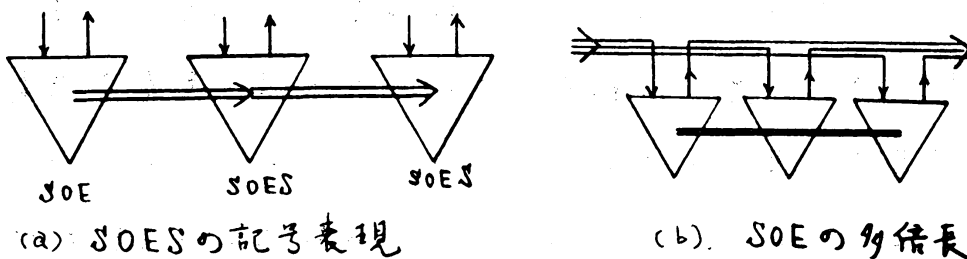


Fig. 2.7. SEE, SOE, SOESの結合

3. 可変長データのリスト

Σ を有限順序集合とする。1 を Σ の最小要素とし、 $\Sigma - \{1\}$ の要素の有限系列の集合を $(\Sigma - \{1\})^*$ とする。 $(\Sigma - \{1\})^*$ の有限部分集合 S の要素を辞書配列に従ってリストすることを見る。この処理は S の各要素に 1 を適当な個数だけ接続して各系列の長さを一定にして処理を行う (Fig. 3.1)。 S を図

H	H L L L
A J K L	A J K L
E F	E F L L
H I J K	H I J K
A B C	A B C L
A B	A B L L
B	B L L L

Fig. 3.1 可変長系列の集合 S と
 S の要素の固定長化

固定長化したものを S' で表わすことにする。 S' のリスト法として 3 種類の方法が考えられる。一つの方法は語長を多倍長化した SOE を用いることである。SOE はビットスライスのアーキテクチャに依っているので充分な個数の SOE を結合することにより、このような多倍長語の SOE を構成できる。他の

2種類の方法では、無制限に多倍長化の機能を用いることなく、多倍長データのソートを行う。

アルゴリズム A は SOE と SOES を用いて、この処理をラックソートの考えに基づいて実行する (Fig. 3.2)。まず HLLL, AJKL, EFLl, HIJK, ABCl, ABLL, BLLL が最後尾の文字に関してソートされ、例えば EFLl, ABCl, ABLL, HLLL, BLLL, HIJK, AJKL となる。今度は、これを後から 2 番目の文字でソートする。この文字が等しい系列 a, b は、このときの系列の並びで a が b の前に並んでいれば a を b の前に出力するようにする。これを SOE を行うために、j の系列に順番を 0, 1, 2, ... と付け、後から 2 番目の文字とこの順番の連接をキーとしてソートする。この結果は EFLl, ABLL, HLLL, BLLL, ABCl, HIJK, AJKL の順になる。以下同様のことを後から 3 番目の文字、4 番目の文字、... と繰り返すことにより ABLL, ABCl, AJKL, BLLL, EFLl, HLLL, HIJK の順に出力され、ソートが完了する。

アルゴリズム B は SEE と SOE を Fig. 3.3 のように組み合わせ、この処理を実行する。このアルゴリズムでは、テーブルアドレス i , i^* を i 1 文字と見做す。図中の $\boxed{1/2}$ は 1 文字から 2 文字への変換器で、1 文字のデータが 2 個到着するのを

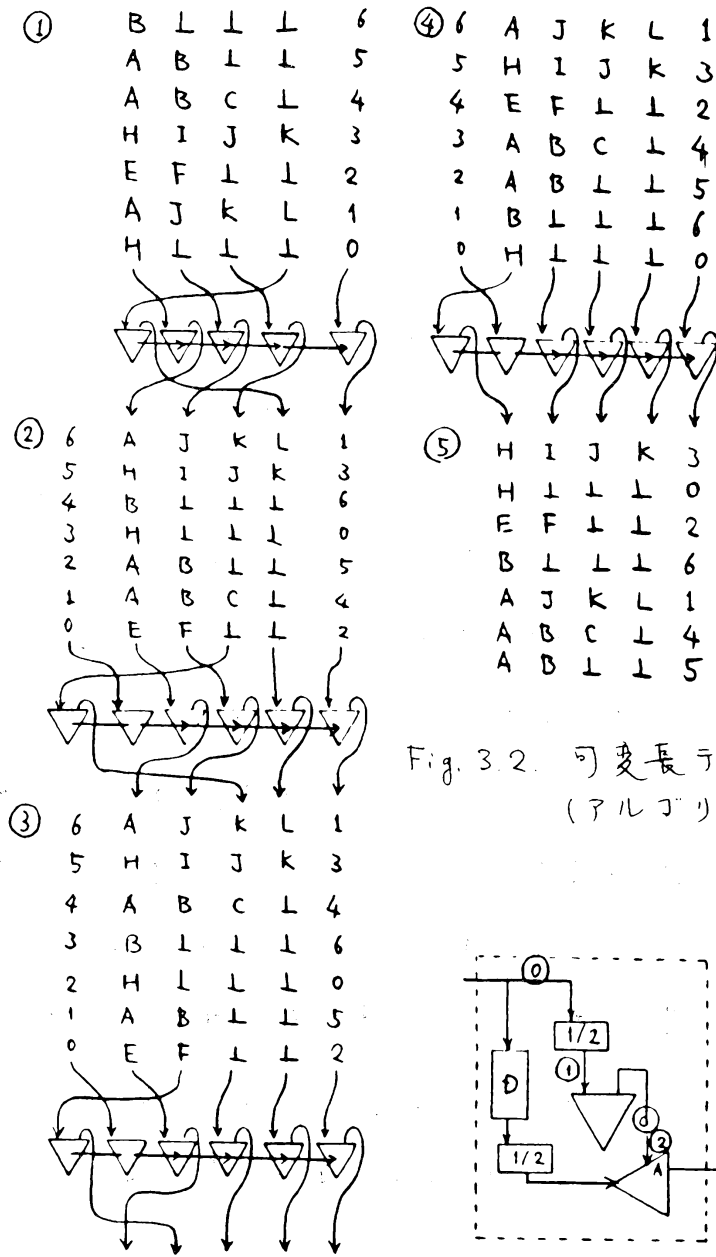


Fig. 3.2 可変長データのソート (アルゴリズム A)

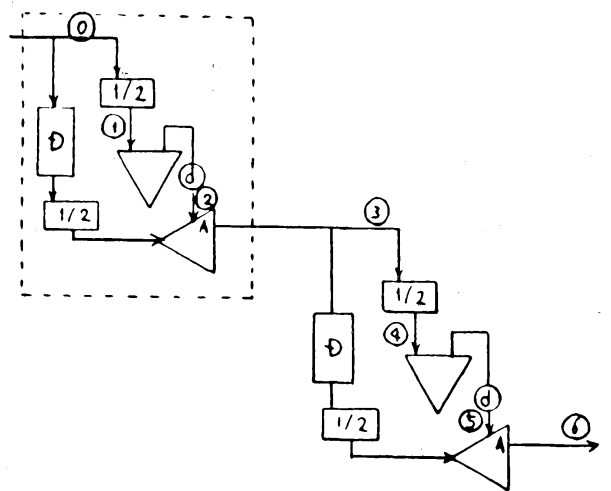


Fig. 3.3 可変長データのソート (アルゴリズム B)

待って、これらを2文字長のデータ1個に構成して出力する。
 図中の①は、重複データの除去器である。図中に①の置
 かれてゐる場所では、データはソートされておき、従って重
 複データは連続する。よって重複データの除去は容易である。
 S_0 のデータは各々 2^L 文字であるとする。そうであるときは、
 各データに1を適当な個数だけ接続すればよい。このとき、
 S_0 のデータのソートを行うには、図中の破線や囲んだ部分
 を1段用ひればよい。物理的に破線や囲んだ部分を1個用意
 する必要はなく、1個を繰り返し用ひることが可能である。

例題の S_0 に対し、図中の②には HLLL AJKLEFLLHIJK
 ABCLABLLBLLL をデータの流れとして入力する。①では
 これが2文字づつまとめられ

(HL)(LL)(AJ)(KL)(EF)(LL)(HI)(JK)(AB)(CL)(AB)(LL)
 (BL)(LL)

となる。これをソートし、重複の除去をしたのが②で、こ
 れはテーブルとしてSEEに格納される。②とテーブルアドレ
 スは

0 1 2 3 4 5 6 7 8 9
 (LL)(AB)(AJ)(BL)(CL)(EF)(HL)(HI)(JK)(KL)

となる。図中の①は、SEE へのターゲットの格納が終了すると同時に SEE への探索キーマスの入力開始されるように時間遅れを調整するためのものである。SEE への入力は ①と同じであり、送りターゲットアドレス出力である ③は、

$$6^* 0^* 2^* 9^* 5^* 0^* 7^* 8^* 1^* 4^* 1^* 0^* 3^* 0^*$$

となる。以下は同様で、④は、

$$(6^* 0^*) (2^* 9^*) (5^* 0^*) (7^* 8^*) (1^* 4^*) (1^* 0^*) (3^* 0^*)$$

となり、⑤とターゲットアドレスは

0	1	2	3	4	5	6
(1^* 0^*)	(1^* 4^*)	(2^* 9^*)	(3^* 0^*)	(5^* 0^*)	(6^* 0^*)	(7^* 8^*)

となり、⑥は、

$$5 \quad 2 \quad 4 \quad 6 \quad 1 \quad 0 \quad 3$$

となる。この並びは、 Σ をソートすると、HLLL は 5+1 番目に、次の AJKL は 2+1 番目に、...、ABLL は 0+1 番目に、そして BLLL は 3+1 番目になることを示している。

4. スtringマッチング

Σ を有限順序集合とし、 Σ^* を Σ の有限系列の集合とする。
 Σ^* の有限部分集合 S をパターン集合、 $t \in \Sigma^*$ をサブジェクト
 スtring とする String マッチングとは、各整数 $i > 0$
 に対して t の i 文字目から始まる部分 String で S に含
 まれ、長さの ρ と ρ 長 $\dots \rho$ のを見つける処理をいふ (Fig. 4.1)。

$$\Sigma = \{ \perp, A, B, C, \dots, \varepsilon \}$$

$$S = \{ \underset{P_1}{C}, \underset{P_2}{CB}, \underset{P_3}{CBC}, \underset{P_4}{CJKL}, \underset{P_5}{FD} \}$$

$$t = AFCBCJKLCAD$$

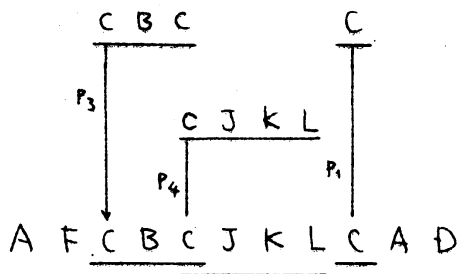


Fig. 4.1 String マッチングの例。

m は S の最長要素長以上とし、 S の各要素 a に対して、 Σ^*
 の要素の内 ρ 長さが m 以下のものを辞書配列に並べたとき、
 a の前には並ぶ長さ m の系列 γ と γ 後に並ぶ ρ を a とし、

先頭部分が a であるような系列の内、最後に並んでゐるものを a^+ と定義する。 a^+ の長さ m の系列となる。 \perp を Σ の最小要素とするとき、 $a = \perp \dots \perp$ のように \perp が任意個並んだ系列に対し、 a^- は存在しない。 Fig. 4.1 で $m=4$ とすると、 $C^- = BZZZ$, $C^+ = CZZZ$, $CB^- = CAZZ$, $CB^+ = CBZZ$, $CBC^- = CBBZ$, $CBC^+ = CBCZ$, $CJKL^- = CJKK$, $CJKL^+ = CJKL$, $FD^- = FCZZ$, $FD^+ = FDZZ$ となる。 S_1 を

$$S_1 = \{ a^-, a^+ \mid a \in S \}$$

とし、 S_1 をソートしたものを S_2 とする。 Fig. 4.1 の例では、順序集合は、

$$S_2 = \{ BZZZ, CAZZ, CBBZ, CBCZ, CBZZ, CJKK, CJKL, CZZZ, FCZZ, FDZZ \}$$

となる。

Σ^* の要素の内、長さが m のものを Σ^m と表わす。 Σ^m は辞書配列に従う順序集合である。 各 $a \in S$ は、 a^- , a^+ によつて Σ^m を最大3つの区間に分ける。

$u \in \Sigma^m$ に対して、

if $u \leq a^-$ then u の先頭部分 a^- と一致しない。

if $a^- < u \leq a^+$ then u の先頭部分が a に一致する.

if $a^+ < u$ then u の先頭部分は a と一致しない.

のような関係がある。 S_2 の要素は順序集合 Σ^m を Fig. 4.2 のように分割する。

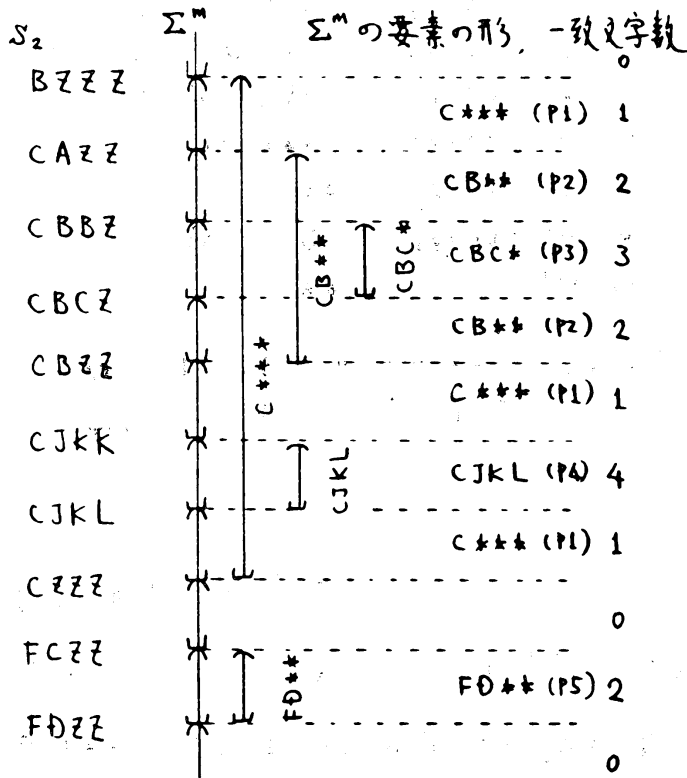


Fig. 4.2 S_2 による Σ^m の分割

後に、4文字長の語表を持つ SEE があり、例の S_2 がこの SEE に格納されている、 $u \in \Sigma^4$ が入力されたとする。このとき、 u と出力されるテーブルアドレスには Fig. 4.3 のような関係

係がある。

テーブルアドレス	u	一致文字数
0, 0* (BZZZ)	****	0
1, 1* (CAZZ)	C*** (P1)	1
2, 2* (CBBZ)	CB** (P2)	2
3, 3* (CBCZ)	CBC* (P3)	3
4, 4* (CBZZ)	CB** (P2)	2
5, 5* (CJJK)	C*** (P1)	1
6, 6* (CJKL)	CJKL (P4)	4
7, 7* (ZZZZ)	C*** (P1)	1
8, 8* (FCZZ)	****	0
9, 9* (FDZZ)	FD** (P5)	2
10, 10*	****	0

Fig. 4.3 S_2 を u をサーチした際の出力アドレスと u の関係

M文字長 (例では $m=4$) の SEE を用いるかわりに、前章のアルゴリズム B と同様に、文字やテーブルアドレスを2つずつ組にして処理する。例えば BZZZ は BZ と ZZ に分割される。 S_2 の各要素をこのように分割したときできる2文字タームの集合をリストし、重複を除去すると、例題の S_2 に対しては、 $\{BZ, CA, CB, CJ, CZ, FC, FD, KK, KL, ZZ\}$ が得られる。このi番目の2文字タームを $(i-1)^*$ の値にコード化する。その結果 S_2 の各要素は Fig. 4.4 (a) のようにコード化される。この

1-トをさらに、 Z^* の要素を2個づつに区切り、同様のコード化を行う。このとき、 $m=2^L$ のストリングマッチングシステムは Fig. 4.4 のような構成で、SEE を1段連結したものである。処理の仕組みは図より明らかであろう。

このストリングマッチングシステムは、SEE のレベル数を L とするとき、表さか、

$$\log m \times L + (m-1)$$

のパイプと見做すことができ、サブジェクトストリングは文字の流れとして1文字づつこのパイプに流れ込み、各文字が流出するときには Fig. 4.4 の出力部に示すようなストリングマッチングの情報が付加されて流れる。サブジェクトストリングが、このパイプ中を流れるだけで、一度に多くのパターンとの照合がとれる点が、本システムの特長である。(Fig. 4.5)

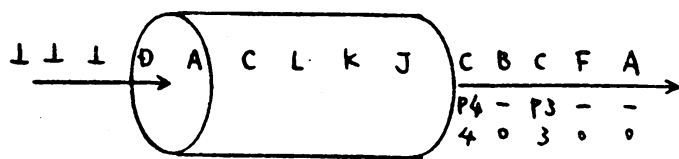


Fig. 4.5 ストリングマッチング
モジュールのスリーム処理

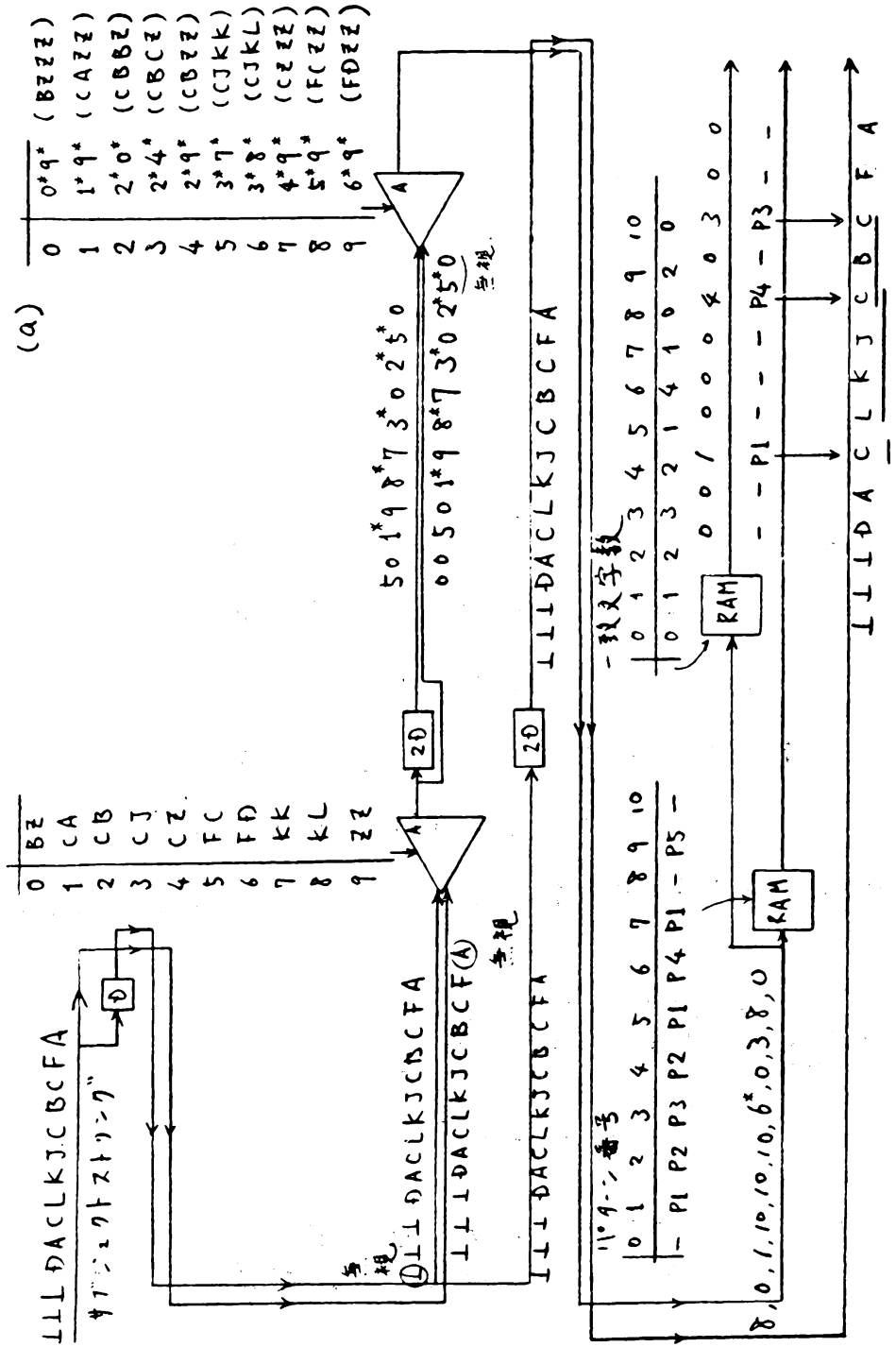


Fig. 4.4. SEE 全用... 構成 (ファスト) リング マチング
 毛ジニニ

5. 結論

データベースの高速処理を目指して開発したSER, SOEが拡張性と実現性の両方において、VLSI化に向けたアーキテクチャであることを述べ、これらがVLSI化されることにより、サーチソートと基本演算と見做し得るようになる場合には、どのような効果か期待され得るかを、ストリングマッチングを例にとり示した。さらに高度な応用として、論理型言語の推論機構部への応用等を検討中である。

本論文中で提案したストリングマッチングシステムは、Fig. 4.5のような1つの機能モジュールと見做すことができ、これを組み合わせることにより、ワードプロセッシングやテキストプロセッシングへの応用が期待される。

参考文献

- (1) Y. Tanaka et al., "Pipeline Searching and Sorting Modules as Components of a Data Flow Database Computer," Proc. IFIP Congress 80, 1980 (Tokyo), pp. 429-432.
- (2) 田中 譲, 'データベースストリング処理方式のデータベースシステム', "情報処理研究, 記号処理 12-14, 1980, pp. 97-103.