数理解析研究所講究録
322巻 1978年 264-285

不成功マッチング処理に適した部分マッチングアルゴリズム

京大・工学部　上林 彌彦
中津 楢男
矢島 脩三

1. まえがき

与えられた文字系列の集合の中から、別に与えられた部分系列の集合の要素を含むものを選び出す操作を部分マッチングと呼ぶ。部分マッチングのための効率の良いアルゴリズムを求めることはオートマトン理論との関連のみならず〔KNUTM74〕，効率良いテキストエディタの作成や、標題やアブストラクトに含まれる単語や熟語による文献検索を行なう場合に非常に有用であると考えられる〔AHO-C7506〕〔ARIKT76〕〔BOYEM7710〕。

与えられた長さnの系列ひが，与えられた長さ m $(n \geq m)$ の系列 ω を部分系列として含むかどうかを調べるには，簡単な方法では $O(m \cdot n)$ だけの計算時間を必要とする。Knuthらはこれを $O(m+n)$ に改良した。この方法を用いて長さnの系列ひが，与えられた部分系列 $\omega_1, \omega_2, \cdots \omega_k$（但し、それらの

長さの総和をmとする）を含むかどうかを調べるには，このアルゴリズムをn回適用するとよく，その計算時間は$O(m+n\cdot n)$となる。HopcroftとKarpはこれを$O(m\cdot k+n)$に改良した。ここでkは入力記号の数である。Ahoらは，それと等価な操作をパタンマッチングマシンと呼ばれる有限オートマトンで表現することによって，計算時間を$O(m+n)$に改良している。

計算の複雑さの次数の上からはこれ以上の改良はできないので，ここでは部分マッチングの起らないための十分条件を利用して計算を途中で打切ったり，入力系列内の文字の飛ばし読みを行なうことにより計算速度の向上をはかる試みについて検討する。特に文献検索システムにおける文献標題の中の単語の部分マッチングへの応用を想定して，その場合に有用な部分マッチングの手法について検討した。この様な場合には，部分マッチングの対象となる系列集合は，与えられた種々の部分系列集合に対して何度も部分マッチ検索が行なわれるので，部分マッチ検索に適した処理をあらかじめ行なっておくことが考えられる。この様な処理の1つとして，ここでは不在文字情報を利用した部分マッチングアルゴリズムを与える。その他，部分マッチングの起らないための十分条件も列挙してみた。また最近BoyerとMooreが，部分マッチン

グをなすべき部分系列がただ1つの場合についてKnuthらの方法を改良するため、系列の飛ばし読みを行なう方法を提唱している〔BOYEM7710〕。ここでは、これらのアルゴリズムの一般化に関した検討も行なっている。

最後に、日本科学技術情報センターの文献検索用磁気テープを用いて、部分マッチングアルゴリズムの実験を行なったのでその結果を示している。それによれば、入力系列（文献標題）内の文字の比較回数は、最も簡単な方法に比べて、Ahoらのパタンマッチングマシンを用いた場合には約1/2～1/3になり、一方不在文字情報を用いた最も簡単な方法でも比較回数は約1/4～1/9になることがわかった。また与えられた部分系列の要素数nが小さい場合、或いは、不成功マッチングの割合の大きい場合には、実行時間もAhoらのパタンマッチングマシンよりも短いことが実験で確かめられた。但し、各プログラムはPL/Iで開発したため、正確な比較とは言い難い。

アルゴリズム理論の研究においては、理論的に可能な計算量の下限を満足するアルゴリズムの開発の他に、その様なアルゴリズムの係数部分をいかに減少させるかという点も、実用上有用であると考えている。

2. 諸定義

$\Sigma=\{a_1, a_2, \cdots, a_k\}$を記号の有限集合とし，$\Sigma$の要素によって生成される可能な全ての系列の集合（空系列も含む）をΣ^*とする。$|\Sigma|$は集合Σの要素数を示し，この場合$|\Sigma|=k$である。Σ^*の要素uについて，uの長さがmである場合，$l(u)=m$で表わす。Σ^*の要素Λとλを結合した系列は単に$\Lambda\lambda$で表わすものとする。Σ^*の要素λ，u，v，ωについて$v=\lambda\omega u$の形で表現できる時，ωはvの部分系列であるという。

〔定義１〕　系列vの系列ωによる部分マッチングとは，Σ^*の要素vとωが与えられた場合に，ωがvの部分系列になっているかどうかを調べることである。このωをキーパタン，vを被検査系列と呼ぶことにする。

$l(v)=n$，$l(\omega)=m$とする。vの各文字を先頭として系列ωがその位置から続くかどうか調べ，ωと一致しない文字を発見すればvの次の文字を先頭にして同様の手続きを繰り返す。この最も簡単な方法によれば，計算時間はほぼ$O(m\cdot n)$となる。次の例は，Knuthらによるもので，$n=2i+1$，$m=i+1$の場合である。比較回数は上記の方法で$(i+1)^2$となる。

〔例　１〕

記号の有限集合：$\Sigma=\{A, B\}$

キーパタン：$w = A^i B$

被検査系列：$\nu = A^{2i} B$

Knuth, Morris, Pratt〔KNUTM74〕は，これを$O(m+n)$に改良した。このアルゴリズムをAhoとCorasickは，パタンマッチングマシンという概念を導入して有限オートマトンを用いて定式化した〔AHO-C7506〕。

〔定義２〕　系列νの系列集合Wによる部分マッチングとはΣ^*の要素νとΣ^*の部分集合Wが与えられた場合に，各$w \in W$がνの部分系列になっているかどうかを調べることである。Wをキーパタン集合という。

Ahoらのパタンマッチングマシンは，キーパタン集合Wによる部分マッチングの方法として，Knuthらの方法をHopcroftとKarpが一般化したもの〔KNUTM74〕を更にオートマトンの形式で表わしたものであり，オートマトンを作る計算時間の面で改良されている。つまり，パタンマッチングマシン(PMM)は，記号の有限集合Σとキーパタン集合Wが与えられると一意に決まる有限オートマトンで，その受理集合はWに等しい。

〔例　２〕

Ahoらの論文で示されている，Σが英語のアルファベットでキーパタン集合$W = \{he, she, his, hers\}$の場合のPMMの状態遷移図を図１に示す。但し簡単のために，状態S_0, S_1,

S_7 への遷移枝は省略してある。

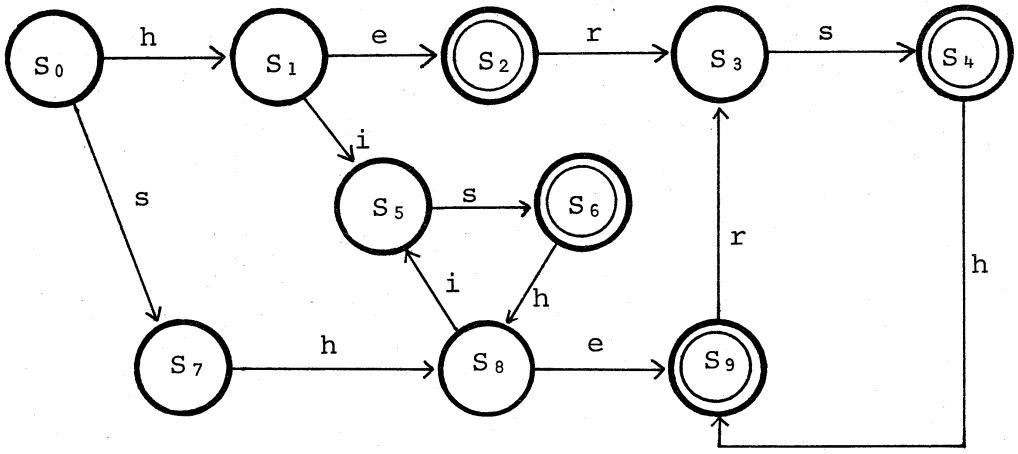


図1　キーパタン集合 W=[he,she,his,hers] に対するパタンマッチングマシンの例

PMMを作るのに要する計算時間は、$m=\sum_{w\in W} l(w)$ として $O(m)$ であり、このマシンに被検査系列 v ($l(v)=n$) を入力して調べるのに $O(n)$ だけの時間がかかり、全体として $O(m+n)$ の時間で処理できる。

Boyerらは、キーパタン集合の要素数が1つの場合について、被検査系列内の文字をとばし読みする方法によってサブリニアアルゴリズムを提案している。Boyerらのアルゴリズムを次の例で示す。

〔例　3〕

記号の有限集合：$\Sigma=\{A,B,C,D\}$

キーパタン集合：$W=\{CABA\}$，被検査系列：$v=$ ABBACB DCABAAC　とし，現在検査中の v の文字を↓で示す。

```
(1) A B B A C B D C A B A A C
      C A B A
(2) A B B A C B D C A B A A C
      C A B A
(3) A B B A C B D C A B A A C
      C A B A
(4) A B B A C B D C A B A A C
            C A B A
(5) A B B A C B D C A B A A C
                  C A B A
(6) A B B A C B D C A B A A C
                  C A B A
(7) A B B A C B D C A B A A C
                  C A B A
(8) A B B A C B D C A B A A C
                  C A B A
                                Match
```

	A	B	C	D
$delta_1$	0	1	3	4
	1	2	3	4
$delta_2$	7	6	3	1

この方法によれば、計算時間の複雑さは同じく$O(m+n)$であるが、被検査系列の全てを調べる必要がないという理由から$O(m+\alpha \cdot n)$ $(0<\alpha<1)$の計算時間で処理できる。ここで、ベクトル$delta_1$と$delta_2$はとばし読み間隔を指定するベクトルであり、$delta_1$は不一致になった文字の情報、$delta_2$は不一致になった位置の情報を利用するものである。

3. 不在文字情報を利用した部分マッチング

系列長の合計がm $(m=\sum_{w \in W} l(w))$である様なパタンマッチ

ングマシンをつくるのに$O(m)$の計算時間がかかり，これにより長さnの系列を検査するのに$O(n)$の時間がかかる。これらの次数をこれ以上下げることは不可能なので更に計算時間を短くするには，これらの次数の係数部分を減少させなければならない。Boyer, Mooreはそのための1つの方法を提唱しているが，ここでは検査される系列に前処理を行なう方法を提案する。文献検索システムの様に，対象とする文献標題集合が与えられていて，利用者の要求するキーパタンを組合せてキーパタン集合を作り，それらのキーパタンを含む文献標題を選び出す場合，文献標題集合をあらかじめ前処理しておいて，キーパタン集合に対する部分マッチングの処理速度を上げるようにすることは有効であると考えられる。論文の標題の長さは平均して50～60文字であるので，その中にアルファベットの全ての文字が現われるというよりは，特定の文字が出現しない可能性が大きい。そこで，文献標題の前処理としてその標題に含まれていない文字を示す情報を付加しておけば，それによって標題の検査を省略したり，キーパタン集合を小さくすることができる。例えば，例2の$W=\{he, she, his, hers\}$の場合では，もし標題の中にnが含まれていないことがわかれば，その標題の検査をする必要はなくなる。また標題の中にeが含まれていないことがわかれば，$W'=\{$

his} だけを対象にすれば良いことがわかる。このため，各標題に対して次に定義する $|\Sigma|$ ビットのベクトル B_0 を用意する。

〔定義３〕　不在文字ベクトル B_0 は，部分マッチングの対象とする各系列について定義されている。$\Sigma=\{a_1, a_2, \cdots a_k\}$ として，系列の中に a_j が含まれていなければ $B_0(j)=1$ とし，そうでなければ $B_0(j)=0$ とする。

英語のアルファベットを対象とする場合には，B_0 は計算機の１語（32ビット/語）で表わすことができる。不在文字を調べるには，あらかじめ標題を１度調べておけば良いので，標題の長さを n として $O(n)$ の計算時間を要する。

〔アルゴリズム１〕　不在文字ベクトルを利用した系列の部分マッチング

(1) 記号集合を $\Sigma=\{a_1, a_2, \cdots, a_k\}$ とし，キーパタン集合を $W=\{w_1, w_2, \cdots, w_k\}$ とし，検査すべき系列をひとする。B_0 はひの不在文字を示す k 個の要素から成るベクトルである。各 $w_i \in W$ について，C_i を次の様に定義する。C_i も k 個の要素から成るベクトルであり，$C_i(j)$ は w_i に a_j が含まれていれば１，そうでなければ０である。

(2) $i=1$，$X=W$ とする。

(3) $B_0 \wedge C_i$ を行ない，その結果が全ての要素が０のベクトルかどうかを調べる。もしそうならば (5) へ，そうでなけれ

ば(4)へ。　（ベクトルの∧（論理積）は，ベクトルの各要素ごとに行なうものとする。）

(4) Xよりw_iを除く。

(5) $i=i+1$。$i=n+1$となれば(6)へ，そうでなければ(3)へ。

(6) $X=\phi$ならばひはWの要素を部分パタンとして含まない。

$X\neq\phi$ならば，Xに対するパタンマッチングマシンを作ってAho.らの方法を適用する。■

文献標題を対象にする場合には上記の方法で十分であるが文献のアブストラクト等の様に対象とする系列が$|\Sigma|$よりもかなり長い場合には，この方法による検査時間の短縮は余り望めない。この場合には系列をある長さeごとに区切って，各部分系列ごとに不在文字ベクトルを用意すればよい。その場合、２つの部分系列にまたがってキーパタンが存在する場合の効率良い処理が必要である。その部分系列内にはキーパタンが存在しない場合でも，キーパタンの先頭部分が含まれる可能性があるので、部分系列の最後のg文字を再検査する必要がある。このgをできるだけ小さくするためにも，不在文字ベクトルが利用できる。

〔アルゴリズム２〕　系列を分割した場合の不在文字ベクトルを利用した系列の部分マッチング

(1) 入力記号集合を$\Sigma=\{a_1,a_2,\cdots,a_k\}$として，キーパタン

集合を $W=\{w_1, w_2, \cdots, w_h\}$，被検査系列をひとする。ひを長さeごとに区切った系列を $v_1, v_2, \cdots, v_p$ とする。各部分系列 v_i に対応する不在文字ベクトルをそれぞれ Bo_i とする。ここで、eはキーパタンの最大長のものより長いことを仮定する。（この仮定は、妥当な仮定である。）

各 $w_i \in W$ について、C_i を w_i に a_j が含まれていれば $C_i(j)=1$ そうでなければ $C_i(j)=0$ となるベクトルと定義する。

(2) $i=1$，$X=W$ とする。

(3) $j=1$ とする。

(4) $Bo_i \wedge C_j$ を利用して、アルゴリズム1と同様に w_j が v_i に含まれていないための十分条件を満足していれば X より w_j を除く。$j=j+1$ として、$j=h+1$ となれば (5) へ，そうでなければ (4) の手続きを繰り返す。

(5) $X=\phi$ ならば v_i の中にWのどの要素も含まれないので、系列 v_i のうしろより g 番目の文字よりWの要素の先頭部分が現われているかどうかを調べる。g は下記の方法で求めることができるが，$\max_{w \in W}[l(w)]-1$ を用いてもよい。

$X \neq \phi$ ならば，系列 v_i の中にXの要素が含まれているかどうかを Aho らの方法で調べる。但し，v_i のうしろより g 番目の文字からは，Wの要素の先頭部分が現われているかどうかを調べる。g の値は次に示す r_k の最大値である。r_k は

各 $w_k = w_k(1)\, w_k(2) \cdots\cdots$ について定義され、$w_k(r_k+1) = a_l$ とした時、$B_{oi}(l) = 1$ となる最小の整数とする。(但し、$r_k \leq l(w_k)-1$)

この時、$g = \max_{w_k \in W} r_k$ で求めることができる。

(6) $i = i+1$ とし、$i = p+1$ となれば検査は終了する。そうでなければ B_{oi} を利用して b_{i-1} のうしろの部分に先頭部分の含まれているWの部分集合に対する処理をした後、$X = W$ として(3)へ戻る。 ■

不在文字ベクトルを更に次のようなものに変更すれば、キーパタンが含まれないための十分条件を改善することができる。〔1〕ちんなどの様に、系列の中によく現われる文字の連についても存在するかどうか調べておく(この存在を調べる操作も部分マッチングなので、線形アルゴリズムでできる)。〔2〕2文字の全ての連について調べると場合の数が勿過ぎるので、たとえば被検査系列内の偶数位置の不在文字ベクトルと奇数位置の不在文字ベクトルの2つを用意する。この組合せによって系列中に現われない文字の連を調べ得る。

次節の実験によれば、この奇偶を考慮した不在文字ベクトルは、実用上非常に有効であることがわかった。この場合、この奇偶を考慮した不在文字ベクトルを、位置情報としても利用することによってBoyerらのアルゴリズム等を改良することができる。つまり、不在文字ベクトルを調べることによ

り、キーパタンwがマッチするのは被検査系列ひの奇数番目の位置に限る、或いは偶数番目の位置に限るといった場合が生じる。それらの場合には、それぞれひの奇数番目、偶数番目の文字だけを調べればよく、より速く処理できる。具体的には、とばし読み間隔を大きくすることができる。例えば、次の例はBoyerらのアルゴリズムによれば、余り効率の良くない例である。

〔例 4〕

記号の有限集合：$\Sigma=\{A,B\}$　キーパタン集合：$W=\{AAAB\}$

被検査系列：ひ＝AAAAAAAAAAAB

奇偶を考慮した不在文字ベクトル：ODD=|0|1|, EVEN=|0|0|

Boyerらのアルゴリズムによれば、12回の比較を必要とするが、不在文字ベクトルを位置情報として利用すれば、キーパタンAAABがマッチするのはキーパタンの右端の文字がひの偶数番目の文字と一致する場合に限る。従って、8回で処理が終了する。その手順を以下に示す。

(1) A A A Ȧ A A A A A A B
　　A A A B

(2) A A A A A Ȧ A A A A B
　　A A Ȧ B

(3) A A A A A A A Ȧ A A B
　　A A A B

(↓は検査中の文字を示す)

```
(4) A A A A A A A A A A A B
                A A A B
(5) A A A A A A A A A A A B
                  A A A B
(6) A A A A A A A A A A A B
                  A A A B
(7) A A A A A A A A A A A B
                  A A A B
(8) A A A A A A A A A A A B
                  A A A B
```

Match

	A	B
$delta_1$	2	0

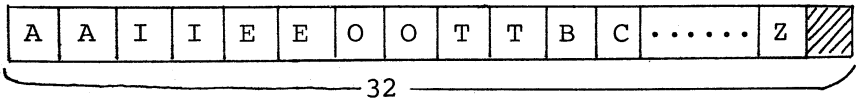
	1	2	3	4
$delta_2$	7	6	5	2

4. 不在文字情報を用いた部分マッチングアルゴリズムの実験

不在文字情報の有用性を確かめるために、日本科学技術情報センター(JICST)の文献検索用磁気テープ(英字・カナ文字モード電気工学編)のデータを利用して実験を行なった。データとしては、JICSTの磁気テープの内、77年度No.3, 8, 9, 13, 16より計算機工学関係の文献1080件余り(但し、日本語文献を除く)を抜き出し、文献標題と文献識別子(JICST文献識別子)及び不在文字情報より成る新しいファイルを作成使用した。ここでは次にあげる4種類のプログラムを作成し与えられたキーパタン集合に対して、部分マッチングでマッチした文献の文献識別子とその時のキーパタンを出力させる

ことにした。4つのプログラムとは，まず，各キーパタンに対して1文字ごとの比較を行なう方法（CBC），部分マッチングの生じないための十分条件を利用しまずキーパタン集合を小さくしてから1文字ごとの比較をする方法（CBCS及びCBCEO），及びAhoらのパタンマッチングマシン（PMM）であり，各々プログラム化を行ない，それぞれ文献標題中の文字の比較回数を調べた。ただし，十分条件としては，CBCSは単なる不在文字ベクトル，CBCEOは奇偶を考慮した不在文字ベクトルを使用している。

不在文字ベクトルの構成は，次のとおりである。



奇偶を考慮した不在文字ベクトルは，同じ構造を持つ2つのベクトルEVEN，ODDより成り，EVENは被検査系列の偶数番目の位置にある文字に対する不在文字ベクトルであり，ODDは奇数番目の位置にある文字に対する不在文字ベクトルである。

PMMのプログラム化は次のようにして行なった。オートマトンの各状態S_kは次の情報を持つ。

EFIN(k)：状態S_kにおける有効入力数。

{INPUT(k,1), NEXT(k,1)}, ···, {INPUT(k,EFIN(k)), NEXT(

k, EFIN(k)}：状態S_kにおける入力記号とそれに対して遷移すべき次状態の組。

OUTN(k)：状態S_kにおける出力数。

OUT(k)：状態S_kにおける出力キーパタン集合へのポインタ。

実験は各種のキーパタン集合に対して行なった。その1例としてキーパタン集合W={NETWORK, SWITCHING, FUZZY, SUPERVISOR, RELATIONAL}に対する、比較回数で分類した文献数の分布を表1に示す。

比較回数＼方法	CBC	CBCS	CBCEO	PMM
0- 100	0	486	794	27
100- 200	22	337	202	271
200- 300	83	122	36	443
300- 400	157	76	34	221
400- 500	202	27	7	94
500- 600	229	19	7	18
600- 700	153	9	6	10
700- 800	93	4	-	3
800- 900	59	4	1	-
900-1000	46	2	-	-
1000-1100	21	1	-	-
1100-1200	8	-	-	-
1200-	14	-	-	-
mean	560	142	59	260
CPU time	58	25	28	41
Execution time	67	42	37	49

表1　比較回数で分類した文献数

また各実験に対する，各方法における，1文献当りの比較回数を表2に示す。

実験	方法 / キーパタン数	CBC	CBCS	CBCEO	PMM
1	3	337	127	52	158
2	5	560	142	59	266
3	3	339	164	74	167
4	4	563	303	127	300
5	6	675	368	153	299
6	9	1011	520	235	301
7	10	1123	520	261	300

表2　1文献当りの平均比較回数

この実験によれば，キーパタン集合の要素数が小さくて，マッチする割合が小さい場合には，PMMよりもCBCS，CBCEOの方が比較回数においても実行時間においても優れていることが明らかになった。また，単なる不在文字ベクトルに比べ奇偶を考慮した不在文字ベクトルが非常に有効であることがわかった。CBCとCBCEOを比較すれば，比較回数は1/4～1/9になり，実行時間も大幅に改善されることがわかり，十分条件によるキーパタンの選択が，部分マッチングのアルゴリズムとして実用的にもかなり有効であることがわかった。従ってアブストラクトなどの長い複検査系列の場合にも，それを区切って，各区間ごとに不在文字ベクトルを用意する方

法も有効であろうと考えられる。一方，キーパタンの数が多くなってくるとPMMが有効であることが確かめられた。現実問題としては，キーパタン集合の要素数はそれほど多くないと考えられるので，不在文字情報だけでも十分実用的と考えることができる。

5．キーパタンの分解による部分マッチングアルゴリズムの改良

3節で述べたアルゴリズムは，テキストエディタ等の様に，被検査系列が頻繁に変化する場合においては不在文字ベクトルを変更する必要があるので余り有効とはいえない。また，4節で示した実験結果によれば，キーパタンの数が多くなればAhoらのパタンマッチングマシンが有効であることが確かめられた。ここでは，キーパタンの分解により被検査系列の一部だけを調べる方法を提案する。Boyerらの示したアルゴリズムの中で，不一致になった文字の情報（$delta_1$）を利用する場合は，ここで述べる方法の特殊な場合である。また，この方法は，キーパタン集合が複数の要素を持つ場合でも，そのまま適用できるといった利点がある。

与えられたキーパタンw（その長さを$l(w)$とする）に対して，wをN個（ただし$N \leq l(w)$）の部分系列$w_1 w_2 \cdots$

w_N に分解する。ここで w_i ($1 \leq i \leq N-1$) は，w の $i \pmod N$ 番目の文字からのみ成る部分系列であり，w_N は，w の $0 \pmod N$ 番目の文字からのみ成る部分系列とする。検査の途中で被検査系列 v が w_i を含むことがわかれば、その位置において v が w を含むかどうかを調べればよい。

〔アルゴリズム3〕 キーパタンの分解による部分マッチング

(1) キーパタン w に対して上記の様に $w_1, w_2, \cdots, w_N$ を作りそれらに関するパタンマッチングマシンMを構成する。

（但し，$w_i = w_j$ $(i \neq j)$ ならば，w_j を無視し，w_i に対する最終状態において，それを記憶しておく）

(2) $j = 0$ とする。

(3) $j = j + N$ とし，$j > \ell(v)$ ならば終了。

(4) 文字 $v(j)$ を，パタンマッチングマシンMに入力する。もし w_k ($k = 1, 2, \cdots, N$) のどれもMによって受理されなければ (3) へ。w_k が受理されれば (5) へ。

(5) 位置 j において，w が v の部分系列になっているかどうか調べ，マッチすれば v は w を部分系列として含んでいる。そうでなければ (3) へ。 ■

ここでステップ (5) において，w が v にマッチするか否かは，マッチする可能性のある場合だけを調べれば良い。アル

ゴリズム3において、$N=\ell(w)$とすれば、Boyer らのアルゴリズムと基本的には同じになり，同様の効率が期待できる。しかし、記号の有限集合Σにおいて，$|\Sigma|$が小さい場合にはBoyer らのアルゴリズムでは効率の悪いことが多い。その場合には，Nを適当に選んでアルゴリズム3を適用すれば、より速く処理することができる。次の例は，$\ell(w)=4$の場合であるが、Boyer らのアルゴリズムによれば12回の比較を必要とするが、アルゴリズム3を適用すれば、$N=4$の場合で10回の比較回数，$N=2$の場合は7回の比較で処理できる。

〔例　5〕　$N=2$ の場合のアルゴリズム3の適用例

記号の有限集合：$\Sigma=\{A, B\}$

キーパタン集合：$W=\{ABAB\}$

被検査系列　　：$v=AAABBAABAB$

パタンマッチングマシンMに入力する文字を↓で示す。

(1) A ↓A A B B A A B A B

to State S_1

(2) A A A ↓B B A A B A B

to State S_3

(3) A A A B B ↓A A B A B

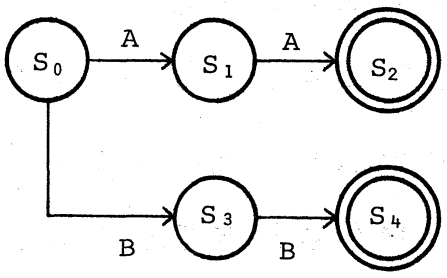
to State S_1

(4) A A A B B A A ↓B A B

to State S_3

(5) A A A B B A A B A ↓B

to State S_4 (Final State)



$w_1=AA$, $w_2=BB$に対応したパタンマッチングマシン

(6) A A A B B A A B A B
Check

(7) A A A B B A A B A B
Match
Check

このアルゴリズムは、特に被検査系列 v の長さが長い場合、キーパタンの長さが長い場合及び入力記号集合の要素数が小さい場合に特に有効であると考えられる。また、キーパタンが複数個ある場合でも、$N \leq \min_{w \in W} l(w)$ とすれば、アルゴリズム3がそのまま適用できる。パタンマッチングマシン M は $m = \sum_{w \in W} l(w)$ として、やはり $O(m)$ で構成することができ、検査に必要な比較回数は、$n = l(v)$ として $n/N + \beta \cdot m$ $(\beta \geq 1)$ である。

6、あとがき

部分マッチングをより効率良く行なうために、不在文字情報を利用したアルゴリズムとその有用性を示した。また、パタンマッチングマシンを利用して Boyer らのサブリニアアルゴリズムの一般化をおこなった。キーパタン集合の要素数が多い場合の部分マッチングに関しても、不成功マッチングのための十分条件を利用して、動的にパタンマッチングマシンを小さくしてゆき、処理の効率を上げる方法についても現在検討中である。

最後に、貴重な御意見・御示唆をいただいた本学矢島研究室の諸氏に感謝致します。

〔参考文献〕

[KNUTM74] Knuth,D.E.,Morris,J.H.Jr., and Pratt,V.R.
"Fast Pattern Matching in Strings", Res. Rep.,
STAN-CS-74-440, Stanford Univ., 1974

[AHO-C7506] Aho,A.V. and Corasick,M.J., "Fast Pattern
Matching:An Aid to Bibliographic Search"
CACM, vol 18, No.6, pp. 333-340, Jun. 1975

[ARIKT76] 有川，武谷，石綿
"パタン・マッチング・マシンを用いる例文検索システム"
昭和51年　情報処理大会　61

[BOYEM7710] Boyer,R.S. and Moore,J.S., "A Fast String
Searching Algorithm", CACM, vol. 20, No. 10
pp. 762-772, Oct. 1977