

文字列集合の共通パターンを求める問題について

愛知教育大学 中津 植男
京大・工学部 上林 彌彦

1. まえがき 本稿では、与えられた文字列集合に対して各要素が共通して持つパターンを求める問題について考察する。ここではパターンとして文字列が定義されているアルファベットの要素と変数の連接を考えている。

例えば、 $\{000, 033, 088, 01515\}$ なる集合は $0x^2x$ という共通パターンを持っている。また、 $\{721327, 331633, 931939\}$ なる集合は x^2yx^R というパターンを持っている。この問題はデータ圧縮や集合の特徴を見つける時などに合う問題である。従来与えられた文字列集合に共通した性質を求める1つの方法として、与えられた文字列集合に対して各要素を含む最小の正規集合を求める問題等が考えられてきた。

また与えられた正系列を受理し、負系列を拒否する状態数最小のオートマトンについても議論されている。

ここでは変数を導入して各要素が共通して持つパターンを

求める問題を考える。この問題は Angluin によって議論されている [1]。

本稿ではまず共通パターンの持つ性質を明らかにし、1変数共通パターンを求めるアルゴリズムを与える。このアルゴリズムは、前処理とデータ構造を工夫することにより従来のアルゴリズムの大巾な改善となっている。2変数以上の共通パターンについては余り良い結果は得られなかった。

2. 基本的定義 Σ を記号の有限集合とする。以下では、

Σ 上で定義された文字列を考える。 Σ の要素を単に定数と呼ぶことがある。 Σ とは異なる記号の有限集合 X ($X \cap \Sigma = \emptyset$) を考える。 X の要素を変数と呼ぶ。

与えられた文字列集合を $S = \{s_1, s_2, \dots, s_m\}$ (但し、 $s_i \in \Sigma^+$, $1 \leq i \leq m$) とする。一般性を失うことなく、各文字列は全て異なり、 $|s_1| \leq |s_2| \leq \dots \leq |s_m|$ であると仮定する。

[定義1] パターン P は $(\Sigma \cup X)^+$ の要素で、長さ有限長のもので定義する。つまり、パターン P は Σ の要素と X の要素を有限個連接したものである。パターン P の長さとは、 P の中に現れる定数の数と変数の数の和とする。 P の長さを、 $|P|$ と書くことにする。

[定義2] パターン P に現れる変数 x を全て、別のパター

んで置きかえることによって新しいパターン q が得られる時 $P \geq q$ と書く。但し、 P の中の同じ変数は全て同じパターンで置き換えるものとする。

〔定義3〕 パターン P が生成する文字列集合 $L(P)$ を次の様に定義する。 $L(P) = \{w \mid w \in \Sigma^+, w \leq P\}$ 。

与えられた文字列集合 S に対して、 $S \subseteq L(P)$ を満たすパターン P を S の共通パターンと呼ぶ。 $S \subseteq L(q) \subset L(P)$ なる2つのパターン P, q が存在する時、 P は冗長な共通パターンであると言う。また、その様なパターン q が存在しない時、 P を冗長でない共通パターンと呼ぶ。

〔例1〕 $\Sigma = \{0, 1\}$, $P = 0x^2x$ とした時 $L(P) = \{w \mid w = 0tt, t \in \{0, 1\}^+\}$ である。また $S = \{1111, 111111\}$ とした時、 $1x1x$ は冗長でない共通パターンであるが、 x^2x は冗長な共通パターンである。

3. 1変数共通パターンの性質 本節では与えられた文字列集合の1変数共通パターンの性質について述べる。

〔性質1〕 (i) 1変数共通パターン P, q に対して

$$q \leq P \iff L(q) \subseteq L(P) \text{ が成り立つ。}$$

〔性質2〕 冗長でない共通パターンは与えられた文字列集合に対して一意には決らない。一般に相異なる1変数共通パ

ターンの数は、冗長でないものだけを考えても、与えられた文字列の長さに対して指数関数的に増大する場合がある。

〔例 2〕 与えられた文字列集合を $\{A_1 = a^{2^k}, A_2 = a^{2^{k+1}}, \dots, A_m = a^{2^{k+n-1}}\}$ とする。 $n = |A_1| = 2^k$ とする。

この時、任意の整数 i ($0 \leq i \leq k$) について、 2^i 個の変数 x と $2^k - 2^i$ 個の定数 a の組合せで得られるパターンは全て冗長でない共通パターンである (性質 1)。従って、冗長でない変数共通パターンの数は、 $\sum_{i=0}^k 2^i C_{2^i} = \sum_{i=0}^k n C_{2^i} = n C_1 + n C_2 + n C_4 + \dots + n C_{2^k} + n C_n$ とあり存在する。

〔性質 3〕⁽¹⁾ 集合 S の 1 変数共通パターン P の中に変数 x が i 回出現しているとする。この時 i ($0 < i \leq |A_1|$) は、整数集合 $\{|A_2| - |A_1|, |A_3| - |A_2|, \dots, |A_m| - |A_{m-1}|\}$ の公約数でなければならない。また、文字列 A_j の中で x に置き換えられた部分列の長さ l_j とした時 $l_{j+1} = l_j + (|A_{j+1}| - |A_j|) / i$ が成り立つ。

〔定義 4〕 2 つのパターン P_1, P_2 について、 $P \in \Sigma^*$ が $P \leq P_1$ が $P \leq P_2$ が $|P| \leq |A_1|$ であり、 $\exists \leq P_1$ が $\exists \leq P_2$ が $P \leq \exists$ となる様なパターン \exists が存在しない時、 P を P_1 と P_2 の積と言い、 $P_1 \wedge P_2$ と書く。上記の条件を満たす P が存在しない時、 $P_1 \wedge P_2$ は定義されない。

〔例 3〕 $S = \{aabb, aababbbb\}$ とする。 $P_1 = aa\alpha$, $P_2 = \alpha b$ とした時、 $P_1 \wedge P_2 = aa\alpha b$ である。また $P_1 = aa\alpha$

$P_3 = \alpha b b$ とした時、 $P_1 \wedge P_3$ は定義されない。

[性質4] 2つの1変数共通パターン P_1, P_2 において、 $P = P_1 \wedge P_2$ が定義されている時、 $L(P) = L(P_1) \cap L(P_2)$ となる。

[性質5] P_1, P_2 を与えられた文字列集合 S の1変数共通パターンとする。この時、 $P_1 \wedge P_2$ が定義されており、 $P_1 \wedge P_2$ が1変数パターンであれば、 $P_1 \wedge P_2$ は S の1変数共通パターンである。逆に、 $P_1 \wedge P_2$ が1変数共通パターンであれば、 P_1, P_2 のいずれか一方は冗長な共通パターンである。

4. 1変数共通パターンを求めるアルゴリズム

本節では与えられた文字列集合 $S = \{s_1, s_2, \dots, s_m\}$ の冗長でない1変数共通パターンを求める効率良いアルゴリズムを示す。Angluinは性質3を利用して、 $n = |s_m|$ とした時、 $O(mn^5)$ の計算時間を要するアルゴリズムを示した⁽¹⁾。ここでは $\text{Max}(O(mn^2), O(mn^3))$ の計算時間を要するアルゴリズムを示す($n = |s_i|$)。ただし、 S の冗長でない1変数共通パターンの数は n に関して指数関数的に増大する場合があるので、ここでは全ての解を示す構造のみを求めるアルゴリズムを与える。

[定義5] 与えられた文字列 $\lambda = \lambda(1)\lambda(2)\dots\lambda(n)$ ($\lambda(i) \in \Sigma$, $1 \leq i \leq n$)に対して $\lambda' = \lambda \lambda$ ($\lambda \in \Sigma$)を考える。 λ' の i -位置指示

系列 (i -th position identifier) W_i とは次の条件を満足する \mathcal{A} の部分系列である。

- ① $W_i = \mathcal{A}'(i) \mathcal{A}'(i+1) \cdots \mathcal{A}'(j)$ ($i \leq j \leq n+1$) である。
- ② $i=j$ の時, $W_i = \mathcal{A}'(i)$ は \mathcal{A}' の中にただ1回だけ現れる記号である。 $i < j$ の時, W_i は \mathcal{A}' の中にただ1回だけ現れる部分系列でありしかも, $\mathcal{A}'(i) \mathcal{A}'(i+1) \cdots \mathcal{A}'(j-1)$ は \mathcal{A}' の中に少なくとも2回以上現れる部分系列である。

[定義6] 文字列 $\mathcal{A} = \mathcal{A}(1)\mathcal{A}(2)\cdots\mathcal{A}(n)$ に対する prefix 木とは次の条件を満足する木である。 $\mathcal{A}' = \mathcal{A}\lambda$ とする。

- ① prefix 木は $n+1$ 個の葉節点を持ち, 各葉節点には $1 \sim n+1$ の整数が1意にラベル付けされている。
- ② 各枝には, $\Sigma \cup \{\lambda\}$ の要素がラベル付けされている。
- ③ prefix 木の根節点よりラベル i の葉節点へ到る道上のラベルを順に連接して得られる系列は, \mathcal{A} の i -位置指示系列である。

[例4] $\Sigma = \{a, b\}$,
 $w = aabab a$ の prefix 木
 を図1に示す。例えば,
 $abab$ は w の 2-位置
 指示系列である。

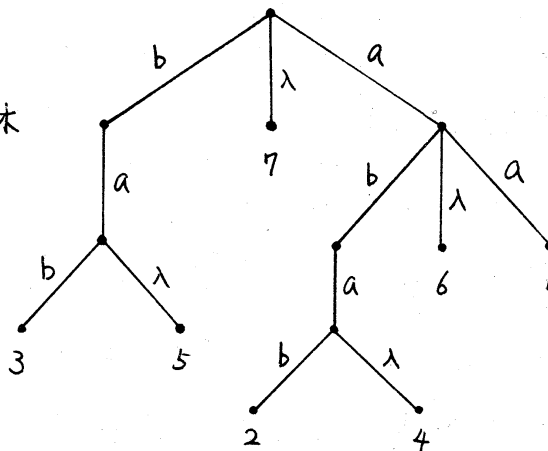


図1 prefix 木の例

〔性質6〕 文字列 Λ の中で繰り返し現れる文字列は全て、 Λ の prefix 木に表現されている [2]。

1 変数共通パターンを求める為に、各文字列の中で繰り返し現れる文字列を求める必要がある。次に、prefix 木を用いて全ての繰り返し現れる文字列を効率良く求めるためのアルゴリズムを示す。Prefix 木の根節点と葉節点以外の節点を内部節点と呼ぶことにする。

〔アルゴリズム 1〕 繰り返し系列を求めるアルゴリズム 4

- (1) 文字列 Λ に対する prefix 木 (T と書く) を構成する。
- (2) 内部節点 P を根節点とする T の部分木の全ての葉節点のラベルの集合を P に対応づける。
- (3) 節点 P の深さを $d(P)$ とし、ステップ (2) によって得られた集合を $\{i_1, i_2, \dots, i_k\}$ とした時、 $\{\Lambda(i_1)\Lambda(i_1+1)\dots\Lambda(i_1+d(P)-1), \dots, \Lambda(i_k)\Lambda(i_k+1)\dots\Lambda(i_k+d(P)-1)\}$ が求める長さ $d(P)$ の繰り返し系列である。

アルゴリズム 1 終り

〔定理 1〕 $|\Lambda| = n$ とした時、アルゴリズム 1 は $O(n^2)$ の計算時間を必要とする。

〔証〕 ステップ (1) は $O(n)$ の時間を必要とする [2]。全ての内部節点 P に対して、ステップ (2) は T をエンドオーダーに 1 度だけトラバースすることによって実行される。 T の節点数は $O(n)$ であることが知られている。従って、 T のトラバース

には $O(n)$ の時間を要する。一方各内部節点は、 $O(n)$ の大きさのラベル集合を持つ。従ってアルゴリズム 1 は全体として $O(n^2)$ の計算時間を必要とする。 Q.E.D.

次に本節で示すアルゴリズムで利用するデータ構造を定義する。定義 7 で示すパターン網は prefix 木を用いて得られた繰り返し系列を利用して簡単に構成できる。

〔定義 7〕 大きさ $l \times l$ の網とは、 $(l+1) \times (l+1)$ 個の節点を持ち、次の条件を満たす有向グラフである。

(1) 各節点は 2 値組 (i, j) ($0 \leq i \leq l, 0 \leq j \leq l$) で表現され、枝は (i, j) より $(i, j+1)$ 或いは $(i+1, j)$ に向けてのみ存在する。

(2) 各枝には $\Sigma \cup X$ の要素がラベル付けされている。

〔定義 8〕 文字列 w に対する (i, j) パターン網とは大きさ $i \times j$ の網であり、定数 i 個、変数 j 個より成る $\{w\}$ の冗長でない 1 変数共通パターンが全て、点 $(0, 0)$ より点 (i, j) に到る有向道で表現されているものを言う。

〔性質 7〕 w の (i, j) パターン網において $(0, 0)$ より (i, j) への有向道が存在するための必要十分条件は、 w の中に長さ $(|w| - i) / j$ の文字列が少なくとも j 回繰り返し出現していることである。但し、文字列が重なってはならない。

〔例 5〕 $w = a a b a b a$ に対する $(2, 2)$ パターン網を図 2 に示す。ここで変数 α_1 は文字列 "ab", α_2 は文字列 "ba" を示

している。w の (2,2) パター
 網を求める為に、図1の
 prefix 木を用い、長さ2で
 少なくとも2回繰り返してw
 の中に現れる文字列を求め
 れば良い。

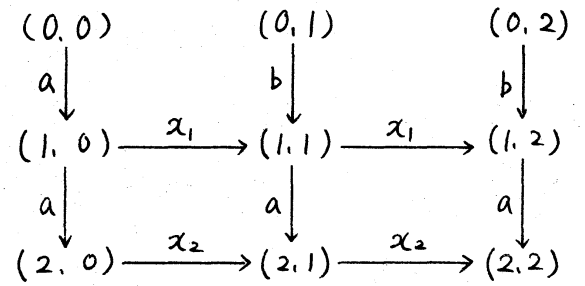


図2 (2,2) パターン網の例

次に文字列wの (i,j) パターン網を構成するためのアルゴリズムを示す。ただし、 $(|w|-i)/j$ は整数とする。

〔アルゴリズム42〕 (i,j) パターン網を求めるアルゴリズム

- (1) 文字列wにアルゴリズム1を適用し、wのprefix木の全ての内部節点Pについて、対応したラベル集合を求める。
- (2) $(i+1) \times (j+1)$ 個の節点を作る。 $h = (|w|-i)/j$ とする。
- (3) $\forall k, l (0 \leq k \leq i-1, 0 \leq l \leq j)$ について、節点 (k, l) から節点 $(k+1, l)$ へ枝を作り、記号 $w(hl+k+1)$ をラベル付けする。
 $u = 0$ とする。
- (4) wのprefix木の深さhの新しい内部節点Pを求める。Pがなければアルゴリズムは終了。
- (5) $u = u + 1$ とする。Pに対応したラベル集合(ステップ(1))を考える。そのラベル集合の各要素 i_t についてステップ(6)を行なう。
- (6) $i_t = 1 + k + hl$ を満足する全ての整数 k, l に対し、点 (k, l)

より点 $(l, l+1)$ の枝をつくり、ラベル α_u をつける。

(7) ステップ (4) へゆく。 アルゴリズム 2 終り

〔定理 2〕 アルゴリズム 2 は、 $|w| = n$ とした時、
 $O(n^2) + O(ij)$ の計算時間を必要とする。

〔証〕 ステップ (1) は、定理 1 より $O(n^2)$ の計算時間を必要とする。ステップ (2) 及びステップ (3) はそれぞれ、 $(l+1) \cdot (j+1)$ 、 $i \cdot (j+1)$ の計算量を要する。ステップ (4) は prefix 木をトラバースすればよいため $O(n)$ の時間で処理できる。ステップ (5) ~ (6) によって、パターン網の右向き枝が作られるが、各枝のラベルは一意に決定される。従って、(5) ~ (6) で必要な計算時間は作られる枝数に比例し、枝数は高々 $(l+1) \cdot j$ 本である。従って全体として、 $O(n^2) + O(3(l+1)(j+1))$ の計算量を要する。Q.E.D.

文字列 $w \in S$ に対する (i, j) パターン網は、 i, j を決めればその構造が一意に決まる。また S の 1 変数共通パターンの内、定数 i 個、 α (変数) j 個より成るパターンがあれば、定義 δ より、それは w の (i, j) パターン網に表現されている。この事実により、 S の各要素に対する (i, j) パターン網の共通部分を求めれば、 S の 1 変数共通パターン網が求まる。公式なアルゴリズムを次に示す。

〔アルゴリズム 3〕 1 変数共通パターンを求めるアルゴリズム

(1) 各 $s_i \in S$ に対して、アルゴリズム 1 を適用する。

(2) 整数集合 $\{|s_2| - |s_1|, |s_3| - |s_2|, \dots, |s_m| - |s_{m-1}|\}$ の公約数の集合 (以後 CD と書く) を求める。

(3) for $j=1$ until $|s_1|$ do begin
 if $j \in CD$ then begin

(4) for $h=1$ until $\lfloor |s_1|/j \rfloor$ do begin

$i \leftarrow |s_1| - h \cdot j$

s_1 について, (i, j) パターン網 $N_1(i, j)$ を作る。

for $k=2$ until m do begin

s_k の (i, j) パターン網 $N_k(i, j)$ と $N_1(i, j)$ の
 共通部分を改めて $N_1(i, j)$ とする。

end

(5) $N_1(i, j)$ において, 点 $(0, 0)$ より点 (i, j) へ到る有向道が存在すれば, $N_1(i, j)$ は解を与える構造である。

(6) end

(7) end

アルゴリズム 3 終り

[定理 3] アルゴリズム 3 は, $\text{Max}(O(mn^2), O(mn^3))$ の計算時間を必要とする。但し $n_1 = |s_1|$, $n = |s_m|$ とする。

((証)) ステップ (1) は定理 1 より, 高々 $O(mn^2)$ の計算時間を要する。ループ (3) ~ (7) 及びループ (4) ~ (6) は, それぞれ最悪の場合, n_1 回及び $\lfloor n_1/i \rfloor$ 回繰り返される。ループ (4) ~ (6) の中で, 各文字列 s_k ごとに (i, j) パターン網が構成される。但し実際には, s_1 の (i, j) パターン網上での操作が行なわれるのみである。ステップ (5) は, $(i+1)(j+1)$ の時間で処理できる。

従って、ステップ(3)~(7)に対して次の評価式を得る。

$$\begin{aligned} & \sum_{j=1}^{n_1} \sum_{k=1}^{\lfloor n_1/j \rfloor} \left\{ (l+1)(j+1) + i(j+1) + (l+1)j + \sum_{k=2}^m (i(j+1) + (l+1)j) + (l+1)(j+1) \right\} \\ & \leq \sum_{j=1}^{n_1} \sum_{k=1}^{\lfloor n_1/j \rfloor} \left\{ 2(l+1)(j+1) + m(2ij + l+j) \right\} < (m+2) \sum_j \sum_h (l+1)(j+1) \\ & = (m+2) \sum_j \sum_h (j+1)(n_1 - h_j + 1) = O(mn_1^3) \end{aligned}$$

従って、全体として定理3の結果を得る。 Q.E.D.

[例6] $S = \{ aababaa, abbabbaa \}$ の1変数共通パターンの中で、定数2、変数2のものを考える。

$\Delta_1 = aababaa$ の(2,2)パターン網は図2に示すとおりである。 $\Delta_2 = abbabbaa$ の(2,2)パターン網を図3に示す。2つの(2,2)パターン網の共通部分を図4に示す。

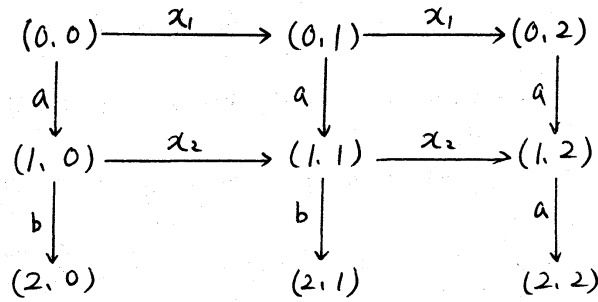


図3 $\Delta_2 = abbabbaa$ の(2,2)パターン網

図4より、 S の1変数共通パターンの1つは、 $a \times \times a$ であることがわかる。また、定数2、変数2個より成る1変数共通パターンはこれ以外にない。

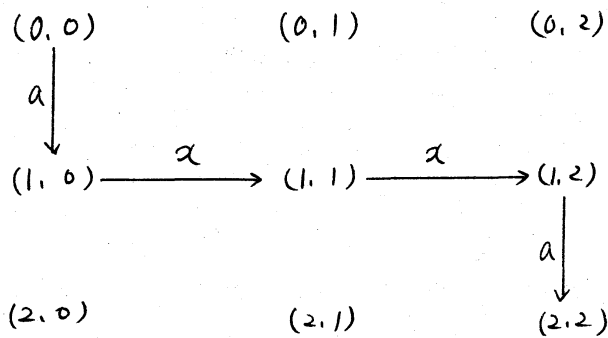


図4. 図2と図3の(2,2)パターン網の共通部分

5. 多変数共通パターンについて

任意の種類の変数を許した場合に、与えられた文字列集合 S に対し、 S の冗長でない共通パターンの内で長さ最大のものを見つけることは NP-hard であることが既に知られている⁽¹⁾。この結果からみても、一般の共通パターンを求めることは非常に難しいと言える。1変数の場合に成立していた性質1も、2変数以上では成立しない。また、1変数共通パターンの場合には、変数と定数の個数を決めれば与えられた文字列に対して、変数が表現する文字列の長さは一意的に決定されたが、2変数以上の場合にはこれも成り立たない。また、性質2の様に実行時間の減少に寄与する性質も判明していない。そこで本節では、変数の種類に制限を加えて、共通パターンの問題を考える。すなわち、 Σ 上の文字列から文字列への同型写像の集合を $\{f_i\}$ とした時、 $\{f_i(x)\}$ なる変数の集合のみを考えることにする。

同型写像の有限集合を $\{f_1, f_2, \dots, f_k\}$ (但し、 f_1 は恒等写像とする) とした時、変数として $f_i(x)$ ($1 \leq i \leq k$) を用いた共通パターンを $O(kmn^3)$ で求めるアルゴリズムを示すことができる。まず、例を用いて我々のアルゴリズムを説明する。

〔例7〕 $\Sigma = \{a, b\}$ とし、変数として x 及び x^R を許した共通パターンを考える。与えられた文字列集合を $S = \{aababa,$

$abb\ a\ a\ a\ b\ b\}$ とする。S の共通パターン内の、定数2個、変数2個より成るパターンを求める為に、(2,2) パターン網を構成する。その為に、 $\Lambda_1 = aababab$ の長さ2の部分列 $\Lambda_{1,2}$ と $\Lambda_{1,2}^R$ の出現位置を全て求める。また、 $\Lambda_2 = abb\ a\ a\ a\ b\ b$ の長さ3の部分列 $\Lambda_{2,3}$ と $\Lambda_{2,3}^R$ の出現位置を全て求める。例えば $\Lambda_{2,3} = abb$ とした時、 $\Lambda_{2,3}$ と $\Lambda_{2,3}^R$ の Λ_2 における出現位置を全て求める為に、図5に示す様な有限オートマトンを考えればよい [3][4]。ここで状態0が

初期状態、状態3,6が最終状態である。 Λ_1, Λ_2 の(2,2)パターン網を各々、図6,7に示す。図6において、 x_1, x_2 は各々長さ2の文字列 ab, ba を示している。図7において、 x_1, x_2, x_3, x_4, x_5 は各々 abb, bba, baa, aab, aab を示している。

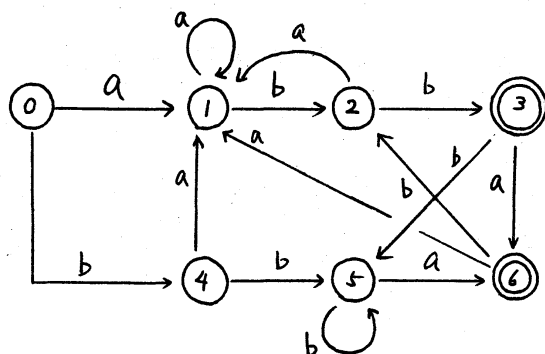


図5 $\{abb, bba\}$ を受理する有限オートマトン

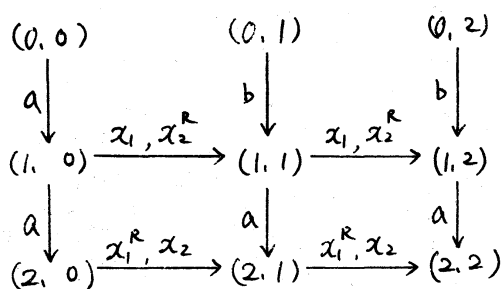


図6 Λ_1 の(2,2)パターン網

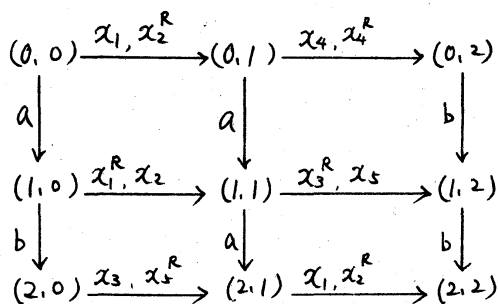


図7 Λ_2 の(2,2)パターン網

図6と図7の共通部分を考えることにより、定数2、変数2個のSの共通パターン $a^2 a^2$, $a^2 a^2$ を得る。

公式なアルゴリズムを次に示す。

[アルゴリズム4]

(1) for $t=1$ until m do begin

(2) for $P=1$ until $|S_t|$ do begin

文字列 S_t の長さ P の部分列を $S_{t,p}$ とする。

全ての $S_{t,p}$ について以下の処理を行なう。

(3) 文字列集合 $\{f_1(S_{t,p}), f_2(S_{t,p}), \dots, f_k(S_{t,p})\}$ を受理する有限オートマトンを構成する。

その有限オートマトンに S_t を入力することにより、文字列 $f_j(S_{t,p})$ ($1 \leq j \leq k$) の出現位置を全て求める。

(4) end

(5) end

(6) アルゴリズム3のステップ(2)以下を適用する。

アルゴリズム4 終り

[定理4] アルゴリズム4の計算時間は、 $O(kmn^3)$ である。但し、 $n = |S_m|$ 、 k は同型写像の数である。

(証) 文字列 S_t の長さ P の部分列は高々 $|S_t| - P + 1$ 種類存在する。ステップ(3)は $O(kP|S_t|)$ の時間で計算できる^{[3][4]}。故にステップ(1)~(5)は全体として、 $O(kmn^3)$ の計算量を必要とする。一方、ステップ(6)では、アルゴリズム3の場合と同様の計算時間を要するが、パターン網の右向き枝には、高々

長種のラベルが付けられる可能性がある。従ってステップ(6)は $O(kmn^3)$ の計算時間を要する。 Q.E.D.

6. あとがき 本稿では1変数共通パターンを求める有効なアルゴリズムと、変数を x 及び y に同型写像を施した変数に制限した場合の2変数共通パターンを求めるアルゴリズムを示した。本稿で述べたパターンの形を拡張して、より一般的なパターンを定義することができる。しかしながら、このようなパターンまで含めて一般的な共通パターンを見つけ出すことは非常に難かしい問題である。

謝辞

御討論頂いた京都産業大学・岩間一雄先生に感謝します。また日頃より貴重な御助言を賜わる京都大学情報工学教室・矢島脩三先生に感謝致します。

[参考文献]

- [1] Angluin, D., "Finding Patterns Common to a Set of Strings", 11th ACM Symposium on Theory of Computing, pp. 130-141, 1979.
- [2] Weiner, P., "Linear Pattern Matching Algorithms", 14th Annual Symposium on Switching and Automata Theory, pp. 1-11, 1973.
- [3] Aho, A.V. and Corasick, M.J., "Efficient String Matching: An Aid to Bibliographic Search", CACM, vol. 8, no. 6, pp. 333-340, June 1975.
- [4] Kambayashi, Y., Nakatsu, N. and Yajima, S., "Hierarchical String Pattern Matching Using Dynamic Pattern Matching Machines", The Proceedings of IEEE COMPSAC'79, pp. 813-818, Nov. 1979.