Theoretical Computer Science 92 (1992) 77–86
Elsevier

Two recognizable string-matching problems over free partially commutative monoids*

Kosaburo Hashiguchi and Kazuya Yamada

Department of Information and Computer Sciences, Toyohashi University of Technology, Tempaku, Toyohashi 441, Japan

Abstract

Hashiguchi, K. and K. Yamada, Two recognizable string-matching problems over free partially commutative monoids, Theoretical Computer Science 92 (1992) 77–86.

The two string-matching problems over free partially commutative monoids are studied and analyzed in detail in order to present efficient linear-time algorithms for solving these two problems over a constant-size alphabet.

1. Introduction

Let Σ be a finite alphabet, and Σ^* the free monoid generated by Σ. λ denotes the empty word. One of the typical string-matching problems over Σ^* is the following:

Given a text string $x \in \Sigma^*$ and a pattern string $y \in \Sigma^*$, decide whether or not y is a factor of x.

Many efficient algorithms for this string-matching problem are known, cf. [1, 2, 5, 9, 14].

Recently many contributions about free partially commutative monoids have also appeared [3, 4, 6, 7, 10–13, 15]. We recall its definition briefly. Let θ be an irreflexive, symmetric binary relation over Σ. $\equiv_\theta$ (or $\equiv$ simply) denotes the smallest equivalence relation over Σ^* such that for any $x, y \in \Sigma^*$, $x \equiv y$ if $x = uabv$ and $y = ubav$ for some $(a,b) \in \theta$ and $u, v \in \Sigma^*$. Then $\equiv$ is a congruence relation. $M(\Sigma, \theta)$ denotes the quotient of Σ^* by the congruence $\equiv$. $M(\Sigma, \theta)$ is the free partially commutative monoid generated by Σ w.r.t. θ, and can be regarded as a model of concurrency control system, or a model of any system with finitely many partially commutative operations. For

* Extended abstract

0304-3975/92/$05.00

any $x, y \in \Sigma^*$, if $x \equiv uyv$ for some $u, v \in \Sigma^*$, then we call y a θ-factor of x; moreover, if $u = \lambda$, then y is a θ-prefix of x, and if $v = \lambda$, then y is a θ-suffix of x.

We study the following two problems over $M(\Sigma, \theta)$. Let $x, y \in \Sigma^*$ be a given text string and a pattern string, respectively.

Problem A: Decide whether or not y is a θ-factor of x.

Problem B: Decide whether or not x has a prefix of which y is a θ-suffix.

Problem B may be regarded as a hybrid problem concerning Σ^* and $M(\Sigma, \theta)$. We analyze these two problems in detail, and obtain two efficient algorithms solving these two problems. The two algorithms have certain similar characters and consist of two parts. The first part consists of constructing functions $\rho_{a,b}$ as in [1] to each $\pi_{a,b}(y)$, where $a, b \in \Sigma$, $a \neq b$, $(a, b) \notin \theta$, and $\pi_{a,b}(y)$ is the string in Σ^* obtained from y by deleting all letters distinct from a and b.

The running time of this part is $O(|y| \cdot |\Sigma^2|)$. The second part of the algorithm for Problem A (Problem B) consists of scanning x once from left to right with proper transitions in the above functions, and deciding whether or not y is a θ-factor of x (y is θ-suffix of some prefix of x). The running time of this part is $O(|x| \cdot |\Sigma^3|)$.

This article is an extended abstract of [8]: only Theorem 7 is a new observation.

2. Main results

Let $\bar{\theta}$ denote the set of pairs (a, b) such that $a \neq b$, $a, b \in \Sigma$ and $(a, b) \notin \theta$. Σ_c is the set of $a \in \Sigma$ such that $(a, b) \in \theta$ for any distinct $b \in \Sigma$. Γ is a binary relation over Σ^* such that for any $u, v \in \Sigma^*, u \Gamma v$ iff for any $(a, b) \in \Sigma(u) \times \Sigma(v)$, $(a, b) \in \theta$. θ^* is a binary relation over Σ^* such that for any $u, v \in \Sigma^*$, $u\, \theta^*\, v$ iff for any $(a, b) \in \Sigma(u) \times \Sigma(v)$, either $a = b$ or $(a, b) \in \theta$.

The congruence $\equiv$ can be characterized by simultaneous equations over Σ^*: the following theorem is fundamental.

Theorem (Cori and Perrin [4]). *For any $u, v \in \Sigma^*$, $u \equiv v$ iff the following conditions hold*:

(1) *For any $a \in \Sigma$, $|u|_a = |v|_a$.*

(2) *For any $(a, b) \in \bar{\theta}$, $\pi_{a,b}(u) = \pi_{a,b}(v)$.*

The following two propositions hold.

Proposition 2.1. *For any $x, y \in \Sigma^*$, y is a θ-factor of x iff the following conditions hold*:

(1) *For any $a \in \Sigma_c$, $|x|_a = |y|_a$.*

(2) *There exists a prefix $x_{b,c}$ of $\pi_{b,c}(x)$ for each $(b, c) \in \bar{\theta}$ for which the following conditions hold*:

(2.1) *$x_{a,b}\pi_{b,c}(y)$ is a prefix of $\pi_{b,c}(x)$;*

(2.2) *For any $(b, c), (b, d) \in \bar{\theta}$, $|x_{b,c}|_b = |x_{b,d}|_b$.*

Proposition 2.2. *For any $x, y \in \Sigma^*$, y is a θ-suffix of some prefix u of x iff the following conditions hold*:

(1) *For any* $a \in \Sigma_c$, $|u|_a \geq |y|_a$;
(2) *For each* $(b,c) \in \bar{\theta}$, $\pi_{b,c}(y)$ *is a suffix of* $\pi_{b,c}(u)$.

We shall first develop the results which we need for solving Problem A.

Proposition 2.3. *Let* $u, y, t \in \Sigma^*$, *and assume that* y *is* θ*-factor of* ut. *Then there exist* $\alpha, \beta, \gamma, \delta \in \Sigma^*$ *such that* (1) $\alpha\beta$ *is a* θ*-suffix of* u, (2) $\alpha\delta \equiv y$, (3) $\gamma\delta$ *is a* θ*-prefix of* t, *and* (4) $\alpha\beta\,\Gamma\,\gamma$ *and* $\beta\,\Gamma\,\delta$.

Definition. Let $u, y \in \Sigma^*$.

(1) An extensible pair of (u,y) is a pair (α,β) such that (i) $\alpha, \beta \in \Sigma^*$, (ii) $\alpha\beta$ is a θ-suffix of u, and (iii) for some $\gamma \in \Sigma^*$, $\alpha\gamma \equiv y$ and $\beta\,\Gamma\,\gamma$.

(2) An extensible, 2-maximal pair of (u,y) is an extensible pair (α,β) of (u,y) with $|\beta|$ maximum, that is, $|\beta| = \max\{|\beta'| \mid \beta' \in \Sigma^*$ and (α',β') is an extensible pair of (u,y) for some $\alpha' \in \Sigma^*\}$.

(3) An extensible, (1,2)-maximal pair of (u,y) is an extensible, 2-maximal pair (α,β) of (u,y) with $|\alpha|$ maximum, that is, $|\alpha| = \max\{|\alpha'| \mid \alpha' \in \Sigma^*$ and (α',β') is an extensible, 2-maximal pair of (u,y) for some $\beta' \in \Sigma^*\}$.

Notation. For any $u, y \in \Sigma^*$, $\langle u, y\rangle$ denotes any extensible, (1–2)-maximal pair of (u,y): see Theorem 2.5.

Proposition 2.4. *Let* $u, y \in \Sigma^*$ *and* (α,β) *be an extensible pair of* (u,y). *Then for any* $a \in \Sigma(\beta)$ and $b \in \Sigma$ with $(a,b) \in \bar{\theta}$, $\pi_{a,b}(\alpha) = \pi_{a,b}(y)$.

Theorem 2.5. *Let* $u, y \in \Sigma^*$.

(1) *Let* (α_1,β_1) and (α_2,β_2) *be two extensible pairs of* (u,y). *Then there exists an extensible pair* (α,β) *of* (u,y) *such that* (i) β_1 *and* β_2 *are* θ*-suffixes of* β, *and* (ii) α_1 *and* α_2 *are both* θ*-prefixes and* θ*-suffixes of* α.

(2) $\langle u, y\rangle$ *is unique up to the congruence* $\equiv$.

Notation. For any $u, v, w, t \in \Sigma^*$, $(u,v) \equiv (w,t)$ means $u \equiv w$ and $v \equiv t$.

Theorem 2.6. *Let* $u, y \in \Sigma^*$, $a \in \Sigma$, $\langle u, y\rangle \equiv (\alpha_1,\beta_1)$ *and* $\langle ua, y\rangle \equiv (\alpha_2,\beta_2)$. *Then* $\alpha_2\beta_2$ *is a* θ*-suffix of* $\alpha_1\beta_1 a$.

We need the following proposition and corollary for efficiency of our algorithm solving Problem A.

Notation. *For any* $\alpha, \beta \subset \Sigma^*$ *and* $B \subset \Sigma$, $\pi_B(\alpha,\beta)$ *denotes* $\langle \pi_B(\alpha), \pi_B(\beta)\rangle$.

Proposition 2.7. *Let* $B, C \subset \Sigma$ *be such that* $B \cup C = \Sigma$ *and* $B\,\Gamma\,C$. *Then for any* $u, y \in \Sigma^*$, $\pi_B(\langle u, y\rangle) \equiv \langle \pi_B(u), \pi_B(y)\rangle$.

Corollary 2.8. *Let $u, y \in \Sigma^*$ and $a \in \Sigma$. Assume that there exist $B, C \subset \Sigma$ such that $B \cup C = \Sigma$, $B \,\Gamma\, C$ and $a \in C$. Then $\pi_B(\langle ua, y \rangle) = \pi_B(\langle u, y \rangle)$.*

Now we shall develop the results for solving Problem B.

Definition. Let $u, y \in \Sigma^*$.

(1) An extensible word of (u, y) is $\alpha \in \Sigma^*$ such that α is a θ-suffix of u and a θ-prefix of y.

(2) A maximal extensible word of (u, y) is an extensible word α of (u, y) with $l(\alpha)$ maximum, that is, $l(\alpha) = \max\{l(\alpha') \mid \alpha' \text{ is an extensible word of } (u, y)\}$.

Notation. $[u, y]$ denotes any maximal extensible word of (u, y): see the following theorem.

Theorem 2.9. Let $u, y \in \Sigma^*$.

(1) *Let $\alpha_1, \alpha_2 \in \Sigma^*$ be two extensible words of (u, y). Then there exists an extensible word $\alpha \in \Sigma^*$ of (u, y) such that α_1 and α_2 are both θ-prefixes and θ-suffixes of α.*

(2) *$[u, y]$ is unique up to the congruence $\equiv$.*

Theorem 2.10. *Let $u, y \in \Sigma^*$ and $a \in \Sigma$. Then $[ua, y] \equiv [[u, y]a, y]$.*

The following proposition and corollary are necessary for efficiency of our algorithm solving Problem B.

Proposition 2.11. *Let $B, C \subset \Sigma$ be such that $B \cup C = \Sigma$ and $B \,\Gamma\, C$. Then for any $u, y \in \Sigma^*$, $\pi_B([u, y]) \equiv [\pi_B(u), \pi_B(y)]$.*

Corollary 2.12. *Let $u, y \in \Sigma^*$ and $a \in \Sigma$. Assume that there exist $B, C \subset \Sigma$ such that $B \cup C = \Sigma$, $B \,\Gamma\, C$ and $a \in C$. Then $\pi_B([ua, y]) \equiv \pi_B([u, y])$.*

3. Algorithms solving Problems A and B

We shall first present algorithms solving Problem B. The following is a rather implicit algorithm solving Problem B, whose correctness is clear from Theorems 2.9 and 2.10.

Algorithm B.1

Input: A text string $x = a_1 \ldots a_n$, $n \geqslant 1$, $a_i \in \Sigma$, $1 \leqslant i \leqslant n$, and a pattern string $y \in \Sigma^+$
Output: "ACCEPT" if y is a θ-suffix of some prefix of x;
"REJECT" otherwise

begin
$i \leftarrow 1$; $t \leftarrow \lambda$; $s \leftarrow$ false;
while $s =$ **false and** $1 \leqslant i \leqslant n$ **do**

begin
$t \leftarrow [ta_i, y]$;
if $|t| = |y|$, **then**
begin
write "ACCEPT";
$S \leftarrow$ **true**
end
else $i \leftarrow i+1$
end
if $S =$ **false**, **then** write "REJECT"
end

Notation. Let $u \in \Sigma^*$. When $u \neq \lambda$, $[u]$ denotes the longest word which is both a proper prefix and a proper suffix of u. We put $[\lambda] = \lambda$.

Definition. Let $(b,c) \in \bar{\theta}$.

(1) $\rho_{b,c}$ is the function from $\mathrm{Pre}(\pi_{b,c}(y))$ to $\mathrm{Pre}(\pi_{b,c}(y))$ such that for any $u \in \mathrm{Pre}(\pi_{b,c}(y))$, $\rho_{b,c}(u) = [u]$.

(2) $\rho_{b,c}^{(1)} = \rho_{b,c}$ and for $k \geqslant 1$, $\rho_{b,c}^{(k)} = \rho_{b,c} \cdot \rho_{b,c}^{(k-1)}$.

(3) $\psi_{b,c}$ is the failure function from $\mathrm{Pre}(\pi_{b,c}(y)) \cdot \{b,c\}$ to $\mathrm{Pre}(\pi_{b,c}(y))$ such that for any $w \in \mathrm{Pre}(\pi_{b,c}(y))$ and $d \in \{b,c\}$,

(3.1) $\psi_{b,c}(wd) = \rho_{b,c}^{(m)}(w)d$ if m is the least positive integer such that $\rho_{b,c}^{(m)}(w)d \in \mathrm{Pre}(\pi_{b,c}(y))$;

(3.2) $\psi_{b,c}(wd) = \lambda$ if such an m does not exist.

For the proof of the following proposition, see [1].

Proposition. *For any* $(b,c) \in \bar{\theta}, w \in \mathrm{Pre}(\pi_{b,c}(y))$ *and* $d \in \{b,c\}$, $\psi_{b,c}(wd)$ *is the longest word in* $\mathrm{Pre}(\pi_{b,c}(y)) \cap (\mathrm{Suf}(wd) - \{wd\})$.

Definition. $G(\Sigma, \bar{\theta})$ is the finite undirected graph whose vertices are letters of Σ and whose edges are those $\{a,b\}$ such that $(a,b) \in \bar{\theta}$. Let $\{C_1, \ldots, C_e\}$ be the set of connected components of $G(\Sigma, \bar{\theta})$, and for each $1 \leqslant i \leqslant e$, let V_i be the set of vertices of C_i.

Notation. For each $1 \leqslant i \leqslant e$, π_i denotes the function π_{V_i}.

Now we have the following more precise implementation of Algorithm B.1.

Algorithm B.2.

Input: A text string $x = a_1 \ldots a_n$, $n \geqslant 1$, $a_i \in \Sigma$, $1 \leqslant i \leqslant n$, and a pattern string $y \in \Sigma^+$
Output: "ACCEPT" if y is a θ-suffix of some prefix of x;
"REJECT" otherwise

begin
  Obtain $\pi_a(y)$ for each $a \in \Sigma_c$ and $\pi_{b,c}(y)$ for each $(b,c) \in \bar{\theta}$;
  Construct $\rho_{b,c}$ for each $(b,c) \in \bar{\theta}$;
  $t_a \leftarrow \lambda$ for all $a \in \Sigma_c$; $t_{b,c} \leftarrow \lambda$ for all $(b,c) \in \bar{\theta}$;
  $s \leftarrow$ false; $i \leftarrow 1$;
  while $s =$ **false and** $1 \leqslant i \leqslant n$ **do**
    (* where $a_i \in V_j$, $1 \leqslant j \leqslant e$, and $B = \bar{V}_j$ *)
  begin
    if $a_i \in \Sigma_c$, **then** $t_{a_i} \leftarrow$ the shortest word of $t_{a_i} a_i$ or $\pi_{a_i}(y)$
    else
    begin
      if for all $b \in \theta(a_i)$, $t_{a_i,b} a_i \in \mathrm{Pre}(\pi_{a_i,b}(y))$, **then**
        $t_{a_i,b} \leftarrow t_{a_i,b} a_i$ for all $b \in \bar{\theta}(a_i)$
      else
      begin
        $t_{a_i,b} \leftarrow \psi_{a_i,b}(t_{a_i,b} a_i)$ for all $b \in \bar{\theta}(a_i)$ with
          $t_{a_i,b} a_i \notin \mathrm{Pre}(\pi_{a_i,b}(y))$;
        $\varepsilon_b \leftarrow \min\{|t_{b,c}|_b \mid (b,c) \in V_j \times V_j \cap \bar{\theta}\}$ for all $b \in V_j$;
        while for some $(b,c) \in V_j \times V_j \cap \bar{\theta}$, $|t_{b,c}|_b > \varepsilon_b$, **do**
        begin
          if $b = a_i$ or $c = a_i$, **then** $t_{b,c} \leftarrow \psi_{b,c}(t_{b,c} a_i)$;
          else $t_{b,c} \leftarrow \rho_{b,c}(t_{b,c})$;
          $\varepsilon_b \leftarrow \min\{\varepsilon_b, |t_{b,c}|\}$
        end
      end
    end
    if $|t_a| = |\pi_a(y)|$ for all $a \in \Sigma_c$ **and**
      $|t_{b,c}| = |\pi_{b,c}(y)|$ for all $(b,c) \in \bar{\theta}$, **then**
    begin
      write "ACCEPT";
      $s \leftarrow$ **true**
    end
    else $i \leftarrow i + 1$
  end
  if $s =$ **false**, **then** write "REJECT"
end

Theorem 3.1. *The running time of Algorithm* B.2 *is* $O(|xy| \cdot |\Sigma|^3)$.

Next we shall present algorithms solving Problem A. We first present the following implicit algorithm solving Problem A.

Algorithm A.1

Input: A text string $x=a_1\ldots a_n$, $n\geqslant 1$, $a_i\in\Sigma$, $1\leqslant i\leqslant n$, and a pattern string $y\in\Sigma^+$

Output: "ACCEPT" if y is a θ-factor of x;
"REJECT" otherwise

begin
  $i\leftarrow 1$; $\alpha\leftarrow\lambda$; $\beta\leftarrow\lambda$; $s\leftarrow$**false**;
  while $s=$ **false and** $1\leqslant i\leqslant n$ **do**
  begin
    $(\alpha,\beta)=\langle\alpha\beta a_i, y\rangle$;
    if $|\alpha|=|y|$, **then**
    begin
      write "ACCEPT"
      $s\leftarrow$true
  end
  else $i\leftarrow i+1$
  end
  if $s=$false, **then** write "REJECT"
end

Definition. Let $u,y\in\Sigma^*$ and $\langle u,y\rangle\equiv(\alpha,\beta)$. Define the following:

(1) $A(u,y)=\Sigma(\beta)$.

(2) $B(u,y)=\{a\in\overline{\Sigma(\beta)}\mid |\alpha|_a=|y|_a\}$.

(3) For each $(a,b)\in\bar{\theta}$, $q(u,y,a,b)=\pi_{a,b}(\alpha)$.

(4) For each $a\in\Sigma$, $\varepsilon(u,y,a)=|\alpha|_a$.

Lemma. (1) $\sum_{a\in\Sigma}\varepsilon(u,y,a)=|\alpha|$.

(2) *For each* $(a,b)\in A(u,y,a,b)\times(A(u,y)\cup B(u,y))\cap\bar{\theta}$, $q(u,y,a,b)=\pi_{a,b}(y)$.

(3) *For each* $a\in A(u,y)\cup B(u,y)$, $\varepsilon(u,y,a)=|y|_a$.

By this lemma, it suffices to compute $\sum_{a\in\Sigma}\varepsilon(u,y,a)$ for each prefix u of x. To do this, we also need $p(u,y,a,b)\in\Sigma$ for each $(a,b)\in\bar{\theta}$: see PROCEDURE NEWSTATE and Algorithm A.2 below. Here for each $(a,b)\in\bar{\theta}$, $p(u,y,a,b)$ is a sufficiently long suffix of $\pi_{a,b}(\alpha\beta)$ and a prefix of $\pi_{a,b}(y)$ so that for any $t\in\Sigma^*$, $q(ut,y,a,b)$ is a suffix of $p(u,y,a,b)t$ when $\{a,b\}-A(ut,y)\neq\emptyset$. Thus, $p(\lambda,y,a,b)=\lambda$, and for each $(a,b)\in\theta\cap\overline{A(u,y)}\times\overline{A(u,y)}$, $p(u,y,a,b)=q(u,y,a,b)$. Here we also note that (a,b) should be rather regarded as a set $\{a,b\}$, and $p(u,y,a,b)$ and $p(u,y,b,a)$ have the same meaning, etc.

We need the following subroutine which computes (1) $A(ua,y), B(ua,y)\subset\Sigma$, (2) for each $(b,c)\in\bar{\theta}$, $q(ua,y,b,c), p(ua,y,b,c)\in\Sigma^*$, and (3) for each $b\in\Sigma$, $\varepsilon(ua,y,b)$, when (4) $a\in\Sigma$, (5) $A(u,y), B(u,y)\subset\Sigma$, (6) for each $(b,c)\in\bar{\theta}$, $q(u,y,b,c), p(u,y,b,c)\in\Sigma^*$ and (7) for each $b\in\Sigma$, $\varepsilon(u,y,b)$ are all given as inputs. Here we recall the definition of $G(\Sigma,\bar{\theta})$, and let $a\in V_j$, $1\leqslant j\leqslant e$.

PROCEDURE NEWSTATE

Input: $a \in V_j$, $1 \leqslant j \leqslant e$; $A, B \subset \Sigma$; $\varepsilon_b \geqslant 0$ for each $b \in \Sigma$; $p(b,c), q(b,c) \in \Sigma^*$ for each $(b,c) \in \bar{\theta}$

begin
  if $a \in (A \cup B) \cap \theta(\Sigma(y) - (A \cup B))$, **then**
  begin
    $A \leftarrow A \cup \{a\}$; $B \leftarrow B - \{a\}$;
    for each $b \in \bar{\theta}(a)$, **do**
      if $p(a,b)\ a \in \mathrm{Pre}(\pi_{a,b}(y))$, **then** $p(a,b) \leftarrow p(a,b)a$
      else $p(a,b) \leftarrow \psi_{a,b}(p(a,b)a)$
  end
  else
  if $a \in \Sigma_c$, **then**
  begin
    $\varepsilon_a \leftarrow \varepsilon_a + 1$;
    if $\varepsilon_a = |y|_a$, **then** $A \leftarrow A \cup \{a\}$
  end
  else
  for all $b \in \bar{\theta}(a)$, **do**
  begin
    $A \leftarrow A - \{b\}$; $B \leftarrow B - \{b\}$;
    if $p(a,b)a \in \mathrm{Pre}(\pi_{a,b}(y))$, **then** $p(a,b) \leftarrow p(a,b)a$
    else $p(a,b) \leftarrow \psi_{a,b}(p(a,b)a)$;
    $q(a,b) \leftarrow p(a,b)$; $\varepsilon_b \leftarrow |q(a,b)|_b$
end;
while for some $(b,c) \in V_j \times V_j \cap \bar{\theta}$, $|q(b,c)|_b > \varepsilon_b$, **do**
begin
  $A \leftarrow A - \{b\}$; $B \leftarrow B - \{b\}$;
  if $b = a$ or $c = a$, **then** $p(b,c) \leftarrow \psi_{b,c}(p(b,c)a)$
  else $p(b,c) \leftarrow \rho_{b,c}(p(b,c))$;
  $q(b,c) \leftarrow p(b,c)$;
  $\varepsilon_b \leftarrow \min\{\varepsilon_b, |q(b,c)|_b\}$
end;
for each $b \in V_j$, **do**
  if $\varepsilon_b = |y|_b$, **then** $B \leftarrow B \cup \{b\}$
end

Now we can present a more precise implementation of Algorithm A.1.

Algorithm A.2

Input: A text string $x = a_1 \ldots a_n$, $n \geqslant 1$, $a_i \in \Sigma$, $1 \leqslant i \leqslant n$, and a pattern string $y \in \Sigma^+$

Output: "ACCEPT" if y is a θ-factor of x;
"REJECT" otherwise

```
begin
  Obtain π_a(y) for each a∈Σ_c and π_{b,c}(y) for each (b,c)∈θ̄;
  Construct ρ_{b,c} for each (b,c)∈θ̄;
  ε_b←0 for all b∈Σ; A←∅; B←∅; p(b,c)←λ and
  q(b,c)←λ for each (b,c)∈θ̄; s←false; i←1;
  while s= false and 1≤i≤n do
    (* where a∈V_j, 1≤i≤e, and D=Σ−V_j *)
  begin
    a←a_i;
    NEWSTATE;
    if Σ_{b∈Σ} ε_b=|y|, then
    begin
      write "ACCEPT"
      s←true
    end
    else i←i+1
  end;
  if s=false, then write "REJECT"
end
```

Theorem 3.2. *The running time of Algorithm* A.2 *is* $O(|xy|\cdot|\Sigma|^3)$.

In Algorithms B.2 and A.2, we need only bounded amount of memory during processing the text string x once from left to right. Thus, the following theorem holds by estimating an upper bound amount of necessary memory.

Here for $y\in\Sigma^*$, $L_A(y,\Sigma,\theta)=\{x\in\Sigma^* \mid y \text{ is a } \theta\text{-factor of } x\}$ and $L_B(y,\Sigma,\theta)=\{x\in\Sigma^* \mid y$ is a θ-suffix of some prefix of $x\}$.

Theorem 3.3. (1) *$L_A(y,\Sigma,\theta)$ can be recognized by a finite deterministic automaton which has at most $|y|\times|\Sigma|^2\times 2^{|\Sigma|+3}$ states.*

(2) *$L_B(y,\Sigma,\theta)$ can be recognized by a finite deterministic automaton which has at most $|y|\times|\Sigma|^2$ state.*

It is left open to decide the numbers of states of the minimal automata which recognize $L_A(y,\Sigma,\theta)$ and $L_B(y,\Sigma,\theta)$ or to obtain better upper bounds of these numbers.

Remark. Our algorithms solving Problems A and B in this paper may be regarded as FPCM versions of the Knuth–Morris–Pratt string-matching algorithm [9] over the free monoids.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

[2] R. Boyer and J.S. Moore, A fast string searching algorithm, *Commun. Assoc. Comput. Mach.* **20** (1977) 262–272.

[3] P. Catier and D. Foata, Problems combinatoires de commutations et de rearrangement, in: Lecture Notes in Mathematics, Vol. 85 (Springer, Berlin, 1969).

[4] R. Cori and D. Perrin, Automates et commutations partielles, *RAIRO Inform. Theor. Appl.* **19** (1985) 21–32.

[5] M. Crochmore, Optimal factor transducers, in: A. Apostolico and Z. Galil, eds., *Combinatorial Algorithms on Words* (Springer, Berlin, 1985) 31–43.

[6] C. Duboc, On some equations in free partially commutative monoids, *Theoret. Comput. Sci.* **46** (1986) 159–174.

[7] K. Hashiguchi, Recognizable closures and submonoids of free partially commutative monoids, to appear in *Theoret. Comput. Sci.*

[8] K. Hashiguchi and K. Yamada, String matching problems over free partially commutative monoids, *Inform. and Comput.*, to appear.

[9] D. Knuth, J. Morris and V. Pratt, Fast pattern matching in strings, *SIAM J. Comput.* **6** (1977) 323–350.

[10] Y. Metivier, On recognizable subsets of free partially commutative monoids, *Theoret. Comput. Sci.* **58** (1988) 201–208.

[11] E. Ochamanski, Regular behaviour of concurrent systems, *EATCS Bull.* (1985) 56–67.

[12] D. Perrin, Words over a partially commutative alphabet, in: A. Apostolico and Z. Galil, eds., *Combinatorial Algorithms on Words* (Springer, Berlin, 1985) 329–340.

[13] D. Perrin, Partial commutations, in: Lecture Notes in Computer Science, Vol. 372 (Springer, Berlin, 1989) 637–651.

[14] P. Weiner, Linear pattern-matching algorithms, in: *Proc. 14th Ann. Symp. on Switching and Automata Theory*, (1973) 1–11.

[15] W. Zielonka, Notes on finite asynchronous automata, *RAIRO Inform. Theor. Appl.* **21** (1987) 99–135.