Theoretical Computer Science 92 (1992) 145–164
Elsevier

A string-matching interpretation of the equation $x^m y^n = z^p$

J. Néraud

LITP, Institut Blaise Pascal and LIR, Université de Rouen, Place Emile Blondel, F-76134 Mont-Saint-Aignan, France

M. Crochemore

LITP, Institut Blaise Pascal, Université Paris-VII, 2 place Jussieu, F-75251 Paris Cedex 05, France

Abstract

Néraud, J. and M. Crochemore, A string-matching interpretation of the equation $x^m y^n = z^p$, Theoretical Computer Science 92 (1992) 145–164.

We consider the following problem:

Instance: a finite alphabet A, a biprefix code $X = \{x, y\}$ whose elements are primitive, $w \in A^*$.

Question: find all maximal factors of w which are prefixes of a word of X^*.

We present an on-line algorithm which solves the problem in time linear in the length of w, after a preprocessing phase applied to the set X.

1. Introduction

In computer science, the concept of factorization plays a prominent part in applications and in the problems that it introduces.

For instance, given a subset X in the free monoid A^*, it is of interest to compute the "rank" of X, i.e. the minimal cardinality of a finite set Y such that $X \subseteq Y^*$ (i.e. every word of X is the concatenation of words belonging to Y). It is established [15] that this problem is NP-complete and if X has cardinality k then there exists an $O(n^k)$ algorithm, where n stands for the sum of the lengths of the words in X.

As another example the problem of the "shortest common superstring" consists, given a set of words X, in constructing a word w of minimal length such that every word of X factorizes w. This is also a classical NP-complete problem [9, 18].

From another point of view, several "pattern-matching" problems are known to be decidable by linear-time algorithms. Let us mention the classical Knuth–Morris–Pratt (KMP) algorithm [12], generalized in [1], and an algorithm computing the longest common factor of two words [5]). All these algorithms make use of

a notion of "failure function" which is also a main feature of the algorithm presented in this paper. "Failure functions" have recently found an application in text formating [10].

Given a biprefix set $X=\{x, y\} \subseteq A^*$ whose elements are primitive, and given a word w, the problem is to find the longest factors of w which are prefixes of words of X^*. We present an algorithm which solves the problem in a time linear in the length of w after a preprocessing phase applied to the set X. A simple modification of our algorithm permits us to compute the longest factors of w which belong to X^* in a time linear in the length of w.

Let us now examine how an on-line algorithm can solve our problem. For a given word $w \in A^*$ let $\varphi(w)$ be the longest suffix of w which is prefix of a word of X^*. Assume that we have computed $\varphi(w)$, for a given prefix w of the input word. Let $a \in A$ such that $\varphi(wa) \neq \varphi(w)a$. The word $\varphi(w)$ inducts an "X-interpretation" (cf. [17]) of $\varphi(wa)a^{-1}$ which consists in fact in considering the word $\varphi(wa)a^{-1}$ as a factor of a word of X^*. According to a result concerning the equations of the form $x^m y^n = z^p$ in a free monoid (cf. [14], [13]), if $\varphi(wa)a^{-1}$ is long enough, then it is possible to precisely describe it. This leads us to introduce two sets of prefixes of words of X^*, namely L and S. Indeed, for every word $w \in A^*$ the word $\varphi(w)$ belongs to $L \cup X^* S$ and, consequently, w belongs to $A^* L \cup A^* X^* S$. In this way, in a preprocessing phase, we shall compute two automata whose behaviours are, respectively, $A^* L$ and $A^* X^* S$. The complexity of the construction of such automata can be expensive, on account of the cardinality of the alphabet A. We solve this problem by introducing two particular suffixes of w, namely $f(w) \in L$ and $g(w) \in S$. In fact, if after the reading of two words w, w' we are in the same state of the automaton whose behaviour is $A^* L (A^* X^* S)$, then $f(w) = f(w')$ $(g(w) = g(w'))$. Consequently, the functions f and g can be defined on the states of the corresponding automata – thus, f and g are "failure functions". This allows us to save space in using "representations" of the above automata. As a consequence, the processing phase consists in the reading of the input word in parallel on the two automata. For every prefix w of the input word, we shall compute $\varphi(w)$ using the preceding "representations".

Our paper contains the full study of the result published in [6].

2. Preliminaries

2.1. Defintions and notations

Given a finite alphabet A, we denote by A^* the free monoid it generates, and by ε the word of length 0. For any arbitrary subsets $X, Y \subseteq A^*$, we denote by XY their (concatenation) product, by X^* the submonoid generated by X (we set $X^+ = X^* - \{\varepsilon\}$), and by XY^{-1} the set: $\{u \in A^* : \exists (x, y) \in X \times Y \; x = uy\}$.

Given a word $w \in A^*$, we denote by *fact*(w), *pref*(w), and *suff*(w) the sets, respectively, of all the factors, prefixes, and suffixes of w, i.e. the sets of all the words

u satisfying the conditions: $w \in A^* u A^*$, $w \in u A^*$, and $w \in A^* u$, respectively. If $u \in suff(w)$, we say that u is a *proper suffix* iff $u \neq w$. If $w \in X^+$, we say that $u \in fact(w)$ is an *X-factor* of w iff $w \in X^* u X^*$. Two words w and w' are *conjugate* iff there exist two other words u and v such that $w = uv$ and $w' = vu$; if u and v belong to X^+ we say that w and w' are *X-conjugate*. w is *primitive* iff $w = x^n$ implies $n = 1$; otherwise, w is *imprimitive*. An *occurrence* of the factor u in $w \in A^*$ is a tuple (t, u, v) such that $w = tuv$; the integer $|t| + 1$ is the *position* of u in the corresponding occurrence.

For every subset $X \subseteq A^*$, we set $pref(X) = \bigcup_{w \in X} pref(w)$. An *X-interpretation* of the word w is a tuple (s, u, p) such that $w = sup$, with $s \in suff(X) - X$, $p \in pref(X) - X$ and $u \in X^*$. We say that X is *biprefix* iff $X \cap XA^+ = A^+X \cap X = \emptyset$.

2.2. A biprefixity result

In [3], the authors study generalizations of the famous defect theorem [13]. The results can be extended to obtain the following proposition [16].

Proposition 2.1. *Let X be a finite subset of A^*. Then there exists a biprefix set Y satisfying the following conditions:*
- *all the elements of Y are primitive words,*
- $X \subseteq Y^*$,
- $|Y| \leqslant |X|$.

The result of Proposition 2.1 justifies the restriction on the instance X in the following factorization problem, solution of which is the purpose of this paper.

Problem.

Instance: a finite alphabet A, a biprefix code $X = \{x, y\}$ whose all elements are primitive, an input word $w \in A^*$.

Question: Find all maximal elements of $fact(w) \cap pref(X^*)$.

2.3. Aho and Corasick's algorithm

Consider the following "pattern-matching" problem:

Instance: a finite alphabet A, a finite subset X or A^*, an input word w;

Question: does any factor of w belong to X?

A solution can consist of an algorithm with two phases. The preprocessing phase constructs a deterministic automaton $\mathcal{A}$, with transition function δ, initial state i, and whose behaviour is A^*X. In the processing phase we shall decide whether w belongs to the behaviour of $\mathcal{A}$. In time and in space, the complexity of the preprocessing phase is $|A| \sum_{x \in X} |x|$, which can clearly be expensive, on account of the cardinality $|A|$ of A.

For a given word w, denote by $f(w)$ the longest proper suffix of w which is prefix of a word of X. Aho and Corasick proved that given two words $w, w' \in A^*$, if $\delta(i, w) = \delta(i, w')$ then $f(w) = f(w')$. This allows us to define the "failure function" f on

the states of $\mathscr{A}$. Aho and Corasick's solution consists in "representing" $\mathscr{A}$ by a pair $(\mathscr{A}', f)$, where $\mathscr{A}'$ is the tree-like automaton whose behaviour is X (note that the states of $\mathscr{A}'$ can be identified as prefixes of X). The complexity of the construction is linear in the sum of the lengths of the words of X, $\sum_{x \in X} |x|$. In the processing phase, it is easy to compute the transitions by the following recursive rule:

(1) If the transition $\delta(p, a)$ is not defined in $\mathscr{A}'$, we set:

$$\delta(p, a) = \begin{cases} \delta(f(p), a) & \text{if } f(p) \text{ is defined;} \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Aho and Corasick established that the processing phase has a complexity linear in the length of the input word. Once more, note that $|A|$ does not appear.

3. Some results on the X-interpretations

The results of this section are in fact consequences of Fine and Wilf's theorem [8] that we first recall.

Theorem 3.1. *Given two words $x, y \in A^*$, if two powers x^p, y^q have a common prefix of length at least $|x|+|y|-g.c.d.(|x|, |y|)$, then x and y are powers of the same word.*

In [14, 13], the authors study the equations of the form $x^m y^n = z^p$ in a free monoid. With the concept of X-interpretation [17], the results can be reformulated as follows [2].

Proposition 3.2. *Let $X = \{x, y\}$ be a biprefix code with primitive elements and such that $|x| \geqslant |y|$. Let $w \in X^+$ such that $|w| \geqslant 2|x| + 2|y|$. If w has an X-interpretation different from $(\varepsilon, w, \varepsilon)$ then one of the two following cases holds:*

(i) *x and y are conjugate and $w \in x^+ \cup y^+$;*

(ii) *x and y are not conjugate, $x^2 y \cup x y^+$ contains a unique imprimitive word z, and w is an X-factor of a word of z^+.*

In the case where x and y are not conjugate words, the following lemma is a direct consequence of Fine and Wilf's result.

Lemma 3.3. *Let $X = \{x, y\}$ be a biprefix code with primitive elements, and such that $|x| \geqslant |y|$. If $x^+ y \cup x y^+$ contains an imprimitive word t, then one of the two following cases holds:*

(i) *$t = x^2 y$ and t is the square of a primitive word;*

(ii) *$t \in x y^k$ and $1 \leqslant k \leqslant (|x|/|y|) + 1$.*

According to Proposition 3.2, we introduce the following notation.

Notation. Let X be a biprefix primitive code, with $|X|=2$.

(1) For a given word $w \in A^*$, we denote by $\varphi(w)$ the longest suffix of w belonging to $pref(X^*)$.

(2) We denote by *cycle* the set defined as follows:
- if x and y are conjugate, we set $cycle = X$;
- if x and y are not conjugate, and if there exists a primitive word z and a positive integer k, such that $z^k \in x^2 y \cup x y^+$, then we set $cycle = \{z^k\}$;
- in all the other cases we set $cycle = \emptyset$.

(3) We denote by d the length of every word belonging to *cycle*.

(4) If x and y are conjugate, we set $L = pref(x^+ \cup y^+)$ and $\underline{L} = pref(X)$. If x and y are not conjugate, we denote by L ($\underline{L}$) the set whose elements are all the prefixes of the X-conjugate words of $cycle^+$ ($cycle$).

(5) We denote by S the set of all the prefixes w of the words of X^* such that $|w| < 3|x| + 2|y|$ and by $\underline{S}$ the set of all the words $w \in X^+$ satisfying the condition: $|w| \geq 4|x| + 2|y|$ and $|wX^{-1}| < 4|x| + 2|y|$.

The introduction of the sets S and L is justified by the following straightforward result.

Lemma 3.4. *The function* φ *and the sets* L, S *and* $\underline{S}$ *satisfy the following properties*:

(1) $X^{-1}\underline{S}X = \underline{S}$;

(2) *for every word* $w \in A^*$, $\varphi(w) \in L \cup X^* S$;

(3) *let* $w \in A^*$ *and* $a \in A$ *such that* $\varphi(wa) \neq \varphi(w)a$; *then* $\varphi(wa)a^{-1} \in S \cup L$;

(4) *let* $w \in A^*$ *and* $a \in A$; *if* $\varphi(wa) \in L - \underline{L}$ *then* $\varphi(wa) = \varphi(w)a$.

Examples 3.5. Let $A = \{a, b\}$, $x = baab$, $y = aba$. xy^+ contains the square of the primitive word $z = baaba$. Hence, we have $cycle = xy^2 = baaba\ baaba$, $\underline{L} = pref(xyy + yxy + yyx)$ *and* $L = pref((xyy + yxy + yyx)^+)$.

(a) Let $w = babbaabab$ (see Fig. 1). We have $\varphi(w) = xab \in L \cap S$, $\varphi(wa) = xy = \varphi(w)a \in L \cap S$, but $\varphi(wb) = b \neq \varphi(w)b$, $\varphi(wb)b^{-1} = \varepsilon \in S \cap L$.

(b) Let $w = (yxy)^2 yxa$ (see Fig. 2). We have $\varphi(w) = w \in L - S$, $\varphi(wa) = (y^2x)^2 y^2 a \neq \varphi(w)a$, $\varphi(wa)a^{-1} \in L - S$, $\varphi(wb) = \varphi(w)b$.

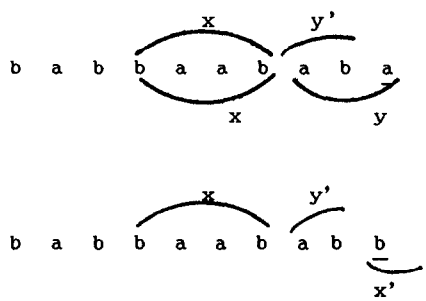

Fig. 1.

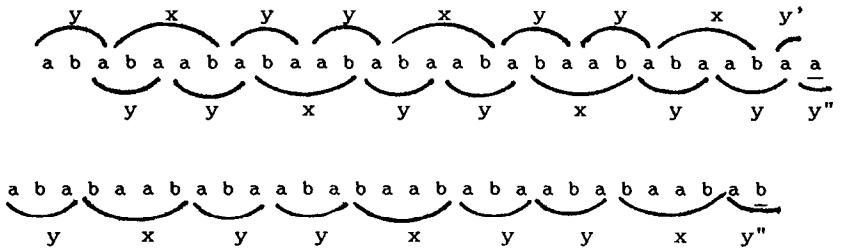

Fig. 2.

According to Lemma 3.4, the preprocessing phase of our algorithm consists in constructing two automata. These automata will allow us to recognize the factors of the input word which, respectively, belong to the sets X^*S and L.

4. Automaton $(\mathscr{S}, g)$

We now indicate the construction of an automaton for recognizing the factors in X^*S.

4.1. The construction

The first step is to construct an automaton recognizing $A^*\underline{S}$, which can be easily done by applying the classical algorithm of Aho and Corasick (cf. Section 2.3). This automaton is represented by a pair $(\mathscr{S}_0, g)$, where $\mathscr{S}_0$ is the tree-like automaton recognizing $\underline{S}$ and g is a failure function. The states of $\mathscr{S}_0$ may be identified as prefixes w of $\underline{S}$, and g is defined as follows: $g(w)$ is the longest proper suffix of w which belongs to S.

Let σ be the transition function of $\mathscr{S}_0$. For every word $w \in \underline{S}$ we define the ε-transition [11]: $\sigma(w, \varepsilon) = X^{-1}w$. With these transitions added to the automaton $\mathscr{S}_0$ we get the automaton $\mathscr{S}$. The justification for such transitions is given by the following: given $w \in \underline{S}$, and $a \in A$, we have $wa \notin pref(\underline{S})$. However, we must be able to distinguish, in terms of the automata, whether $wa \in pref(X^*)$ or not.

4.2. Properties of the automata

Lemma 4.1. *Let $w \in \underline{S}$ and $w' = X^{-1}w$. For every word $t \in A^*$ and for every word $p \in S$ (with p not an X-factor of wt), the two following properties are equivalent:*

(i) *p is a proper suffix of wt;*

(ii) *p is a proper suffix of $w't$.*

Proof of Lemma 4.1. Since $w't$ is a suffix of wt, trivially, claim (ii) implies claim (i).

Let p be a proper suffix of wt belonging to S. According to the definition, we have $|p| < 3|x| + 2|y|$. But, since $w' \in X^{-1}w$, we have $|w'| \geqslant 4|x| + 2|y| - |x|$. Hence, $|w't| \geqslant 3|x| + 2|y|$; thus, p is a suffix of $w't$. □

As a consequence, the failure function g can be extended to a total function $g: A^* pref(X^*) \to S$. Indeed, for all the words $w \in A^*$, we have $g(w) = g(\sigma(\varepsilon, w))$. Thus, A^*X^+ is the behaviour of the automaton represented by the pair $(\mathscr{S}, g)$. We shall compute the transitions of this automaton by applying the following recursive rule (see Fig. 3):

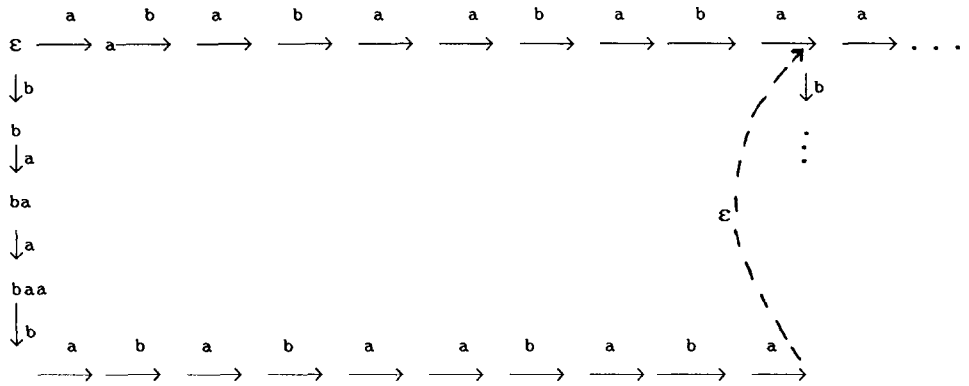

Fig. 3. $x = baab$ and $y = aba$. Two "branches" of the tree-like automaton $\mathscr{S}$.

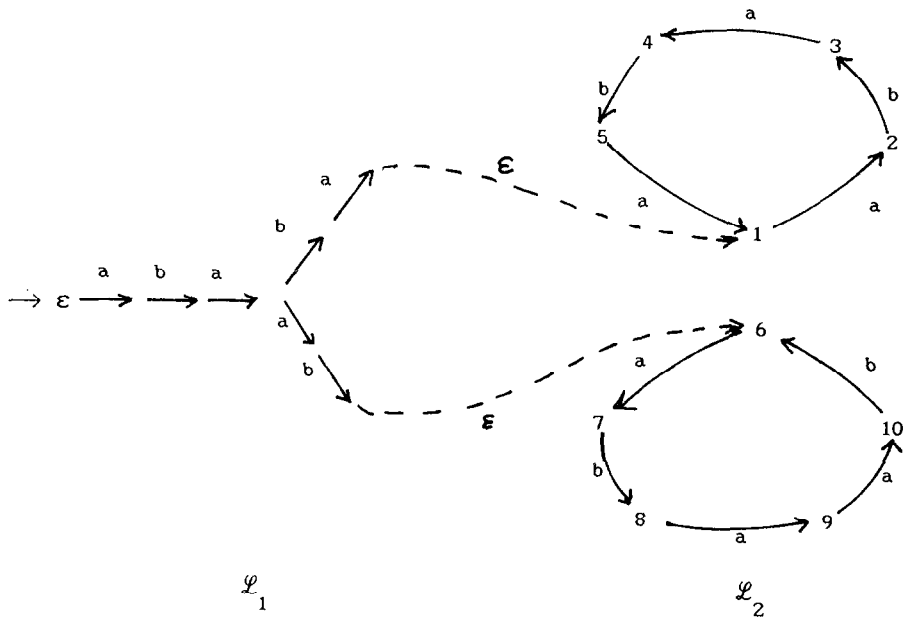

Fig. 4. Automata $\mathscr{L}_1$, $\mathscr{L}_2$: the case where x and y are conjugate, $x = ababa$, $y = abaab$.

(2) If the transition $\sigma(p,a)$ is not defined in $\mathscr{S}$, we set:

$$\sigma(p,a)=\begin{cases}\sigma(g(p),\ a) & \text{if } g(p) \text{ is defined}\\ \varepsilon, & \text{otherwise.}\end{cases}$$

5. Automaton $(\mathscr{L},f)$

The second step of the preprocessing is to construct an automaton $\mathscr{L}$, whose behaviour is A^*L, with transition function λ.

5.1. The construction

Automaton $\mathscr{L}_1$: In a first step we construct an automaton recognizing $A^*\underline{L}$ by applying the algorithm of Aho and Corasick. This automaton is represented by a pair $(\mathscr{L}_1,f)$, where $\mathscr{L}_1$ is the tree-like automaton whose behaviour is $\underline{L}$, and f is the failure function. The states of $\mathscr{L}_1$ may be identified as elements of $\underline{L}$ (ε being the initial state), and f is defined as follows: for every word $w\in\underline{L}$, $f(w)$ is the longest proper suffix of w which is prefix of $\underline{L}$. Let λ_1 be the transition function of $\mathscr{L}_1$.

Automaton $\mathscr{L}_2$: The second step is to construct an automaton $\mathscr{L}_2$ recognizing L. Let λ_2 be the transition function of $\mathscr{L}_2$ and let Q be the set of the states.

(a) *Case 1*: x and y are not conjugate words (cf. Fig. 5). Let w be the unique word in *cycle* and, for all the integers $i\in\{1,\ldots,|w|\}$ let w_i be the letter of w with position i. We set:

- $Q=\{0,\ldots,|w|-1\}$,
- all the states are terminal,
- initial state: 0,
- the transitions are defined as follows:

$$\lambda_2(i,w_{i+1})=i+1 \quad (0\leqslant i\leqslant|w|-2) \quad \text{and} \quad \lambda_2(|w|-1,w_{|w|})=0.$$

Case 2: x and y are conjugate words (cf. Fig. 4). Recall that, in this case, we have $|x|=|y|$.

- $Q=\{0,\ldots,2|x|\}$,
- all the states are terminal,
- initial state: 0,
- the transitions are defined as follows:

For all the integers $i\in\{1,\ldots,|x|\}$, let x_i (y_i) be the letter of x (y) with position i. We set:

$$\lambda_2(i,x_i)=i+1 \quad (1\leqslant i\leqslant|x|-1),$$

$$\lambda_2(i+|x|,y_i)=|x|+i+1 \quad (1\leqslant i\leqslant|x|-1),$$

$$\lambda_2(|x|,x_{|x|})=1 \quad \text{and} \quad \lambda_2(2|x|,y_{|x|})=|x|+1,$$

$$\lambda_2(0,\varepsilon)=1 \quad \text{and} \quad \lambda_2(0,\varepsilon)=|x|+1.$$

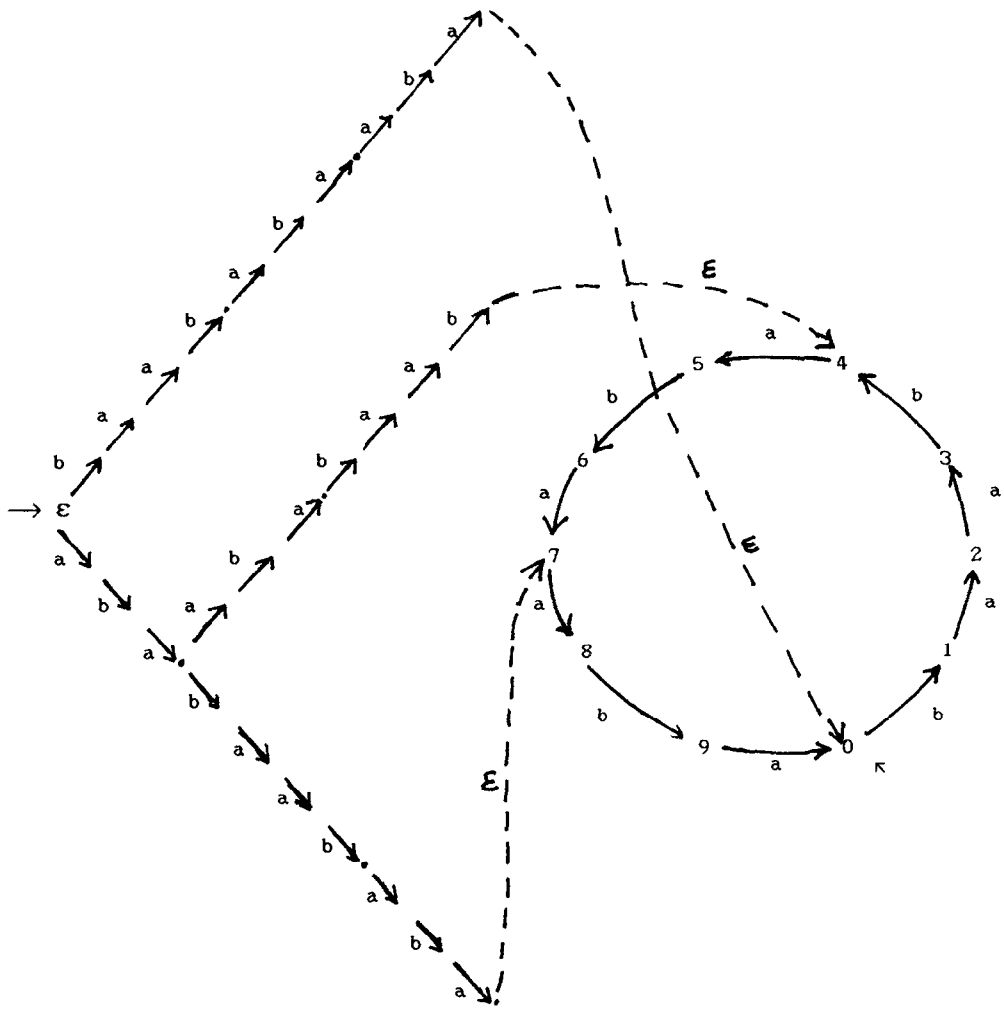

Fig. 5. Automata $\mathscr{L}_1$, $\mathscr{L}_2$: the case where x and y are not conjugate, $x = baab$, $y = aba$, xy^2 imprimitive.

Automaton $\mathscr{L}$: Let w be an X-conjugate word of a word of *cycle*. By construction there exists a unique state $q \in \lambda_2(0, X^+)$ such that (q, w, q) is a path of $\mathscr{L}_2$ and we set $\theta(w, \varepsilon) = q$. The automaton $\mathscr{L}$ is defined as the disjoint union of $\mathscr{L}_1$ and $\mathscr{L}_2$. Its transition function is the union of λ_1, λ_2 and θ, and its initial state is ε, the initial state of $\mathscr{L}_1$.

5.2. *Properties of the automaton* $\mathscr{L}$

As in Section 4.2, we shall extend the function f to all the states of the automaton $\mathscr{L}$.

Lemma 5.1. *Let $w \in L$ and $a \in A$ such that $wa \notin L$. Then the longest proper suffix of wa belonging to L is an element of $\underline{L}$.*

Proof of Lemma 5.1. Since $wa \notin L$, the length of $f(wa)a^{-1}$ is upper-bounded by the minimal period (cf. [7]) of every word belonging to *cycle*. Hence, $f(wa)a^{-1}$ belongs to $\underline{L}$. □

Lemma 5.2. *Let w and w' be two words of L such that $\lambda(\varepsilon, w) = \lambda(\varepsilon, w')$. If $|w| \leqslant |w'|$, then w is an X-factor of w' and a suffix of w'.*

Proof of Lemma 5.2. The result is trivial if $w \in \underline{L} - L$. In the other case, we can assume that w' is the shortest word such that $\lambda(\varepsilon, w) = \lambda(\varepsilon, w')$. By construction of $\mathscr{L}$, we have $w' = x_1 \dots x_n . x'$, with $x_1 \dots x_n$ X-conjugate of a word of *cycle* and $x' \in pref(X)$. By construction of the automaton $\mathscr{L}$, the word w belongs to the set:

$$(x_1 \dots x_n)^+ x' + (x_n . x_1 \dots x_{n-1})^+ x_n . x' + (x_{n-1} . x_n . x_1 \dots x_{n-2})^+ x_{n-1} . x_n . x' + \cdots$$

$$+ (x_2 . x_{n-1} . x_n \dots x_1)^+ x_2 \dots x_n . x'.$$

Hence,

$$w \in (x_1 \dots x_n)^+ x' + x_n (x_1 \dots x_n)^+ x' + x_{n-1} . x_n (x_1 \dots x_n)^+ x' + \cdots$$

$$+ x_2 \dots x_{n-1} (x_1 \dots x_n)^+ x'.$$

Thus, w' is an X-factor of w and a suffix of w. □

Lemma 5.3. *Let $w, w' \in L$ such that $\lambda(\varepsilon, w) = \lambda(\varepsilon, w')$. Then for every word $t \in A^*$ and for every word $p \in \underline{L}$, the following properties are equivalent:*

(i) *p is a proper suffix of wt;*

(ii) *p is a proper suffix of $w't$.*

Proof of Lemma 5.3. If $w \in \underline{L} - L$ then we have $w' = w$ and the result is trivial.

Assume that $w, w' \notin L$, with $|w| \leqslant |w'|$, without loss of generality. According to Lemma 5.2, the word w is a suffix of w'; thus, wt is a suffix of $w't$. Hence, claim (ii) implies claim (i).

Moreover, if p is a proper suffix of wt (with $p \in \underline{L}$), then we have $|p| \leqslant d$. Hence, p is a proper suffix of $w't$. □

Given a word $w \in A^*$, let L_w be the sets of the words w' such that $\lambda(\varepsilon, w) = \lambda(\varepsilon, w')$. As a consequence of Lemma 5.3, L_w is totally ordered by the relation $\geqslant$ defined by:

$$w \geqslant w' \text{ iff } w \text{ is a suffix of } w'.$$

A similar remark can be made in Section 4.2. This can be got closer with a result concerning the suffix automaton [4].

As a consequence, the failure function f can be extended to a total function $A^*L \to \underline{L}$. Thus, A^*L is the behaviour of the automaton represented by the pair $(\mathscr{L}, f)$. We shall compute the transitions of this automaton by applying the following recursive rule:

(3) If the transition $\lambda(p, a)$ is not defined in $\mathscr{L}$, then

$$\lambda(p, a) = \begin{cases} \lambda(f(p), a) & \text{if } f(p) \text{ is defined,} \\ \varepsilon, & \text{otherwise.} \end{cases}$$

We finally add to the preprocessing phase the computation of the lengths of all the words $w \in S \cup \underline{L}$.

6. A special case of failure

6.1. Description

Recall that we denoted by $\varphi(w)$ the longest suffix of w belonging to $pref(X^*)$. Assume that we have $\varphi(wa) \neq \varphi(w)a$, with $a \in A$ and $\varphi(wa) \notin S$. According to Proposition 3.2, the word $w' = \varphi(wa)a^{-1}$ belongs to $L - \underline{L}$. Let w_1 be the shortest suffix of w such that w_1 is an X-factor of w and such that w' is a suffix of w_1. We have also $w_1 \in L$.

We now study this condition more precisely.

The following comes from the definition of the period of a word:

(4) Let t, t' be two conjugate words, and let $p \in pref(t^+)$, $q \in pref(t^+)$.

For every $r \in A^*$, we have $pr \in pref(t^+)$ iff $qr \in pref(t'^+)$.

We now establish two lemmas.

Lemma 6.1. *Let $w \in L - \underline{L}$ and $a \in A$ such that $wa \notin L$. Assume that there exists a suffix w' of w such that $w'a$ is a prefix of a word in X^*. If $w' \notin S$ then there exists a word $w'' \in L \cap X^*$ such that $w' = w''.x \wedge y$,* where *$x \wedge y$ is the longest common prefix of x and y.*

Proof of Lemma 6.1. Let w'' be the longest prefix of w' which belongs to X^*. If w' is an X-factor of w then clearly, $w' \in L$. Otherwise, since $|w'| > 3|x| + 2|y|$, and again according to Proposition 3.2, we have $w' \in L$. Assume that $w' \neq w''.x \wedge y$. Then one of the following cases may occur:

(1) $w' = w''.x \wedge y.r$ $(r \in A^+)$. Since $X = \{x, y\}$ is a biprefix set, there exists a unique word $t \in X^+$, such that $x \wedge y.r$ and $x \wedge y.ra$ are two prefixes of t. This implies $w'a \in L$, which contradicts $wa \notin L$ (according to (4)).

(2) $w' = w''r$, with $ra \in pref(x \wedge y)$; thus, $w'a \in L$, which, once more, contradicts $wa \notin L$.

In each case, we obtain a contradiction. Hence, we obtain $w' = w''.x \wedge y$. □

Lemma 6.2. *Let $x \wedge y$ be the longest common prefix of x and y and let $w = u\underline{w}(x \wedge y) \in L$, with $u \in X^*$, where $\underline{w}$ is an X-conjugate word of a word belonging to cycle. Assume that for a given $a \in A$ we have $wa \notin L$ and let $t \in A^*$, $p \in pref(X^*)$, such that $wat \notin X^* p$. Then the following conditions are equivalent:*

(i) *p is a proper suffix of wat;*

(ii) *p is a proper suffix of $\underline{w}(x \wedge y)at$.*

Proof of Lemma 6.2. Trivially, claim (ii) implies claim (i). Conversely, let p be a proper suffix of *wat*. Assume that $\underline{w}.x \wedge y.at$ is a suffix of p. According to Lemma 6.1, since X is a biprefix set, we have $p \in X^* \underline{w}.x \wedge y.at$; thus, p is an X-factor of *wat*, which contradicts the hypothesis of Lemma 6.2. Consequently, p is a proper suffix of *wat*. □

As a consequence of the preceding results, we introduce the following notation: Let $w \in A^*$, and $a \in A$. We say that (w, a) satisfies the condition (5) iff:

(i) Let w' be the longest suffix of w which belongs to L. The word w' has an X-interpretation $(r, d, x \wedge y)$, where $x \wedge y$ is the longest common prefix of the words x and y,

(ii) $x \wedge y.a \in pref(X)$,

(iii) $d.x \wedge y.a \notin S$.

6.2. *Deciding whether the condition* (5) *holds*

First, we introduce three new notations.

Sets start(i): Assume that $xy^+ \cup x^2 y$ contains an imprimitive word z^k (with z primitive). Since all the X-conjugate words of the word of *cycle* have the same smallest period, each of them has exactly k X-interpretations. Let (r_i, d_i, p_i) $(1 \leqslant i \leqslant k)$ be different X-interpretations of z^k. We set $start(i) = \lambda(\varepsilon, z^k r_i . X^* . x \wedge y)$ $(1 \leqslant i \leqslant k)$.

Assume that x and y are conjugate: $x = uv$, $y = vu$. We set $start(1) = \lambda(\varepsilon, y + uy)$, $start(2) = \lambda(\varepsilon, x + vx)$, and $k = 2$.

Clearly, the sets $start(i)$ $(1 \leqslant i \leqslant k)$ will be computed in the preprocessing phase.

Function π. According to Lemma 6.2, we define the following partial function π: Let $w \in L$, such that there exists $i \in [1, k]$, with $\lambda(\varepsilon, w) \in start(i)$. We denote by $\pi(w)$ the unique suffix of w which satisfies $\pi(w) = \underline{w}.x \wedge y$, where $\underline{w}$ is an X-conjugate word of a word of *cycle*.

According to this definition, the function can be defined on the states of $\mathscr{L}_2$; thus, it can be computed in the preprocessing phase.

Function shift: With the preceding notation, let $w \in A^*$, and let $D(w)$ be the subset of A^*, whose elements satisfy the following:

- every word in $D(w)$ is a suffix of w and belongs to L;

– if w_1 and w_2 are two elements of $D(w)$, with $|w_1| < |w_2|$, then w_1 is not an X-factor of w_2.

Clearly, the cardinality of $D(w)$ is k. Moreover, since X is a biprefix set, at most one of the elements of $D(w)$ belongs to $X^* . x \wedge y$. If such a word w' exists, then there exists an integer i such that $\lambda(\varepsilon, w) \in start(i)$. We set $shift(w, i) = s$, with $w = sw'$. Since X is a biprefix code, the following lemma is clear.

Lemma 6.3. *Let $w \in L$, and let w' be a prefix of w. With the preceding notations, assume that there exists an integer $i \in [1, k]$ such that $\lambda(\varepsilon, w) \in start(i)$ and $\lambda(\varepsilon, w') \in start(i)$. Then we have $shift(w, i) = shift(w', i)$.*

Let $w \in L$, with $w \geqslant d$ (cf. notations in Section 3), and let $\underline{w}$ be the prefix of w whose length is d. As a consequence of Lemma 6.3, if there exists an integer $i \in [1, k]$ such that $\lambda(\varepsilon, w) \in start(i)$, then we have $shift(w, i) = shift(\underline{w}, i)$. This allows us to compute the values of *shift* in the preprocessing phase.

As consequence, from an algorithmic point of view, we shall decide whether the condition (5) holds as follows.

(6) The pair $(w, a) \in A^* \times A$ satisfies the condition (5) iff the following holds:
  (i) There exists a positive integer $i \in \{1, \ldots, k\}$ such that $\lambda(\varepsilon, w) \in start(i)$;
  (ii) $\sigma(\pi(w), a) \neq \emptyset$;
  (iii) if w' is the longest suffix of w which belongs to L then $(shift(w, i))^{-1} w' \notin S$.

6.3. Example

Let $x = baababaab$ and $y = aba$. xy^2 contains the imprimitive word $(baaba)^3$; thus, $cycle = \{xy^2\}$ and $k = 3$ (see Fig. 6).

Since $x \wedge y = \varepsilon$, the sets $start(i)$ $(1 \leqslant i \leqslant 3)$ are the following:

$$start(1) = \lambda(\varepsilon, xyy + yxy + yyx) = \{0, 9, 12\},$$

$$start(2) = \lambda(\varepsilon, xyyba + xyybaaba + yxyab) = \{2, 5, 14\},$$

$$start(3) = \lambda(\varepsilon, xyybaab + xyybaababa + yyxa) = \{4, 7, 10\}.$$

The function π takes the following values:

$$\pi(0) = \pi(5) = \pi(10) = xyy,$$

$$\pi(2) = \pi(7) = \pi(12) = yxy,$$

$$\pi(4) = \pi(9) = \pi(14) = yyx.$$

shift takes the values shown in Table 1.

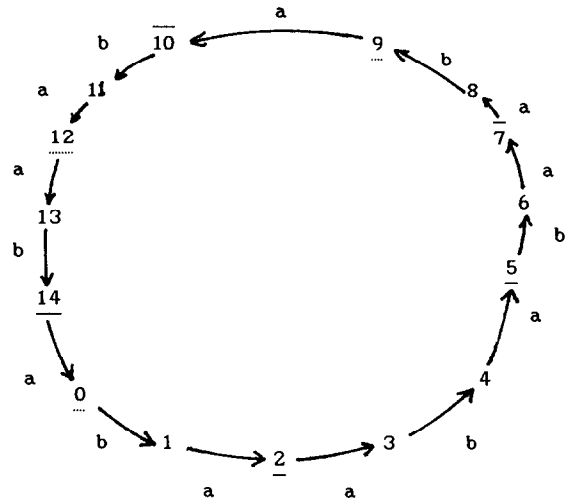

Fig. 6.

Table 1

	i		
w	1	2	3
xy^2	ε	*ba*	*baab*
yxy	ε	*ab*	*ababaab*
y^2x	ε	*abaab*	*a*

Let $w=y(xyy)^2baaba\in L$. We have $wa\notin L$, $\lambda(\varepsilon, w)\in start(2)$, and $shift(w, 2)=shift(yxy, 2)=ab$. Hence, the condition (5) is satisfied, and: $\pi(w)=xyy$, $\pi(w)a\in S$.

7. The processing phase

7.1. Notation

According to Lemma 3.4, for a given word $w\in A^*$, $\varphi(w)$ is the longest of the two following words:

- the longest suffix of w belonging to L,
- the longest suffix of w belonging to X^*S.

The first word belongs in fact to $(L-\underline{L})\cup\underline{L}$ and the second to $(X^*S-S)\cup S$. This leads us to introduce the following new notation.

Notation. Let $w \in A^*$.

(1) Let w' be the longest suffix of w belonging to L. If $|w'| \geq d$ (cf. Section 3) then we set $L(w) = \{w'\}$, otherwise we set $L(w) = \emptyset$.

(2) We denote by $l(w)$ the set whose unique element is the longest suffix of w belonging to $\underline{L}$.

(3) If $|\varphi(w)| \geq 3|x| + 2|y|$ then we set $S(w) = \{\varphi(w)\}$, otherwise we set $S(w) = \emptyset$.

(4) We denote by $s(w)$ the set whose unique element is the longest suffix of w belonging to S.

Moreover, if (w, a) satisfies the condition (5), we set $S_1(wa) = \{d.x \wedge y.a\}$; otherwise we set $S_1(wa) = \emptyset$.

The following result follows from the definitions.

Lemma 7.1. *$\varphi(w)$ is the longest word belonging to $L(w) \cup l(w)S(w) \cup s(w)$.*

7.2. Computation of $L(w)$, $l(w)$, $S(w)$ and $s(w)$

Our on-line algorithm must read all the prefixes of the input word. In the initialization step, we set $L(w) = S(w) = \emptyset$, and $\varphi(w) = \varepsilon$. Assume that we have read the prefix w, and let $a \in A$.

Lemma 7.2. *The tuple $(L(wa), l(wa), S(wa), s(wa))$ can be computed using the tuple $(L(w), l(w), S(w), s(w))$ and the functions f, g, π, shift.*

Proof of Lemma 7.2. Let $w \in A^*$ and $a \in A$. Different cases may occur:

(1) One of the two following conditions holds:

- $L(w) \neq \emptyset$ and $L(wa) = L(w)a$,
- $L(w) = \emptyset$ and $l(wa) = l(w)a$.

In examining the sets $S(wa)$ and $s(wa)$ two new cases may occur:

(1.1) One of the two following cases holds:

- $S(w) \neq \emptyset$ and $S(wa) = S(w)a$,
- $S(w) = \emptyset$ and $s(wa) = s(w)a$.

In the second case, we have $S(wa) \neq \emptyset$ iff the length of its element is at least $3|x| + 2|y|$. If this condition holds then we have $S(wa) = s(w)a$.

(1.2) One of the two following cases holds:

- $S(w) \neq \emptyset$ and $S(wa) \neq S(w)a$,
- $S(w) = \emptyset$ and $s(wa) \neq s(w)a$.

(1.2.1) Assume that $S(w) \neq \emptyset$ and $S(wa) \neq S(w)a$.

Assume that $L(w) \neq \emptyset$ and let $w_1 \in S(w)$, $w_2 \in L(w)$. Since $|w_1| \geq |w_2|$, the word w_2 is a suffix of w_1. Since $w_2 a \in L$, according to (4), the word $w_1 a$ belongs to L; thus, $S(wa) = \{w_1 a\}$, which contradicts the condition (1.2.1). Hence, we have $L(w) = \emptyset$. Moreover:

$$L(wa) = \begin{cases} l(w)a & \text{if the word in } l(wa) \text{ has a length at least } d, \\ \emptyset & \text{otherwise.} \end{cases}$$

In a similar way,

$$\begin{cases} S(wa)=\emptyset, \\ \text{there exists an integer } n \text{ such that } s(wa)=\{g^n(w)a\}. \end{cases}$$

(1.2.2) Assume that $S(w)=\emptyset$ and $s(wa)\neq s(w)a$.

If $S(wa)\neq\emptyset$, then we have $S(wa)=s(w)a$. But, since $s(wa)\neq s(w)a$, the word in $s(wa)$ has length at most $3|x|+2|y|-1$; thus, it cannot belong to $S(wa)$. Hence:

$$\begin{cases} S(wa)=\emptyset, \\ \text{there exists an integer } n \text{ such that } s(wa)=\{g^n(w)a\}. \end{cases}$$

(2) One of the following conditions holds:

- $L(w)\neq\emptyset$ and $L(wa)\neq L(w)a$,
- $L(w)=\emptyset$ and $l(wa)\neq l(w)a$.

According to (4), with the first condition, we have $L(wa)=\emptyset$. With the second condition, there exists an integer n such that $l(wa)=\{f^n(w)a\}$.

(2.1) One of the two following cases holds:

- $S(w)\neq\emptyset$ and $S(wa)=S(w)a$,
- $S(w)=\emptyset$ and $s(wa)=s(w)a$.

In the second case:

$$S(wa)=\begin{cases} S_1(wa) & \text{iff condition (5) holds,} \\ \text{otherwise} & \begin{cases} s(w)a & \text{if the element of this set has length } 3|x|+2|y|, \\ \emptyset, & \text{otherwise.} \end{cases} \end{cases}$$

(2.2) One of the two following cases holds:

- $S(w)\neq\emptyset$ and $S(wa)\neq S(w)a$,
- $S(w)=\emptyset$ and $s(wa)\neq s(w)a$.

Here

$$\begin{cases} S(wa)=\begin{cases} S_1(wa) & \text{iff condition (5) holds,} \\ \emptyset, & \text{otherwise,} \end{cases} \\ \text{there exists an integer } n \text{ such that } s(wa)=\{g^n(w)a\}. \end{cases}$$

Since the conditions $L(wa)=L(w)a$, $l(wa)=l(w)a$, $S(wa)=S(w)a$ and $s(wa)=s(w)a$ can easily be decided in examining the transitions of automata $\mathscr{S}$ and $\mathscr{L}$, we obtain our lemma. □

7.3. *Scheme of the algorithm*

From an algorithmic point of view, the preceding results lead to the introduction of function that we denoe by *suffix*. Given the tuple of sets $(L(w), l(w), S(w), s(w))$, and given the letter a, *suffix* computes the tuple $(L(wa), l(wa), S(wa), s(wa))$.

The different cases are summarized in the arrays of Tables 2 and 3. The results will be easily translated in terms of length.

Table 2

	$L(wa)$	$l(wa)$
$L(w)\neq\emptyset$, $L(wa)=L(w)a$	$L(w)a$	–
$L(w)=\emptyset$, $l(wa)=l(w)a$	$l(w)a\cap A^d$	$l(w)a$
$L(w)\neq\emptyset$, $L(wa)\neq L(w)a$	$\emptyset$	$\{f(wa)\}$
$L(w)=\emptyset$, $l(wa)\neq l(w)a$	$\emptyset$	$\{f(wa)\}$

Table 3

	$S(wa)$	$s(wa)$
$S(w)\neq\emptyset$, $S(wa)=S(w)a$	$S(w)a$	$\{g(wa)\}$
$S(w)=\emptyset$, $s(wa)=s(w)a$	[a] $S_1(w)a \wedge (s(w)a\cap A^{3\lvert x\rvert+2\lvert y\rvert})$	$s(w)a$
$S(w)\neq\emptyset$, $S(wa)\neq S(w)a$	$S_1(w)a \wedge (s(w)a\cap A^{3\lvert x\rvert+2\lvert y\rvert})$	$\{g(wa)\}$
$S(w)=\emptyset$, $s(wa)\neq s(w)a$	$S_1(wa)$	$\{g(wa)\}$

[a] Given two sets of words E and F, we set:

$$E \wedge F = \begin{cases} E & \text{if } F=\emptyset, \text{ or } F \text{ if } E=\emptyset \\ E & \text{if } E\neq\emptyset \text{ and } F\neq\emptyset \end{cases}$$

We can now give the scheme of an algorithm for computing the longest factor w which belongs to $pref(X^*)$ (with minimal position). After reading every prefix of the input word, let q_s (q_L) be the corresponding state of $\mathscr{S}$ ($\mathscr{L}$).

Algorithm

begin

$q_s \leftarrow \varepsilon$; $q_L \leftarrow \varepsilon$; $w \leftarrow \varepsilon$;
while not end of the input word **do**

begin
read the next letter a;
$q_s \leftarrow \sigma(q_s, a)$; $q_L \leftarrow \lambda(q_L, a)$; {step 1}
$(L(wa), l(wa), S(wa), s(wa)) \leftarrow suffix(L(wa), l(wa), S(wa), s(wa))$;
$\varphi(wa) \leftarrow$ longest word of $L(wa)\cup l(wa)\cup S(wa)\cup s(wa)$; {step 2}
if (w, a) satisfies condition (5) **then** $q_s \leftarrow \sigma(\varepsilon, \pi(w)a)$); {step 3}
$w \leftarrow wa$;
if $\varphi(w)$ strictly longer than the previous one **then**
memorize its length and position
end
end.

- In step (1), q_s and q_L are computed by the transition functions in $(\mathscr{L}, f)$ and $(\mathscr{S}, g)$. Step (2) is justified by Lemmas 7.1 and 7.2. Step (3) is the direct consequence of Lemma 6.2.
- In step (2), any element of $L(wa) \cup l(wa) \cup S(wa) \cup s(wa)$ is determined by its length and its position.

7.4. *The full algorithm*

For every prefix w of the input word, the longest word in $S(w) \cup s(w)$ $(L(w) \cup l(w))$ is characterized by its length, which we denote by ℓ_s (ℓ_L). We set $\ell_\varphi = |\varphi(w)|$, and we denote by $w(i)$ the letter of a with position i. The longest factor of w which belongs to $pref(X^*)$ and whose position is minimal is characterized by its length $\ell_{\max}$ and its position $i_{\max}$.

In the full algorithm, each of the steps 1, 2, 3 has been divided in smaller steps, namely 1.1, 1.2, 1.3,...

Algorithm

begin
  $q_s \leftarrow \varepsilon$; $q_0 \leftarrow q'_L \leftarrow q_L \leftarrow \varepsilon$;
  $i \leftarrow 0$; $i_{\max} \leftarrow 0$; $l_{\max} \leftarrow l_\varphi \leftarrow l_s \leftarrow l'_L \leftarrow l_L \leftarrow 0$;
  while $i < |w|$ **do**
    begin
      $\ell_\varphi \leftarrow 0$;
      $i \leftarrow i+1$; $a \leftarrow w(i)$;
      $\ell'_L \leftarrow \ell_L$; $q'_L \leftarrow q_L$; {step of memorization}
      while $(\sigma(q_s, a) = \emptyset$ **and** $q_s \neq \varepsilon)$ **do**
        begin $q_s \leftarrow g(q_s)$ {1.1}; $\ell_s \leftarrow |q_s|$ {3.1} **end;**
      if $\sigma(q_s, a) \neq \emptyset$ **then** {otherwise we have $q_s = \varepsilon$}
        begin $q_s \leftarrow \sigma(q_s, a)$ {1.2}; $\ell_s \leftarrow \ell_s + 1$ {3.2} **end;**
      while $(\lambda(q_L, a) = \emptyset$ **and** $q_L \neq \varepsilon)$ **do**
        begin $q_L \leftarrow f(q_L)$ {2.1}; $\ell_L \leftarrow |q_L|$ {3.3} **end;**
      if $\lambda(q_L, a) \neq \emptyset$ **then** {otherwise we have $q_s = \varepsilon$}
        begin $q_L \leftarrow \lambda(q_L, a)$ {2.2}; $\ell_L \leftarrow \ell_L + 1$ {3.4} **end;**
      if (there exists $j \in [1, k]$ such that $q'_L \in start(j)$ **and** $\sigma(\pi(q'_L), a) \neq \emptyset$
        and ℓ'_L-*shift*$(q_0, j) + 1 > \ell_s)$ **then** {cf. 3.6}
          begin
            $\ell_s \leftarrow \ell'_L - shift(q_0, j) + 1$; {3.5}
            $q_s \leftarrow \sigma(\pi(q'_L), a)$;
          end
      $\ell_\varphi \leftarrow \max(\ell_s, \ell_L)$;
      if $\ell_\varphi \geqslant \ell_{\max}$ **then begin** $\ell_{\max} \leftarrow \ell_\varphi$; $i_{\max} \leftarrow i - \ell_{\max} + 1$ **end**
      if (q'_L is a state of $\mathscr{L}_1$ **and** q_L is a state of $\mathscr{L}_2$) **then** $q_0 \leftarrow q_L$ {3.6}
    end
end.

8. The complexity

8.1. The preprocessing phase

Since $\underline{L} \subset pref(\underline{S})$, as a direct consequence of [1], we have the following result.

Proposition 8.1. *The preprocessing has a complexity in time and in space proportional to* $|pref(\underline{S})|$.

8.2. The processing phase

Theorem 8.2. *Let* $X = \{x, y\}$ *be a biprefix code whose elements are primitive words, and let* $w \in A^*$. *Then the computation of the longest factors of* w *which belongs to* $pref(X^*)$ *can be implemented so that it works in time* $O(|w|)$.

Proof.

- Step (1) allows us to process the word w by applying the rules (2) and (3) at most $2 \cdot 2|w|$ times. Indeed, if v is the prefix of the word w already processed, the numbers $2|v| - |q_s|$ and $2|v| - |q_L|$ increase strictly [1]; moreover, $\underline{L} \subset S$.
- The length of the elements of S and $\underline{L}$, and of the partial function *shift*, are computed in the preprocessing phase. Since q_L and q_s are computed in step (1), the complete process of step (2) requires a time linear in $|w|$.
- Since step (3) requires constant time, the complete process of our algorithm requires time $O(|w|)$. □

References

[1] A. Aho and M. Corasick, Efficient string matching: an aid to bibliographic search, *Comm. ACM* **18** (6) (1975) 333–340.

[2] E. Barbin and M. Lerest, Sur la combinatoire des codes à deux mots, *Theoret. Comput. Sci.* **41** (1985) 61–80.

[3] J. Berstel, D. Perrin, J.F. Perrot and A. Restivo, Sur le théorème du défaut, *J. Algebra*, **60** (1) (1979) 169–180.

[4] M. Crochemore, Transducers and repetition, *Theoret. Comput. Sci.* **45** (1986) 63–86.

[5] M. Crochemore, Longest common factor of two words, in: *TAPSOFT'87*, Lecture Notes in Computer Science, Vol. 249 (Springer, Berlin, 1987) 26–36.

[6] M. Crochemore and J. Néraud, Unitary monoid with two generators: an algorithmic point of view, in: *CAAP'90*, Lecture Notes in Computer Science, Vol. 431 (Springer, Berlin, 1990) 117–131.

[7] J.P. Duval, Périodes et répétitions des mots du monoide libre, *Theoret. Comput. Sci.* **9** (1979) 17–26.

[8] N. Fine and H. Wilf, Uniqueness theorem for periodic functions, *Proc. Amer. Math. Soc.* **16** (1965) 109–114.

[9] M. Garey and D. Johnson, *Computers and intractability. A Guide to the Theory of* NP-*Completeness* (W.H. Freeman, 1978).

[10] D. Hirschberg and L. Larmore, New applications of failure functions, *J. ACM* **34** (3) (1987) 616–625.

[11] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).

[12] D. Knuth, M. Morris and V. Pratt, Fast pattern matching in strings, *SIAM J. Comput.* **6** (1977) 323–350.
[13] A. Lentin, *Equations dans le monoide libre* (Gautier Villars, Paris, 1972)
[14] A. Lentin and M.P. Shützenberger, A combinatorial problem in the theory of free monoids, in: *Proc. University of North California* (1967) 128–144.
[15] J. Néraud, Elementariness of a finite set of words is co-NP-complete. *Theoret. Inform. Appl.* **24** (5) (1990) 459–470.
[16] J. Néraud, On the deficit of a finite set of words, *Semigroup Forum* **41** (1990) 1–21.
[17] M.P. Schützenberger, A property of finitely generated submonoids, in: G. Pollak, ed., *Algebraic Theory of Semigroups* (North-Holland, Amsterdam, 1979) 545–576.
[18] J. Tarhio and E. Ukkonen, A greedy approximation algorithm for constructing shortest common superstrings, *Theoret. Comput. Sci.* **57** (1988) 131–145.