INFORMATIQUE THÉORIQUE ET APPLICATIONS

JEAN NÉRAUD

Detecting the morphic images of a word : improving the general algorithm

Informatique théorique et applications, tome 31, n° 1 (1997), p. 1-14

<http://www.numdam.org/item?id=ITA_1997__31_1_1_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 31, n° 1, 1997, pp. 1-14)

DETECTING THE MORPHIC IMAGES OF A WORD: IMPROVING THE GENERAL ALGORITHM (*)

by Jean NÉRAUD (1)

Abstract. – *This paper takes place in a series of studies concerning the general problem which consists in detecting all the morphic images of a given word* $R \in (\Delta \cup \Sigma)^*$ *in an arbitrary text* $w \in \Sigma^*$. *Applying the naive algorithm to this problem requires time* $O(|w|^{|\Delta|+2})$.

We conjugate the concept of rank *of a pattern, namely the integer* $r(R)$ *(a measure of the complexity of its construction in term of periodicity) [N 95.1], with one of our previous results concerning one-variable pattern [N 95] to prove that the problem may be solved by a* $O(|w|^4 + |R|^3 + |w|^{|\Delta|} r(R) \ln|w| \ln|R|)$*-time algorithm.*

Asymptotically and in the case where Δ *has at least four letters, this leads to a* $O(|w|^{|\Delta|} \ln^2|w|)$*-time algorithm, a direct generalization of the results in [N 95].*

Résumé. – *Cet article s'intègre dans une série d'études du problème général qui consiste à détecter toutes les images homomorphes d'un mot donné* $R \in (\Delta \cup \Sigma)^*$ *dans un texte arbitraire* $w \in \Sigma^*$. *Pour résoudre ce problème, l'algorithme naïf opère avec une complexité en* $O(|w|^{|\Delta|+2})$.

Nous associons le concept de rang *d'un motif (c'est-à-dire l'entier* $r(R)$ *qui mesure la complexité de sa reconstruction en terme de périodicité) [N 95.1], avec un de nos précédents résultats concernant les motifs sur une variable [N 95], pour prouver que le problème peut être résolu en appliquant un algorithme de complexité* $O(|w|^4 + |R|^3 + |w|^{|\Delta|} r(R) \ln|w| \ln|R|)$.

Asymptotiquement, et dans le cas où Δ *possède au moins quatre lettres, on obtient une complexité en* $O(|w|^{|\Delta|} \ln^2|w|)$, *une généralisation directe du résultat de [N 95].*

1. INTRODUCTION

Given a finite alphabet Σ, a recursive subset L of Σ^* (the free monoid generated by Σ) and a string $w \in \Sigma^*$, the pattern matching problem consists in deciding whether or not there exists a substring of w which belongs to L. Many classical problems are concerned with this definition.

(*) Received October 1993, accepted September 1996.

(1) LIR, LITP, Université de Rouen, Faculté des Sciences, place E. Blondel, 76821 Mont Saint-Aignan Cedex, France.

e-mail: Jean.Néraud@dir.univ-rouen.fr

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/97/01/$ 7.00/

This paper is part of a series where we evaluate the complexity of the pattern matching problem, relative to a special class of languages L described by the morphic images of words.

Formally, given two finite disjoint sets, namely Σ, the alphabet of the "constants", and Δ, the alphabet of the "variables", we consider a word, say R, over $\Delta \cup \Sigma$. The language described by R, namely $L(R)$, is the set of all the words $\phi(R) \in \Sigma^*$, where ϕ stands for any morphism from $(\Delta \cup \Sigma)^*$ into Σ^* satisfying $\phi(a) = a$ for every "constant" $a \in \Sigma$. For instance, take $\Sigma = \{a, b, c, d\}$, $\Delta = \{X, Y, Z\}$ and consider the "pattern" $R = X^2aXYbZ \in (\Delta \cup \Sigma)^*$. By definition, the word $(ab)^2a^2bcbd$ belongs to $L(R)$. Indeed, such a string may be constructed by substituting ab for X, c for Y, and d for Z in the preceding word R.

Many scientific domains are concerned by the applications of that topic: indeed, beside the classically mentioned examples of text editing, or Molecular Biology, matching patterns with variables is also a helpful investigation tool in several domains of Computer Science, such as Rewriting Theory, or in the framework of infinite words [B 92].

From the point of view of the computational complexity, as shown in [An 80], given $R \in (\Delta \cup \Sigma)^*$, and given an arbitrary word $w \in \Sigma^*$, the problem of deciding whether $w \in \Sigma^* L(R) \Sigma^*$ or not is NP-complete. In fact, applying the naive algorithm requires time $O(|w|^{|\Delta|+2})$. This algorithm examines each of the $O(|w|^2)$ factors of w, and makes use of a diophantine equation in the lengths of the candidates $\phi(X)$, with $X \in \Delta$.

In some special cases of patterns, different results have been published:

- In the case of periodicities, *i.e.* one-variable patterns of type X^k, or two-variable patterns of the $(XY)^kX$ (k being a positive integer), different $O(|w|\ln|w|)$-time algorithms have been implemented (*cf.* e.g. [C 81], [AP 83], [ML 85] or [R 85]). Howether, the corresponding algorithms do not allow all the corresponding morphisms ϕ, and all the corresponding occurrences of $\phi(R)$ to be located in the input word w.
- For a one-variable pattern with constants, we constructed a $O(|w|^2\ln|w|)$-time algorithm for detecting all the morphisms, and all the corresponding occurrences [N 95].
- In the same paper we presented a $O(|w|^2\ln^2|w|)$-time algorithm for matching arbitrary two-variable patterns without constants, like XYX^2XY, or $X^2Y^3X^5Y$, ...

However, the question of improving the complexity of the general algorithm remained open.

In our new study, we conjugate the preceding result for one-variable patterns [N 95], with recent results concerning our so-called notion of *rank* of an arbitrary word $R \in (\Delta \cup \Sigma)^*$ [N 95.1] (and which allowed a $O(|R|^3 + |w|^4 + |w|^{|\Delta|+1} r(R))$-time improvement). Now, we show that the general problem may be decided in time $O(|w|^{|\Delta|} r(R) \ln|w| \ln(R))$, (where $r(R)$ stands for the rank of R), after a $O(|w|^4)$-time preprocessing phase on the word w itself, and a $O(|R|^3)$-time preprocessing phase on the pattern.

It is of interest to note that, from the point of view of asymptotic complexity, the integer $r(R)$ is negligible beside the length of the word w. That is, for $|\Delta| \geq 4$, we obtain a $O(|w| \ln^{|\Delta|} |w|)$-time algorithm, a direct generalization of the result in [N 95]. Once more our method is applied in an exhaustive way, so we may compute all the morphism ϕ, and all the occurrences of the word $\phi(R)$. We now present the contents of our paper.

Section 2 is concerned with the basic definitions used in our paper. The terminology of free monoids is settled, and the general problem of detecting morphic images is stated.

We recall the notion of *rank of a pattern*, which was introduced in [N 95.1], and a result concerning a one-variable prefix-pattern [N 95]: these two theoretical concepts which are the main feature of our new improvement.

In Section 3, we present conventions of implementations. In particular in a preprocessing phase, we compute the so-called "factor map" and "periodicity'map" of an arbitrary word, which avoid useless steps of computation and comparisons in the processing phase: this is the second feature of our method.

As a consequence, we obtain our main algorithm, which is presented in Section 4.

2. PRELIMINARIES

2.1. Definitions and notations

We adopt the standard notations of the free monoid theory: given a word w in Σ^* (the free monoid generated by Σ), we denote by $|w|$ its length, the empty word being the word of length 0.

Given two words u, $w \in \Sigma^*$, we say that u is a *factor* (resp. *prefix, suffix*) of w iff we have $w \in \Sigma^* u \Sigma^*$ ($u\Sigma^*$, $\Sigma^* u$). Given a factor f of w, an occurrence of f is every tuple (u, f, v) such that $w = ufv$. Given such an occurrence, the integer $|u| + 1$ is called the *position* of f. For each integer

$i \in [1, |w|]$, $w[i]$ stands for the letter of position i in w. Given a subset Γ of Σ, we denote by $|w|_\Gamma$ the number of occurrences of the letters of Γ in Σ.

A non-empty word w is primitive iff $w = x^n$ implies $n = 1$.

2.2. Matching morphic images of a word

Consider the following general problem (P):

Instance: – $\Delta = \{X_1, \ldots, X_m\}$ the finite alphabet of "variables"
– Σ, a finite alphabet of "constants", with $\Delta \cap \Sigma = \emptyset$
– R, a word over the alphabet $\Delta \cup \Sigma$ (the "pattern")
– $w \in \Sigma^*$, the "text"

Question: Decide whether or not a non-erasing morphism exists

$$\phi : (\Delta \cup \Sigma)^* \to \Sigma^*$$

(*i.e.* with ϵ not in $\phi(\Sigma \cup \Delta)$), such that the two following conditions hold:

– for every letter $a \in \Sigma$, we have $\phi(a) = a$
– $\phi(R)$ is a factor of w.

As shown in [An 80], if no bound for $|\Delta|$ is given, this problem is NP-complete. The preceding mapping ϕ will be called a *solution*. As already discussed:

CLAIM 1: *Given a constant alphabet, Δ, and given an arbitrary instance of the preceding problem, deciding whether or not a solution exists may be done in time $O(|R|^3 + |w|^{|\Delta|+1} r(R))$.*

Our new method of computation of the solutions, which is presented in Section 3 and Section 4, rests upon two previous results [N 95], [N 95.1]. It is now convenient to recall the main feature of these results.

2.3. The rank of a word

Let $w \in \Sigma^*$, and let $p \in [1, |w|]$. Consider the following equivalence $E^{[p]}$ in $[1, |w|] \times [1, |w|]$:
we have $(i, j) \in E^{[p]}$ iff the following property holds:

CONDITION 1: (i) $i + p - 1 \leq |w|$, $j + p - 1 \leq |w|$;
(ii) $w[i] \ldots w[i + p - 1] = w[j] \ldots w[j + p - 1]$.

In other words, the factor of w with length p and position i is equal to the factor with length p and position j. In [C 81], the author presents an $O(|w|\ln|w|)$-time algorithm for constructing all the equivalences $E^{[p]}$. This algorithm, in fact, detects any "maximal periodicity" in the word w.

Generating equivalence $E^{[1]}$

Clearly, R is completely determined by the preceding equivalence $E^{[1]}$. In [N 95.1], we present a method for constructing this equivalence, by making use of a (minimal) generating system of binary relations, which finally leads to the concept of *rank* of R.

Given a tuple of integers $(i, j, k) \in [1, |R|]^3$, with $i < j$, and $k \geq 1$, we denote by $\theta_{i,j}^{[k]}$ the symetric binary relation with elements $\theta_{i,j}^{[k]} = \{\{i, j\}, \{i+1, j+1\}, \ldots, \{i+k-1, j+k-1\}\}$, and by $\Theta(R)$ the set of the preceding relations $\theta_{i,j}^{[k]}$.

Given an arbitrary subset Θ_0 of $\Theta(R)$, we denote by Θ_0^* the equivalence they generate, *i.e.* the smallest equivalence containing all the elements of Θ_0, and we define the *rank* of R as the integer: $r(R) = \min\{|\Theta_0| : \Theta_0^* = E^{[1]}\}$.

PROPOSITION 1 [N 95.1]: *Given a word $R \in \Delta$, computing its rank (and computing a corresponding generating system of $E^{[1]}(R)$) requires time $O(|R|^3)$.*

Example 1: Let $\Delta = \{X, Y, Z\}$, and $R = XY\,ZXY\,ZXY\,XZY\,XZ$. The classes corresponding to the non-trivial equivalences $E^{[k]}$ are the following:

$E^{[1]}$: {1, 4, 7, 9, 12}, {2, 5, 8, 11}, {3, 6, 10, 13}

$E^{[2]}$: {1, 4, 7}, {9, 12}, {2, 5}, {8, 11}, {3, 6}, {10}, {13}

$E^{[3]}$: {1, 4}, {7}, {9}, {12}, {2, 5}, {8, 11}, {3, 6}, {10}, {13}

$E^{[4]}$: {1, 4}, {7}, {9}, {12}, {2, 5}, {8}, {11}, {3}, {6}, {10}, {13}

$E^{[5]}$: {1, 4}, {7}, {9}, {12}, {2}, {5}, {8}, {11}, {3}, {6}, {10}, {13}

We obtain the following minimal generating system:

$\Theta_0 = \{\theta_{1,4}^{[5]}, \theta_{8,11}^{[3]}, \theta_{7,9}^{[1]}, \theta_{3,6}^{[1]}\}$, thus we have $r(R) = 4$.

As an application we obtained a first improvement of the general algorithm for detecting all the morphic images of a word $R \in (\Delta \cup \Sigma)^*$ (*cf.* Claim 1).

The main feature of the algorithm consists in detecting the tracing into the word w itself of the binary relations of a minimal generating system of R.

2.4. One-variable prefix pattern

Let $\phi : (\{X\} \cup \Sigma)^* \to \Sigma^*$ be a morphism. Consider the following condition:

CONDITION 2: (i) $\phi(a) = a$, for every $a \in \Sigma$;
(ii) $\phi(X) \in (uv)^* u$

We say that ϕ satisfies Condition 2 with respect to the pair of words (u, v). The following result leads to implement an algorithm for detecting in time $O(|w|\ln|w|^2)$ all the occurrences of a one-variable pattern:

LEMMA 2 ([N 95], Section 3.2): *Given two words $w \in \Sigma^*$, $R \in (\{X\} \cup \Sigma)^*$, and given a pair of factors of w, $(u, v) \in \Sigma^* \times \Sigma^+$, with uv a primitive word, a unique one-unknown linear diophantine constraint (C) exists, such that the following property holds:*
For every morphism $\phi : (\{X\} \cup \Sigma)^* \to \Sigma^*$ satisfying Condition 2 with respect to (u, v), if $\phi(R)$ is a prefix of w, then the length of the word $\phi(X)$ satisfies (C).

The proof of Lemma 2 contains a precise computation of the constants that take place directly in the construction of constraint (C). We now recall the lists of these constants:

Let $(t_k)_{1 \leq k \leq n}$ be the unique sequence of words in Σ^* such that $R = t_0 X t_1 X t_2 \ldots X t_n$. Set $w' = t_0^{-1} w$:

– We denote by $i_{\max}$ the greatest positive integer (if it exists) such that $t_i \in (vu)^* v$, for each integer $i \in [1, i_{\max} - 1]$, and such that $t_{i_{\max}}$ is a prefix of a word in $(vu)^* u$ (otherwise, we set $i_{\max} = 0$).

– If $i_{\max} = n$, we denote by p the greatest integer such that $(uv)^p ut_n$ is a prefix of w'. Otherwise, p stands for the greatest integer such that $(uv)^p ut_n$ is a prefix of w'. With the second condition:

– if $t_{i_{\max}} \in (vu)^* v$, then q stands for the greatest non-negative integer such that $(vu)^q$ is a prefix of $t_{i_{\max}+1}$, otherwise, we denote by q the greatest non-negative integer such that $(vu)^q$ is a prefix of $t_{i_{\max}} (uv)^2$.

3. THE PREPROCESSING PHASE: CONVENTION OF IMPLEMENTATION

As already discussed, the results of Proposition 1 and Lemma 2 are the main features of our new algorithm. Howether, putting to use these results efficiently will require a special technique of implementation. We now

proceed to describe this technique. We shall assume that, given an arbitrary table, say t, the access to an arbitrary element $t[i]$ requires constant time.

From an algorithmic point of view, we represent words by linked lists of characters, sets of integers being also represented by lists. With this convention, concatening two words will be done in constant time.

3.1. The factor map

We shall also represent the occurrence of a factor u of w (with $u \neq \varepsilon$) by the unique pair of integers (i, j) such that $w = w[1] \dots w[i-1]uw[j+1] \dots w[|w|]$. In this way, the lexicographic ordering over the elements of $[1, |w|] \times [1, |w|]$ leads to a definition of a total ordering over the $O(|w|^2)$ different occurrences of the factors of the word w. Let $f_1 \prec f_2 \dots \prec f_p$ be the corresponding sequence of factors of w [1].

These factor occurrences, with respect to the monoidal structure of Σ^*, are controlled by introducing the five, following tables, namely *index, occ, fact, concat,* and *power*:

1. Given a factor occurrence (i, j), $index[i, j]$ stands for the smallest of the integers $k \in [1, p]$ such that f_k is equal to the factor of occurrence (i, j).
2. Given an integer $k \in [1, p]$, we denote by $occ[k]$ the smallest occurrence of the factor f_k and we set $fact[k] = f_k$.
3. For each factor occurrence (i, j), and for each integer $i' \in [i, j]$, we set:

$$concat[index[i, i'], index[i', j]] = index[i, j]$$

   Clearly, this leads to representing all the concatenations of pairs of (non-empty) factors (u_1, u_2) such that u_1u_2 remains a factor of w.
4. Given a factor occurrence $(i, j) \in [1, |w|]$ and given an integer $a \geq 2$, we set $power[[i, j], a] = index[i', j']$ iff $(fact[index[i, j]])^a$ remains a factor of w with occurrence (i', j').

According to these definitions, constructing the tables *index, occ* and *fact* requires time $O(|w|^2)$, and computing table *concat* will require time $O(|w|^3)$.

[1] This representation is different of the lexicographic ordering introduced over the factors themselves in [CR 94], p. 162).

Moreover, it is well-known that, by making use of the binary representation of the positive integer a, computing u^a requires $O(\ln a)$ concatenations. As a consequence, computing table *power* requires time $O(|w|^3 \ln |w|)$.

We say that the tuple of tables (*index, occ, fact, concat, power*) is the *factor map* of w [2].

CLAIM 2: *Given a word $w \in \Sigma^*$, computing its factor map requires time $O(|w|^3 \ln |w|)$.*

Moreover, the following result allows the amount of the reconstruction of the word $\phi(R)$ to be evaluated, for every morphism ϕ.

LEMMA 3: *With the convention of implementation, given a word $R \in (\Delta \cup \Sigma)^*$, and given a generating system Θ_0 of R, with $|\Theta_0| = r(R)$, for every morphism $\phi : (\Delta \cup \Sigma)^* \to \Sigma^*$, constructing the word $\phi(R)$ requires time $O(r(R) \ln |R|)$.*

Indeed, according to [N 95.1]; the construction of $\phi(R)$ comes down to the construction of the word R itself.

This computation is done by making use of a recursion tree, whose vertices different from the root may be identified to the elements of Θ_0. By definition, with every relation $\theta \in \Theta_0$, we associate the occurrence of a factor $F(\theta)$, of type $(uv)^m u$, that is $(uv)^a$, with $a = m + \frac{|u|}{|uv|}$ where m stands for a positive integer. More precisely, given a relation $\theta \in \Theta_0$, and given a factor f of the word $F(\theta)$, two sequences of words exist, namely $(f_i)_{0 \leq i \leq k}$, $(g_i)_{1 \leq i \leq k}$ such that both the following conditions hold:

(1) We have $f = (f_0 g_1 f_1 \dots g_k f_k)^a$, with $a \in \mathbf{Q}^+$.

(2) For each integer $i \in [1, k]$, there exists a relation θ_i, which is a son of θ in the preceding tree, and such that g_i is a factor of the word $F(\theta_i)$.

The factors g_i are computed recursively, when the factors f_i, and the rational number a may be associated to each vertex of the recursion tree. Since each vertex is examined at most once time, the computation of R may be done by applying $O(|\Theta_0| \ln |R|)$ concatenations.

[2] Note that we need only the comonents of these table corresponding to factors of w to be affected.

3.2. The periodicity map

The computation of the $E^{[p]}$-classes in [C 81], given a pair of integers $(i, p) \in [1, |w|] \times [1, |w|]$, allows to determined the three following tables to be determined, namely exp, c and SQR:

1. Let e be the greatest integer such that $(i, i + (e - 1)p) \in E^{[p]}$. We set $\exp[i, p] = e$.
2. We denote by $c[p, i]$ the unique $E^{[p]}$-equivalence class of i. We shall represent the class $c[p, i]$ by the smallest of its elements.
3. Given a factor occurrence with index $k \in [1, p]$, and given an integer $i \in [1, |w|]$, we denote by $SQR[k, i]$, the set of the factor occurrences (i, j) that satisfy the two following conditions:

   – $x = w[i] \dots w[j]$ is a primitive word.

   – x^2 is a prefix of the word $fact[k] \cdot (w[i] \dots w[|w|])$.

   According to [CR 91]:

CLAIM 3: *We have* $|SQR(w[i] \dots w[|w|]| \sim O(\ln|w|))$.

Constructing each set $SQR[k, i]$ may be done in time $O(|w|)$ by applying the KMP-algorithm [KMP 77]. This leads to the table SQR in time $O(|w|^4)$.

We say that the tuple of corresponding tables (exp, SQR, c) is the *periodicity map* of the word w:

CLAIM 4: *Given a word w, constructing its periodicity map may be done in time $O(|w|^4)$.*

Moreover, by construction:

LEMMA 4: *Let $w \in \Sigma^*$, and let (i, j), (i_1, j_1), (i_2, j_2) be three occurrences of factors of w. Let $u = w[i_1] \dots w[j_1]$ and $v = w[i_2] \dots w[j_2]$. After the computation of the periodicity map of w, each of the two following conditions may be decided in constant time:*

1) *Deciding whether the word $w[i] \dots w[j]$ belongs to $(uv)^* u$.*

2) *Computing the greatest integer e such that $(uv)^e u$ is a prefix of $w[i] \dots w[j]$.*

3.3 A special factorization of R

We complete the presentation of our preprocessing phase by indicating a particular decomposition of the word R, which allows to determine a candidate for the morphism ϕ in the processing phase to be determined.

Given the word $R \in \Delta^*$, for each integer $a \in [1, m]$, denote by p_a the smallest position of the letter X_a in the word R. Let j be the unique integer in $[1, m]$ such that $p_j = \max\{p_a : 1 \leq a \leq m\}$. We set $X = X_j$, and $\Delta_1 = \Delta \backslash \{X\}$ (in other words, X is the last letter to appear the first time when reading R from left to right). Without loss of generality, we may assume that we have $j = m$, thus $\Delta_1 = \{X_1, \ldots, X_{m-1}\}$.

Clearly, there exists a unique pair of words (R_1, R_2), such that the following condition holds:

CONDITION 3: *We have* $R = R_1 R_2$ *with* $R_1 \in (\Delta_1 \cup \Sigma^*)$, *and* $R_2 \in X(\Delta \cup \Sigma)^*$.

More precisely, the word R_2 may be factorized as $XT_1X \ldots XT_n$, with $n \geq 1$, and $T_a \in (\Delta_1 \cup \Sigma)^*$.

4. THE PROCESSING PHASE

In all the sequel of our paper, we assume that we have computed the factor map and the periodicity map of the words w, and R in a processing phase. Set $\Delta = \{X_1, \ldots, X_m\}$. Let $R \in (\Delta \cup \Sigma)^*$, $w \in \Sigma^*$. It is convenient to introduce the following condition:

Let $i \in [1, |w|]$, $(x_a)_{1 \leq a \leq m-1} \in [1, |w|]^{m-1}$, and let $\phi : (\Delta \cup \Sigma)^* \to \Sigma^*$ be a morphism. We say that ϕ satisfies Condition 4 with respect to the tuple $(i, (x_a))$ iff the following condition holds:

CONDITION 4: (i) *for each integer* $a \in [1, m-1]$, *we have* $\phi(X_a) = x_a$;
(ii) $\phi(R)$ *is a prefix of* $w[i] \ldots w[|w|]$.

Let $i \in [1, |w|]$ such that there exists a morphism ϕ, with $\phi(R)$ a prefix of $w[i] \ldots w[|w|]$. Let $(x_a)_{1 \leq a \leq m-1}$ be a $(m-1)$-uple in $[i, |w|]^{m-1}$, and let ϕ be a morphism satisfying Condition 4, with respect to the type $(i, (x_a))$. Trivially, the integer $j = i + |\phi(R_1)|$, is the position of the factor $\phi(R_2)$ in the corresponding occurrence of $\phi(R)$.

We shall explain how to decide whether or not $\phi(R_2)$ is a prefix of $w[j] \ldots w[|w|]$. With respect to the preceding notations, consider a pair of

words (u, v) such that $\phi(X) \in (uv)^* u$ (recall that, according to [CR 91], we get $O(\ln|w|)$ candidates for such a pair).

4.1. Constructing constraint (C)

This construction rests upon the computation of the constants $i_{\max}$, p, q, in the proof of Lemma 2. Recall that we set $R_2 = XT_1X \ldots XT_n$, with $T_a \in (\Delta_1 \cup \Sigma)^*$.

LEMMA 5 : *Given the preceding integer tuple* $(i, (x_a)_{1 \leq a \leq m-1})$, *given the family of words* (T_a) $(1 \leq a \leq n)$, *and given the preceding pair of words* (u, v)*:*

(1) *Computing* $i_{\max}$ *requires time* $O(|R|\ln|R|)$.

(2) *Computing the integers* p, q *requires time* $O(r(R))$.

Proof of Lemma 5: As a corollary of Lemma 4, after the computation of the factor map and the periodicity map of the word w, given an integer tuple $(x_a)_{1 \leq a \leq m-1}$, and given an integer $a \in [1, n]$:

– computing the word $\phi(T_a)$ may be done in time $O(r(R))$ (indeed, T_a is a factor of R, and we have $T_a \in \Delta_1^*$).

– after that, deciding whether or not we have $\phi(T_a) \in (uv)^* u\,(\phi(T_a) \in \text{pref}\,(uv)^* u))$ requires constant time.

It is well-known that computing the greatest integer a that satisfyies the preceding condition may be done in time $O(\ln|R_2|)$ by applying a method of binary search in the sequence $\phi(T_a)$ (*see* e.g. [K 73] or [CLR 90]). Note that computing all the words $\phi(T_a)$ is not necessary. This establishes assertion (1).

Moreover, once more according to Lemma 4, after the computation of the word $T_{i_{\max}+1}$ the computation of the integers p and q may be done in constant time by making use of table exp in the periodicity map, hence the second assertion follows. ■

4.2. The new algorithm

Since solving the preceding constraint (C) allows (in constant time) a candidate for the morphism ϕ to be completely determined, we get a new improvement of the general naive algorithm for deciding whether or not Problem (P) has solutions. This algorithm is informally described as follows:

Having, in the preprocessing phase, computed the factor maps and the periodicity maps of the words R and w, for each prefix w_0 of the input word

w, and for all the $(m-1)$-uple of integers $(x_1, \ldots, x_{m-1}) \in [|w_0|, |w|]$, we shall apply each of the following steps:

- In Step 1, we compute a candidate for the word $\phi(R_1)$, and the corresponding word $\phi(T_1)$.
- After that, Step 2 consists in investigating the corresponding set of primitive words t such that t^2 is a prefix of $\phi(T_1) \cdot w_0^{-1} w$, as follows:
  - For each of the preceding words x, in Step 2.1, we compute the corresponding constraint (C), and in Step 2.2, we shall decide whether or not (C) has at least one solution.
  - If the answer is positive, then we must decide whether or not the corresponding word $\phi(R_2)$ is a prefix of $w_0^{-1} w$ (Step 2.3). This will be done by applying in time $O(r(R))$ the "comparison phase" which was presented in [N 95.1], Section 4.2. Note that if (C) is an inequation, the verification step may be done by considering the longest of its solutions [3].

The different steps are effectively implemented by making use of the factor maps, and periodicity maps of the words R and w. In this way, the computation of a factor f is done by the computation of its smallest occurrence in the lexico-graphic ordering.

The main scheme of this algorithm is described as follows:

Algorithm

input: $R \in (\Delta \cup \Sigma)^*$
$w \in \Sigma^*$;
output: answer to Problem (P)
$\{m = |\Delta|\}$

{Preprocessing phase}
compute the factor maps and the periodicity maps of w and R
and a generating system of the preceding words R, R_1, R_2, T_1
compute the preceding words R_1, R_2
$S_1 \leftarrow |R_1|_\Sigma$

{Processing phase}
answer $\leftarrow$ *FALSE*;
for each integer $i \in [1, |w|]$ **do**
  for each $(m-1)$-uple $(x_1, \ldots, x_{m-1} \in [i, |w|]^{m-1})$ **do**
    {Step 1}

[3] Moreover, an interval of solutions may be given. With such a convention, we get the same result of complexity.

compute the corresponding word $\phi(T_1)$
$j \leftarrow i + \sum_{1 \leq i \leq m-1} x_i + S_1$ {position of $\phi(T_1)$}
$k \leftarrow$ the unique integer such that $\phi(T_1) = fact[k]$
{Step 2}
for each word y in $SQR[k, j]$ **do**
compute the corresponding tuple of integers ($|u|$, $|v|$, $i_{\max}$, p, q) {Step 2.1}
if at least one solution x for Constraint (C) **then** {Step 2.2}
$x_k \leftarrow x$
{Step 2.3}
for each relation $\theta_{i,j}^{[k]}$ in the generating system of R_2 **do**
compute the integers i', j', k' as indicated in [N 95.1] (Section 4)
if $(i', j') \in E^{[k]}(w)$ **then** *answer* $\leftarrow$ *TRUE* **endif**
endfor
endif
endfor
endfor
endfor
endalgorithm

Complexity of algorithm 3

- As we indicated above, the preprocessing phase requires time $O(|w|^4 + |R|^3)$.
- Let $i \in [1, |w|]$, and let $(x_1, \ldots, x_{m-1} \in [i, |w|]^{m-1})$. According to Lemma 3, since the words T_a are factors of R, the computation of the words $\phi(T_a)$ $(1 \leq a \leq n)$ requires time $r(R)\ln|R|$. Computing the integer j is done in constant time, the access to the set $SQR[k, j]$ itself being obtained in constant time.
- According to Lemma 5, computing the integers $i_{\max}$, $|u|$, $|v|$ in Step 2.1 requires time $O(r(R)\ln|R|)$. Clearly, solving the corresponding constraint (C) requires constant time.
- Finally, in Step 2.3, deciding whether the corresponding candidate ϕ is a solution of Problem (P) requires $O(r(R))$ comparisons.
- Now, according to [CR 91], we have $|SQR[k, j]| \sim O(\ln|w|)$ times. Moreover, by construction, we consider $O(|w|^{|\Delta|})$ tuples $(i, x_1, \ldots, x_{k-1})$.

As a consquence,

THEOREM 6: *Given a pair of arbitrary words w, R, deciding whether or not there exists a morphism ϕ such that $\phi(R)$ is a factor of w, may be implemented so that it requires time $O(|w|^4 + |R|^3 + |w|^{|\Delta|} r(R)\ln|w|\ln|R|)$.*

Moreover, this method allows all the morphisms ϕ to be exhaustively computed, with all the occurrences of the corresponding word $\phi(R)$.

REFERENCES

[An 80] D. ANGLUIN, Finding Patterns Common to a Set of Strings, *Journal of Computer and Syst. Sci.*, 1980, *21*, pp. 46-62.

[AC 75] A. AHAO and M. CORACICK, Efficient String Matching: An Aid to Bibliographic Search, *Comm. ACM*, 1975.

[AP 83] A. APOSTOLICO and F. P. PREPARATA, Optimal off-line detection of repetitions in a string, *Theoret. Comput. Sci.*, 1983, *22*, pp. 297-315.

[B 92] K. BAKER, Open problems on avoidable and unavoidable patterns, manuscript (Université de Rouen, France).

[C 81] M. CROCHEMORE, An optimal algorithm for computing the repetitions in a word, *Information Proc. Letters*, 1981, *12*, pp. 244-250.

[CN 89] M. CROCHEMORE and J. NÉRAUD, Unitary monoid with two generators: an algorithmic point of view, in: *Proceedings of CAAP'90*, 1990.

[CLR 90] T. TCORMEN, C. LEISERSON and R. RIVEST, "Introduction to Algorithm", The MIT Press, 1990.

[CR 91] M. CROCHEMORE and W. RYTTER, Periodic prefixes of string, in: *Acts of Sequences'91*, 1991.

[CR 94] M. CROCHEMORE and W. RYTTER, "Text Algoriths", Oxford University Press, 1994.

[GS 83] Z. GALIL and J. SEIFERAS, Saving space in fast string-matching, *SIAM J. Comput.*, 1980, pp. 417-438.

[JSSY 95] TAO JIANG, A. SALOMAA, K. SALOMAA and SHENG YU, Decision problems for patters, *Journ. of Comput. Syst. Sci.*, 1995, *50*(1), pp. 53-63.

[K 73] D. KNUTH, "Sorting and Searching", vol. 3, of "The Art of Computer Programming". Addison Wesley 1969, Second edition, 1981.

[KMP 77] D. KNUTH, J. MORRIS and V. PRATT, Fast pattern matching in string, *SIAM J. Comput.*, 1977, *6*, No. 2, pp. 323-350.

[Lo 83] M. LOTHAIRE, "Compbinatorics on words", Encyclopedia of Mathematics and appl., Addison Wesley Publish. Company, 1983.

[ML 85] G. MAIN and J. LORENTZ, Linear time recognition of squarefree strings, in "Combinatoric Algorithms on Words", A. Apostolica and Z. Galil eds., NATO ASI, Springer Verlag, Berlin, 1985 (1), pp. 5-37.

[N 95] J. NÉRAUD, Algorithms for detecting morphic images of a word, *Info. and Comput.*, 1995, *120*, No. 1, pp. 126-148.

[N 95.1] J. NÉRAUD, Detecting morphic images of a word: On the rank of a pattern, *Acta Info.*, 1994.

[R 85] O. RABIN, Discovering repetitions in strings, in "Combinatoric Algorithms on Words", A. Apostolico and Z. Galil eds., NATO ASI, Springer Verlag, Berlin, 1985.