String Matching Algorithms and Automata

Imre Simon*

Instituto de Matemática e Estatística
Universidade de São Paulo
05508-900 São Paulo, SP, Brasil
<is@ime.usp.br>

Abstract. In this paper we study the structure of finite automata recognizing sets of the form A^*p, for some word p, and use the results obtained to improve the Knuth-Morris-Pratt string searching algorithm. We also determine the average number of nontrivial edges of the above automata.

1 Introduction

It has been known for a long time that sequential string matching algorithms, such as the Morris and Pratt [15] or the Knuth, Morris and Pratt [14] algorithms are intimately related to finite automata recognizing sets of the form A^*p, where $p \in A^*$ is the pattern to be found. This approach, found in [1, 8, 18, 7, 2], was not further developed probably because such an automaton has $|A||p|$ edges while the before mentioned algorithms work in time $O(|p| + |t|)$, for a text t and pattern p, with a proportionality constant independent of the alphabet size.

In the next section we shall show that most edges of the automaton for A^*p are not essential. More precisely, we shall classify the edges of $\mathcal{A}$ as either forward, backward or trivial and shall show that $\mathcal{A}$ has exactly $|p|$ forward edges and at most $|p|$ backward edges. Furthermore, knowledge of these edges is sufficient to simulate the automaton in linear time, without dependence of the alphabet size.

Using these ideas we show an implementation of automaton $\mathcal{A}$ which leads to an improvement of the Knuth, Morris and Pratt algorithm, in the sense that for any pattern p and text t the set of character comparisons of our algorithm is a (usually proper) subset of the set of character comparisons of the Knuth, Morris and Pratt algorithm. Besides, our algorithm is a real-time string matching algorithm, with delay bounded by $|A|$. Using techniques of Galil [10] this delay can be reduced to be bounded by a constant independent of the alphabet size.

We remark that Hancart [12, 13] has shown the surprising result that any "reasonable" implementation of our automaton leads to a linear pattern matching algorithm.

In section 4 we make a detailed study of the automaton $\mathcal{A}$ and compute the average number of its nontrivial edges. This is achieved by using an easy and elementary method to count the number of un-bordered words of a given length over a fixed alphabet.

* This work was done with partial support from FAPESP, CNPq and the Fibonacci Institute.

2 Finite automata recognizing sets of the form A^*p

Let p be a nonempty word in A^* and let $\mathcal{A}$ be the reduced finite automaton recognizing the set A^*p. Assume that p has length m and let us denote by $p[i]$ the i-th letter of p, i.e. $p = p[1]p[2]\ldots p[m]$. The factor $p[i]\ldots p[j]$ of p will be denoted by $p[i..j]$.

Initially we present a construction of $\mathcal{A}$ done purely in syntactic terms. Recall that any set $X \subseteq A^*$ defines a coarsest right congruence on A^*, its *syntactical right congruence,* for which X is a union of congruence classes. These congruence classes can be considered the states of the reduced automaton accepting X, the initial state being the class containing the empty word 1 and the final states being the classes which contain some word of X. Furthermore, for words u and v the congruence is defined by

$$u \equiv v \ (\operatorname{mod} X) \text{ iff } \forall x \in A^* \ : \ ux \in X \text{ iff } vx \in X.$$

In the sequel we apply these concepts to the set $X = A^*p$.

For a word u we denote by $\operatorname{pre}(u), \operatorname{suf}(u), \operatorname{fat}(u)$ the set of prefixes, suffixes and factors of u; in other words, $\operatorname{pre}(u) = uA^{*-1}$, $\operatorname{suf}(u) = A^{*-1}u$ and $\operatorname{fat}(u) = A^{*-1}uA^{*-1}$, where we use the quotient notation developed by Eilenberg [9]. For words u, v the *overlap of* u *and* v (in that order) is the longest word in $\operatorname{suf}(u) \cap \operatorname{pre}(v)$. It is denoted by $u \rightleftharpoons v$. The *borders of* u are the nonempty words in $\operatorname{suf}(A^{-1}u) \cap \operatorname{pre}(uA^{-1})$. In other words, v is a border of u if $0 < |v| < |u|$ and $u \in vA^* \cap A^*v$. A word is *un-bordered* if the set of its borders is empty. Note that the empty word is un-bordered. We denote by $\operatorname{alpha}(u)$ the set of letters which appear in the word u.

The reader will verify without difficulty that

$$u \equiv v \ (\operatorname{mod} X) \text{ iff } u \rightleftharpoons p = v \rightleftharpoons p.$$

Clearly p has $m+1$ prefixes and no two of them are congruent $\operatorname{mod} X$, hence we can denote the states of $\mathcal{A}$ by $0, 1, \ldots m$ with 0 the initial state, with m the final state and satisfying the property that $0p[1..i] = i$, that is to say, the prefix of length i of p takes automaton $\mathcal{A}$ from state 0 to state i. Figure 1 represents automaton $\mathcal{A}$ for the word $p = abacabadabacaba$.

Clearly A^*p is a definite language, in the sense of [5, 16]; hence $\mathcal{A}$ is a definite automaton, i.e. for every sufficiently long word $w \in A^*$ the map defined by w on the states of $\mathcal{A}$ is a constant function.

Besides, automaton $\mathcal{A}$ has the following basic properties. For every word w, $0w = |w \rightleftharpoons p|$, i.e. w takes the initial state to the state corresponding to the length of the longest suffix of w which is also a prefix of p. Furthermore, the triple (i, a, j) is an edge of $\mathcal{A}$, i.e. state i goes to state j under letter a, iff $p[1..i]a \rightleftharpoons p = p[1..j]$.

Notice next that most edges of the automaton $\mathcal{A}$ end at the initial state 0; we shall call such edges *trivial*. The non-trivial edges of $\mathcal{A}$, (i, a, j), with $j \neq 0$, will be called *forward* or *backward* according $j = i+1$ or $j \leq i$, respectively. Clearly, all edges of $\mathcal{A}$ have been classified. We need another definition: the *length* of the

q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
qa	1	1	3	1	5	1	7	1	9	1	11	1	13	1	15	1
qb	0	2	0	2	0	6	0	2	0	10	0	2	0	14	0	2
qc	0	0	0	4	0	0	0	4	0	0	0	12	0	0	0	4
qd	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	8

Fig. 1. The reduced automaton $\mathcal{A}$ recognizing A^*p, for $p = abacabadabacaba$.

backward edge (i, a, j) is the integer $i - j$. We state now one of the main results of the paper.

Theorem 1 *For every $p \in A^*$, of length m, the reduced automaton $\mathcal{A}$ recognizing A^*p has exactly m forward edges, its number of backward edges is at least $|\mathrm{alpha}(p)|$ and at most m.*

The extreme values in theorem 1 are actually obtained. To see this, consider the word $p = a_1^{q_2} a_2 a_1^{q_3} a_3 \cdots a_1^{q_k} a_k$, with $0 \leq q_3, q_4, \ldots, q_k < q_2$. It is not difficult to see that $\mathcal{A}$ has exactly k backward edges. On the other hand, a very interesting sequence of words is given by the so called *sesquipowers*, of which the word of figure 1 is an example. Consider a sequence of letters $(a_1, a_2, \ldots)$ and construct the words $p_1 = a_1$ and $p_{n+1} = p_n a_{n+1} p_n$, for $n \geq 1$. The reader will verify that, unless $a_i = a_{i+1}$ for some $i \leq n$, the automaton $\mathcal{A}_n$ has exactly $|p_n|$ backward edges.

Actually, we can characterize an even more precise set which contains always m elements, which is easily constructed, and from which the set of edges of $\mathcal{A}$ can be obtained. The *return set* of a word p, denoted $\mathrm{Ret}(p)$ is a subset of $[1..m] \times A \times [1..m]$ which contains an element for every $k \in [1, m]$. The element corresponding to k is $(k + j, p[j + 1], j + 1)$, where $j \in [0..m - 1]$ is the unique value given by

$$p[k + 1..k + j] = p[1..j] \text{ and } p[k + j + 1] \neq p[j + 1] \text{ or } k + j = m.$$

The element corresponding to k can be obtained as follows. Let us spell word p in p itself, as far as possible, beginning to match $p[1]$ with $p[k + 1]$; assume that we succeed to match j letters but not more. Then the triple $(k+j, p[j+1], j+1)$ belongs to the return set of p. Notice that whatever the resulting value of j, the length of the triple corresponding to k is always $k - 1$. It follows that $\mathrm{Ret}(p)$ has exactly one triple of each length $0, 1, \ldots, m - 1$. Hence, $|\mathrm{Ret}(p)| = m$.

We have now:

Theorem 2 *The triple (i, a, j) is a backward edge of $\mathcal{A}$ iff it belongs to the return set of p and no other triple (i, a, k), with $k > j$, belongs to* $\mathrm{Ret}(p)$.

3 Application to string matching

In this section we informally explore the consequences of our study of the automaton $\mathcal{A}$ recognizing A^*p for string matching algorithms.

Let us first note that the Knuth-Morris-Pratt algorithm can be easily described in terms of the automaton $\mathcal{A}$. Indeed, that algorithm, based on character comparisons, computes a failure function which is to be used whenever the matching of p with the text t comes to a mismatch or reaches an occurrence of p in t. More precisely, assume that comparison of $t[i]$ with $p[j]$ resulted in a mismatch. Then we substitute j by fail(j) and repeat the algorithm. Unless fail$(j) = 0$, in which case we abandon the current value of i and re-initialize everything with $i+1$ for i and 1 for j.

But what is fail(j)? It is defined, for $1 \leq j \leq m$, as being the greatest k for which $p[1..k-1]$ is a border of $p[1..j-1]$ and for which $p[k] \neq p[j]$ or is 0 if no such k exists. By analogy, we also define fail$(m+1)$, to be used after a complete match of p has been found, as being 1 plus the length of the longest border of p. One can verify that fail$(j) = 0$ if there are no backward edges issuing from state $j-1$ or is the terminus of the shortest return edge issuing from $j-1$. Besides, the label of such edge is always $p[\text{fail}(j)]$ in consequence of the definiteness of our automaton, hence comparing $t[i]$ with that label nothing else is but comparing $t[i]$ with $p[\text{fail}(j)]$.

As an illustration, we show on figure 2 the failure function of the word of figure 1.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
fail(j)	0	1	0	2	0	1	0	4	0	1	0	2	0	1	0	8

Fig. 2. The failure function of $p = abacabadabacaba$.

Thus, the Knuth-Morris-Pratt algorithm can be seen as being based on a representation of the automaton $\mathcal{A}$. Such representation is achieved through the failure function. This strategy succeeds because of the following two properties of the definite automaton $\mathcal{A}$.

- every edge terminating at state j has label $p[j]$,
- if $0 \neq \text{fail}(j) = k$ then for every edge $(k-1, a, l)$, $\mathcal{A}$ also has the edge $(j-1, a, l)$.

In other words, iterating successively the failure function one recovers all the edges of $\mathcal{A}$ but some of them might be recovered more than once.

The foregoing discussion illustrates why is the use of the automaton $\mathcal{A}$ possibly more efficient than its simulation by the failure function? We realize a gain whenever we are looking for an edge with label c and the successive values of the failure function lead to letters alternating between, say, a and b, before hitting a c. In this situation the automaton simulation examines only once the transition a and only once the transition b, while the Knuth-Morris-Pratt algorithm can spend up to $\Omega(\log m)$ consecutive jumps of the failure function before finding the correct transition labeled by c. Examples when this behavior arises can be constructed by suitable sesquipowers. Take, for instance, the sesquipower u defined by the sequence (c, a, b, a, b, a, b) and then search for the pattern $p = udu$ in the text $t = (uc)^n$.

This analysis can be resumed by saying that our string matching algorithm simulating the automaton $\mathcal{A}$ is a real-time algorithm with delay bounded by $|A|$, while the Knuth-Morris-Pratt algorithm does not work in real-time.

It remains to be seen whether the automaton $\mathcal{A}$ can be constructed and simulated in linear time. This is indeed the case and this also leads to some interesting considerations.

First, we shall describe a data structure to store and search the edges which corresponds to the Knuth-Morris-Pratt algorithm. This is achieved by storing, for every state j the number back(j) of backward edges with origin in j and storing the set of backward edges with origin in j in a vector or list edge(j) in increasing order of length. With this data structure and exploiting the properties of automaton $\mathcal{A}$ one obtains:

Theorem 3 *The automaton $\mathcal{A}$ can be constructed for a pattern p and simulated for any text t using a set of letter comparisons which is a (proper) subset of the letter comparisons performed by the Knuth-Morris-Pratt algorithm. It follows that the construction and the simulation of automaton $\mathcal{A}$ can be done in time and space proportional to $|p| + |t|$, where the constant of proportionality does not depend on the size of the alphabet.*

We mention now a surprising property discovered by Christophe Hancart [12, 13]. He proved that the time to spell a text t in the automaton $\mathcal{A}$ is bounded by at most $2|t| - 1$ comparisons even if *all* the edges issuing from the current state are consulted while computing each transition. Note that the bound $2|t| - 1$ is the same as the one obtained in the Knuth-Morris-Pratt algorithm! This intuitively means that the edges in the automaton are arranged so sparsely that one never "crosses" with more than $2|t| - 1$ edges when spelling a word t. It also implies that the edges in the lists edge(j) can be stored in any order whatsoever without compromising the time bound $2|t| - 1$ necessary to spell the word t. Actually, the edges in edge(j) can even be randomly permuted *before* consulting each list.

We conclude this section with an interesting problem. Does there exist some representation, comparable to the ones we have seen for the automaton of A^*p, for the suffix automaton of a word, as described by Blumer et. al. [4]? Such a structure would be especially interesting if it provided a representation of the suffix automaton in linear space which could be traversed in linear time with constants independent of the alphabet size. As far as we know the existence of such a representation is an open question.

4 The Average Number of Backward Edges

In this section we will deepen our analysis of the return set of a word p in order to obtain the average number of backward edges of automata $\mathcal{A}$ recognizing A^*p, when p runs over the set of words of length m. We shall show that as m goes to infinity the limit of the average number of edges divided by m tends to a constant depending only on $|A|$ and whose value is always less than 1. To achieve our objectives we introduce an easy and elementary method for counting the un-bordered words of a given length over a finite alphabet.

We begin with an alternate characterization of the return set of p. We call the *neighbor set* of p the set $\mathrm{nb}(p) = \mathrm{pre}(p)A - \mathrm{pre}(p)$. The reason behind the terminology is that if we represent the words of A^* in a regular tree then the vertices corresponding to $\mathrm{nb}(p)$ are exactly those (graph-theoretic) neighbors of the path spelling p which do not belong to the path itself.

Theorem 4 *For every word $p \in A^*$ of length m,*

$$\mathrm{Ret}(u) = \{\, (|u|, a, |v|) \mid a \in A,\ ua \in \mathrm{nb}(p) \textit{ and } v \textit{ is a border of } ua \,\}.$$

It follows that for any p the total number of borders of the neighbor set of p is exactly m.

Note that theorems 2 and 4 show that the values of the function $f(u)$, the number of borders of word u, considered on the neighbor set of p give important information on $\mathcal{A}$. Indeed, state j has a backward edge with label a iff the a-neighbor of $p[1..j]$ has a non-null f-value. Hence, the number of backward edges is exactly the number of neighbors whose f-value is non-null. Since the sum of the f-values on all neighbors is exactly m, the more neighbors with f-values greater than 1 we have and the greater those f-values are the less backward edges we have in $\mathcal{A}$.

In the sequel we shall denote by q the cardinality of the nonempty alphabet A. Let us denote by $U(r,q)$ the number of un-bordered words of length r over a q-letter alphabet, i.e.

$$U(r,q) = |\{\, u \in A^r \mid A^{-1}u \Rightarrow uA^{-1} = 1 \,\}|.$$

Let us denote by $T(r,q)$ the total number of borders of words of length r over a q-letter alphabet, i.e.

$$T(r,q) = |\{\, (u,v) \mid u \in A^r, \quad 0 < |v| < r, \quad u \in vA^* \cap A^*v \,\}.$$

We shall also need the values of $U(r,q)$ and $T(r,q)$ relative to q^r, the total number of words of length r over A, i.e.

$$u(r,q) = U(r,q)/q^r \text{ and } t(r,q) = T(r,q)/q^r.$$

We have now a consequence of theorem 4 and the analysis following it.

Proposition 5 *The average number $E(m,q)$ of backward edges of reduced automata recognizing A^*p, for words p of length m over a q-letter alphabet is*

$$E(m,q) = 2m + q + 1 - q(t(m+1,q) + u(m+1,q)) - \sum_{r=0}^{m}(t(r,q) + u(r,q)).$$

Now we make an excursion in order to compute the numbers $t(r,q)$ and $u(r,q)$. Initially we state an easy lemma.

Lemma 6 *Let v be the shortest border of u or 1 if u is un-bordered. Then, $u \in vA^*v$, i.e. v is a non-overlapping border of u.*

Our initial aim is to determine the number $U(r,q)$ of un-bordered words of length $r \geq 0$ over an alphabet of $q \geq 1$ symbols. These values can be computed using results in [11, 17], but our arguments are simpler then the previous ones.

Theorem 7 *Let $q \geq 1$. The numbers $U(r,q)$ satisfy the following recurrence:*

$$U(r,q) = \begin{cases} 1 & \text{if } r = 0, \\ qU(r-1,q) & \text{if } r \text{ is odd}, \\ q^2U(r-2,q) - U(r/2,q) & \text{if } r \text{ is even}. \end{cases}$$

Proof. Assume first that $r > 0$ is odd, say $r = 2n+1$. Then every $u \in A^{2n+1}$ has a factorization $u = v_1av_2$ for appropriate words $v_1, v_2 \in A^n$ and a letter $a \in A$. Due to lemma 6 u is un-bordered iff v_1v_2 is un-bordered. It follows that $U(2n+1,q) = qU(2n,q)$.

Assume now that $r > 0$ is even, say $r = 2n+2$. Then every $u \in A^{2n+2}$ has a factorization $u = v_1abv_2$ for appropriate words $v_1, v_2 \in A^n$ and letters $a, b \in A$. Due to lemma 6 u is un-bordered iff v_1v_2 is un-bordered and $v_1a \neq bv_2$. Further, let $u = v_1abv_2$ be such that $|v_1| = |v_2|$ and $a, b \in A$. Then v_1v_2 is un-bordered iff v_1a is un-bordered. It follows that $U(2n+2,q) = q^2U(2n,q) - U(n+1,q)$. The proof is complete. ∎

An immediate consequence of theorem 7 is that the function $U(r,q)$ is a polynomial of degree r in q. Further, $U(r,1)$ is 1 for $r = 1$ and is zero otherwise. It follows that $u(r,q)$ is a polynomial in q^{-1}; further, we have the following result.

Theorem 8 *For every $q > 0$ the limit*

$$u(q) = \lim_{r\to\infty} u(r,q)$$

exists and is a series in q^{-1}. Besides, the terms of degree at most n in q^{-1} of $u(q)$ and $u(2n,q)$ are the same. Finally, for every $q \geq 2$, $(q-2)/(q-1) < u(q) < (q-1)/q$, .

Proof. The second sentence follows from the fact that the recurrence in theorem 7 implies that for every $m \geq n$, $u(n,q)$ and $u(m,q)$ have the same terms of degree at most $n/2$ in q^{-1}. This implies the first sentence. To see the last sentence it suffices to observe that, for $r \geq 4$, $U(r,q) \leq q^r - q^{r-1} - q^{r-3}(q-1)$, since (a) every word in $aA^{r-2}a$, for $a \in A$, has a border and there are q^{r-1} such words, (b) every word in $abA^{r-4}ab$, for $a, b \in A$, $a \neq b$, has a border and there are $q^{r-3}(q-1)$ such words, and (c) the sets in (a) and (b) are disjoint and nonempty. It follows that $u(r,q) < 1 - 1/q = (q-1)/q$. The proof of the lower bound is a bit more elaborate. It is based on the fact that $u(r,q) \leq 1$ and this allows to obtain a recursive sequence, say $l(r,q)$, whose value is at most $u(r,q)$ and which tends to a value strictly greater than $(q-2)/(q-1)$ when r tends to ∞. ∎

The initial terms of $u(q)$ can be easily computed with Maple [6]. The series begins as follows:

$u(q) = 1 - q^{-1} - q^{-2} + q^{-6} + q^{-9} + q^{-12} - q^{-14} + q^{-15} - q^{-17} + q^{-18} - q^{-20} - q^{-23} - q^{-26} - q^{-28} - q^{-29} + q^{-30} - q^{-31} - q^{-32} + q^{-33} - q^{-34} - 2\,q^{-35} + q^{-36} + \cdots$.

Table 1 shows the values of $u(q)$ for small values of q.

q	2	3	4	5	6	7	8
$u(q)$	.267787	.556980	.687748	.760065	.805577	.836743	.859379

Table 1. Values of $u(q)$ for small q.

The value of $t(r,q)$ is easier to obtain.

Proposition 9 *For every $q, r > 0$, $t(r,q) = \sum_{p=1}^{r-1} 1/q^p$. Consequently, for $q > 1$, $t(q) = \lim_{r\to\infty} t(r,q) = 1/(q-1)$.*

The above results and proposition 5 can be used to compute $E(m,q)$ which is a polinomial in q for every m. In view of theorem 1 it is interesting to compare $E(m,q)$ to m. The next result shows that the sequence $E(m,q)/m$, $m = 1, 2, \ldots$, converges to a value strictly less than one, which is dependent on q.

Theorem 10 *For every $q \geq 2$, the limit $e(q)$,*

$$e(q) = \lim_{m\to\infty} E(m,q)/m = 2 - t(q) - u(q),$$

hence $e(q)$ is a series in q^{-1}. Further, for every $q \geq 2$, $e(q) < 1$.

Again, the initial terms of the series $e(q)$ can be easily computed. They are as follows and were used to compute the first few numerical values of $e(q)$ shown in table 2.

$e(q) = 1 - q^{-3} - q^{-4} - q^{-5} - 2\,q^{-6} - q^{-7} - q^{-8} - 2\,q^{-9} - q^{-10} - q^{-11} - 2\,q^{-12} - q^{-13} - 2\,q^{-15} - q^{-16} - 2\,q^{-18} - q^{-19} - q^{-21} - q^{-22} - q^{-24} - q^{-25} - q^{-27} - 2\,q^{-30} - 2q^{-33} + q^{-35} - 2\,q^{-36} + \cdots$.

q	2	3	4	5	6	7	8
$e(q)$	.732213	.943020	.978916	.989935	.994423	.996590	.997764

Table 2. Values of $e(q)$ for small q.

TEMPORAL NOTES The main property of this paper, namely that the automaton $\mathcal{A}$ has at most $|p|$ backward edges was discovered by the author in 1989, based on the work of Knuth, Morris and Pratt [14] and especially Duval [8], while trying to prepare a text covering the most important algorithms on words in everyday use. Unfortunately he was unable to conclude that text, actually he was even unable to publish these results in due time. In the meanwhile, some renderings of the ideas of the first three sections appeared, based on lectures he gave in Paris, Naples and São Paulo. These include the book of Beauquier, Berstel and Chrétienne [3], an examination problem in the Concourse 1990 to the École Normale Superieure and the Doctoral Dissertation of Hancart [12, 13]. He sincerely hopes that he will be able to publish the complete proofs of the results here reported before such a long time passes again!

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. M. V. A. Andrade. Métodos eficientes para reconhecimento de padrões em texto (in portuguese). Master's thesis, UNICAMP, Campinas, Brazil, 1993.
3. D. Beauquier, J. Berstel, and P. Chrétienne. *Éléments d'algorithmique.* Manuels Informatiques Masson. Masson, Paris, 1992.
4. A. C. Blumer, J. A. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen, and J. Seiferas. The smallest automaton recognizing the subwords of a text. *Theoretical Comput. Sci.*, 40:31–55, 1985.
5. J. A. Brzozowski. Canonical regular expressions and minimal state grahps for definite events. In *Proc. of the Symp. of Math. Theory of Automata*, pages 529–561, New York, 1962. Polytechnic Institute of Brooklyn.
6. B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual.* Springer-Verlag, New York, 1991.
7. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The M.I.T. Press, Cambridge, Mass., 1990.
8. J.-P. Duval. Contribution a la combinatoire du monoïde libre. Thèse, Faculté des Sciences de L'Université de Rouen, 1980.

9. S. Eilenberg. *Automata, Languages, and Machines, Volume A*. Academic Press, New York, 1974.
10. Z. Galil. String matching in real time. *J. ACM*, 28:134–149, 1981.
11. L. J. Guibas and A. M. Odlyzko. Periods in strings. *J. Comb. Th. A*, 30:19–42, 1981.
12. C. Hancart. *Analyse Exacte et en Moyenne d'Algorithmes de Recherche d'un Motif dans un Texte*. PhD thesis, Université Paris 7, Paris, 1993.
13. C. Hancart. On Simon's string searching algorithm. *Inf. Process. Lett.*, 47:95–99, 1993.
14. D. E. Knuth, J. H. Morris, Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6:323–350, 1977.
15. J. H. Morris, Jr. and V. R. Pratt. A linear pattern–matching algorithm. Research report 40, University of California, Berkeley, 1970.
16. M. Perles, M. O. Rabin, and E. Shamir. The theory of definite automata. *IEEE Trans. Electronic Computers*, EC-12:233–243, 1963.
17. M. Regnier. Enumeration of bordered words, le langage de la Vache-Que-Rit. *R.A.I.R.O. Informatique Théorique*, 26:303–317, 1992.
18. R. Sedgewick. *Algorithms*. Addison-Wesley, Reading, MA, 1983.

This article was processed using the LaTeX macro package with LLNCS style