Information Processing Letters 30 (1989) 209–214
North-Holland

27 February 1989

SEPARATING STRINGS WITH SMALL AUTOMATA

J.M. ROBSON

Computer Science Department, Australian National University, G.P.O. Box 4, Canberra, A.C.T. 2601, Australia

Communicated by P.C. Poole
Received 15 July 1988

Given any two distinct strings, there is a finite automaton which accepts one but not the other and has a number of states much less than the length of either string. In the case of two strings of the same length, this considerably strengthens the best previously known result.

Keywords: Finite automaton, separating strings, rational function

1. Introduction

In this paper we consider the question of how large a finite automaton may be necessary to separate two strings u and v each of length less than or equal to n, that is to accept one but not the other. This question arose in connection with work of H. Johnson on data compression [3] and has previously been studied by Goralcik and Koubek [1]. They established that $\Theta(\log n)$ states are necessary and sufficient in the case of strings over a unary alphabet or of strings of different lengths. They also showed that the result is independent of the alphabet size as soon as that size exceeds one.

Thus we can limit our attention to two strings u and v of the same length n over the alphabet $\{0, 1\}$, though for technical reasons connected with the definition of a "completion" of a string (Section 3), we actually consider the two strings U and V over $\{0, 1, \#\}$ formed by appending the symbol $\#$ to u and v respectively. We use a notation such as $u\#$ for this operation of appending a symbol to a string. We denote the elements of U (V) by u_i (v_i), $i = 1, \ldots, n$, $u_n = \#$ $(= v_n)$ and denote substrings as follows: $u_i..u_j$. Clearly U and V are exactly as easy or hard to separate as u and v.

The main result to be proved is that U and V can be separated by a machine with $O(n^{2/5}\log^{3/5}n)$ states. This result will be proved in Section 3. In Section 2 we give some definitions and preparatory results, including a simple proof that U and V can be separated by a machine with $O((n \log n)^{1/2})$ states, a proof which will be modified to produce the main proof.

2. A simple $O((n \log n)^{1/2})$ upper bound

A concept which is of central importance is that of a "periodic" string. We first define this concept and then prove three simple lemmas.

Definition. A string S $(s_0..s_{l-1})$ has a period p if $s_i = s_{i+p}$ for all i such that $0 \leqslant i \leqslant l-p$.

Definition. A string is "periodic" if it has a period not greater than half its length.

In the sequel whenever "the" period of a string is mentioned, the least period is meant.

Lemma 1. *If* $S0$ *(remember that this is* S *concatenated with the symbol* 0) *is periodic, then* $S1$ *is not.*

Proof. Assume the contrary and let p_i be the period of Si and l the length of S. Now it is easy to show that $s_{(l+ap_1) \bmod p_0} = 1$ for any integer a and similarly that $s_{(l+bp_0) \bmod p_1} = 0$ for any integer b. Now choosing a and b so that $ap_1 + bp_0 = \gcd(p_0, p_1)$ yields a contradiction. □

Lemma 2. *If S has period p and U contains two occurrences of S starting at u_i and u_j then $|i-j| \geqslant p$.*

Proof. Immediate. □

Lemma 3. *For every $\alpha < 1$, if $u_i..u_{i+1-l}$ is nonperiodic and $l \leqslant n^\alpha$, there exists j such that*

$$u_k..u_{k+1-l} \neq u_i..u_{i+1-l}$$
for every k such that $k \neq i$ but $k \equiv i$ (modulo j); (1)

moreover there exists a c depending only on α such that j may be chosen less than cn log *n/l.*

Proof. By Lemma 2 the number of occurrences of $u_i..u_{i+1-l}$ in U is bounded above by $2n/l$. Each such occurrence can cause (1) to fail for less than $(1-\alpha)^{-1}$ prime numbers $j > n/l$ since $n/l \geqslant n^{1-\alpha}$ and $|k-i|$ cannot have $(1-\alpha)^{-1}$ prime factors greater than $n^{1-\alpha}$. Thus there must be a j satisfying (1) amongst the first $2n/l(1-\alpha)$ primes greater than n/l. The conclusion follows by the prime number theorem (e.g. [2]). □

We are now nearly ready to prove the simple upper bound but first we need another definition.

Definition. A machine M "finds" a string S on input S it enters an accepting state Q for the first time upon reading the last element of S.

One significance of this definition is that the action of M on S is independent of its transitions from Q so that we can "compose" machines by identifying the Q of one machine with the start state of another and the composite machine will find the concatenation of the strings found by the individual machines.

All the machines presented in this paper will separate strings by the following technique. First find some prefix $u_0..u_i$ of U with a machine M which does not accept $v_0..v_i$. If M never enters the state Q on V then make M's transition on Q be to always stay in Q and M already separates U and V. Otherwise if M first enters Q at v_j, compose with M a machine which separates $u_{i+1}..u_n$ and $v_{j+1}..v_n$; since these strings have different lengths we know that they can be separated by a machine with O(log n) states.

Now comes the upper bound:

Theorem 1. *U and V can be separated by a machine with $O((n \log n)^{1/2})$ states.*

Proof. Suppose the first difference between U and V is $u_i \neq v_i$. If $i < (n \log n)^{1/2}$ then trivially a

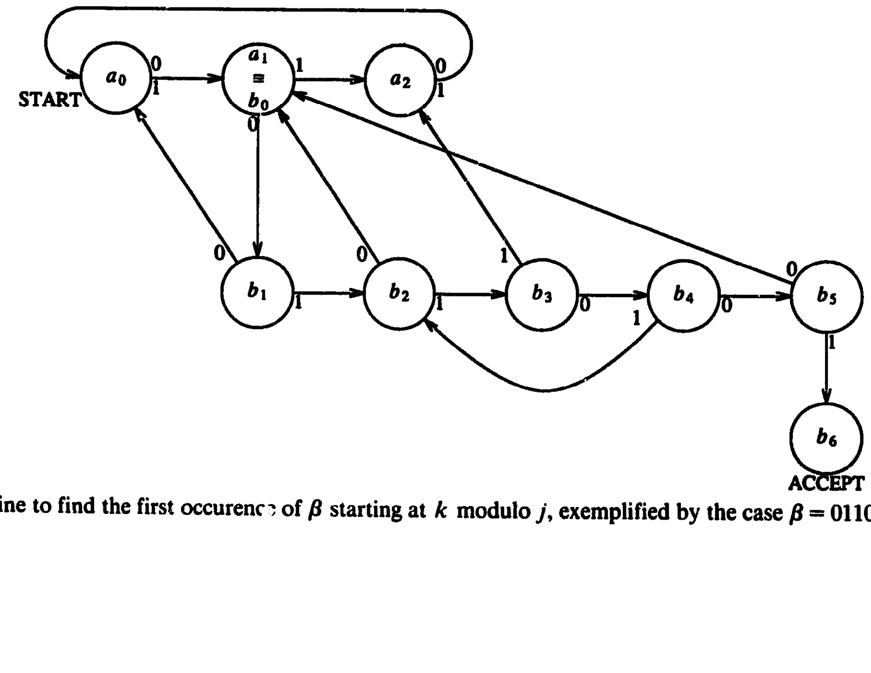

Fig. 1. A machine to find the first occurence of β starting at k modulo j, exemplified by the case $\beta = 011001$, $k = 1$, $j = 3$.

machine with $(n \log n)^{1/2}$ states can find $u_0..u_i$ and not accept $v_0..v_i$. Otherwise consider the two substrings $u_{i-(n \log n)^{1/2}}..u_i$ and $v_{i-(n \log n)^{1/2}}..v_i$; by Lemma 1 at least one of these is not periodic; suppose it is in U. Choose j for this substring according to Lemma 3. Now a machine which locates the first occurrence of this substring starting at the appropriate position modulo j will find $u_0..u_i$ but not accept $v_0..v_i$. One such machine to do this is illustrated in Fig. 1 and has $j + (n \log n)^{1/2}$ states. This machine deals with the general case of finding the first occurence of a string β to occur starting at position k modulo j using $j + |\beta|$ states. It consists of two parts A and B where A simply counts symbols modulo j except that its "count $\equiv k$" state is identified with b_0 the start state of B which checks for an occurrence of β. When B finds a string γ which it recognises as a nonprefix of β the correct successor state is:

$$\begin{cases} a_{(|\gamma|+i) \bmod j} & \text{if } |\gamma| < j, \text{ or} \\ \text{the } \gamma' \text{ successor of } b_0 & \text{otherwise,} \end{cases}$$

where γ' is γ with its first j symbols removed. □

Corollary 1. *If $u_j \neq v_j$ then U and V can be separated in $O((j \log j)^{1/2} + \log n)$ states.*

Proof. Again let the first difference between U and V be at position $i \leqslant j$. The machine constructed as above to separate $u_0..u_i$ and $v_0..v_i$ will have $O((i \log i)^{1/2})$ states and will find $u_0..u_i$. □

3. An improved upper bound

The lowering of the upper bound of Theorem 1 stems from the observation that the submachine B of Fig. 1 accomplishes more than is required of it: it separates the substring β from every other string except those of which β is a prefix whereas it only needs to separate it from those few strings which actually occur as substrings of U starting at the appropriate position modulo j. However we cannot hope to find $u_0..u_i$ using a machine with fewer states than the nonperiodic substring ending at i as is clear from considering the nonperiodic string 01^k. These remarks motivate the following definitions.

Definition. If a substring $s_i..s_j$ of S has a suffix with period p and $s_{j-2p+1}..s_{k+1}$ is the shortest substring of S starting at s_{j-2p+1} and not having period p, then $s_i..s_{k+1}$ is a "completion" of $s_i..s_j$ (with respect to p); a substring is also a "completion" of itself.

Note that any substring of U or V which has a periodic suffix has a completion with respect to the period of that suffix because of the # concatenated to u and v.

Definition. A machine M "identifies" the prefix $s_0..s_{i-1}$ of the string S $(s_0..s_l)$ with respect to the set of strings $\{t_j \mid 1 \leqslant j \leqslant m\}$ if there exists a set W of strings $\{w_j \mid 1 \leqslant j \leqslant n\}$ and a set Q of distinct states $\{Q_j \mid 1 \leqslant j \leqslant n\}$ of M such that

(i) input w_j causes M to enter Q_j,

(ii) no proper prefix of any w_j causes M to enter any state in Q,

(iii) each t_j has some element of W as a prefix,

(iv) one w_j has the form $s_0..s_k$ for $k \geqslant i-1$.

In other words, if given input S, M reads at least the prefix before entering one of the states Q and knowing which Q_j was first entered tells us exactly what symbols were read up to that point, on the assumption that the input provided was either S or some t_j. In Fig. 1 submachine B could be replaced by any machine which identifies the prefix $u_{i-(n \log n)^{1/2}}..u_i$ of $u_{i-(n \log n)^{1/2}}..u_n$ with respect to the other prefixes of U starting at congruent positions modulo j and the prefix of V $v_{i-(n \log n)^{1/2}}..v_n$. The states Q_j would be identified with states of A or M as appropriate to the associated string w_j.

We will prove an upper bound on the number of states needed to identify a string terms of the length of the string and the size m of the set of "other" strings. First we need a lemma on machines which skip over periodic strings.

Lemma 4. *If S consists of k repetitions of a nonperiodic string s of length l, then a machine of $< 2k + l$ states can find the last element of S.*

Proof. The machine simply needs to locate the first occurrence of s at the position $(k-1)l$ modulo j for some j greater than or equal to k and having no factors in common with l. There must be at least one such j in the range $[k, k+l-1]$. We call this machine $SKIP(s, k)$. □

Theorem 2. *For any positive α the prefix $s_0..s_{i-1}$ of the string S $(s_0..s_{l-1}\#)$ can be identified with respect to the set $\{t_j\# \mid 1 \leqslant j \leqslant m\}$ by a machine of $O((im \log n)^{1/2} + m^2 \log n + n^{\alpha})$ states where n is the maximum of the lengths of S and the t_j.*

Proof. We divide S into "parts" and associate with each part (regarded as a prefix of the remainder of S) a machine which identifies it with respect to all those $t_j\#$ whose first difference from S lies within that part. Then the required machine is obtained by composing the machines for the parts. We define the parts iteratively.

To obtain the "part" starting at s_a (either $a = 0$ or the previous part ended at s_{a-1}) suppose that s_b is the first point where any of the t_j differs from S (excluding those that have already differed from S before s_a). There are three cases to consider depending on the periodicity of the prefix P of $s_a..s_{b-1}$ defined as $P = s_{b-1-((b-a)\log n)^{1/2}}..s_{b-1}$.

Case A. P is not periodic. The "part" is $s_a..s_b$ and the machine to identify it is constructed as in Section 1 to find $s_a..s_{b-1}$ and then switch on the next symbol. The number of states is $O(((b-a)\log n)^{1/2})$.

Case B. P is periodic with period p and periodic pattern π. In this case we first find the smallest c such that $s_c..s_{b-1}$ has period p; unless $c = a$, $s_{c-1}..s_{c+((b-a)\log n)^{1/2}-2}$ is a nonperiodic substring of length $((b-a)\log n)^{1/2}$ and we locate that, again using $O(((b-a)\log n)^{1/2})$ states. The continuation depends on p.

Case B1. $p > (i/m)^{1/2}$. Again the "part" is $s_a..s_b$ and the continuation consists of $SKIP(\pi, k)$ for k the number of occurrences of π remaining $(k > (b-a)/p)$ followed by a straightforward machine to read the remaining $< p$ symbols. The total number of states

$$O\left(((b-a)\log n)^{1/2} + \frac{(b-a)}{(i/m)^{1/2}}\right).$$

Case B2. $p \leqslant (i/m)^{1/2}$. Here the "part" is the completion of $s_a..s_b$ with respect to p. Suppose that X of the t_j differ from S for the first time within this part and that the numbers of repetitions of π are τ_i $(0 \leqslant i \leqslant X)$ for S and these t_j. First we choose a number q such that no $\tau_i \equiv \tau_j$ (modulo q) unless $\tau_i = \tau_j$. We can choose such a q which is $O(X^2 \log n + n^{\alpha})$ since each $\tau_i - \tau_j$ must have fewer than α^{-1} prime factors greater than n^{α}. Now we construct a machine which counts occurrences of π modulo q with a check whenever the count is τ_i for any i to confirm that the next p symbols actually are π or otherwise to enter an appropriate state Q_j. The modular counting process is accomplished by $X+2$ $SKIP(\pi, (\tau_i - \tau_{i-1} - 1) \bmod q)$ machines (assuming that the τ_i are sorted modulo q, that $\tau_{-1} = 0$ and that $\tau_{X+1} = q + 1$). This machine is illustrated in Fig. 2. The number of states is

$$O\left(\sum_{i=0}^{X+1} (\tau_i - \tau_{i-1}) \bmod q + (X+2)p\right)$$
$$= O(q + Xp)$$
$$= O\left(X^2 \log n + n^{\alpha} + X(i/m)^{1/2}\right).$$

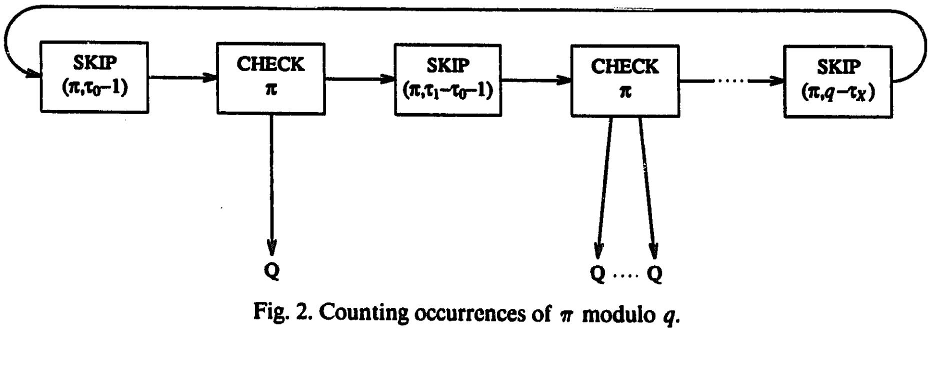

Fig. 2. Counting occurrences of π modulo q.

Now we sum the states over the machines for all the parts.

Cases A and B1. For each part the number of states is

$$O\left(((b-a)\log n)^{1/2}+\frac{(b-a)}{(i/m)^{1/2}}\right).$$

The number of parts is at most m and the sum over all of them of $(b-a)$ is at most i. The sum of $(b-a)^{1/2}$ will be maximised when there are exactly m equal values of $(b-a)$. Thus the total number of states from these two cases is

$$O\left(\log^{1/2}n\ m(i/m)^{1/2}+\frac{i}{(i/m)^{1/2}}\right)$$
$$=O\left((im\log n)^{1/2}\right).$$

Case B2. For each part the number of states is $O(X^2\log n+n^\alpha+X(i/m)^{1/2})$. X must be at least 1 and the sum of the X over all these parts is at most m. This gives a total number of states of:

$$O\left(m^2\log n+n^\alpha+m(i/m)^{1/2}\right)$$
$$=O\left(m^2\log n+n^\alpha+(im)^{1/2}\right).$$

Adding the two bounds gives the result. □

Corollary 2. *If the conditions of Theorem 2 are satisfied and also all except M of the strings $t_j\#$ differ from S within their first i' symbols, then the bound in Theorem 2 can be replaced by* $O((iM\log n)^{1/2}+M^2\log n+n^\alpha+i')$.

Proof. Use a machine which has i' states to check that its first i' input symbols are those of S and then behaves like the machine given by Theorem 2 for the remaining M strings (or more precisely for their suffixes after removing the first i' symbols.) □

Now we are ready to prove the main theorem.

Theorem 3. *Two distinct strings u and v of length n can be separated by a finite automaton with* $O(n^{2/5}\log^{3/5}n)$ *states.*

Proof. We intend to use Corollary 2 with i, i' and M respectively equal to $n^{3/5}\log^{2/5}n$, $n^{2/5}\log^{3/5}n$ and $(n/\log n)^{1/5}$. We could do this if we could find a substring S of U or V whose length was $n^{3/5}\log^{2/5}n$, which was not periodic and whose prefix of length $n^{2/5}\log^{3/5}n$ was also not periodic. Such a substring does not necessarily exist but if we relax the condition on the length and periodicity of S to allow any length in the range $[n^{3/5}\log^{2/5}n, 2n^{3/5}\log^{2/5}n]$ while still allowing any period greater than $\frac{1}{2}n^{3/5}\log^{2/5}n$ then we can always find an S except in one simple case where u and v are easily separated. We proceed as follows. First choose the nonperiodic substring S' of length $n^{3/5}\log^{2/5}n$ ending at the point i where U and V first differ and as in Theorem 1 assume that this is in U. If the prefix of S' of the required length is nonperiodic then S' is the desired S. Otherwise work back from the start of S' until the periodicity of the prefix is broken; if this never happens then U and V are easy to separate so assume that it does happen at u_k. If $i-k<2n^{3/5}\log^{2/5}n$ then $u_k..u_i$ is the desired S; otherwise $u_k..u_{k+n^{3/5}\log^{2/5}n}$ is (note that in the latter case the nonperiodicity of S is guaranteed by Lemma 1's mirror image). □

Next we show that this substring S or a completion of it can be found using the number of states asserted in the conclusion of the theorem. We choose an integer j obeying two conditions: firstly, as in the proof of Theorem 1, the chosen substring (in this case S) never occurs earlier in U at a position congruent modulo j to the real position; and secondly the prefix of S of length $n^{2/5}\log^{3/5}n$ occurs in such positions at most $(n/\log n)^{1/5}$ times. We claim that we can choose a j satisfying these two conditions such that $j=O(n^{2/5}\log^{3/5}n)$. The reason is the limit on the periodicity of the string and the prefix; the string can occur at most $2(n/\log n)^{2/5}$ times in U and the prefix at most $2(n/\log n)^{3/5}$ times; each occurrence of the string can rule out up to 2 primes j greater than $n^{1/3}$ whereas it takes $(n/\log n)^{1/5}$ occurrences of the prefix to rule out one prime and each occurrence can affect only 2 primes greater than $n^{1/3}$. Thus there is a suitable j in the first $4(n/\log n)^{2/5}$ primes greater than $n^{1/3}$.

Now we use the same idea as illustrated in Fig. 1. The machine A counts modulo j and whenever it finds a count congruent to the start of S it starts machine B but now B is the machine guaranteed by Corollary 2 to identify the suffix of U starting with S with respect to all the earlier suffixes of U or V starting at the congruent positions (including the suffix of V starting at the same position as S does in U). Each of the states Q of B associated with a string w (apart from the one actually reached when starting at S) is identified with a state of A or B as specified in the comments on Fig. 1.

Finally, having found the end of S or of some completion of S, there are three possible conclusions:

(i) if this is at the end of S' we are already finished because this machine finds a prefix of U but does not accept the prefix of V of the same length;

(ii) if it is after the end of S' then this must be because some periodic pattern at the end of S' continued in U but not in V and this machine would have entered one of its Q states at that point in V giving us a machine which finds a prefix of V but does not accept the corresponding prefix of U;

(iii) otherwise this machine finds a point within a periodic pattern of period $< n^{2/5}\log^{3/5}n$ which continues to within S' and is broken there; compose a machine to read until the first break in this periodic pattern reaching a position k within S' and then compose a machine to separate the suffixes $u_{k+1}..u_n$ and $v_{k+1}..v_n$ according to Corollary 1. In each case we obtain a machine which separates U and V in $O(n^{2/5}\log^{3/5}n)$ states.

4. Conclusion

The upper bound on the number of states required by a finite automaton separatng two n-element strings has been substantially reduced from $o(n)$ [1] to $O(n^{2/5}\log^{3/5}n)$ but there is still a large gap between this upper bound and the known lower bound of $\Omega(\log n)$. Ch. Choffrut conjectures (in a private communication) that the upper bound can be reduced to $O(n^{\epsilon})$ for any positive ϵ, a conjecture which this author regards as unlikely to be provable by the methods developed here.

References

[1] P. Goralcik and V. Koubek, On discerning words by automata, *13th Internat. Colloquium on Automata, Languages and Programming*, Lecture Notes Comput. Sci. **226** (Springer, Berlin, 1986) 116–122.

[2] M.N. Huxley, *The Distribution of Prime Numbers* (Oxford, 1972).

[3] J.H. Johnson, Rational equivalence relations, *13th International Colloquium on Automata, Languages and Programming*, Lecture Notes Comput. Sci. **226** (Springer, Berlin, 1986) 167–176.