Automatic Complexity of Strings.

Article · January 2001

Source: DBLP

CITATIONS
53

READS
269

2 authors:

Jeffrey Shallit
University of Waterloo
471 PUBLICATIONS **7,720** CITATIONS
SEE PROFILE

Ming-wei Wang
Microsoft
27 PUBLICATIONS **447** CITATIONS
SEE PROFILE

Automatic Complexity of Strings

Jeffrey Shallit* and Ming-wei Wang

Department of Computer Science
University of Waterloo
Waterloo, N2L 3G1 Canada
shallit@graceland.uwaterloo.ca
m2wang@math.uwaterloo.ca

Abstract. We define a new measure of complexity for finite strings, called *automatic complexity* and denoted $A(x)$. Although $A(x)$ is analogous to Kolmogorov-Chaitin complexity, it has the advantage of being computable. We give upper and lower bounds for $A(x)$, and estimate it for some specific strings.

1 Introduction

We are interested in a computable measure of complexity for finite strings x over a finite alphabet, typically $\{0, 1\}$. Any such measure should reflect, in some sense, how "complicated" the string x is.

Of course, any such discussion must start with Kolmogorov-Chaitin complexity [11] $C(x)$, which (roughly speaking) measures the complexity of a string x as the size of the shortest pair

$$(T, y) = (\text{Turing machine description, input})$$

such that T on input y outputs x. Not only does $C(x)$ measure the complexity of x, but also the pair (T, y) can be viewed as the optimal way to compress the string x.

However it has three major deficiencies (the first two are equivalent):

1. It is uncomputable! It is known that "$C(x) < n$" is computably enumerable, but "$C(x) \geq n$" is not computably enumerable.
2. There is no effective procedure for finding a compression pair (T, y).
3. K depends somewhat on the particular model of universal Turing machine chosen, and is defined in a machine-independent way only up to an additive constant.

One consequence of deficiency (3) above is that since $C(xx) = C(x) + O(1)$, with the constant depending on the particular model of universal Turing machine chosen, it doesn't make sense to ask if $C(xx) > C(x)$ for any, most, or all strings x. We will see below, however, that in the measure of complexity proposed in

* Supported in part by a grant from NSERC

this paper, we have $A(xx) \geq A(x)$ for all strings x, and in fact there are infinitely many strings x for which this inequality is strict; see Theorems 4 and 13. [1]

It would be nice to find a measure without these deficiencies. Turing machines are extremely powerful, and this suggests that we could replace the Turing machine with a less powerful model and hope to find a computable measure.

For example, we could consider replacing the Turing machine with a context-free grammar (CFG). We choose, perhaps arbitrarily, some measure of the complexity of a context-free grammar, and then ask for the smallest grammar G such that $L(G) = \{x\}$.

If we demand that the context-free grammar be in Chomsky normal form (i.e., all productions are of the form $A \to BC$ or $A \to a$ where A, B, C are variables and a is a terminal), and use the number of variables as the measure of a grammar's size, then then we get a well-known measure of complexity associated with "word chains". Diwan [8] was apparently the first to study this measure; for other papers see [6, 15, 2, 1, 7].

In this paper we consider replacing the Turing machine with a deterministic finite automaton, or DFA.

Given a string x, in analogy with the word chain problem mentioned above, we might seek to find a smallest DFA M such that $L(M) = \{x\}$. But this is clearly uninteresting, since if $|x| = n$, a smallest such DFA always has exactly $n + 2$ states. Hence we consider relaxing the requirement somewhat.

If a DFA M has the property that it accepts a string x, but no other strings of length $|x|$, we say M *accepts* x *uniquely*. In this paper, we examine the consequences of the following definition. We define $A(x)$, the *automatic complexity* of x, to be the smallest number of states in any DFA M that accepts x uniquely. Of course, there may be many such DFA's with the smallest number of states. We do not care how M behaves on strings that are shorter or longer than x. More formally,

Definition 1. *Let $\Sigma = \{0, 1\}$ and $x \in \Sigma^*$ with $|x| = n$. Define $A(x)$ to be the smallest number of states in any DFA M such that $L(M) \cap \Sigma^n = \{x\}$.*

An earlier paper of the first author and Y. Breitbart [16] explored a similar notion of descriptional complexity for languages. However, that measure turns out to be uninteresting for the case of a single string.

There is a connection between the measure studied in this paper and the so-called "separating words" problem, which, given two strings w and x, both

[1] The question whether $C(xx) > C(x)$ is relevant to questions raised by "scientific" creationists putting forth a theory of "intelligent design". It has been claimed, for example, that "*...there is no more information in two copies of Shakespeare's* Hamlet *than in a single copy. This is of course patently obvious, and any formal account of information had better agree.*" (William Dembski, *Intelligent Design: The Bridge Between Science and Theology*, Intervarsity Press, 1999, Chapter 6, p. 158. I am grateful to Wesley Elsberry for pointing this out.) But in fact all that the Kolmogorov theory can claim is that $C(xx) = C(x) + O(1)$, which is not the same as $C(xx) = C(x)$. By Dembski's reasoning we would also have $C(x^n) = C(x)$ for all n, and this is clearly untrue; in fact it is easy to see that $C(x^n) - C(x)$ is unbounded as $n \to \infty$.

of length $\leq n$, asks for the number $B(w,x)$ of states in the smallest DFA M such that M *separates* w from x, i.e., M accepts exactly one of $\{w,x\}$. It is known that $B(w,x) = O(\log n)$ if $|w| \neq |x|$, and $B(w,x) = O(n^{2/5}(\log n)^{3/5})$ if $|w| = |x|$; see, for example, [10,13,14]. Given w, the function $A(w)$ can be viewed as measuring the size of the smallest DFA M such that M separates w from $\Sigma^{|w|} - \{w\}$.

2 Basic results

Clearly $A(x) \leq |x| + 1$, since we can uniquely accept any string of length $|x|$ with a chain of $|x|$ states that loops back to the start state, plus one additional "dead" state. It follows that A is computable, since we can simply examine all finite automata with $|x| + 1$ or fewer states, and test each DFA by brute force to see if x is accepted uniquely. As we will see below, it is possible to improve this algorithm somewhat, but we still do not know if $A(x)$ is computable in time polynomial in $|x|$.

It is possible, however, that $A(x)$ is significantly smaller than $|x|+1$. Roughly speaking, there are two ways to save states. The first is to use a loop. For example, the DFA in Figure 1 shows that $A(0^9 1^8) \leq 8$. (Unspecified transitions go to a "dead state" which is not shown.)

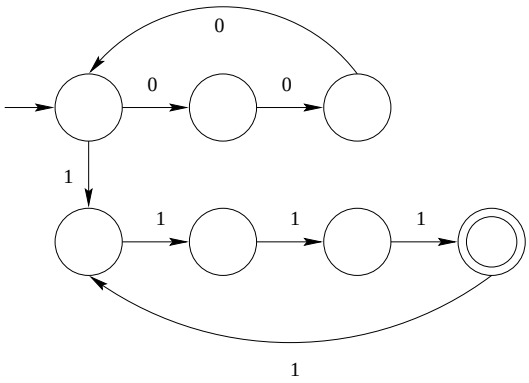

Fig. 1. Automaton uniquely accepting $0^9 1^8$.

The second way to save states is through reuse. For example, you can reuse states, if the string is of the form $x\,y\,z\,\overline{y}^R\,w$, as shown in Figure 2. (By $\overline{y}$ we mean the string obtained by changing 0 to 1 and vice versa.)

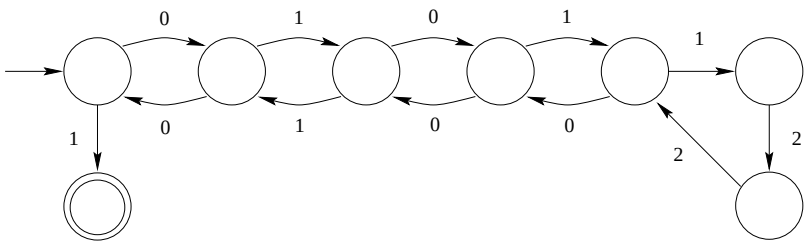

Fig. 2. Automaton uniquely accepting 010112200101

Hopefully the reader is already convinced that this definition is somewhat natural and worthy of study.[2] Let us first see if the definition is useful; for example, can we use the measure as a data compression technique?

The answer is yes, in the following sense.

Theorem 2. *Given a description of a DFA M which uniquely accepts x, and the length $n = |x|$, we can efficiently recover x.*

Proof. By "efficiently", we mean polynomial in the description size of M and n, the length of x.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA uniquely accepting x. Let $Q = \{q_1, q_2, \ldots, q_r\}$, with $r = |Q|$. Create a directed graph $G = (V, E)$ with vertex set V defined as follows:

$$V = \{p_{i,j} \ : \ 1 \leq i \leq r, \ 0 \leq j \leq n\}.$$

Place a directed edge $(p_{i,j}, p_{k,l})$ labeled a if $\delta(q_i, a) = q_k$ and $l = j + 1$. Note that G is acyclic.

Since M uniquely accepts x, there exists a single index t with $q_t \in F$ such that there is exactly one path from $p_{1,0}$ to $p_{t,n+1}$, and for all $u \neq t$, there is no path from $p_{1,0}$ to $p_{u,n+1}$. We can now find this path using, for example, depth-first search, in $O(|V| + |E|) = O((n+1)|Q||\Sigma|)$ time. ■

Given a DFA, we can also efficiently decide if it uniquely accepts a given x.

Theorem 3. *Given a DFA M with r states and a string x of length $n \geq 1$, we can determine in $O(n + r^3 \log n)$ steps whether M uniquely accepts x.*

Proof. Let $M = (Q, \Sigma, \delta, q_1, F)$, where $Q = \{q_1, q_2, \ldots, q_r\}$. We can determine if M accepts x by simply simulating it on x, which can be done in $O(n)$ time.

Now create a matrix $M = (a_{i,j})_{1 \leq i,j \leq n}$ where $a_{i,j} = \text{Card}\{b \in \Sigma \ : \ \delta(q_i, b) = q_j\}$. Then an easy induction gives that if $M^k = (c_{i,j,k})_{1 \leq i,j \leq n}$, then $c_{i,j,k} = \text{Card}\{x \in \Sigma^k \ : \ \delta(q_i, x) = q_j\}$. Now compute $\sum_{j : q_j \in F} c_{1,j,|x|}$. This sum is 1 iff M uniquely accepts x.

[2] But if not, there are some alternatives that also may be of interest. For example, we could define $B(x)$ to be the smallest number of states in any DFA M such that x is the lexicographically least string of length $|x|$ accepted by M.

Thus it suffices to compute M^k efficiently. To do so we can use the familiar "binary method" of exponentiation; see, for example, [3]. Furthermore, during the computation of M^k, we can always reduce an entry that is ≥ 2 to 2. The result is a matrix M' with entries in $\{0, 1, 2\}$ with the property that if an entry of M^k is 0 or 1, so is the corresponding entry of M', and if an entry of M^k is 2 or more, the corresponding entry in M' is 2. Since the sizes of the entries of M' are bounded by 2, it follows that this computation can be done in $O(r^3 \log n)$ bit operations. ■

Our last theorem of this section is the following:

Theorem 4. *We have $A(xx) \geq A(x)$ for all strings x.*

The following simple proof was shown to us at the DCAGRS 2000 workshop in London, Ontario, by Kai Salomaa:

Proof. Consider the DFA $M = (Q, \Sigma, \delta, q_0, F)$ minimizing $A(xx)$. Then we know there is only one path of length $|xx|$ from q_0 to a state of F, and this path is labeled xx. Let $q = \delta(q_0, x)$. Construct a new DFA $M' = (Q, \Sigma, \delta, q_0, \{q\})$. Then we claim M' uniquely accepts x. For if not, there exists another string $w \neq x$, $|w| = |x|$, such that $\delta(q_0, w) = q$. Then $\delta(q_0, wx) \in F$, and so M accepts wx, another string of length $|xx|$, and $wx \neq xx$. This contradiction proves that $A(x) \leq A(xx)$, as desired. ■

In Theorem 13 below we show that in fact $A(xx) > A(x)$ for infinitely many strings x.

3 Upper bounds

In this section we prove some upper bounds on $A(x)$.

Theorem 5. *Let $x \in \Sigma^*$ with $|\Sigma| = k \geq 2$ and $|x| = n$. Suppose $n > k^t + t - 1$. Then $A(x) \leq n + 2 - t$.*

Proof. If $n > k^t + t - 1$, then $x = a_1 a_2 \cdots a_n$ has at least $k^t + 1$ subwords of length t. Hence some subword of length t appears at least twice in x. Let y be a longest repeated subword, and let the first two occurrences of y be denoted y' and y'' (they may overlap).

Then we have the two factorizations shown in Figure 3.

$x =$

u	y'	v	w
u	v'	y''	w

Fig. 3. Two factorizations of x.

where $y = y' = y''$. Furthermore, since y is a longest repeated subword, we know that either $w = \epsilon$ or the first letter of v differs from the first letter of w.

By a classic theorem of Lyndon & Schützenberger [12], the equality $yv = v'y$ implies that there exist strings r, s and an integer $e \geq 0$ such that

$$y = (rs)^e r$$
$$v = sr$$
$$v' = rs.$$

Thus $x = u(rs)^{e+1}rw$. It follows that the first letter of s differs from the first letter of w, so we can accept x uniquely with a DFA as in Figure 4.

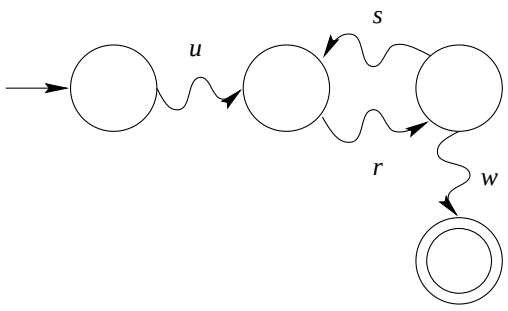

Fig. 4. DFA uniquely accepting $x = u(rs)^{e+1}rw$.

The total number of states is $|ursw| + 2 = n + 2 - |y|$. ■

Theorem 6. *Let $x \in \{0,1\}^n$. Then*

$$A(x) \leq \frac{3}{4}n + (\log n)\sqrt{\frac{n}{8}}$$

for almost all strings x.

Proof. (Sketch.) The idea is to write $x = x'ax''$ where $|x'| = |x''| = \lfloor \frac{n}{2} \rfloor$ and $a \in \{\epsilon, 0, 1\}$. Then the expected number of mismatches between x' and x''^R is $\frac{n}{4} + O(1)$, with standard deviation $\sqrt{\frac{n}{8}} + O(1)$. We can now build a DFA for x', and attempt to reuse states corresponding to the mismatches between x' and x''^R, as in Figure 2. ■

4 Lower bounds

First, we show by a simple counting argument the existence of a constant C such that almost all strings x of length n satisfy $A(x) > C\frac{n}{\log n}$.

More precisely, we prove

Theorem 7. *Suppose* $|\Sigma| = k \geq 2$, *and let* $0 < \varepsilon, \delta < 1$ *be fixed. If* n *is sufficiently large, then*

$$A(x) \geq (1-\delta)\varepsilon \frac{\log k}{k} \frac{n}{\log n}$$

for all strings $x \in \Sigma^n$, *with at most* $k^{\varepsilon n}$ *exceptions.*

Proof. It is easy to see there are at most q^{qk+1} essentially distinct automata with $\leq q$ states and exactly one final state. (The factor q^{qk} comes from the transition function, and the factor q comes from the assignment of final states. Note we can simulate a DFA with $\leq q$ states by one with exactly q states, by simply adding non-connected states, if necessary.) Each of these automata uniquely accepts at most one string of length n. Thus if

$$q^{qk+1} < k^{\varepsilon n}, \tag{1}$$

then at most $k^{\varepsilon n}$ different strings of length n can be represented. Now a routine calculation shows that if $q < (1-\delta)\varepsilon \frac{\log k}{k} \frac{n}{\log n}$, then the inequality (1) holds. ■

It is possible to improve this bound as follows:

Theorem 8. *We have* $A(x) \geq n/13$ *for almost all strings* $x \in \{0,1\}^n$.

Proof. Suppose M is a DFA with $A(x)$ states that uniquely accepts x. Let $n = |x|$. Consider the transition diagram D of M, which is a labeled directed graph whose vertices are the states of M and whose (labeled) edges correspond to transitions. We define the *accepting path* P for x to be the sequence of $n+1$ edges traversed in this graph. Note that the first element of P is an edge labeled ϵ that enters the initial state q_0 of M. We define the *abbreviated accepting path* P' to be the sequence of edges obtained from P by considering each edge in order and deleting it if it has previously been traversed. The idea is to encode P' in a space-efficient manner so that x can be recovered.

The outdegree of each vertex encountered along P' is ≤ 2, since M is a DFA. We claim the indegree of each vertex is ≤ 2. If not, then let v be a vertex with indegree ≥ 3. Then there are at least three distinct edges entering v, say g_1, g_2, g_3. Let x_1 be a prefix of x such that the edge g_1 is used in the last transition when the DFA processes x_1. (If $v = q_0$, the initial state, we may have $x_1 = \epsilon$.) Let x_1x_2 be a prefix of x such that g_2 is used in the last transition when processing x_1x_2, $x_2 \neq \epsilon$. Let $x_1x_2x_3$ be a prefix of x such that g_3 is used in the last transition when processing $x_1x_2x_3$, $x_3 \neq \epsilon$. Finally, let x_4 be such that $x = x_1x_2x_3x_4$. Then $x' := x_1x_3x_2x_4$ is also accepted by M, and $|x| = |x'|$. If $x = x'$, then $x_2x_3 = x_3x_2$. Then, by a theorem of Lyndon & Schützenberger [12], there exist a string $z \neq \epsilon$ and integers $i, j \geq 1$ such that $x_2 = z^i$, $x_3 = z^j$. Now if there is a path labeled z^i going from v to v, and a path labeled z^j from v to v, then there is a path labeled $z^{\gcd(i,j)}$ from v to v. But then $g_2 = g_3$, a contradiction. Hence $x \neq x'$, contradicting the hypothesis that x is accepted uniquely.

Now consider the vertices visited by $P' = (e_0, e_1, \ldots, e_t)$, the abbbreviated accepting path for x. Each vertex v is of exactly one of the following types:

Type 1. There is exactly one edge e_i of P' entering v and there is exactly one edge e_{i+1} leaving v.

Type 2. There are exactly two edges, e_i and e_j, $i < j$, entering v, and exactly one edge e_{i+1} leaving v.

Type 3. There is one edge, e_i, entering v, and exactly two edges, e_{i+1} and e_j, $i < j$, leaving v.

Type 4. There are exactly two edges, e_i and e_j, entering v, with $i < j$, and there are exactly two edges, e_{i+1} and e_{j+1}, leaving v.

We now describe a space-efficient encoding E of P' which will avoid recording the state numbers. Instead, we record the labels of the edges along with some additional information that tells us what type each vertex is, and allows us to recover how these vertices are connected.

If $P' = (e_0, e_1, \ldots, e_t)$, then we define $E(i, n)$ to be a certain encoding, over the alphabet $\{0, 1, [,]_0,]_1, *, +\}$ of the edges $(e_i, \ldots, e_n)$. We also define a_i to be the label of the edge e_i corresponding to the symbol causing the transition. The meaning of the symbols is as follows: 0 and 1 represent the labels on the edges of P'. A left bracket [represents a vertex that is the target of a backedge. A right bracket ($]_0$ or $]_1$) represents a backedge labeled with its subscript. The symbol + represents a vertex of outdegree 2, and the symbol $*$ (introduced later) represents a final state.

The base case is when $i > n$, in which case we define $E(i, n) = \epsilon$. For the inductive definition there are four cases, depending on the type of the vertex reached by the directed edge e_i, given in Figures 5–8.

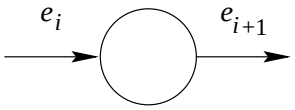

Fig. 5. Vertex of type 1: $E(i, n) := a_i \; E(i + 1, n)$

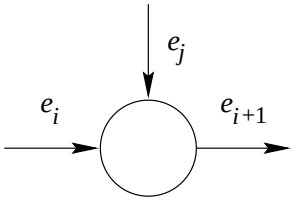

Fig. 6. Vertex of type 2: $E(i, n) = a_i \, [\, E(i + 1, j - 1) \,]_{a_j} \; E(j + 1, n)$

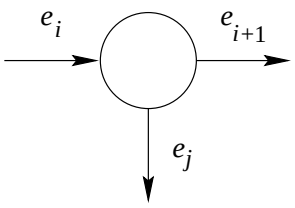

Fig. 7. Vertex of type 3: $E(i,n) = a_i + E(i+1,n)$

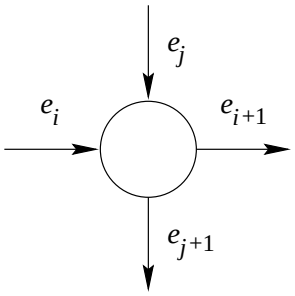

Fig. 8. Vertex of type 4: $E(i,n) = a_i\,[\,+E(i+1,j-1)\,]_{a_j}E(j+1,n)$

Finally, if $P' = (e_0, e_1, \ldots, e_t)$, we define $E(x)$ to be $E(0,t)$ with a symbol $*$ inserted after the symbol leading to the (unique) accepting state, followed by the symbol #, followed by the base-2 representation of $n = |x|$, followed by ##. Thus $E(x)$ is a self-delimiting encoding of x over the 8-symbol alphabet $\{0, 1, +, *, [,]_0,]_1, \#\}$. We consider some examples of this encoding.

Figure	String	Encoding
Figure 1	$0^9 1^8$	$[+00]_0 1[111*]_1 \#10001\#\#$
Figure 10	0110100110	$[+0[+1[+10]_1]_0]_0 110*\#1010\#\#$
Figure 11	01101001100101	$0[1*101[001+]_1]_0 \#1110\#\#$

We leave it to the reader to verify that P' can be reconstructed from $E(0,t)$ and x can be reconstructed from $E(x)$. It is easy to prove by induction that $|E(a,b)| \leq 2(b-a+1)$. Now P' has at most $2A(x)$ edges with nonempty labels, so we find $|E(0,t)| \leq 4A(x)+2$. It follows that $|E(x)| \leq 4A(x) + 6 + \log_2 |x|$. Since E is over an 8-letter alphabet, it can be recoded over $\{0,1\}$ using three bits for each symbol. It follows that $C(x) \leq 12A(x)+18+3\log_2 |x|$. On the other hand, it is known that $C(x) \geq |x| - \log_2 |x|$ for almost all x. Hence $A(x) \geq |x|/13$ for almost all x. ■

Remark. We have not tried to optimize the constant 13 in Theorem 8. H. Petersen informs us (personal communication) that 13 can be reduced to 7.

We can improve the lower bound for certain kinds of strings, as follows:

Theorem 9. *Suppose $w \in \Sigma^*$ is kth-power-free for some integer $k \geq 2$, i.e., w contains no subword of the form x^k with $x \neq \epsilon$. Then $A(w) \geq \frac{|w|+1}{k}$.*

Proof. Let $w = a_1 a_2 \cdots a_n$ be uniquely accepted by some DFA $M = (Q, \Sigma, \delta, q_0, A)$, and define $p_i := \delta(q_0, a_1 a_2 \cdots a_i)$ for $0 \leq i \leq n$.

Suppose some state is visited at least $k+1$ times on the acceptance path for w. Then there exist indices $i_1, i_2, \ldots, i_{k+1}$ such that

$$p_{i_1} = p_{i_2} = \cdots = p_{i_{k+1}}.$$

Define

$$\begin{aligned} w_0 &= a_1 a_2 \cdots a_{i_1} \\ w_1 &= a_{i_1+1} \cdots a_{i_2} \\ w_2 &= a_{i_2+1} \cdots a_{i_3} \\ &\vdots \\ w_k &= a_{i_k+1} \cdots a_{i_{k+1}} \\ w_{k+1} &= a_{i_{k+1}} \cdots a_n. \end{aligned}$$

Then M uniquely accepts $w = w_0 w_1 w_2 w_3 \cdots w_{k+1}$. However, it also accepts, for example, $w' = w_0 w_2 w_1 w_3 \cdots w_{k+1}$. But $|w'| = |w|$. If $w_1 \neq w_2$, this gives a contradiction. Hence $w_1 = w_2$. By a similar argument we find $w_i = w_j$ for $1 \leq i, j \leq k$. It follows that $w = w_0 w_1^k w_{k+1}$, and so w contains a k'th power, a contradiction.

Thus we have shown that no state can be visited $k+1$ times on the acceptance path for w. Now for $0 \leq i < |Q|$ let b_i be the number of times state q_i is visited on the acceptance path for w. Then we have

$$\sum_{0 \leq i < |Q|} b_i q_i = n + 1.$$

But by the argument above $0 \leq b_i \leq k$. Thus

$$n + 1 = \sum_{0 \leq i < |Q|} b_i q_i \leq \sum_{0 \leq i < |Q|} k q_i = k|Q|.$$

It follows that $|Q| \geq (n+1)/k$, and so $A(w) = |Q| \geq (n+1)/k$, as desired. ∎

Remark. If $|\Sigma| \geq 2$ then there are infinitely many cube-free strings. For example, if $\mathbf{t} = 01101001\cdots$ denotes the infinite Thue-Morse word, then every prefix is cube-free. If $|\Sigma| \geq 3$ then there are infinitely many square-free strings [5].

5 Some specific examples

In this section we determine the automatic complexity for some particular examples. There are interesting connections to number theory.

Theorem 10. *We have* $A(0^n 1^n) = O(\sqrt{n})$.

Proof. Assume $n \geq 1$. Let $r = \lfloor \sqrt{n} \rfloor$, so $r^2 \leq n < (r+1)^2$. Write $n = r^2 + a$. Then $0 \leq a \leq 2r$ and $r \geq 1$. Then we can accept $0^n 1^n$ with a DFA of the form given in Figure 9. (Unspecified transitions go to a "dead state" which is not shown.)

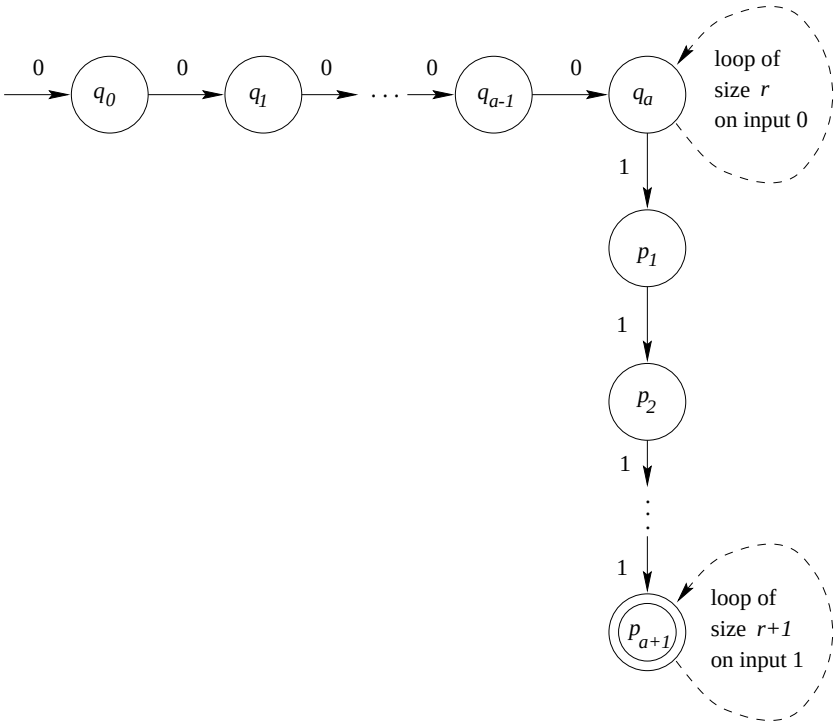

Fig. 9. Automaton uniquely accepting $0^n 1^n$, where $n = r^2 + a$.

This DFA does indeed accept $0^n 1^n$ because

1. We go from state q_0 to state q_a on 0^a;
2. We then go around the loop at q_a r times;
3. Next on 1^a we go from q_a to p_{a+1};
4. Finally, we go around the loop at p_{a+1} $r-1$ times.

This path accepts $0^a(0^r)^r 11^a(1^{r+1})^{r-1} = 0^{r^2+a}1^{r^2+a}$.

On the other hand, we claim that this DFA accepts no other string of length $2n$. Suppose it did. Then any accepting path must go around the loop on q_a b times and the loop on p_{a+1} c times. Then

$$2n = a + br + a + 1 + c(r+1).$$

Since $n = r^2 + a$, it follows that $2r^2 - 1 = br + c(r+1)$. Reducing modulo r, we get $c \equiv -1 \pmod{r}$. Thus $c \in \{r-1, 2r-1, \ldots\}$. But if $c \geq 2r-1$ then the string would be of length $\geq (2r-1)(r+1)+2a+1 = 2r^2+r-1+2a+1 \geq 2n+r > 2n$, a contradiction.

Finally, our DFA uses $a+1+r-1+a+1+r = 2r+2a+1 \leq 6r+1 \leq 6\sqrt{n}+1$ states. ■

We now show that the bound of $O(\sqrt{n})$ is tight. First we state the following lemma:

Lemma 11. *Let c, d be integers ≥ 1. Suppose the linear diophantine equation $N = xc+yd$ is solvable in integers, i.e., suppose* $\gcd(c,d) \mid N$. *If $N > 2cd-c-d$, then the linear diophantine equation $N = xc + yd$ has at least two solutions in non-negative integers x, y.*

The proof is easy and left to the reader. This result (in a more general form) has recently been proved independently by Beck & Robins [4].

We now prove

Theorem 12. *Any DFA that uniquely accepts 0^n1^n must have at least $\sqrt{n} - 1$ states.*

Proof. Suppose M is a DFA with $< \sqrt{n} - 1$ states that uniquely accepts 0^n1^n. Define $p_i = \delta(q_0, 0^i)$ for $0 \leq i \leq n$. Since M has $< n$ states, some state must be repeated, and thus there must be a "loop" of $r \geq 1$ states that is repeated j times, for some integer $j \geq 0$. There may also be a "tail" at the beginning, and at the end we may not go around the "loop" an integral number of times. Let $s = n - rj$. Then $r, s < \sqrt{n} - 1$.

Similarly, define $r_i = \delta(p_n, 1^i)$ for $0 \leq i \leq n$. By the same argument there must be a "loop" of $u \geq 1$ states that is repeated k times, for some integer $k \geq 0$. Let $t = n - ku$. Then $t, u < \sqrt{n} - 1$.

Since M accepts 0^n1^n uniquely, it must be the case that the equation $ra + ub = 2n - s - t$ has exactly one solution $(a, b) = (j, k)$. Then, by Lemma 11, we have $2n - s - t \leq 2ru - r - u$. Thus $2n - 2(\sqrt{n} - 1) \leq 2n - s - t \leq 2ru - r - u \leq 2(\sqrt{n} - 1)(\sqrt{n} - 1) - 2$. But then $2\sqrt{n} + 2 \leq 0$, a contradiction. ■

We can now exhibit infinitely many strings for which $A(xx) > A(x)$.

Theorem 13. *Let $x = 0^n1$. Then $A(xx) = \Omega(\sqrt{n})$, but $A(x) = O(1)$.*

Proof. It is clear that $A(x) = O(1)$, since we can accept 0^n1 uniquely with a 3-state DFA. However, mimicking the lower bound proof of Theorem 12 above, it is easy to see that $A(0^n10^n1) = \Omega(\sqrt{n})$. ■

It is possible to generalize Theorem 10. We need some technical lemmas. The first concerns solvability of certain linear diophantine equations.

Lemma 14. *Let $k \geq 1$, and let $n_1, n_2, \ldots, n_k$ be positive integers, relatively prime in pairs. Let $r \geq 0$ be an integer. Define $P := n_1 n_2 \cdots n_k$. If $r = 0$, the linear diophantine equation*

$$a_1 \frac{P}{n_1} + a_2 \frac{P}{n_2} + \cdots + a_k \frac{P}{n_k} = (n_1 - 1)\frac{P}{n_1} + (n_2 - 1)\frac{P}{n_2} + \cdots + (n_k - 1)\frac{P}{n_k} - rP \tag{2}$$

has a unique solution

$$(a_1, a_2, \ldots, a_k) = (n_1 - 1, n_2 - 1, \ldots, n_k - 1)$$

in non-negative integers. If $r \geq 1$, then (2) has no solutions in non-negative integers.

Proof. By induction on k. If $k = 1$ Eq. (2) becomes $a_1 = n_1 - 1 - rn_1$. If $r = 0$ this equation has the unique solution $a_1 = n_1 - 1$, but if $r \geq 1$ then clearly there are no solutions in non-negative integers.

Now assume the result is true for $1, 2, \ldots, k-1$. We prove it for k. Consider Eq. (2) mod n_k. We get

$$a_k \frac{P}{n_k} \equiv -\frac{P}{n_k} \pmod{n_k}.$$

Since the n_i are pairwise relatively prime, it follows that $a_k \equiv -1 \pmod{n_k}$. Since a_k is a non-negative integer, we can therefore write $a_k = jn_k - 1$ for some integer $j \geq 1$.

Now substitute $a_k = jn_k - 1$ in Eq. (2). After a little easy algebra, we get

$$a_1 \frac{P}{n_1} + a_2 \frac{P}{n_2} + \cdots + a_{k-1} \frac{P}{n_{k-1}} = (n_1 - 1)\frac{P}{n_1} + (n_2 - 1)\frac{P}{n_2} + \cdots + (n_{k-1} - 1)\frac{P}{n_{k-1}} - (j + r - 1)P \tag{3}$$

By induction Eq. (3) has a solution iff $j + r - 1 = 0$. But $j \geq 1$. Hence $j = 1$ and $a_k = n_k - 1$, and hence Eq. (3) has a solution iff $r = 0$. If $r = 0$, by induction the solution is $(a_1, a_2, \ldots, a_{k-1}) = (n_1 - 1, n_2 - 1, \ldots, n_{k-1} - 1)$.

Lemma 15. *(a) If $M^{1/k} > 2B$, then*

$$\frac{M}{M^{1/k} - B} < M^{\frac{k-1}{k}} + 2BM^{\frac{k-2}{k}}.$$

(b) If $0 < B < A$ and $k \geq 1$, then

$$(A - B)^k \geq A^k - kA^{k-1}B.$$

Proof. (a) We have

$$(M^{1/k} - B)(M^{\frac{k-1}{k}} + 2BM^{\frac{k-2}{k}}) = M + BM^{\frac{k-2}{k}}(M^{1/k} - 2B) > M.$$

(b) An easy induction on k proves that if $0 < x < 1$ and $k \geq 1$ then $(1-x)^k \geq 1 - kx$. Now let $x = B/A$ and multiply by A^k.

For our last lemma, we will need a certain number-theoretic function. For $t \geq 1$, define $f(t)$ to be the least integer n such that every set of n consecutive positive integers contains a subset of size t that is pairwise relatively prime. Then, for example, $f(4) = 6$, since the set $\{2, 3, 4, 5, 6\}$ contains no subset of 4 relatively prime integers, while it is easy to check that every set of 6 consecutive positive integers does.

It seems quite difficult to estimate f precisely. However, the following lemma follows easily from results of Erdős and Selfridge [9]:

Lemma 16. *For all $\delta > 0$ and t sufficiently large we have $f(t) < t^{2+\delta}$.*

Proof. Erdős and Selfridge defined $F(n,k)$ to be the largest subset of pairwise relatively prime integers in $\{n+1, n+2, \ldots, n+k\}$, and proved that $\min_{n \geq 0} F(n,k) > k^{1/2-\epsilon}$. Now let $k = t^{2+5\epsilon}$ for some $\epsilon < 1/10$. We find

$$\min_{n \geq 0} F(n, t^{2+5\epsilon}) > (t^{2+5\epsilon})^{1/2-\epsilon} > t^{1+\epsilon/2-5\epsilon^2} > t,$$

since $\epsilon < 1/10$. Hence for all $0 < \epsilon < 1/10$ and all t sufficiently large, any $t^{2+5\epsilon}$ consecutive integers contains a pairwise relatively prime subset of cardinality $> t$. In other words, $f(t) < t^{2+\delta}$ where $\delta = 5\epsilon$. ■

We are now ready to prove

Theorem 17. *Let $a_1, a_2, \ldots, a_k$ be k distinct symbols. Then $A(a_1^n a_2^n \cdots a_k^n) = O(n^{1-1/k})$, where the constant in the big-O may depend on k.*

Proof. The idea is as follows: we choose k pairwise relatively prime integers, each $\leq n^{1/k}$, say $n_1, n_2, \ldots, n_k$. Let $P = n_1 n_2 \cdots n_k$. We then form a DFA similar to that in Figure 9, with k loops, one on each a_i, $1 \leq i \leq k$, of size P/n_i. Each loop is preceded by a "tail" of length $n - (P/n_i)(n_i - 1) = n - P + P/n_i$. By Lemma 14, this DFA uniquely accepts $a_1^n a_2^n \cdots a_k^n$.

The total number of states is $\leq N$, where

$$N := 1 + k(n-P) + 2 \sum_{1 \leq i \leq k} \frac{P}{n_i}. \tag{4}$$

By Lemma 5 we can choose the pairwise relatively prime numbers n_i such that $n^{1/k} - k^{2+\delta} < n_i < n^{1/k}$. Setting $A = n^{1/k}$ and $B = k^{2+\delta}$ in Lemma 15 (b) we obtain

$$P = n_1 n_2 \cdots n_k \geq n - k^{3+\delta} n^{\frac{k-1}{k}}.$$

Hence

$$k(n-P) < k^{4+\delta} n^{\frac{k-1}{k}}. \tag{5}$$

On the other hand, setting $M = P$ and $B = k^{2+\delta}$ in Lemma 15 (a) we obtain

$$\frac{P}{n_i} < P^{\frac{k-1}{k}} + 2k^{2+\delta} P^{\frac{k-2}{k}} \tag{6}$$

for all n sufficiently large. Combining Eqs. (4)–(6), we obtain $N = O(k^{4+\delta} n^{\frac{k-1}{k}})$, as desired. ■

6 Infinite words

Up to now we have been dealing with finite words. However, it is also interesting to consider the case of infinite words. In this paper, by an infinite word we will mean a one-sided, right-infinite word, i.e., a map from $\mathbb{N}$ to Σ. For an infinite word $\mathbf{x}$ we are interested in computing

$$I(\mathbf{x}) = \liminf_{x \text{ is a prefix of } \mathbf{x}} \frac{A(x)}{|x|}$$

and

$$S(\mathbf{x}) = \limsup_{x \text{ is a prefix of } \mathbf{x}} \frac{A(x)}{|x|}.$$

for "interesting" infinite words $\mathbf{x}$.

We start with the Thue-Morse word $\mathbf{t}$. Let μ be a morphism defined by $\mu(0) = 01$, $\mu(1) = 10$. Then $\mathbf{t} = t_0 t_1 t_2 \cdots = \lim_{n\to\infty} \mu^n(0)$. We define $T(r) = t_0 t_1 \cdots t_{r-1}$, the prefix of $\mathbf{t}$ of length r.

Theorem 18. *We have*

$$I(\mathbf{t}) \geq \frac{1}{3}$$

and

$$S(\mathbf{t}) \leq \frac{2}{3}.$$

Proof. The lower bound for $I(\mathbf{t})$ follows immediately from Theorem 9, since, as is well-known, the Thue-Morse word is cube-free.

For the upper bound, we break the argument up as follows. We claim that we can accept $T(m)$ using $h(m)$ states, where h is given in the table below.

m	$h(m)$
$2 \cdot 2^{2n} \leq m \leq 3 \cdot 2^{2n}$	$m + 3 - 2^{2n}$
$3 \cdot 2^{2n} < m < 4 \cdot 2^{2n}$	$2 \cdot 2^{2n} + 2$
$4 \cdot 2^{2n} \leq m < 5 \cdot 2^{2n}$	$m + 2 - 2 \cdot 2^{2n}$
$5 \cdot 2^{2n} \leq m \leq 6 \cdot 2^{2n}$	$m + 1 - 2 \cdot 2^{2n}$
$6 \cdot 2^{2n} < m < 8 \cdot 2^{2n}$	$4 \cdot 2^{2n} + 2$

For $2 \cdot 2^{2n} \leq m \leq 3 \cdot 2^{2n}$, we use the fact that $T(2 \cdot 2^{2n}) = T(2^{2n})\overline{T(2^{2n})^R}$, which allows us to reuse $2^{2n} - 1$ states, as illustrated in Figure 10.

For $3 \cdot 2^{2n} \leq m < 4 \cdot 2^{2n}$, we use the fact that $T(4 \cdot 2^{2n}) = T(2^{2n})(\overline{T(2^{2n})})^2 T(2^{2n})$, which allows us to reuse 2^{2n} states in an inner loop and $m - 3 \cdot 2^{2n}$ states in an outer loop, as illustrated in Figure 11.

For $4 \cdot 2^{2n} \leq m < 5 \cdot 2^{2n}$, it is easiest to give the encoding of the corresponding machine, as introduced in Section 4:

$$t_0[t_1 \cdots t_{m-3\cdot 2^{2n}-1} * t_{m-3\cdot 2^{2n}}[t_{m-3\cdot 2^{2n}+1} \cdots t_{2^{2n+1}-1} + t_{2^{2n+1}} \cdots t_{m-2^{2n+1}-1}]_{t_{m-2^{2n+1}}}]_{t_{3\cdot 2^{2n}}}$$

This is illustrated in Figure 12.

For $5 \cdot 2^{2n} \leq m \leq 6 \cdot 2^{2n}$, we use the fact that $T(5 \cdot 2^{2n}) = (T(2^{2n})\overline{T(2^{2n})T(2^{2n})})^{5/3}$, as illustrated in Figure 13.

For $6 \cdot 2^{2n} \leq m \leq 8 \cdot 2^{2n}$, we use the fact that $T(m) = T(2^{2n+3} - m)x\overline{x}^R$, where $x = t_{2^{2n+3}-m} \cdots t_{2^{2n+2}}$, as illustrated in Figure 14.

In the figures that follow, unspecified transitions go to a "dead state" which is not shown. ■

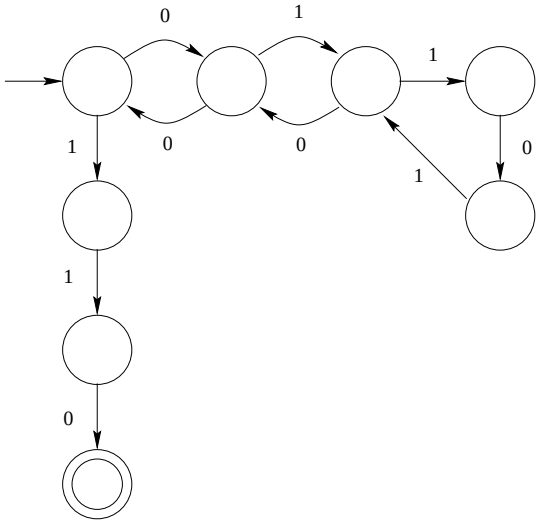

Fig. 10. Automaton uniquely accepting t_{10}

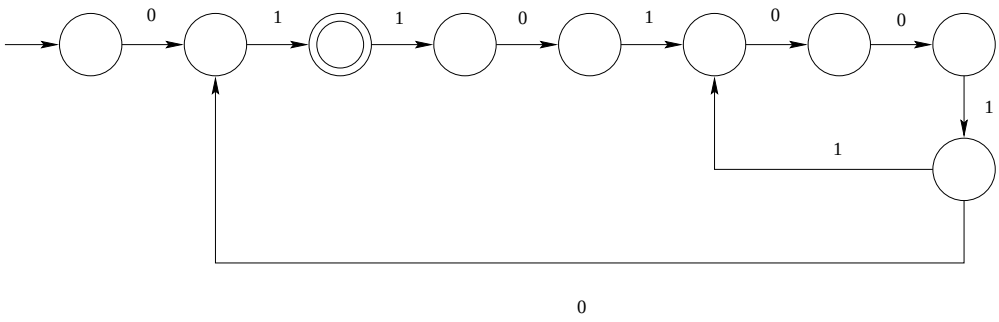

Fig. 11. Automaton uniquely accepting t_{14}

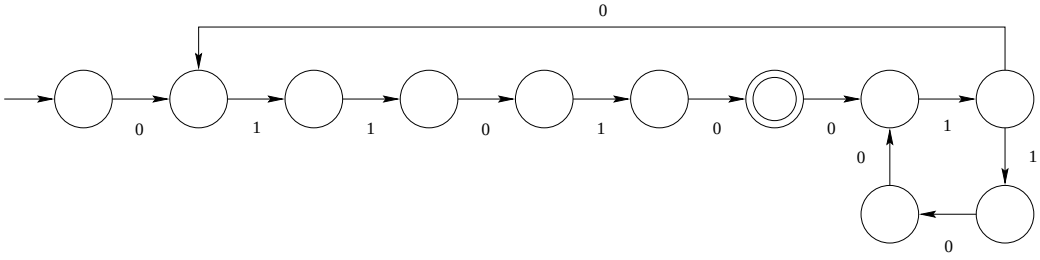

Fig. 12. Automaton uniquely accepting t_{18}

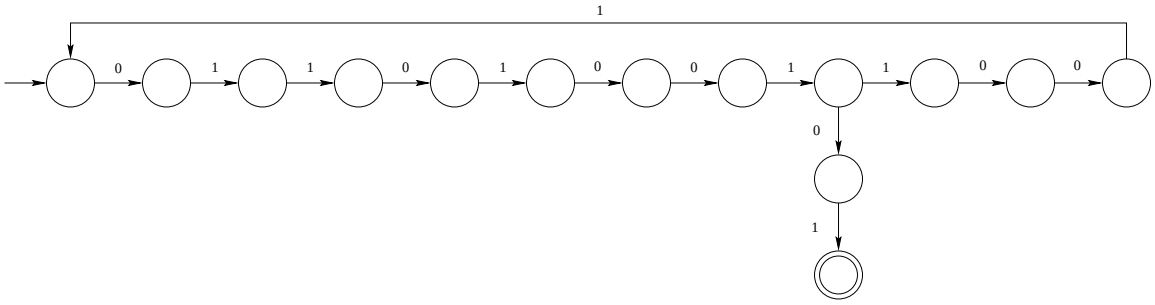

Fig. 13. Automaton uniquely accepting t_{22}

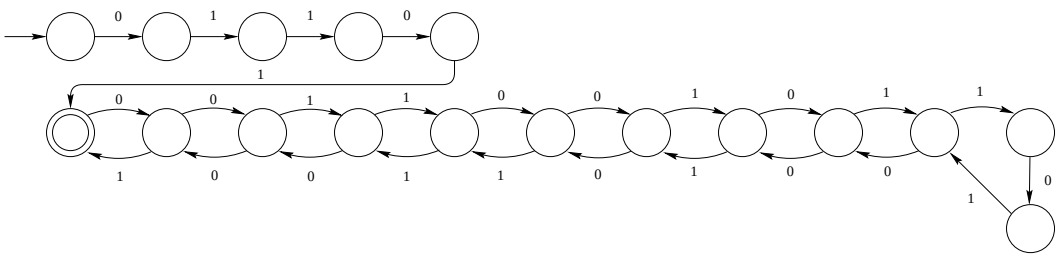

Fig. 14. Automaton uniquely accepting t_{26}

If we consider the Thue-Morse word on three symbols [5], we can get a sharper result.

Theorem 19. *Let* $\mathbf{u} = 102120102012\cdots$ *be the infinite Thue-Morse word on three symbols, generated by* $1 \to 102$, $0 \to 12$, $2 \to 0$. *Then*

$$I(\mathbf{u}) = \frac{1}{2}.$$

Proof. The lower bound comes from Theorem 9. For the upper bound, we claim that if we let rs be a prefix of $\mathbf{u}$ with $|r| = |s| = 2^{2k}$ for some $k \geq 0$, then r differs in every single position from s^R. Hence we may reuse states in analogy with Figure 2.

To prove the claim, let $\mathbf{u} = u_0u_1u_2\cdots$. Let $r = u_0u_1\cdots u_{2^{2k}-1}$ and $s = u_{2^{2k}}\cdots u_{2^{2k+1}-1}$. It is known that $u_i = (2t_i + t_{i+1}) \bmod 3$, where $\mathbf{t} = t_0t_1t_2\cdots$ is the Thue-Morse word. For $0 \leq i < 2^{2k+1}$ we have $t_i = 1 - t_{2^{2k+1}-i-1}$. Now $\mathbf{t}$ is cube-free, so $t_{i-1}t_it_{i+1} \notin \{000, 111\}$. Hence $t_i + t_{i+1} \neq 2t_{i-1}$, and $t_j \in \{0, 1\}$ for $j \geq 0$, so $t_i + t_{i+1} \not\equiv 2t_{i-1} \pmod 3$. Adding t_i to this last incongruence, we have

$$\begin{aligned} u_i &\equiv 2t_i + t_{i+1} \not\equiv t_i + 2t_{i-1} \\ &\equiv 2(1 - t_i) + (1 - t_{i-1}) \\ &\equiv 2t_{2^{2k+1}-i-1} + t_{2^{2k+1}-i} \\ &\equiv u_{2^{2k+1}-i-1} \pmod 3. \end{aligned}$$

This proves that r differs in every position from s^R. ■

One might wonder if $S(\mathbf{x}) = 0$ implies that $\mathbf{x}$ is ultimately periodic. The answer is no, as the following example shows:

Theorem 20. *Let* $\mathbf{v} = 0^2\,1\,0^4\,1\,0^{16}\,1\,0^{256}\,10^{65536}\,1\cdots$. *Then* $S(\mathbf{v}) = 0$.

Proof. First, we need the following lemma.

Lemma 21. *Let $y \in \{0,1\}^*$. Then for all $r, s \geq 1$ we have $A(y10^r10^s) \leq |y| + 6\sqrt{m}$, where $m = \max(r, s)$.*

Proof. Suppose $r \geq s$. (The proof for $r < s$ is similar and is left to the reader.) Define $t := \lfloor \sqrt{r} \rfloor$ and $u = \min(\lfloor s/t \rfloor, t)$. Now construct the following DFA:

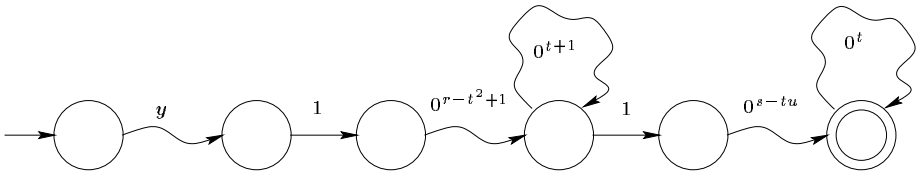

Fig. 15. DFA uniquely accepting $y10^r10^s$

We claim this DFA uniquely accepts $y10^r10^s$. It clearly accepts this string, by going around the first loop $t-1$ times and the second loop u times. To see that acceptance is unique, consider going around the first loop $k \geq 0$ times and the second loop $j \geq 0$ times. This gives a string of length $|y|+1+r-t^2+1+k(t+1)+1+s-tu+jt$. Setting this equal to the desired length of $|y|+1+r+1+s$, we get the linear diophantine equation

$$r - t^2 + 1 + k(t+1) + s - tu + jt = r + s;$$

in other words, $k(t+1) + jt = t^2 - 1 + tu$. Now consider this equation modulo t. We find $k \equiv -1 \pmod{t}$. Suppose $k \geq 2t-1$. Then $t^2 - 1 + tu = k(t+1) + jt \geq (2t-1)(t+1) + jt$. Simplifying, we obtain $u \geq t + 1 + j$. But $j \geq 0$, so $u \geq t+1$, contradicting the definition of u. It follows that $k = t-1$, and hence $j = u$, as desired.

Our DFA has $N := |y| + 1 + r - t^2 + 1 + t + 1 + s - tu + t - 1 + 1$ states. Since $\sqrt{r} - 1 \leq t \leq \sqrt{r}$, it follows that $r - t^2 \leq 2\sqrt{r} - 1$ and

$$\begin{aligned} s - tu &\leq s - t\min(\lfloor s/t \rfloor, t) = \max(s - t\lfloor s/t \rfloor, s - t^2) = \max(s \bmod t, s - t^2) \\ &\leq \max(t, r - t^2) \leq \max(\sqrt{r}, 2\sqrt{r} - 1) \leq 2\sqrt{r} - 1. \end{aligned}$$

Hence $N < |y| + 6\sqrt{r}$. ■

Now we can complete the proof of Theorem 20. Every sufficiently long prefix of $\mathbf{v}$ is of the form

$$x = 0^2\,1\,0^{2^2}\,1\,0^{2^{2^2}}\,1\,0^{2^{2^3}}\,1 \cdots 1\,0^{2^{2^n}}\,1\,0^a$$

where $0 \leq a \leq 2^{2^{n+1}}$. Let $y = 0^2\,1\,0^{2^2}\,1\,0^{2^{2^2}}\,1\,0^{2^{2^3}}\,1 \cdots 1\,0^{2^{2^{n-1}}}$, $r = 2^{2^n}$, and $s = a$. Then $|x| = 2^{2^n} + a + O(2^{2^{n-1}})$, while Lemma 21 states that $A(x) \leq 6\sqrt{2^{2^n} + a} + O(2^{2^{n-1}})$. It follows that $A(x)/|x| = O(1/\sqrt{x})$, and so $S(\mathbf{v}) = 0$. ■

7 Open Problems

There are many open problems related to this work. For example, is $A(x)$ computable in polynomial time?

8 Acknowledgments

We thank Holger Petersen, Kai Salomaa, and particularly the anonymous referees for their helpful criticism and suggestions.

References

1. I. Althöfer. Tight lower bounds for the length of word chains. *Inform. Process. Lett.* **34** (1990), 275–276.
2. A. Arnold and S. Brlek. Optimal word chains for the Thue-Morse word. *Inform. Comput.* **83** (1989), 140–151.
3. E. Bach and J. Shallit. *Algorithmic Number Theory.* The MIT Press, 1996.
4. M. Beck and S. Robins. A formula related to the Frobenius problem in two dimensions. Unpublished manuscript, January, 2000.
5. J. Berstel. Sur la construction de mots sans carré. *Séminaire de Théorie des Nombres* (1978–1979), 18.01–18.15.
6. J. Berstel and S. Brlek. On the length of word chains. *Inform. Process. Lett.* **26** (1987/88), 23–28.
7. M. Bousquet-Mélou. The number of minimal word chains computing the Thue-Morse word. *Inform. Process. Lett.* **44** (1992), 57–64.
8. A. A. Diwan. A new combinatorial complexity measure for languages. Technical report, Computer Science Group, Tata Institute, Bombay, 1986.
9. P. Erdős and J. L. Selfridge. Complete prime subsets of consecutive integers. In R. S. D. Thomas and H. C. Williams, editors, *Proceedings of the Manitoba Conference on Numerical Mathematics*, pp. 1–14. 1971.
10. P. Goralčík and V. Koubek. On discerning words by automata. In L. Kott, editor, *Proc. 13th Int'l Conf. on Automata, Languages, and Programming (ICALP)*, Vol. 226 of *Lecture Notes in Computer Science*, pp. 116–122. Springer-Verlag, 1986.
11. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 1997.
12. R. C. Lyndon and M. P. Schützenberger. The equation $a^M = b^N c^P$ in a free group. *Michigan Math. J.* **9** (1962), 289–298.
13. J. M. Robson. Separating strings with small automata. *Inform. Process. Lett.* **30** (1989), 209–214.
14. J. M. Robson. Separating words with machines and groups. *RAIRO Inform. Théor. App.* **30** (1996), 81–86.
15. P. Roth. A note on word chains and regular languages. *Inform. Process. Lett.* **30** (1989), 15–18.
16. J. Shallit and Y. Breitbart. Automaticity I: Properties of a measure of descriptional complexity. *J. Comput. System Sci.* **53** (1996), 10–25.