Information Processing Letters 110 (2010) 396–398

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Two-way deterministic automata with two reversals are exponentially more succinct than with one reversal ☆

Marcin Balcerzak *, Damian Niwiński

Faculty of Mathematics, Informatics, and Mechanics, Warsaw University, Poland

ARTICLE INFO

Article history:
Received 22 June 2009
Received in revised form 6 March 2010
Accepted 8 March 2010
Available online 10 March 2010
Communicated by M. Yamashita

Keywords:
Formal languages
Regular languages
Two-way finite automata
Bounded number of reversals

ABSTRACT

We prove that limiting the number of reversals from two to one can cause an exponential blow-up in the size of two-way deterministic automaton.

1. Introduction

In the seminal paper introducing the concept of finite automaton [1], Rabin and Scott showed the equivalence between the two-way and one-way models; this equivalence was independently proved by Shepherdson [2]. It is well known, however, that transforming a two-way automaton into a one-way one can cause an exponential blow-up in the number of states, and this holds also in deterministic case. Possibly the simplest example is given by a family of languages $M_k = (a+b)^* a (a+b)^{k-1}$, for $k \in \mathbb{N}$. Indeed, a minimal deterministic (one-way) automaton recognizing M_k has 2^k states, but it is straightforward to come up with an equivalent two-way automaton with $\mathcal{O}(k)$ states. Moreover, this obvious automaton performs only one reversal in its computation on each input word. In this note, we extend this observation by showing that decreasing the number of reversals from 2 to 1 can also cause an exponential blow-up in the number of states. To this end, we exhibit a family of languages L_k (over a fixed alphabet $\{a, b, c\}$), such that any deterministic two-way automaton recognizing L_k and limited to one reversal must have at least 2^k states, but there is an equivalent 2-reversal automaton with $\mathcal{O}(k)$ states. This family has the additional property that L_k can be recognized by a 1-reversal *nondeterministic* automaton with the number of states polynomial in k. This can be viewed as a small step towards showing a super-polynomial gap between deterministic and nondeterministic two-way automata, which is a celebrated open problem (see [3,4]).

One of the first steps in this direction was done by Sipser [5], who showed that a deterministic automaton that can reverse direction only at the ends of a word (called sweeping) may require exponentially more states than a one-way nondeterministic one. This result was extended by Hromkovič and Schnitger [4] for two-way automata with the number of reversals bounded by a constant (called oblivious). These authors also showed an exponential gap between determinism and nondeterminism for automata with the degree of non-obliviousness bounded by $o(n)$, where n is the length of the input word. For more general models, a notable result was obtained by Hromkovič [6], who exhibited a language recognized by a two-way deterministic two-head finite automaton in linear time, which cannot be recognized by any two-way

☆ Work supported by Polish government grant No. N206 008 32/0810.
* Corresponding author.
E-mail addresses: marcin.t.balcerzak@gmail.com (M. Balcerzak), niwinski@mimuw.edu.pl (D. Niwiński).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2010.03.008

nondeterministic sensing multihead finite automaton with number of reversals bounded by n^a, where n is the length of the input word, and $a \in \mathbb{R}$, $0 < a < \frac{1}{3}$.

2. Limiting number of reversals from two to one

We assume that the reader is familiar with the concepts of one-way and two-way (deterministic) finite automata (see, e.g., [7]). We adopt a slightly more liberal definition of acceptance than [7]: an automaton accepts an input word whenever it enters an accepting state and at the same time falls off the right end *or* the left end of the word. We abbreviate two-way (resp. one-way) deterministic finite automaton by 2DFA (resp. 1DFA). *Reversal* is any change of the moving direction of the tape head. We are interested in (deterministic) automata which, on any input word, perform only a bounded number of reversals, say no more than k. We then refer to k-reversal 2DFA.

Before giving the construction, we present a family of languages that at first glance "require" two reversals, but in fact can be done with one; this illustrates the (relative) difficulty of the problem.

Example 2.1. With an alphabet $\{a, b, c\}$, let $L_k = \{((a+b)^*c)^{k-1}(a+b)^*a(a+b)^{k-1}c((a+b)^*c)^*\}$. Words in this language consist of blocks of type of $(a+b)^*c$, and the k-th block (from the beginning) has the property that its $(k+1)$-th symbol from the end is a. It is easy to recognize L_k with a 2-reversal 2DFA with $\mathcal{O}(k)$ states that goes just behind the k-th block, returns to check if this special block has an a on the distinguished position, and then continues forward to check the rest of the word. (Note that the word must end with c.) However, there exists a 1-reversal 2DFA with $\mathcal{O}(k)$ states recognizing the same language. After passing by the k-th block, the automaton continues moving to the right, but it starts to count the number of consecutive blocks modulo k. When reaching the end of the word, the automaton knows the residue of the total number of blocks modulo k. Then it turns back and moves left, still counting the blocks (modulo k). Note that the automaton can now identify each $\alpha \cdot k$-th block from the beginning, for $\alpha \geqslant 1$. In each such block (which is "suspected" of being the k-th block), the automaton deterministically checks whether an a symbol is where it has to be. If so, the automaton continues going left and counting blocks, and it enters an accepting state after passing by the $k-1$ blocks. If that is the left end of the word, the automaton ends and accepts. Otherwise, the next block (to the left) is again a suspected block. So the automaton continues the procedure.

Now we state the main result.

Theorem 2.2. *There exists a family of languages $\{L_k \mid k \in \mathbb{N}\}$, such that any 2-reversal 2DFA recognizing L_k has $\mathcal{O}(k)$ states, but any equivalent 1-reversal 2DFA has $\Omega(2^k)$ states.*

Proof. For an alphabet $\Sigma = \{a, b, c\}$ we select the language

$$L'_k = \big((a+b)^k(a+b)^*c\big)^k\big((a+b)^k(a+b)^*c\big)^* \times c\,(a+b)^k(a+b)^*c$$

Words in this language consist of at least $k+1$ blocks of type of $(a+b)^k(a+b)^*c$, and the last block is preceded by an additional c. Let us call a block *good* if it has the form $(a+b)^*\,a\,(a+b)^{k-1}c$, i.e., its $(k+1)$-th symbol from the end is a. Let L_k be a subset of L'_k consisting of those words in which the number of good blocks, besides the last one, is in interval $[0, k]$ (call this number q), and the q-th symbol in the last block (from the beginning) is a. (The last block may be good or not.)

For example, the following word is in L_4

abaabbc baabbbbc abaabbc bbbaaaac abbabbc c bbabbac

Indeed, it contains three good blocks (not counting the last one), and the 3rd symbol in the last block is a.

The language L_k is recognized by a 2-reversal 2DFA with $\mathcal{O}(k)$ states that goes to the end of the word, returns determining q, provided that $q \in [0, k]$, and then moves forward once again to check the letter on q-th position in the last block.

We will show that an equivalent 1-reversal 2DFA needs at least 2^k states.

Suppose to the contrary that A is a 2DFA with $< 2^k$ states that accepts L_k making only one reversal. We will find two words, $u \in L_k$ and $v \notin L_k$, such that A cannot distinguish between them.

Clearly, for each accepted word, the automaton must first go to the right end of the word (otherwise it would not make sure that the last letter is c). Note first that, by assumption, there exist two different words $x, y \in (a+b)^k$ such that, starting from the initial state and going to the right, the automaton enters the same state after reading x and y. We can then extend them by the same word $z \in (a+b)^*$, $|z| < k$, such that exactly one of the blocks xzc, yzc is good, but the automaton enters the same state, say q', after reading both of them. Making the same reasoning for the state q', we can find two blocks $x'z'c$ and $y'z'c$ say, such that exactly one of them is good, but the automaton does not distinguish between them when starting from the state q'. In consequence, when starting from the initial state, the automaton does not distinguish between 4 words

$$xzc\,x'z'c \qquad yzc\,x'z'c$$
$$xzc\,y'z'c \qquad yzc\,y'z'c$$

Continuing this reasoning, we can, for each pattern $\alpha \in \{0,1\}^k$, construct a word consisting of k consecutive blocks

$$x_\alpha = B_1 \ldots B_k$$

such that the distribution of good blocks among the blocks $B_1, \ldots, B_k$ corresponds to the pattern α (i.e., B_i is good iff $\alpha_i = 1$, where α_i is the i-th letter of the pattern α), but, after reading each x_α, the automaton enters the same state, say q_c.

Now consider all possible extensions of the words x_α by the last block of the form cwc, where $w \in (a+b)^k$. If the automaton A starts a computation in the state q_c at the left end of a word cwc, it must go to the right end, and then return, leaving this word at the left end in some state depending on w, say $p(w)$. However, by assumption

on A, there are more words in $(a+b)^k$ than possible states $p(w)$. So, we can find two words $w, w' \in (a+b)^k$, such that $p(w) = p(w')$, but $w_i = a$ and $w'_i = b$, for some i. Now take any word x_α which has exactly i good blocks. Then $x_\alpha cwc \in L_k$, while $x_\alpha cw'c \notin L_k$ – this follows directly from the definition of L_k. However, the automaton A does not distinguish between these two words. Indeed, after reading the prefix x_α it arrives into the state q_c. Then it reaches the right end of the word, and returns. After leaving the last block (preceded by c), it enters the same state $p(w) = p(w')$. It now scans backward the prefix x_α, and leaves both words at the left end in the same state. This remark completes the proof. □

Remark 2.3. It is straightforward to construct a *nondeterministic* automaton with $\mathcal{O}(k^2)$ states recognizing the language L_k and making only one reversal. (The automaton guesses the number q of good blocks, verifies that the q-th letter of the last block is a, and then, moving backward, verifies that the guess has been correct.) So the example of Theorem 2.2 also shows that, for 1-reversal automata, determinization causes a super-polynomial blow up.

We conjecture that Theorem 2.2 generalizes to any finite number of reversals, that is, an exponential blow-up of the number of states may occur each time when the number of reversals decreases by one.

References

[1] Michael O. Rabin, Dana Scott, Finite automata and their decision problems, IBM Journal 3 (1959) 114–135.
[2] John C. Shepherdson, The reduction of two-way automata to one-way automata, IBM Journal 3 (1959) 198–200.
[3] William J. Sakoda, Michael Sipser, Nondeterminism and the size of two-way finite automata, in: 10th ACM STOC, 1978, pp. 275–286.
[4] Juraj Hromkovič, Georg Schnitger, Nondeterminism versus determinism for two-way finite automata: Generalization of Sipser's separation, in: ICALP, in: Lecture Notes in Computer Science, vol. 2719, Springer, 2003, pp. 439–451.
[5] Michael Sipser, Lower bounds on the size of sweeping automata, in: STOC '79: Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, ACM, 1979, pp. 360–364.
[6] Juraj Hromkovič, Fooling a two-way nondeterministic multihead automaton with reversal number restriction, Acta Informatica 22 (1985) 589–594.
[7] John E. Hopcroft, Jeffrey D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison–Wesley, Reading, MA, 1979.