Nondeterminism versus Determinism for Two-Way Finite Automata: Generalizations of Sipser's Separation*

Juraj Hromkovič[1] and Georg Schnitger[2]

[1] Lehrstuhl für Informatik I, Aachen University RWTH, Ahornstraße 55, 52074 Aachen, Germany. Fax: ++49-241-8888216. jh@I1.Informatik.RWTH-Aachen.de
[2] Fachbereich Informatik, Johann-Wolfgang-Goethe Universität, Robert-Mayer-Straße 11-15, 60054 Frankfurt am Main, Germany

Abstract. Whether there exists an exponential gap between the size of a minimal deterministic two-way automaton and the size of a minimal nondeterministic two-way automaton for a specific regular language is a long standing open problem and surely one of the most challenging problems in automata theory. Twenty four years ago, Sipser [M.Sipser: Lower bounds on the size of sweeping automata. ACM STOC '79, 360-364] showed an exponential gap between nondeterminism and determinism for the so-called sweeping automata which are automata whose head can reverse direction only at the endmarkers. Sweeping automata can be viewed as a special case of oblivious two-way automata with a number of reversals bounded by a constant.
Our first result extends the result of Sipser to general oblivious two-way automata with an unbounded number of reversals. Using this extension we show our second result, namely an exponential gap between determinism and nondeterminism for two-way automata with the degree of non-obliviousness bounded by $o(n)$ for inputs of length n. The degree of non-obliviousness of a two-way automaton is the number of distinct orders in which the tape cells are visited.

Keywords: Finite automata, nondeterminism, descriptional complexity of regular languages

1 Introduction

Finite automata are the simplest uniform computing model and hence a base for the study of fundamental questions concerning computation and complexity. One of the central topics of theoretical computer science and especially of complexity theory is devoted to the comparison of nondeterministic computation and of deterministic computation. But not only the famous P $\stackrel{?}{=}$ NP problem seems to be hard, surprisingly, one is even not able to capture the computational power of nondeterminism for fundamental models of finite automata. To contribute to

* Supported by DFG grants HR 1416-1 and SCHN 50312-1.

J.C.M. Baeten et al. (Eds.): ICALP 2003, LNCS 2719, pp. 439–451, 2003.

the study of the relative power of nondeterminism and determinism in finite automata is the main goal of this paper.

The "classical" one-way deterministic finite automaton (1dfa) was independently introduced in [6,8,11] and the one-way nondeterministic finite state automaton (1nfa) was proposed by Rabin and Scott [13], who proved that for any 1nfa there is an equivalent 1dfa by the well-know subset construction. Let, for every regular language, $s(L)$ be the size of the minimal 1dfa that accepts L, and let $ns(L)$ be the size of a minimal 1nfa that accepts L. The subset construction [13] assures $s(L) \leq 2^{ns(L)}$ for every regular language L. Already more than 30 years ago Meyer and Fisher [9] and Moore [12] found regular languages with an exponential gap between $s(L)$ and $ns(L)$.

The most natural generalization of a 1dfa [1nfa] is the two-way deterministic [nondeterministic] finite automaton – 2dfa [2nfa]. Two-way automata recognize only regular languages and their size may be considerably smaller than the size of one-way automata [12]. The following question is natural. Let, for every regular language L, $s_2(L)$ denote the size of a minimal 2dfa accepting L, and let $ns_2(L)$ denote the size of a minimal 2nfa that accepts L.

Does there exists a polynomial p, such that

$$s_2(L) \leq p(ns_2(L))$$

for every regular language L?

Unfortunately, this question is still open and so it became one of the fundamental, most challenging open problems on the border between automata theory and complexity theory. The importance of this problem is underlined by its relation to the famous open question, whether deterministic logarithmic space (DLOG) is a proper subset of nondeterministic logarithmic space (NLOG). Berman [1] and Sipser [16] showed that if one proves an exponential gap between nondeterminism and determinism for two-way automata and the words involved in the proof are polynomial in length, then DLOG $\neq$ NLOG.

The first (at least partially successful) attempt to attack this problem was done by Sakoda and Sipser [14], who proved an exponential gap between nondeterminism and determinism for special automata which are allowed to read the input several times from the left to the right. Sipser [16] generalized this result to the so-called sweeping automata which are two-way finite automata whose head may reverse (change the direction of its movement) only at the endmarkers. More precisely, Sipser found a sequence $\{B_n\}_{n=1}^{\infty}$ of regular languages with

$$ns(B_n) = O(n) \text{ and } s_2(B_n) \geq 2^n$$

for every $n \in \mathcal{N}$. Recently Leung [7] proved a maximal possible exponential gap between nondeterminism and determinism in the sweeping automata model for a sequence of regular languages over[1] $\{0, 1\}$.

[1] Note, that the size of the alphabets of the Sipser languages B_n grows with n.

The above mentioned results do not solve the problem for general two-way automata because Micali [10] showed that deterministic sweeping automata may require a number of states that is exponential in $s_2(L)$ for some specific regular languages L.

Our hypothesis is that there is an exponential gap between $ns(B_n)$ and $s_2(B_n)$, where B_n are the languages of Sipser [16]. We are not able to prove it here, but the main goal of this paper is to prove an exponential gap between nondeterminism and determinism for more powerful versions of two-way finite automata than sweeping automata.

First we observe, that the number of reversals of any deterministic sweeping automaton is bounded by its number of states and so it is a constant with respect to the input length. Hence, one possibility to extend Sipser's result is to solve the problem for two-way finite automata with a constant number of reversals.

Another possibility is to say that a sweeping automaton is a special restricted version of oblivious two-way automata. Obliviousness for a two-way automaton means that, for every input length n, the order of tape cells visited by the reading head of the automaton is the same for all inputs of length n. Our experience with proving lower bounds on complexity of specific problems says that the real hardness of proving lower bounds starts with non-obliviousness. There are many examples of computing models where one can prove good lower bounds for the oblivious versions of these models by transparent arguments, but for the non-oblivious versions the proofs are very technical or even no known technique for proving lower bounds works. The considered problem of proving exponential lower bounds on $s_2(L)$ is exactly of this kind. Since non-obliviousness seems to be the core of the hardness of proving lower bounds for several fundamental computing models, Ďuriš et. al. [2] propose to start to measure the degree of non-obliviousness and to investigate the tradeoff between complexity and the degree of non-obliviousness.

Here, we introduce the degree of non-obliviousness of a 2dfa A as a function $f_A : \mathcal{N} \to \mathcal{N}$, where $f_A(n)$ is the number of different orders of the indexes of the tape cells appearing in computations of A on inputs of length n. The main result of this paper says that

> *There is an exponential gap between 1nfa's and 2dfa's with the degree of non-obliviousness bounded by $o(n)$.*

This paper is organized as follows. In Section 2 an exponential gap between nondeterminism and determinism for oblivious two-way automata[2] is established. This is done by proving an exponential lower bound for B_n. Section 3 shows how to prove the main result of this paper. We explain there the proof idea by showing the exponential gap between nondeterminism and determinism for the degree 2 of non-obliviousness. The technical details for the proof for any sublinear non-obliviousness are moved to the appendix.

[2] Note that these automata may have the maximal possible (linear) number of reversals.

2 Oblivious Two-Way Automata

The goal of this section is to extend the result of Sipser to oblivious two-way automata. We do it by the reduction method, i.e., we prove that the existence of a small oblivious two-way deterministic automaton for B_n implies the existence of a small deterministic sweeping automaton for B_n.

In what follows we always assume, that the input tape of an automaton contains $¢w\$$ for any input word w, where ¢ is called the left endmarker and \$ is called the right endmarker. If a two-way automaton reads ¢ [\$] it may not move to the left [right]. Moreover, one assumes that each two-way automaton has exactly one accepting state q_{accept} and exactly one rejecting state q_{reject}. There is no more possible from these two special states.

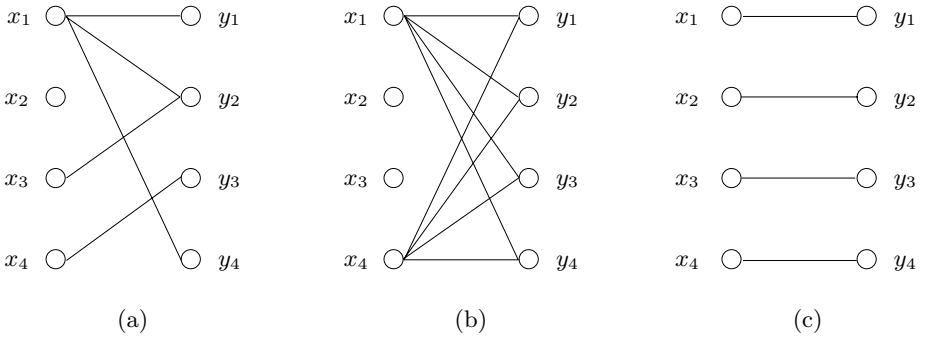

Fig. 1.

Now, let us describe the Sipser's language B_n. Let Σ_n be an alphabet of 2^{n^2} symbols where each symbol of Σ_n represents a bipartite graph of $2n$ vertices $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$ with edges that lead from x-vertices to y-vertices only. Figure 1 shows examples of three symbols of Σ_4. The symbol at Figure 1(c) corresponds to the bipartite graph

$$(\{x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4\}, \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}).$$

The bipartite graph of $2n$ vertices that contains exactly the edges (x_i, y_i) for $i = 1, \ldots, n$ is called the dummy symbol of Σ_n and denoted by d_n. The concatenation of two symbols a and b of Σ_n represent a graph of $3n$ vertices that is obtained by identifying y_i of a with x_i of b for every $i \in \{1, ...n\}$. For instance, the concatenation of the bipartite graphs in Figure 1 (from the left to the right) results in the graph in Figure 2. Thus, any word w of Σ_n corresponds to a graph $G(w)$ of $n \cdot (|w|+1)$ vertices. The language B_n consists of words $w \in (\Sigma_n)^+$ such that $G(w)$ contains a path of length $|w|$ that connects one of the n "left-most" vertices of $G(w)$ with one of the n "right-most" vertices of $G(w)$. For instance, the graph in Figure 2 corresponds to a word in B_n because there is a path from x_1 to y_1 of length 3.

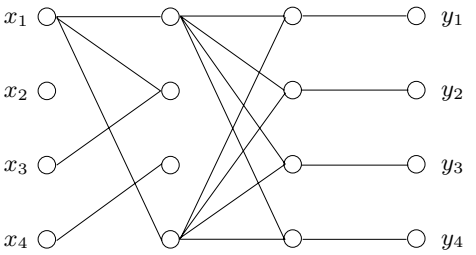

Fig. 2.

Our first useful observation is devoted to the dummy symbol d_n. Let h be a homomorphism defined by

$$h(a) = a \text{ for all } a \in \Sigma_n - \{d_n\}, \text{ and } h(d_n) = \epsilon.$$

where ϵ is the empty word.

Observation 1 *For any* $w \in \Sigma_n^*$,

$$h(w) \in L \text{ iff } w \in L.$$

Next we need to define some basic terms related to the computations of a 2dfa. A *configuration* of a 2dfa A is a triple $(q, ¢w\$, i)$, where q is the state of A, $¢w\$$ is the content of the tape and i is the position of the reading head of A on the tape.

We always assume that ¢ is on position 0 of the tape and hence \$ is on position $|w| + 1$. The pair (q, i) is called the *internal configuration* of configuration $(q, ¢w\$, i)$. A *computation* of A is any sequence of configurations $C_1, \ldots, C_m$ such that A can move from C_i to C_{i+1} in one step for $i = 1, \ldots, m-1$. Any subsequence $C_k, C_{k+1}, \ldots, C_l$ of $C_1, \ldots, C_m$, $1 \leq k < l \leq m$, is called a computation part of the computation $C_1, \ldots, C_m$. Let q_0 be the initial state of A. The *computation of A on input w* is a computation of A that starts in the configuration $(q_0, ¢w\$, 0)$ and finishes either in an accepting state or in a rejecting state. W.l.o.g. we may assume that any 2dfa has exactly one accepting state and exactly one rejecting state and that there are no more possible moves from these states.

For any configuration $C = (q, ¢w\$, i)$, we set $state(C) = q$ and $pos(C) = i$. Any part $C_1, \ldots, C_v$ of a computation $B_1, B_2, \ldots, B_r, C_1, \ldots, C_v, D_1, \ldots, D_s$ is called a *cycle*, if $\text{state}(C_1) = \text{state}(C_v)$ and A reads the same symbol during this computation part (i.e., all tape cells visited in this computation part contain the same symbol).

Fact 1 *If $C_1, \ldots, C_v$ is a computation part with $state(C_1) = state(C_v)$ and $pos(C_1) = pos(C_v)$, then the computation containing $C_1, \ldots, C_v$ is infinite and so it cannot be an accepting computation.*

A cycle $D = C_1, \ldots, C_v$ is called a *simple* cycle if

$$|\{\text{state}(C_1), \text{state}(C_2), \ldots, \text{state}(C_v)\}| = v - 1,$$

i.e., the state $\text{state}(C_1) = \text{state}(C_v)$ is the only state that occurs twice in $C_1, \ldots, C_v$.

We denote $\text{pos}(C_v) - \text{pos}(C_1)$ by $\text{move}(C_1, \ldots, C_v) = \textit{move(D)}$. If $\text{move}(D) > 0$, we say that D goes to the right, and if $\text{move}(D) < 0$, we say that D goes to the left. Let

$$\text{left}(D) = \min\{\text{pos}(C_i) | i = 1, \ldots, v\}, \text{ and}$$
$$\text{right}(D) = \max\{\text{pos}(C_i) | i = 1, \ldots, v\}.$$

We denote by

$$\textit{diff}(C_1, \ldots, C_v) = \text{right}(D) - \text{left}(D)$$

the length of the part of the tape that is scanned during D. We observe that a simple cycle cannot cover many positions.

Observation 2 *Let $D = C_1, \ldots, C_v$ be a simple cycle of a computation of a 2dfa $A = (Q, \Sigma, \delta, q_0, q_{accept}, q_{reject})$. Then*

$$\text{diff}(D) \leq |Q|$$

and so $|move(D)| < |Q|$.

Observation 3 *Let $C = C_1, \ldots, C_m$ be a part of a computation of a 2dfa A with k states. If $|pos(C_m) - pos(C_1)| \geq k$, $pos(C_i)$ lies between $pos(C_1)$ and $pos(C_m)$ for all $i = 1, \ldots, m$, and all cells on the positions from $pos(C_1)$ to $pos(C_m)$ contain the same symbol, then C contains a simple cycle.*

Let $C = C_1, C_2, \ldots, C_m$ be a part of a computation of a 2dfa. The sequence

$$\text{e-Traj}(C) = \text{pos}(C_1), \text{pos}(C_2), \ldots, \text{pos}(C_m)$$

is called the *exact trajectory* of C. The *trajectory* of C is the maximal subsequence $\text{Traj}(C) = \alpha_1, \ldots, \alpha_s$ of $\text{e-Traj}(C)$ such that

$$\alpha_1 = \text{pos}(C_1), \alpha_s = \text{pos}(C_m) \text{ and } \alpha_i \neq \alpha_{i+1} \text{ for } i = 1, \ldots, s-1.$$

Definition 1. *Let A be a 2dfa. We say that A is an* oblivious *2dfa, if for every $n \in \mathcal{N}$, the trajectories of all computations of A on words of length n are the same.*

Observe that sweeping automata can be transformed to oblivious 2dfa's with the trajectories

$$(0, 1, 2, \ldots, |w|, |w| + 1, |w|, \ldots, 2, 1, 0)^j$$

for a $j \leq |Q|$ and $w \in \Sigma^*$. One can easily design a transformation to an oblivious 2dfa that causes at most a quadratic growth of the size of a given sweeping automaton.

Now, we are ready to present the main result of this section.

Theorem 1. *Let n be a positive integer. Every oblivious 2dfa that accepts B_n has at least 2^n states.*

Proof Outline. Let A_n be an oblivious 2dfa that accepts B_n, and let A_n have k states. We will show that there exists a sweeping 2dfa S_n that accepts B_n and has at most k states. To construct S_n we need first to show that the trajectory T_m of A_n on inputs of length m is "nice" in the sense of the following two facts. Let us call the first k symbols and the last k symbols of the input the *border of the input.*

Fact 2 *Let A_n have its head on ¢ or \$ in a configuration of a computation on an input $w \in B_n$ of length $m > 2k$. Then in the next $(k+1)^2+1$ steps A_n either finishes its computation or leaves the border of the input.*

Proof. Fact 2 is a direct consequence of Fact 1.

To show that T_n must be nice we consider the input $w_m = (d_n)^m \in B_n$.

Fact 3 *Let $C = C_1, \ldots, C_r$ be a computation part of A_n on w_m where $pos(C_1) = 0$ $[pos(C_r) = m+1]$, $pos(C_r) = k+1$ $[pos(C_r) = m-k-1]$ and $pos(C_i) \leq k+1$ $[pos(C_i) \geq m-k-1]$ for all $i = 1, 2, \ldots, r-1$. Then C contains a simple cycle C' that goes to the right [to the left].*

Proof. Fact 3 is a direct consequence of Observation 3.

Combining Fact 2 and Fact 3 one can obtain the following characterization of the trajectory T_m on w_m (and so on every word of length m from Σ_n).

Lemma 1. *Let T' be any part of T_m on w_m that starts on ¢ [\$] and ends on \$ [¢]. Then after at most $(k+1)^2+1$ steps T' leaves the border and starts to move in a simple cycle going to the right [left] until T' reaches \$ [¢] (Figure 3).*

Thus, the computation $C(w)$ of A_n on $w_m = (d_n)^m$ alternates between short computation parts on the borders of the input and crossings of the input from the left to the right or from the right to the left in a simple cycle. The following lemma provides a property of $C(w)$ that is crucial for the construction of the sweeping 2dfa S_n with k states and $L(S_n) = B_n$.

Lemma 2. *Let $C_i, C_{i+1}, \ldots, C_l$ be a part of $C(w_m)$ with the following properties:*

1. *$pos(C_i) > 0$ and $pos(C_l) < m+1$ $[pos(C_i) < m+1$ and $pos(C_l) > 0]$*
2. *$pos(C_i) < pos(C_j) < pos(C_l)$ $[pos(C_i) > pos(C_j) > pos(C_l)]$ for $j \in \{i+1, i+2, \ldots, l-1\}$, and*
3. *$|pos(C_l) - pos(C_i)| \geq 2k$.*

Then the position $pos(C_i)$ cannot be visited again in $C(w)$ before the endmarker \$ [¢] has been visited.

Now, we are ready to outline the construction of S_n. For every input

$$x = x_1 x_2 \ldots x_r \in (\Sigma_n)^r,$$

S_n simulates the work of A_n on the input

$$\text{virtual}(x) = (d_n)^{2k} x_1 (d_n)^{2k} x_2 (d_n)^{2k} \ldots (d_n)^{2k} x_r (d_n)^{2k}$$

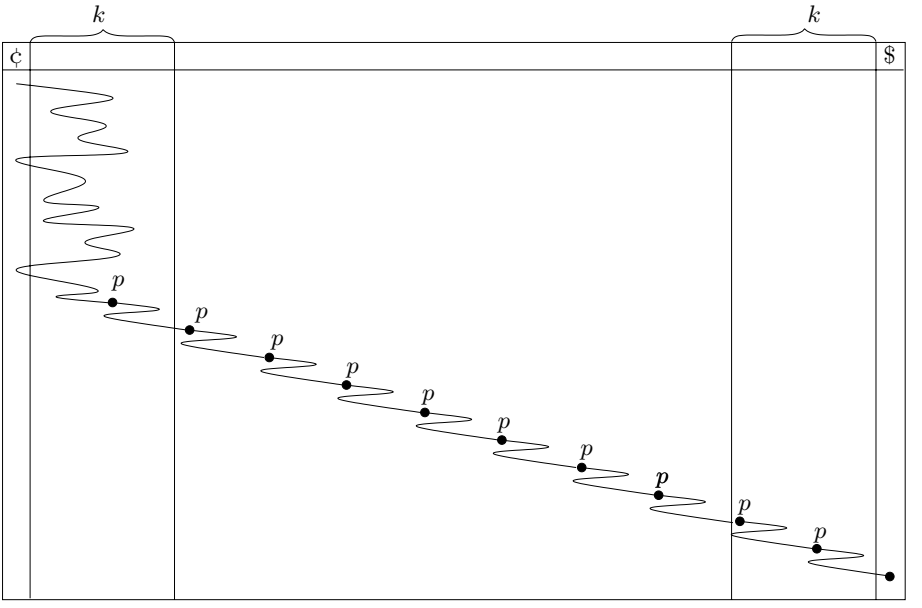

Fig. 3.

in the following way. If A_n reads a symbol x_i in a state q and after that it moves to the right [to the left] in a state p, then S_n looks in a table saying what happens when A_n enters the word $(d_n)^{2k}$ from the left [right] in the state p. There are only three possible situations (Figure 4):

(i) A_n finishes the computation in q_{accept} or q_{reject} without leaving the subword $(d_n)^{2k}$.
(ii) A_n crosses $(d_n)^{2k}$ and leaves it on the other side in a state s.
(iii) A_n leaves $(d_n)^{2k}$ and returns to x_i in a state h.

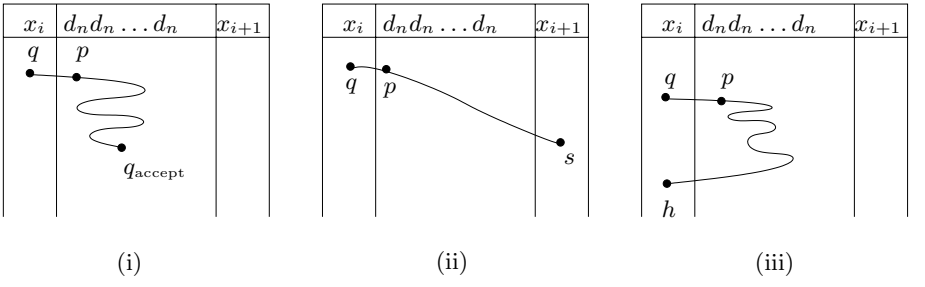

Fig. 4.

If (i) happens, S_n enters the corresponding state q_{accept} or q_{reject} without moving its head. If (ii) happens, then S_n moves the head to the right [left] to the position x_{i+1} [x_{i-1}] in the state s. If (iii) happens, S_n exchanges the state q for the state h without moving its input head.

One can easily observe that the table describing the behaviour of A_n on $(d_n)^{2k}$ can be stored in the transition function of S_n and that S_n uses the same set of states as A_n. The automaton S_n accepts B_n because of Observation 1 that claims

$$x \in B_n \Leftrightarrow \text{virtual}(x) \in B_n.$$

It remains to show that S_n is a sweeping 2dfa. But this is a direct consequence of Lemma 2 that claims that in a crossing of A_n on virtual(x) from the left to the right [from the right to the left] one cannot return from x_{i+1} to x_i [from x_{i-1} to x_i] before visiting \$ [¢], i.e., if S_n simulates the work of A_n on virtual(x) in the above described way then it makes reversals on the endmarkers only. Hence, S_n is a sweeping 2dfa.

This completes the proof of Theorem 1. □

3 Bounded-Degree Non-oblivious Automata

In this section we present our main result and a proof idea.

Theorem 2. *Let n be a positive integer. Any 2dfa that accepts B_n with $o(n)$ degree of non-obliviousness has at least $2^{\Omega(n)}$ states.*

The idea of the proof is again the reduction to sweeping automata. We have to show that if there is a "small" 2dfa with sublinear degree of non-obliviousness for B_n, then there is a small sweeping 2dfa for B_n. Let D_n be a minimal 2dfa with a sublinear degree of non-obliviousness that accepts B_n. Let D_n have k_n states.

To simplify our argument we use a concept based on the following technical assertions.

Lemma 3. *For every $n \in \mathcal{N} - \{0\}$, there exists a positive integer r_n such that, any 2dfa accepting B_n and working in the sweeping manner on all inputs of length of at most r_n has at least $2^{\Omega(n)}$ states.*

Fact 4 *Let E be a 2dfa that accepts B_n and behaves in the sweeping manner on inputs of length r_n. Then there exists a 2dfa F with $L(E) = L(F) = B_n$, $size(F) = O(size(E))$, and F behaves in the sweeping manner on all inputs of lengths at most r_n.*

Following Lemma 3 and Fact 4 it is sufficient to show that the existence of a "small" 2dfa D_n with sublinear degree of non-obliviousness for B_n implies the existence of a small 2dfa that accepts B_n and works in the sweeping manner on all inputs of length r_n. First we outline the proof for the degree 2 of non-obliviousness and then we give an idea how to generalize it for proving Theorem 2.

Let $L(D_n) = B_n$, size$(D_n) = k_n$, and let the degree of non-obliviousness of D_n be at most 2.

Consider the work of D_n on inputs of the length

$$m = 3 \cdot (2k_n + 1) \cdot r_n.$$

Let T_1 and T_2 be the two possible trajectories of D_n on inputs of this length. Let T_1 be the trajectory on $(d_n)^m$. Following Lemma 1 and Lemma 2 the trajectory T_1 consists of crossings between ¢ and \$ in which the head never moves back to a position in the distance $2k_n$ from the current position. Consider the set of inputs

$$X_1 = \{(d_n)^{2k_n} x_1 (d_n)^{2k_n} x_2 \dots x_{r_n - 1} (d_n)^{2k_n} x_{r_n} y \mid \\ x_i \in \Sigma_n \text{ for } i = 1, \dots, n, y \in \{d_n\}^*, |y| = m - r_n \cdot (2k_n + 1)\}$$

We distinguish two possibilities with respect to the trajectories T_1 and T_2.

1. Assume all words in X_1 have trajectory T_1. Then in a similar way as in the proof of Theorem 1 one can construct a 2dfa H_n that for every input $x = x_1 x_2 \dots x_{r_n}$ of length r_n simulates the work of D_n on

   $$\text{virtual}(x) = (d_n)^{2k_n} x_1 (d_n)^{2k_n} x_2 \dots x_{r_n} (d_n)^{2(k_n+1) \cdot r_n} \in X_1.$$

   Thus, H_n computes in the sweeping manner on inputs of length r_n. Since H_n simulates the work of D_n on virtual(y) for each $y \in \Sigma_n^*$ (i.e., for any input length), H_n accepts B_n. Since size(H_n) = size(D_n), Lemma 4 and Fact 4 imply

   $$\text{size}(D_n) = 2^{\Omega(n)}.$$

2. Assume a word $z \in X_1$ has trajectory T_2. A reasonable generalization of Lemma 1 and Lemma 2 shows that the computations on all words with trajectory T_2 behave as described in Lemma 2 on the second half of these inputs. Since this is the case for T_1 too, D_n behaves "nicely" on the second half of all inputs. Considering the language

   $$X_2 = \{(d_n)^{2(k_n+1)r_n} x_1 (d_n)^{2k_n} x_2 \dots x_{r_n - 1} \ (d_n)^{2k_n} x_{r_n} (d_n)^{2k_n} \mid \\ x_i \in \Sigma_n \text{ for } i = 1, \dots, n\}$$

   the proof can be completed in a similar way as in the first case.

Now we explain how to generalize this proof idea for proving Theorem 2.

Outline of the Proof of Theorem 2

Let D_n be a 2dfa that accepts B_n. Let the degree of non-obliviousness of D_n be bounded by a function $f(n) = o(n)$. Let Size$(B_n) = k(= k_n)$.

The idea of the proof is to show that there are some "nice" trajectories for some large group of words and then to use these trajectories to construct an automaton F_n with $L(F_n) = B_n$ that works in the sweeping manner on inputs of length r_n.

To formalize our idea we need the following terminology.

Definition 2. *Let T be a trajectory of the computation $C(w)$ of D_n on an input word w. Let $w = xyz$ for some $x, y, z \in (\Sigma_n)^*$ and $|y| \geq 2k$. We say that T* behaves nicely on the subword *y of w if*

1. *T always crosses y from the left to the right or from the right to the left in a simple cycle (Lemma 1), and*
2. *in each crossing from the left to the right [from the right to the left] if T visits a position j of the tape, then in the rest of this crossing T does not visit any position $j - i$ [$j + i$] for $i > k$ (see Lemma 2).*

The assertion of the following lemma can be proved in the same way as Lemma 1 and Lemma 2.

Lemma 4. *Let $w = x(d_n)^k y(d_n)^k z \in (\Sigma_n)^+$ for a word $y \in \{d_n\}^*$, $|y| \geq 2k$. Then the trajectory of the computation of D_n on w behaves nicely on y.*

Observation 4 *Let T be a trajectory of D_n on two different words xyz and uvw with $|x| = |u|$, $|y| = |v|$ and $|z| = |w|$. If T behaves nicely on y, then T behaves nicely on v.*

Thus, if T behaves nicely on y from xyz we can say that T *behaves nicely on the (tape) interval* $[|x| + 1, \ldots, |x| + |y|]$.

Let c_n be a positive integer such that

$$f(m) < m/((2k + 1) \cdot r_n \tag{1}$$

for very $m \geq c_n$. Now, we study the computations of D_n on words of the length

$$m = (2k + 1) \cdot c_n \cdot r_n + 2k.$$

Since $m > c_n$, (1) implies that $f(m) < c_n$. Let $T_1, T_2, \ldots, T_{f(m)}$ be all trajectories of D_n on inputs of the length m. Note, that some of these trajectories may be very complex and so they may be very far from being "nice" on any subpart of input. Our idea is to consider some classes $X_1, X_2, \ldots, X_{C_n}$ of "nice" words and to show that the words of at least one of these classes have nice trajectories. Consider the sets:

$$Y = \left\{ (d_n)^{2k} x_1 (d_n)^{2k} x_2 \ldots x_{r_n - 1} (d_n)^{2k} x_{r_n} \mid x_i \in \Sigma_n \right\}$$

$$X_i = \left\{ (d_n)^{(2k+1) \cdot r_n \cdot (i-1)} y (d_n)^{(2k+1) \cdot r_n \cdot (c_n - i) + 2k} \mid y \in Y \right\}$$

for all $i = 1, 2, \ldots, c_n$. Observe (Figure 5) that for $i \neq j$, the intervals of non-dummy symbols of X_i and X_j do not overlap.

Since the number of intervals $[z_i, z_{i+1}]$ is larger than the number of trajectories, one can show that there exists a $j \in \{1, \ldots, c_n - 1\}$ such that all words of X_j have trajectories that behave nicely on the interval $[z_{j-1} + k + 1, \ldots, z_j]$ (Figure 5).

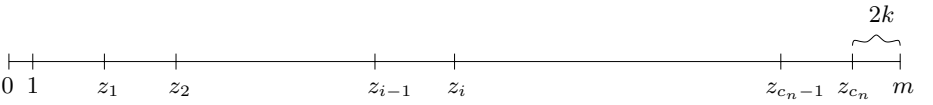

Fig. 5. Two words in X_i have their non-dummy symbols in the interval $[z_{i-1} + k, \ldots, z_i - k]$, where $z_i = (2k+1) \cdot r_n \cdot (i-1)$ for $i = 1, \ldots, c_n$.

Then the construction of F_n can be done as described as follows. For simplicity, consider $j = 2$. The automaton F_n working on some $y = y_1 \ldots y_{r_n}, y_i \in \Sigma_n$ for $i = 1, \ldots, r_n$ simulates the work of D_n on

$$\text{virtual}(y) = (d_n)^{(2k+1)\cdot r_n}(d_n)^{2k} y_1 \ldots (d_n)^{2k} y_{r_n} (d_n)^{(2k+1)\cdot r_n (c_n-2)+2k} \in X_2.$$

To simulate the work on the block $(d_n)^{2k}$ between y_i and y_{i+1} one uses the strategy described in the construction of S_n in the proof of Theorem 1. If F_n reads ¢ it simulates the work of D_n on ¢$(d_n)^{(2k+1)\cdot r_n}$ and if F_n reads \$ then it simulates the work of D_n on $(d_n)^{(2k+1)\cdot r_n (c_n-2)2k}$\$. These simulations can be performed by one step of F_n because the tables describing the input-output behaviour of D_n when entering ¢$(d_n)^{(2k+1)r_n}$ from the left or $(d_n)^{(2k+1)\cdot r_n (c_n-2)+2k}$\$ from the right can be saved in the description of the transition function of F_n. Since all trajectories behave nicely on the interval $[z_1 + k + 1, \ldots, m - k]$, F_n becomes a sweeping automaton on inputs of length r_n.

□

References

1. P. Berman: A note on sweeping automata. In: *Proc. 7th ICALP. Lecture Notes in Computer Science* 85, Springer 1980, pp. 91–97.
2. P. Ďuriš, J. Hromkovič, S. Jukna, M. Sauerhoff, G. Schnitger: On multipartition communication complexity. In: *Proc. STACS '01. Lecture Notes in Computer Science* 2010, Springer 2001, pp 206–217.
3. J. Hromkovič: *Communication Complexity and Parallel Computing*. Springer 1997.
4. J. Hromkovič, G. Schnitger: On the power of Las Vegas II: Two-way finite automata. *Theoretical Computer Science* 262 (2001), 1–24.
5. J. Hromkovič, J. Karhumäki, H. Klauck, G. Schnitger, S. Seibert: Measures of nondeterminism in finite automata. In: *Proc. 27th ICALP,Lecture Notes in Computer Science* 1853, Springer-Verlag 2000, pp. 194–21, full version: *Information and Computation* 172 (2002), 202–217.
6. D. A. Huffman: The synthesis of sequential switching circuits. *J. Franklin Inst.* 257, No. 3–4 (1954), pp. 161–190 and pp. 257–303.
7. H. Leung: Tight lower bounds on the size of sweeping automata. *J. Comp. System Sciences*, to appear.
8. G. M. Mealy: A method for synthesizing sequential circuits. *Bell System Technical Journal* 34, No. 5 (1955), pp. 1045–1079.
9. A. Meyer and M. Fischer: Economy in description by automata, grammars and formal systems. In: *Proc. 12th SWAT Symp.*, 1971, pp. 188–191.

10. S. Micali: Two-way deterministic automata are exponentially more succinct than sweeping automata. *Inform. Proc. Letters* 12 (1981), 103–105.
11. E. F. Moore: Gedanken experiments on sequential machines. In: [14], pp. 129–153.
12. F. Moore: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata. *IEEE Trans. Comput.* 10 (1971), 1211–1214.
13. M. O. Rabin, D. Scott: Finite automata and their decision problems. In: *IBM J. Research and Development*, 3 (1959), pp. 115–125.
14. W. J. Sakoda, M. Sipser: Nondeterminism and the size of two-way finite automata. In: *Proc. 10th ACM STOC*, 1978, pp. 275–286.
15. C. E. Shannon and J. McCarthy: *Automata Studies.* Princeton University Press, 1956.
16. M. Sipser: Lower bounds on the size of sweeping automata. *J. Comp. System Sciences* 21 (1980), 195–202.