Tight Lower Bounds on the Size of Sweeping Automata

Hing Leung

Department of Computer Science, New Mexico State University, Las Cruces, New Mexico 88003
E-mail: hleung@cs.nmsu.edu

Received March 5, 1999; revised May 25, 2001

A sweeping automaton is a two-way deterministic finite automaton which makes turns only at the endmarkers. We say that a sweeping automaton is degenerate if the automaton has no left-moving transitions. We show that for each positive integer \( n \), there is a nondeterministic finite automaton \( A_n \) over a two-letter alphabet such that \( A_n \) has \( n \) states, whereas the smallest equivalent nondegenerate sweeping automaton has \( 2^n \) states.

Key Words: deterministic finite automata; nondeterministic finite automata; two-way finite automata; sweeping automata; descriptional complexity.

1. INTRODUCTION

The simplest machine model for denoting regular languages is the one-way deterministic finite automaton (DFA). It is well known that the use of nondeterminism and two-way movements of the tape head would not change the class of languages denoted.

In the literature, the transition function of a DFA is usually required to be a total function. If the transition function of a DFA is allowed to be a partial function, then we say that the automaton is an incomplete DFA. The reader is referred to [4] for the definitions of DFAs and NFAs (one-way nondeterministic finite automaton) with the exception that NFAs will be allowed to have a set of starting states.

Trade-offs in the succinctness of different machine models for denoting the same languages were studied in a number of research papers. Meyer and Fischer [6] showed that for each positive \( n \), there is an \( n \)-state NFA such that the corresponding smallest equivalent DFAs have \( 2^n \) states. The same result was also obtained by Moore [7] using a different family of NFAs.

Sakoda and Sipser [9] raised an open question regarding the trade-off in the succinctness between two-way nondeterministic finite automata (2NFA) and two-way
deterministic finite automata (2DFA). Specifically, they asked whether there exists a polynomial \( p \) such that for every \( n \)-state 2NFA there is an equivalent \( p(n) \)-state 2DFA.

A partial negative answer has been provided by Sipser in [10]. He introduced sweeping automata as a restricted model of 2DFA in which turns can occur only at the endmarkers. In Section 2, a sweeping automaton is formally defined in such a way that the transition function is allowed to be a partial function. We say that a sweeping automaton is degenerate if the automaton has no left-moving transitions. Thus, a degenerate sweeping automaton is limited to making only one sweep of the input from left to right. That is, a degenerate sweeping automaton is the same as an incomplete DFA except that the input is delimited by two endmarkers for a sweeping automaton. We observe (in Section 2) that any \( n \)-state incomplete DFA can be converted to an equivalent \( n \)-state degenerate sweeping automaton and vice versa. Sipser [10] showed that for each positive \( n \), there is a language \( B_n \) such that \( B_n \) is accepted by an \( n \)-state NFA whereas it is not accepted by any nondegenerate sweeping automaton with fewer than \( 2^n \) states.

Note that every \( n \)-state NFA can be converted by the subset construction to an equivalent incomplete DFA (hence a degenerate sweeping automaton) of at most \( 2^n - 1 \) states. By adding one dummy state with a left-moving transition, we obtain a nondegenerate sweeping automaton with at most \( 2^n \) states. Thus, we deduce that a smallest nondegenerate sweeping automaton denoting \( B_n \) has \( 2^n \) states and a smallest degenerate sweeping automaton denoting \( B_n \) has \( 2^n - 1 \) states. Therefore, Sipser’s result achieves the largest trade-off possible in the number of states between NFA and sweeping automata.

If should be noted that \( B_n \) is over an alphabet of size \( 2^{n^2} \). It is argued [10] that with some clever encoding of the alphabet into a binary alphabet, the resulting language still requires \( 2^n \) states on a nondegenerate sweeping automaton and can be recognized by an \( O(n) \)-state NFA.

In this paper, we show that for each positive \( n \), there is an NFA \( A_n \) over a binary alphabet such that any equivalent nondegenerate sweeping automaton requires at least \( 2^n \) states. Our result is tight in the sense that no family of NFAs over a unary alphabet can provide the same largest trade-off in sizes between NFAs and sweeping automata. This is because Chrobak [3] showed that any \( n \)-state NFA over a unary alphabet can be simulated by a DFA of \( O(e^{\sqrt{n \log n}}) \) states.

In fact, the NFAs \( A_n \) over a binary alphabet were introduced in [5]. For any positive integer \( n \), \( A_n \) has \( n \) states, whereas the smallest equivalent DFA has \( 2^n \) states. The \( n \)-state NFA \( A_n \) is very simple and compact in that it has only \( 2n \) transitions. In [5], we showed another strong succinctness property of \( A_n \). It is shown that \( A_n \) is exponentially ambiguous and yet any equivalent polynomially ambiguous NFA would require at least \( 2^n - 1 \) states. It is easy to obtain an unambiguous NFA of \( 2^n - 1 \) states by eliminating the dead state from the subset construction. The same construction gives an equivalent incomplete DFA of \( 2^n - 1 \) states.

\(^2\)Sipser did not introduce the terminology of nondegenerate sweeping automata. Sweeping automata, according to Sipser [10], are required to include both left-moving and right-moving transitions.
It is shown [1, 2, 10] that the question of the trade-off in succinctness between 2NFA and 2DFA (or 2NFA and sweeping automata) is related to the famous open question of whether deterministic log space (denoted L) is properly contained in nondeterministic log space (denoted NL). Specifically, it is noted [1, 10] that if the strings involved in the proof of the exponential lower bound result are polynomial in length, then L \neq NL.

There are some interesting differences between the proof techniques used in our paper and in Sipser’s [10]. In the works of Sipser, Berman, and Micali [1, 8, 10], the proofs relied heavily on the use of a substring d of length 2^n denoting a sequence of consecutive numbers from 0 to 2^n - 1 such that d is not in the language considered but the removal of any proper substring from d would result in a string in the language. In contrast, the crucial substring w_1 w_2 (Section 3) involved in our proof is of length at most 4n+2 instead of 2^n. This feature may become important when we try to extend the result to prove that L \neq NL. However, the strings g w_1 w_2 g (Section 3) considered in our current proof are not guaranteed to be polynomial in length, where g = 0^{2n-2}a0^{2n-2}. Specifically, the string a may be very long.

We believe that A_n is a good candidate to be considered in the trade-off question between NFA and 2DFA. That is, we conjecture that any equivalent 2DFA would require an exponential number of states.

In Section 2, we define the NFAs A_n and give the definitions and notation for 2DFA and sweeping automata. In Section 3, we prove the 2^n lower bound for sweeping automata.

2. DEFINITIONS AND NOTATION

2.1. NFA A_n

For any positive integer n, we define an NFA A_n = (P, \Sigma, \delta_A, \{p_1\}, \{p_1\}) (Fig. 1), where P = \{p_1, p_2, ..., p_n\} is the set of states, p_1 is the only starting state and the only final state, \Sigma = \{0, 1\}, and \delta_A is defined as follows:

- \delta_A(p_1, 0) = \{p_1, p_2\}
- \delta_A(p_i, 0) = \{p_{i+1}\} for 2 \leq i \leq n - 1
- \delta_A(p_n, 0) = \{p_1\}
- \delta_A(p_1, 1) = \emptyset
- \delta_A(p_i, 1) = \{p_i\} for 2 \leq i \leq n.

We denote the language of A_n by L_n, which is (0+0(1*0)^{n-1})*. It is easy to see that L_n = L_\Sigma^* = L_n^*.

We say that y \in \Sigma^* is live with respect to a language L iff \exists x, z \in \Sigma^* such that xyz \in L; otherwise y is dead with respect to L. Sometimes, we may simply say that a string is live (or, dead) without mentioning explicitly the underlying language, which could also be the language of an automaton.
FIG. 1. Transition diagram of $A_n$.

Since every state in $A_n$ is reachable (from the starting state $p_1$) and useful (that is, can reach the final state $p_1$), we see that a string $y \in \Sigma^+$ is live (with respect to $L_n$) iff there exist some states $p_i, p_j$ such that $p_j \in \delta_A(p_i, y)$. In other words, a string $y$ is live iff $\delta_A(P, y) \neq \emptyset$.

In [5], it is shown that the smallest DFA equivalent to $A_n$ has $2^n$ states. Thus, the smallest equivalent incomplete DFA has $2^n - 1$ states.

2.2. 2DFA

A 2DFA is a 7-tuple $(Q, \Sigma, \bot, \bot, \delta, q_1, F)$, where $Q = \{q_1, q_2, \ldots, q_k\}$ is the set of states, $\Sigma$ is the alphabet set, $\bot \notin \Sigma$ and $\bot \notin \Sigma$ are left and right endmarkers delimiting the input string, $q_1$ is the starting state, and $F$ is the set of accepting states.

The transition function $\delta$ is a partial function from $Q \times (\Sigma \cup \{\bot, \bot\})$ to $Q \times \{L, R\}$. An input string $w = a_1a_2\ldots a_n$, where $a_i \in \Sigma$ for $1 \leq i \leq n$, is presented to the 2DFA as $\bot a_1a_2\ldots a_n \bot$. The 2DFA is started in state $q_1$ on the symbol $a_1$. If $w$ is the empty string $\varepsilon$, then the 2DFA is started in state $q_1$ on the right endmarker $\bot$.

The input string $w$ is accepted if, from the initial configuration in which the automaton is in state $q_1$ while reading $a_1$ (or $\bot$ if $w = \varepsilon$), the 2DFA enters into a configuration with the state in $F$ while reading $\bot$ after a sequence of moves. The sequence of moves may be empty if $w = \varepsilon$. More accurately, if the sequence of moves is not empty, the 2DFA must signal acceptance by entering a state in $F$ when it makes a right move on the symbol $a_n$ (before detecting that it has reached the $\bot$ symbol).

Let $q_s, q_t \in Q$ and $a \in \Sigma \cup \{\bot, \bot\}$. When $\delta(q_s, a) = (q_t, L)$, the meaning is that the 2DFA, when reading symbol $a$ while at state $q_s$, would move the tape head to the left and change the state to $q_t$. Similarly, the meaning of $\delta(q_s, a) = (q_t, R)$ is that the 2DFA, when reading symbol $a$ while at state $q_s$, would move the tape head to the right and change the state to $q_t$. Another possibility is that $\delta(q_s, a)$ could be undefined.

Let $w \in \Sigma^+$. We write $\delta(q_s, w) = (q_t, L)$ to denote that the 2DFA, when started at the leftmost symbol of $w$ at state $q_s$, eventually leaves $w$ moving to its left while entering state $q_t$; we write $\delta(q_s, w) = (q_t, R)$ to denote that the 2DFA, when started at the leftmost symbol of $w$ at state $q_s$, eventually leaves $w$ moving to its right while
Thus, the 2DFA will exhibit the same behavior on strings that hang or loops within states. On the other hand, any automaton can be converted to an equivalent incomplete DFA with at most \( n \) transitions without changing the language denoted. Thus, any are not useful for a degenerate sweeping automaton, we can remove these transitions.

Moving transitions. Since the right-moving transitions on the endmarker symbols are not useful for a degenerate sweeping automaton, we can remove these transitions without changing the language denoted. Thus, any \( n \)-state degenerate sweeping automaton can be converted to an equivalent incomplete DFA with at most \( n \) states. On the other hand, any \( n \)-state incomplete DFA can be considered as an

Similarly, we write \( \delta(w, q_i) = (q_i, L) \) to denote that the 2DFA, when started at the rightmost symbol of \( w \) at state \( q_i \), eventually leaves \( w \) moving to its left while entering state \( q_i \); we write \( \delta(w, q_i) = (q_i, R) \) to denote that the 2DFA, when started at the rightmost symbol of \( w \) at state \( q_i \), eventually leaves \( w \) moving to its right while entering state \( q_i \). Again, it is possible that \( \delta(w, q_i) \) may be undefined.

Let \( a \in \Sigma \). Using the previous notation, \( \delta(q_i, a) = \delta(a, q_i) \).

Let \( w \in \Sigma^+ \). We write \( \delta_1(w) \) to denote \( (\delta(q_1, w), \delta(q_2, w), \ldots, \delta(q_k, w)) \). Even if some of the \( k \) components of \( \delta_1(w) \) are not defined, we still consider \( \delta_1(w) \) to be defined. That is, \( \delta_1(w) = \delta_1(w') \) if corresponding entries of the \( k \)-tuples are either both undefined or both defined and equal. Similarly, we write \( \delta_k(w) \) to denote \( (\delta(w, q_1), \delta(w, q_2), \ldots, \delta(w, q_k)) \). Next, we write \( \delta(w) \) to denote \( (\delta_1(w), \delta_2(w)) \).

Thus, the 2DFA will exhibit the same behavior on strings \( w \) and \( w' \) if \( \delta(w) = \delta(w') \).

For \( Q' \subseteq Q \), let \( \delta_\le(Q', w) \) denote \( \{q_j \mid q_j \in Q', \delta(q_j, w) = (q_j, R)\} \) and \( \delta_\ge(w, Q') \) denote \( \{q_j \mid q_j \in Q', \delta(w, q_j) = (q_j, L)\} \). We write \( \eta(w) = (\# \delta_\le(Q, w), \# \delta_\ge(w, Q)) \).

We define a partial ordering on ordered pairs of natural numbers such that \( (p, q) \leq (i, j) \) iff \( p \leq i \) and \( q \leq j \). It is easy to see that \( \eta(ww') \leq \eta(w) \) and \( \eta(w'w) \leq \eta(w) \) for \( w, w' \in \Sigma^+ \).

We say that a live string \( w \) is minimal with respect to a 2DFA if, for all live strings \( w', \eta(w) \leq \eta(w) \) implies \( \eta(w') = \eta(w) \). Note that by definition a dead string cannot be minimal. It is easy to see that minimal strings exist when the 2DFA accepts a nonempty string.

### 2.3. Sweeping Automata

Conceptually, a sweeping automaton is a 2DFA which makes turns only at the endmarkers. Formally, a sweeping automaton is specified in the same way as a 2DFA. One way to define a sweeping automaton is by imposing strict syntactic requirements on the transition function so that the 2DFA is guaranteed to perform sweeping actions on any given input. Our definition of a sweeping automaton is more general. Whether a 2DFA is a sweeping automaton depends on the behavior of the automaton during the processing of each input string. As in the case of a 2DFA, a sweeping automaton is started in the initial state on the leftmost symbol of the input, which is the symbol to the right of the left endmarker \( \leftarrow \). We require that for each given input string, a sweeping automaton makes turns only at the endmarkers. Moreover, the first sweep has to be from left to right. For the special case when the input string is an empty string, any 2DFA can only sweep from one endmarker to another endmarker and thus behaves like a sweeping automaton.

We say that a sweeping automaton is degenerate if the automaton has no left-moving transitions. Since the right-moving transitions on the endmarker symbols are not useful for a degenerate sweeping automaton, we can remove these transitions without changing the language denoted. Thus, any \( n \)-state degenerate sweeping automaton can be converted to an equivalent incomplete DFA with at most \( n \) states. On the other hand, any \( n \)-state incomplete DFA can be considered as an
Therefore, for each \( g \) that, for each \( b \) but not for 2DFA. Similarly, we introduce the concepts of

Hence, \( g \) has the property that are distinguishable and the smallest DFA for

...0w

and redefine \( d \) as

\( d (w) = (\#d_\ldots, \ldots) \).

3. MAIN RESULT

Consider a nondegenerate sweeping automaton \( B = (Q, \Sigma, \rightarrow, \rightarrow, \delta, q_1, F) \) accepting the language \( L_\omega \) introduced in Section 2.1, where \( Q = \{q_1, q_2, \ldots, q_k\} \). We want to show that the number of states \( k \) is at least \( 2^\omega \).

Let \( A \) be the automaton \( A_\omega = (P, \Sigma, \delta_\omega, \{p_1\}, \{p_2\}) \) of Section 2.1. The following definition is taken from [5]. Let \( P' \subseteq P \). We define \( w_{p'} = \Sigma^+ \) to be \( w_1w_20w_30w_40 \ldots \) if \( p_1 \in P' \) and \( w_1 = 1 \) otherwise. It is easily verified [5] that, for each \( P \subseteq P \), \( \delta_\omega(p, w_{p'}) = \emptyset \) if \( p \notin P \) and \( p \in \delta_\omega(p, w_{p'}) \subseteq P_2 \) if \( p \in P_2 \).

Therefore, for each \( P_1 \subseteq P \),

\( P_1 \cap P_2 = \emptyset \) implies \( \delta_\omega(P_1, w_{p_2}) = \emptyset \)

and

\( P_1 \cap P_2 \subseteq \delta_\omega(P_1, w_{p_2}) \subseteq P_2 \).

Hence, \( \delta_\omega(P, w_{p_1}w_{p_2}) \neq \emptyset \) iff \( P_1 \cap P_2 \neq \emptyset \), which proves the following lemma.

LEMMA 3.1. For any \( P_1, P_2 \subseteq P \), \( w_{p_1}w_{p_2} \) is live iff \( P_1 \cap P_2 \neq \emptyset \).

Suppose \( P_1, P_2 \subseteq P \) and \( P_1 \neq P_2 \). Let \( p \in P_1 - P_2 \). We observe that the two strings \( 0^{n-1}w_{p_1} \) and \( 0^{n-1}w_{p_2} \) are inequivalent according to the equivalence relation defined in the Myhill–Nerode theorem [4] since \( 0^{n-1}w_{p_1}w_{p_2}0^{n-1} \notin L_\omega \) whereas \( 0^{n-1}w_{p_2}w_{p_1}0^{n-1} \notin L_\omega \). This is because \( \delta_\omega(p_1, 0^{n-1}w_{p_1}w_{p_1}0^{n-1}) = \delta_\omega(p, w_{p_1}w_{p_1}0^{n-1}) = \delta_\omega(P_1, w_{p_1}0^{n-1}) = \delta_\omega(p_1, 0^{n-1}) \equiv \{p_1\} \) and \( \delta_\omega(p_1, 0^{n-1}w_{p_2}w_{p_1}0^{n-1}) = \delta_\omega(P_2, w_{p_2}w_{p_1}0^{n-1}) = \delta_\omega(p_2, w_{p_2}0^{n-1}) = \delta_\omega(p_2, w_{p_2}0^{n-1}) = \delta_\omega(\emptyset, 0^{n-1}) = \emptyset \).

Therefore, the \( 2^\omega \) strings in the set \( \{0^{n-1}w_{p'} \mid P' \subseteq P \} \) are distinguishable and the smallest DFA for \( L_\omega \) has \( 2^\omega \) states.

Let \( \alpha \) be a minimal string with respect to \( B \) and let \( g = 0^{2n-2}0^{2n-2} \). Since \( 0^{2n-2} \) has the property that \( \delta_\omega(p, 0^{2n-2}) = P \) for each \( p \in P \), so do \( g \) and \( gg \) since \( \alpha \) is live.

Hence, \( g \) and \( gg \) are live, and so they are also minimal since \( \alpha \) is.
LEMMA 3.2. For any $P_1, P_2 \subseteq P$, $gw_{P_1}w_{P_2}g$ is minimal with respect to $B$ iff $P_1 \cap P_2 \neq \emptyset$.

Proof. By Lemma 3.1 and the fact that $g$ is minimal, we only need to show that $gw$ is live if $y$ is live, and this is true since $\delta_\epsilon(p, g) = P$ for each $p \in P$.

LEMMA 3.3. $\delta_\epsilon(Q, gg) = \delta_\epsilon(Q, g)$ and $\delta_\epsilon(gg, Q) = \delta_\epsilon(g, Q)$.

Proof. As noted earlier, $g$ and $gg$ are minimal. The lemma then follows from the facts that $\delta_\epsilon(Q, gg) \subseteq \delta_\epsilon(Q, g)$ and $\delta_\epsilon(gg, Q) \subseteq \delta_\epsilon(g, Q)$.

Let $\delta_\epsilon(Q, g) = \{q_1, q_2, \ldots, q_s\}$ and $\delta_\epsilon(g, Q) = \{q_1, q_2, \ldots, q_t\}$, where $\eta(g) = (s, t)$. Note that $s$ cannot be 0. Otherwise $g$ cannot be live, since a string is only accepted while reading the right endmarker. On the other hand, it is possible that $t$ may be 0.

Define a matrix $D_\epsilon$ over the field of integers mod 2 with rows indexed by $\{q | q \in \delta_\epsilon(Q, gw_{P_1}) \neq P \subseteq P\}$ and columns indexed by $\{(w_{P_2}, g, q_i) | 1 \leq i \leq s, \emptyset \neq P_2 \subseteq P\}$ such that $D_\epsilon[q, (w_{P_2}, g, q_i)] = 1$ if $\delta(q, w_{P_2}g) = (q_i, R)$, and 0 otherwise.

Define a matrix $D_\epsilon$ over the field of integers mod 2 with rows indexed by $\{q_1, gw_{P_1} \subseteq P\}$ and columns indexed by $\{q | q \in \delta_\epsilon(w_{P_2}g, Q), \emptyset \neq P_2 \subseteq P\}$ such that $D_\epsilon[(q_1, gw_{P_1}), q] = 1$ if $\delta(gw_{P_1}, q) = (q_1, L)$, and 0 otherwise.

Obtain from $D_\epsilon$ by elementary row operations a matrix $E_\epsilon$ over the field of integers mod 2 with rows indexed by $\{gw_{P_1} | \emptyset \neq P \subseteq P\}$ such that the row indexed by $gw_{P_1}$ is obtained by adding those rows of $D_\epsilon$ indexed by states in $\delta_\epsilon(Q, gw_{P_1})$.

Obtain from $D_\epsilon$ by elementary column operations a matrix $E_\epsilon$ over the field of integers mod 2 with columns indexed by $\{w_{P_2}g | \emptyset \neq P_2 \subseteq P\}$ such that the column indexed by $w_{P_2}g$ is obtained by adding those columns of $D_\epsilon$ indexed by states in $\delta_\epsilon(w_{P_2}g, Q)$.

LEMMA 3.4. Suppose $t > 0$. For any nonempty subsets $P_1, P_2$ of $P$, we have $P_1 \cap P_2 \neq \emptyset$ iff $E_\epsilon[gw_{P_1}, (w_{P_2}g, q_i)] = 1$ and $E_\epsilon[(q_1, gw_{P_1}), w_{P_2}g] = 1$ for all $1 \leq i \leq s$ and $1 \leq j \leq t$.

Proof. (Only If) Assume $P_1 \cap P_2 \neq \emptyset$. Since $g$ is minimal and by Lemma 3.2 $gw_{P_1}w_{P_2}g$ is minimal, we deduce that $gw_{P_1}$ is also minimal. Thus, $s = \#\delta_\epsilon(Q, g) = \#\delta_\epsilon(Q, gw_{P_1}) = \#\delta_\epsilon(Q, gw_{P_1}w_{P_2}g)$. Observe that $\delta_\epsilon(Q, gw_{P_1}, w_{P_2}g) = \delta_\epsilon(Q, gw_{P_1}w_{P_2}g) = \delta_\epsilon(Q, g, gw_{P_1}) = \delta_\epsilon(Q, g, gw_{P_1}w_{P_2}g)$. Let $1 \leq i \leq s$. Since $\#\delta_\epsilon(Q, gw_{P_1}) = s$, there exists a unique $q' \in \delta_\epsilon(Q, gw_{P_1})$ such that $\delta(q', w_{P_2}g) = (q_i, R)$. That is, for $q \in \delta_\epsilon(Q, gw_{P_1}), D_\epsilon[q, (w_{P_2}g, q_i)] = 1$ iff $q = q'$. Therefore, $E_\epsilon[gw_{P_1}, (w_{P_2}g, q_i)] = \sum D_\epsilon[q, (w_{P_2}g, q_i)] | q \in \delta_\epsilon(Q, gw_{P_1}) = 1$. Similarly, we can show that $E_\epsilon[(q_1, gw_{P_1}), w_{P_2}g] = 1$ for all $1 \leq j \leq t$.

(If) Assume $E_\epsilon[gw_{P_1}, (w_{P_2}g, q_i)] = 1$ and $E_\epsilon[(q_1, gw_{P_1}), w_{P_2}g] = 1$ for all $1 \leq i \leq s$ and $1 \leq j \leq t$. By Lemma 3.2 and the fact that $g$ is minimal, it suffices to show that $gw_{P_1}w_{P_2}g$ is live. By Lemma 3.3, we have $\delta_\epsilon(q_1, (q_2, \ldots, q_t), g) = \delta_\epsilon(\delta_\epsilon(q_1, (q_2, \ldots, q_t), g), w_{P_2}g) = \delta_\epsilon(\delta_\epsilon(\delta_\epsilon(Q, g), g), w_{P_2}g) = \delta_\epsilon(\delta_\epsilon(Q, gw_{P_1}w_{P_2}g), g)$ which is $\{q_1, \ldots, q_t\}$ by the
assumption that $E_x[\text{gw}_{P_1}, (w_{P_2} g, q_i)] = 1$ for $1 \leq i \leq s$. Thus, there exists $x$ such that $\delta(q_i, (\text{gw}_{P_1}, w_{P_2} g)) = (q_i, R)$ for all $1 \leq i \leq s$. Similarly, there exists $y$ such that $\delta((\text{gw}_{P_1}, w_{P_2} g), q_i) = (q_j, L)$ for all $1 \leq j \leq t$. Therefore, $\delta(g(\text{gw}_{P_1}, w_{P_2} g)^\vee g) = \delta(g g)$. Since $g g$ is live, $g(g(\text{gw}_{P_1}, w_{P_2} g)^\vee g)$ is live and hence $\text{gw}_{P_1} w_{P_2} g$ is live.

**Lemma 3.5.** Let $F$ be a matrix over the field of integers mod 2 with rows and columns indexed by nonempty subsets of $P$ such that if $P_1 \cap P_2 \neq \emptyset$ then $F[P_1, P_2] = 1$; otherwise $F[P_1, P_2]$ has an arbitrary value of 0 or 1. Then $\text{rank}(F) \geq 1 + \# \{(P_1, P_2) | P_1 \neq \emptyset, P_2 \neq \emptyset, P_1 \cup P_2 = P, P_1 \cap P_2 = \emptyset, F[P_1, P_2] = 0\}$.

**Proof.** We can index the rows and columns of $F$ by $n$-bit positive binary numbers in the order of increasing values such that any $n$-bit positive binary number $b_n b_{n-1} \ldots b_1$ corresponds to the nonempty set $P$ with the property that $p_i \in P$ iff $b_i = 1$ for $1 \leq i \leq n$. See Fig. 2 for a picture of $F$.

The entries labeled “1” are required to be 1. The entries labeled “?” are permitted to be 0 or 1. The entries labeled “?” consist of some entries that are required to be 1 and some that are permitted to be 0 or 1. It is easy to see that the rows that have a zero entry in the position labeled “?” together with the row of all 1’s, are linearly independent. We are done since the positions labeled “?” are entries $(P_1, P_2)$ such that $P_1 \neq \emptyset, P_2 \neq \emptyset, P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$.

**Lemma 3.6.** Suppose $t > 0$. Then $\text{rank}(E_{\ldots} + \text{rank}(E_{\ldots}) \geq 2^n$.

**Proof.** Recall that each column of $E_{\ldots}$ is indexed by $(w_{P_2} g, q_i)$, where $\emptyset \neq P_2 \subseteq P$ and $1 \leq i \leq s$. By selecting from $E_{\ldots}$ one column for each group of $s$ columns indexed by $\{(w_{P_2} g, q_i) | 1 \leq i \leq s\}$, we obtain a matrix $F_{\ldots}$ over the field of integers mod 2 with rows and columns indexed by nonempty subsets of $P$. The selections are done as follows. If $P_2 = P$, then we select any arbitrary column from the $s$ columns.
Similarly, nonempty subsets \( E \) stated in Lemma 3.4 that if \( \operatorname{rank}(E) \) of \( (w_{g}, q_{g}) \) \| 1 \leq i \leq s \}. Otherwise, suppose \( P_{1} \neq P \). Let \( P_{1} = P - P_{2} \). There are two cases. The first case is when \( E_{\text{m}} [(w_{g}, (w_{g}, q_{g})] = 1 \) for all \( 1 \leq i \leq s \). We select again an arbitrary column from the \( s \) columns of \( E_{\text{m}} \) indexed by \( \{w_{g}, q_{g}\} \| 1 \leq i \leq s \}. The second case is when there exists an \( 1 \leq i \leq s \) such that \( E_{\text{m}} [(w_{g}, (w_{g}, q_{g})] = 0 \). We select the column indexed by \( (w_{g}, q_{g}) \) from \( E_{\text{m}} \) as the column indexed by \( P_{1} \) for \( F_{\text{m}} \). It follows from the properties of \( E_{\text{m}} \) as stated in Lemma 3.4 that if \( P_{1} \cap P_{2} \neq \emptyset \) then \( F_{\text{m}} [P_{1}, P_{2}] = 1 \). That is, \( F_{\text{m}} \) has the structure of \( F \) as given in Lemma 3.5. Similarly, we can construct \( F_{\text{m}} \) from \( E_{\text{m}} \), which also has the structure of \( F \) as given in Lemma 3.5. Let

\[
\mathcal{P} = \{(P_{1}, P_{2}) \mid P_{1} \neq \emptyset, P_{2} \neq \emptyset, P_{1} \cup P_{2} = P, P_{1} \cap P_{2} = \emptyset\}.
\]

By Lemma 3.4, if \( (P_{1}, P_{2}) \in \mathcal{P}, \), then either one or both of \( F_{\text{m}} [P_{1}, P_{2}] \) and \( F_{\text{m}} [P_{1}, P_{2}] \) is 0. Thus, by Lemma 3.5,

\[
\operatorname{rank}(F_{\text{m}}) + \operatorname{rank}(F_{\text{m}}) \geq 2 + \#(P_{1}, P_{2}) \in \mathcal{P} \mid F_{\text{m}} [P_{1}, P_{2}] = 0
\]

\[
+ \#(P_{1}, P_{2}) \in \mathcal{P} \mid F_{\text{m}} [P_{1}, P_{2}] = 0
\]

\[
\geq 2 + \#(P_{1}, P_{2}) \in \mathcal{P} \mid F_{\text{m}} [P_{1}, P_{2}] = 0 \quad \text{or} \quad F_{\text{m}} [P_{1}, P_{2}] = 0
\]

\[
= 2 + \# \mathcal{P}
\]

\[
= 2 + (2^{n} - 2)
\]

\[
= 2^{n}.
\]

Therefore \( \operatorname{rank}(E) + \operatorname{rank}(E) \geq 2^{n} \) since \( F_{\text{m}} \) and \( F_{\text{m}} \) are obtained from \( E_{\text{m}} \) and \( E_{\text{m}} \) respectively by elementary operations.

**Lemma 3.7.** \( \bigcup \{\delta_{\text{m}} (Q, gw) \mid \emptyset \neq P \subseteq P\} \) and \( \bigcup \{\delta_{\text{m}} (w_{g}, Q) \mid \emptyset \neq P \subseteq P\} \) are disjoint.

**Proof.** It suffices to show that \( \delta_{\text{m}} (Q, gw) \) and \( \delta_{\text{m}} (w_{g}, Q) \) are disjoint for all nonempty subsets \( P_{1} \) and \( P_{2} \) of \( P \). Since \( gw \) is live, it is minimal because \( g \) is. In order that \( \delta_{\text{m}} (Q, gw) \) have the same cardinality as \( \delta_{\text{m}} (Q, gw) \), \( \delta(q, 0) \) must be moving to the right for every state \( q \in \delta_{\text{m}} (Q, gw) \). Similarly, \( 0w_{g} \) is live and minimal. For every state \( q' \in \delta_{\text{m}} (w_{g}, Q) \), we deduce that \( \delta(0, q') \) must be moving to the left. Therefore, \( \delta_{\text{m}} (Q, gw) \) and \( \delta_{\text{m}} (w_{g}, Q) \) are disjoint.

**Theorem 3.1.** Any nondegenerate sweeping automaton denoting \( L_{n} \) has at least \( 2^{n} \) states.

**Proof.** There are two cases to consider. The first case is when \( t > 0 \). Since matrix \( E_{\text{m}} \) is derived from \( D_{\text{m}} \) by elementary row operations, we have \( \operatorname{rank}(D_{\text{m}}) \geq \operatorname{rank}(E_{\text{m}}) \). Since the rows of matrix \( D_{\text{m}} \), are indexed by states in \( \bigcup \{\delta_{\text{m}} (Q, gw) \mid \emptyset \neq P \subseteq P\} \), we have \( \# \bigcup \{\delta_{\text{m}} (Q, gw) \mid \emptyset \neq P \subseteq P\} \geq \operatorname{rank}(D_{\text{m}}) \geq \operatorname{rank}(E_{\text{m}}) \). Similarly, \( \# \bigcup \{\delta_{\text{m}} (w_{g}, Q) \mid \emptyset \neq P \subseteq P\} \geq \operatorname{rank}(D_{\text{m}}) \geq \operatorname{rank}(E_{\text{m}}) \). By Lemmas 3.6 and 3.7, the number of states is at least \( \operatorname{rank}(E_{\text{m}}) + \operatorname{rank}(E_{\text{m}}) \geq 2^{n} \).

The second case is when \( t = 0 \). Consider the processing of a string \( gw_{P} x \). Note that \( \delta_{\text{m}} (p_{1}, gw_{P} x) = \delta_{\text{m}} (p_{1}, w_{P} x) = \delta_{\text{m}} (p_{1}, x) \). That is, \( gw_{P} x \) is in \( L_{n} \) iff \( x \) is in \( L_{n} \).
Since $t = 0$, the automaton is not allowed to perform another sweep from right to left once the right endmarker is reached. It has to decide if $x$ is in $L_n$ in only one sweep from left to right starting with the state $\delta_\omega(q_1, gw_{p_1})$. Since $x$ is an arbitrary string and the smallest incomplete DFA for $L_n$ has $2^n - 1$ states, we conclude that there are at least $2^n - 1$ states in the sweeping automaton that behaves in a one-way manner from left to right. If the automaton is not degenerate, it must have at least one more state with a left-moving transition. Thus, a nondegenerate sweeping automaton has at least $2^n$ states.

In Section 2, we required that, when a sweeping automaton wants to signal acceptance, it must enter a final state while moving right on the last symbol $a_n$ before detecting the right endmarker $\rightarrow 1$. We observe that the proof of the main result does not rely on this specific requirement for acceptance. The result is still valid if we relax the acceptance criterion allowing the sweeping automaton to signal acceptance after it has detected the right endmarker.

ACKNOWLEDGMENT

The author thanks an anonymous referee for pointing out the mistakes in earlier versions of the paper and for the numerous valuable suggestions which helped improve the presentation of this paper.

REFERENCES