Theoretical Computer Science 119 (1993) 267–291
Elsevier

Partial orders on words, minimal elements of regular languages, and state complexity

Jean-Camille Birget

Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE 68588-0115, USA

Communicated by D. Perrin
Received June 1991
Revised November 1991

Abstract

Birget, J.-C., Partial orders on words, minimal elements of regular languages, and state complexity, Theoretical Computer Science 119 (1993) 267–291.

The classical partial orders on strings (prefix, suffix, subsegment, subsequence, lexical, and dictionary order) can be generalized to the case where the alphabet itself has a partial order. This was done by Higman for the subsequence order, and by Kundu for the prefix order. Higman proved that for any language L, the set MIN(L) of minimal elements in L with respect to the generalized subsequence order is finite. Kundu proved that for any regular language L, the set MIN(L) of minimal elements in L with respect to the generalized prefix order is also regular. Here we extend his result to the other orders and give upper bounds for the number of states of the finite automata recognizing MIN(L).

The main contribution of this paper, however, is the proof of *lower bounds*. The upper bounds are shown to be tight; in particular, if L is recognized by a deterministic finite automaton with n states then any deterministic (or even *nondeterministic*) finite automaton recognizing MIN(L) needs *exponentially* many states in n; here, MIN is taken with respect to a generalized prefix, suffix, or subsegment order (with a partially ordered alphabet of 4 letters, whose Hasse diagram contains just one edge) or with respect to the ordinary subsequence order.

We also give a new proof of a theorem of Sakoda and Sipser about the complementation of nondeterministic finite automata.

1. Introduction and definitions

Let Σ be a finite alphabet, and let Σ^* be the set of all finite strings (or "words") over Σ (including the empty string ε). The length of a string w is denoted $|w|$; the cardinality of a set S is denoted $|S|$. Several partial orders on Σ^* are well known, e.g., the prefix

Correspondence to: J.-C. Birget, Computer Science and Engineering, University of Nebraska, Ferguson Hall, Lincoln, NE 68588-0115, USA. Email: birget@cse.unl.edu.

order, the suffix order, the subsegment order, the subsequence order, the lexical order, and the dictionary order; in the latter two, one assumes that a total order on the alphabet Σ has been given beforehand.

Sukhamay Kundu [8] introduced a *generalized prefix order* on Σ^*, assuming that a *partial order* has been put on the alphabet Σ beforehand:

Definition 1.1 (Kundu [8]). Let $(\Sigma, \leqslant)$ be a partially ordered finite alphabet. The *generalized prefix order* on Σ^* is defined as follows: two words $u = a_1 \dots a_h$ and $v = b_1 \dots b_k$ (with $a_1, \dots, a_h, b_1, \dots, b_k \in \Sigma$) are ordered $u \leqslant v$ if and only if $h \leqslant k$ and $a_1 \leqslant b_1, \dots, a_h \leqslant b_h$.

Note that if $(\Sigma, \leqslant)$ is just the equality relation, the generalized prefix order reduces to the ordinary prefix order.

Definition 1.2. For any partial order $(\Sigma^*, \leqslant)$ and any set $L \subseteq \Sigma^*$, we define $\mathrm{MIN}(L) = \{w \in L \mid \text{there is no } x \in L \text{ such that } x < w\}$ (the set of *minimal elements* of L).

Kundu proves [8] that if $L \subseteq \Sigma^*$ is a *regular* language then $\mathrm{MIN}(L)$ is also regular (where MIN is taken with respect to the generalized prefix order determined by a partially ordered alphabet $(\Sigma, \leqslant)$). In his proof, if L has a deterministic finite automaton with n states then $\mathrm{MIN}(L)$ has a deterministic finite automaton with $\leqslant n(4^{n-1} + 3)$ states; he also proves an order-n^2 lower bound. He asks whether there exists some exponential lower bound. In this paper we will give a slightly improved upper bound (namely, $1 + (n-1)2^{n-2}$), and we will show that this new bound is optimal, by constructing examples for which the upper bound is reached; the construction and analysis of these examples constitutes the main part of this paper. We will also show that even a *nondeterministic* finite automaton needs exponentially many states to recognize $\mathrm{MIN}(L)$.

Kundu's idea of generalizing the prefix order can also be applied to other classical orders. We give exact definitions below; later, we will show that for these orders the MIN operator preserves regularity, and we will analyze the number of states needed to recognize $\mathrm{MIN}(L)$; we will see that, except for the generalized lexical and dictionary orders, the number of states grows exponentially when one goes from L to $\mathrm{MIN}(L)$, even if nondeterminism is allowed.

Definition 1.3. Assume that $(\Sigma, \leqslant)$ is a *partially ordered finite alphabet*. Let $u = a_1 \dots a_h$ and $v = b_1 \dots b_k$ be two words (with $a_1, \dots, a_h, b_1, \dots, b_k \in \Sigma$). Then we define the following partial orders on Σ^*:

Generalized dictionary order: $u \leqslant v$ iff either u is a prefix (in the ordinary sense) of v, or we can write $u = pa_i x$ and $v = pb_i y$, where p is the longest common prefix (in the ordinary sense) of u and v, and $a_i < b_i$ (where $i - 1$ is the length of p).

Note that if $(\Sigma, \leqslant)$ is a total order then the generalized dictionary order reduces to the *ordinary dictionary order* (and the definition is formally the same); on the other

hand, if the partial order on Σ is just the *equality* relation, the generalized dictionary order reduces to the *ordinary prefix order.*

Remark 1.4. If in this definition one replaces "u is a prefix (in the ordinary sense) of v" by "u is a generalized prefix of v", one still obtains the same generalized dictionary order.

Generalized lexical order: $u \leqslant v$ iff either $|u| < |v|$, or $|u| = |v|$ and $u \leqslant v$ with respect to the generalized dictionary order.

Generalized subsegment order: $u \leqslant v$ iff $h \leqslant k$ and there exists $0 \leqslant j \leqslant k-h$ such that $a_1 \leqslant b_{j+1}$, $a_2 \leqslant b_{j+2}, \ldots, a_{h-1} \leqslant b_{j+h-1}$, $a_h \leqslant b_{j+h}$.

If $(\Sigma, \leqslant)$ is the equality relation on Σ then this defines the ordinary subsegment (or "substring") order.

Generalized subsequence order (Higman [3]): $u \leqslant v$ iff $h \leqslant k$ and there exist h numbers $1 \leqslant i_1 < i_2 < \cdots < i_{h-1} < i_h \leqslant k$ such that $a_1 \leqslant b_{i_1}$, $a_2 \leqslant b_{i_2}, \ldots, a_{h-1} \leqslant b_{i_{h-1}}, a_h \leqslant b_{i_h}$.

Generalized suffix order is defined in a way similar to the generalized prefix order.

In this paper we use *nondeterministic finite automata* (NFA), viewed as structures $(Q, \Sigma, \cdot, q_0, F)$, where Q is the set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accept states, and $\cdot$ is the next-state relation (i.e., a function from $Q \times \Sigma$ into the power set of Q; for $q \in Q$, $a \in \Sigma$, the set of next states is $q \cdot a$); we will follow [4]. A *partial deterministic finite automaton* (DFA) is an NFA whose next-state relation is a partial function (i.e., $q \cdot a$ contains at most one element of Q). A DFA is *complete* if the next-state relation is a total function (i.e., $q \cdot a$ contains exactly one element of Q). In this paper, DFA means "partial DFA" (unless we explicitly say "complete").

2. Results about the state-complexity of MIN(L)

We first state and prove upper-bound results. We then state the lower-bound results; the constructions of rather complicated families of examples that prove these lower bounds are given in Section 3.

Theorem 2.1 (Upper bounds). *Let $(\Sigma, \leqslant)$ be a partially ordered finite alphabet and let $L \subseteq \Sigma^*$ be a regular language which is recognized by a deterministic finite automaton with n states. Assume $n > 2$ (otherwise the problem is trivial).*

(1) *With respect to the generalized prefix or suffix orders*: MIN(L) *is recognized by a deterministic finite automaton whose number of states is* $\leqslant (n-1)2^{n-2}+1$.

(2) *With respect to the generalized subsegment order*: MIN(L) *is recognized by a deterministic finite automaton whose number of states is* $\leqslant (n-2)2^{n-3}+2$.

(3) *With respect to the generalized subsequence order*: MIN(L) *is recognized by a deterministic finite automaton whose number of states is* $\leqslant (n-2)2^{n-3}+2$. *Moreover, for every L (not necessarily regular),* MIN(L) *is a finite set (by a classical theorem of Higman).*

(4) *With respect to the generalized dictionary order*: $\mathrm{MIN}(L)$ *is recognized by a deterministic finite automaton with* $\leqslant n$ *states. For the generalized lexical order*: $\mathrm{MIN}(L)$ *is recognized by a deterministic finite automaton with* $\leqslant n^2$ *states.*

Proof of Theorem 2.1. (1) *Generalized prefix order*: Let $\mathscr{A}=(Q,\Sigma,\cdot,q_0,\{f\})$ be a deterministic finite automaton recognizing L. Here "$\cdot$" denotes the next-state function. We can assume that all out-edges of the accept states of $\mathscr{A}$ have been dropped and that all accept states have been coalesced into a single accept state (which we denoted by f above). To avoid trivial cases, we assume $q_0\neq f$.

A deterministic finite automaton recognizing $\mathrm{MIN}(L)$ can be defined as follows:

Set of states: $\{\boldsymbol{f}\}\cup\{(q,S)\in Q\times 2^Q \mid q\neq f,\ q\notin S,\ f\notin S\}$; here $\boldsymbol{f}$ is a new symbol. *Start state*: $(q_0,\emptyset)$. *Accept states*: $\{\boldsymbol{f}\}$.

Next-state function:

First, given a state (q,S) and a letter $a\in\Sigma$, let us define

$$S^{q,a}=\{q\cdot b\in Q \mid b\in\Sigma,\ b<a\}\cup\{p\cdot c\in Q \mid p\in S,\ c\in\Sigma,\ c\leqslant a\}.$$

The next state, reached from (q,S) on input a is

$$(q,S)\cdot a=\begin{cases}(q\cdot a,S^{q,a}) & \text{if } f\neq q\cdot a,\ q\cdot a\notin S^{q,a},\ f\notin S^{q,a}, \text{ and } q\cdot a\neq\emptyset;\\ \boldsymbol{f} & \text{if } q\cdot a=f \text{ and } f\notin S^{q,a};\\ \emptyset & \text{otherwise (i.e., } q\cdot a\in S^{q,a} \text{ or } f\in S^{q,a} \text{ or } q\cdot a=\emptyset).\end{cases}$$

We also define $\boldsymbol{f}\cdot a=\emptyset$ for all $a\in\Sigma$.

The number of states of this automaton is $1+(n-1)2^{n-2}$, where $n=|Q|$. One can check that the language recognized is $\mathrm{MIN}(L)$ with respect to the generalized prefix order. Intuitively, the automaton works as follows: Suppose that, starting in state $(q_0,\emptyset)$ and reading an input string $w\in\Sigma^*$, the state (q,S) has been reached; then $q_0\cdot w=q$ (with respect to the automaton $\mathscr{A}$); and S contains all those elements of Q that can be reached from q_0 when $\mathscr{A}$ reads any string $u\in\Sigma^*$ satisfying $|u|=|w|$ and $u<w$ (strict generalized prefix order). The definition of $S^{q,a}$ and of $(q,S)\cdot a$ has been devised accordingly. If, after reading w, we have reached a state (q,S) with $f\in S$ but $f\neq q$, then w (and any continuation of w) cannot be minimal in L because some strict generalized prefix of w is in L; similarly, if $q\in S$ but $q\neq f$ then w cannot be minimal in L because some strict generalized prefix of w leads to the same state in $\mathscr{A}$ as w. On the other hand, if w is such that $q=f\notin S$ then w belongs to $\mathrm{MIN}(L)$.

Generalized suffix order: Let $\mathscr{A}=(Q,\Sigma,\cdot,q_0,F)$ be a deterministic finite automaton recognizing L. We can assume that all in-edges of the start state have been dropped. We assume $q_0\notin F$, otherwise the problem is trivial.

A deterministic finite automaton for $\mathrm{MIN}(L)$ (with respect to the generalized subsegment order) is defined as follows:

Set of *states*: $\{(q_0,\emptyset)\}\cup\{(q,T)\in Q\times 2^Q \mid q_0\neq q,\ q\notin T,\ q_0\in T\}$. *Start state*: $(q_0,\emptyset)$. *Accept states*: $\{(f,T)\in Q\times 2^Q \mid F\cap T=\emptyset,\ q_0\in T, f\in F\}$.

Next-state function: First, given a state (q, S) and a letter $a \in \Sigma$, let us define

$$S_{q,a} = \{q_0\} \cup \{q \cdot b \in Q \mid b \in \Sigma,\ b < a\} \cup \{p \cdot c \in Q \mid p \in S,\ c \in \Sigma,\ c \leqslant a\}.$$

The next state reached from (q, S) on input $a \in \Sigma$ is

$$(q, S) \cdot a = \begin{cases} (q \cdot a, S_{q,a}) & \text{if } q \cdot a \notin S_{q,a} \text{ and } q \cdot a \neq \emptyset; \\ \emptyset & \text{otherwise (i.e., if } q \cdot a \in S_{q,a} \text{ or } q \cdot a = \emptyset). \end{cases}$$

The number of states of this automaton is $1 + (n-1)2^{n-2}$, where $n = |Q|$; one can check that it recognizes MIN(L). The intuition is like in the case of the generalized prefix order; the main difference is that $S_{q,a}$ is defined to contain q_0 always.

(2) *Generalized subsegment order*: Let $\mathscr{A} = (Q, \Sigma, \cdot, q_0, \{f\})$ be a deterministic finite automaton recognizing L. We can assume that all in-edges of the start state, and all out-edges of accept states have been dropped, and that all accept states have been coalesced into a single state f. We assume $q_0 \neq f$, otherwise the problem is trivial.

A deterministic finite automaton for MIN(L) (with respect to the generalized subsegment order) is defined as follows (where $\boldsymbol{f}$ is a new symbol):

Set of *states*: $\{\boldsymbol{f}\} \cup \{(q_0, \emptyset)\} \cup \{(q, T) \in Q \times 2^Q \mid q_0 \neq q \neq f,\ q \notin T,\ q_0 \in T,\ f \notin T\}$. *Start state*: $(q_0, \emptyset)$. *Accept states*: $\{\boldsymbol{f}\}$.

Next-state function: First, given a state (q, S) and a letter $a \in \Sigma$, let us define

$$S_{q,a} = \{q_0\} \cup \{q \cdot b \in Q \mid b \in \Sigma,\ b < a\} \cup \{p \cdot c \in Q \mid p \in S,\ c \in \Sigma,\ c \leqslant a\}.$$

The next state reached from (q, S) on input $a \in \Sigma$ is

$$(q, S) \cdot a = \begin{cases} (q \cdot a, S_{q,a}) & \text{if } f \neq q \cdot a,\ q \cdot a \notin S_{q,a},\ f \notin S_{q,a}, \text{ and } q \cdot a \neq \emptyset; \\ \boldsymbol{f} & \text{if } f = q \cdot a, \text{ and } f \notin S_{q,a}; \\ \emptyset & \text{otherwise (i.e., if } q \cdot a \in S_{q,a} \text{ or } f \in S_{q,a} \text{ or } q \cdot a = \emptyset). \end{cases}$$

We also define $\boldsymbol{f} \cdot a = \emptyset$ for all $a \in \Sigma$.

The number of states of this automaton is $2 + (n-2)2^{n-3}$, where $n = |Q|$; one can check that it recognizes MIN(L). The intuition is like in the case of the generalized prefix and suffix orders.

(3) *Generalized subsequence order*: That MIN(L) is always a finite set (no matter what L is) follows from the following famous theorem of Higman [3]: *With respect to the ordinary subsequence order, Σ^* contains no infinite subset of pairwise incomparable elements.* Higman's theorem (1952) has been rediscovered many times; see [5] for an overview; see also [7, pp. 105–109] (where a short proof is given).

MIN(L), being finite, has a finite automaton, of course; but, in general, this automaton cannot be effectively constructed from a description of L (indeed, $L = \emptyset$ iff MIN(L) $= \emptyset$; so, if L is described by a Turing machine, the emptiness problem of recursively enumerable languages would become decidable).

Let us construct a deterministic finite automaton recognizing MIN(L) when L is regular. We will obtain an exponential upper bound on the number of states of this automaton; in Section 3 we will also show that this upper bound is tight.

Let $\mathscr{A}$ be a deterministic finite automaton recognizing L, as in the proof of Theorem 2.1(2). A deterministic finite automaton for MIN(L) (with respect to the generalized subsequence order) is obtained as follows (where f is a new symbol):

Set of *states*: $\{f\} \cup \{(q_0, \emptyset)\} \cup \{(q, T) \in Q \times 2^Q \mid q_0 \neq q \neq f,\ q \notin T,\ q_0 \in T,\ f \notin T\}$. *Start state*: $(q_0, \emptyset)$. *Accept states*: $\{f\}$.

Next-state function: First, given a state (q, S), and $a \in \Sigma$, we define

$$S_{q,a} = \{q\} \cup S \cup \{q \cdot b \in Q \mid b \in \Sigma,\ b < a\} \cup \{p \cdot c \in Q \mid p \in S,\ c \in \Sigma,\ c \leqslant a\}.$$

(For the ordinary subsequence order $S_{q,a}$ becomes $\{q\} \cup S \cup \{p \cdot a \in Q \mid p \in S\}$.)
The next state reached from (q, S) on input $a \in \Sigma$ is:

$$(q, S) \cdot a = \begin{cases} (q \cdot a, S_{q,a}) & \text{if } f \neq q \cdot a,\ q \cdot a \notin S_{q,a},\ f \notin S_{q,a}, \text{ and } q \cdot a \neq \emptyset; \\ f & \text{if } f = q \cdot a,\ f \notin S_{q,a}; \\ \emptyset & \text{otherwise (i.e., } f \in S_{q,a} \text{ or } q \cdot a \in S_{q,a} \text{ or } q \cdot a = \emptyset). \end{cases}$$

We also define $f \cdot a = \emptyset$ for all $a \in \Sigma$.

This automaton has $2 + (n-2)2^{n-3}$ states and recognizes MIN(L). Intuitively, suppose that from $(q_0, \emptyset)$, reading $w \in \Sigma^*$, one reaches (q, S); then $q = q_0 \cdot w$ (with respect to $\mathscr{A}$), and S contains those elements of Q that $\mathscr{A}$ can reach from q_0 by reading a *strict* generalized subsequence of w.

(4) *Generalized dictionary order*: Let $\mathscr{A}$ be as in the proof of Theorem 2.1(1); in addition, let us assume that from every state of $\mathscr{A}$ one can reach the accept state. We construct a deterministic finite automaton for MIN(L) as follows:

The state set Q, start state q_0, accept state f, are the same as for $\mathscr{A}$.
The next-state function $\cdot$ is defined by

$$q \cdot a = \begin{cases} q \cdot a & \text{if for all } b \in \Sigma \text{ such that } b < a \text{ we have: } q \cdot b = \emptyset \text{ (in } \mathscr{A}); \\ \emptyset & \text{otherwise (i.e., there exists } b \in \Sigma \text{ such that } b < a, \\ & \text{and } q \cdot b \neq \emptyset). \end{cases}$$

Intuitively, this automaton processes a word $w \in \Sigma^*$ the same way as $\mathscr{A}$, until it finds a letter a for which there exists $b \in \Sigma$ such that $b < a$, and $q \cdot b$ is defined (recall that we are always using partial automata); if this happens, the new automaton rejects (by having no next state); in this case the input indeed cannot be minimal in L (recall that we assumed that from any state one can reach an accept state).

We mentioned already that if $(\Sigma, \leqslant)$ is just the equality relation, the generalized dictionary order becomes the ordinary prefix order; hence MIN(L) can be infinite. When $(\Sigma, \leqslant)$ is a total order, the generalized dictionary order is the ordinary dictionary order; in that case MIN(L) contains either exactly one element or is empty (the latter is possible, e.g., when $\Sigma = \{a, b\}$ with $a < b$ and $L = a^* b$).

Generalized lexical order: If L is recognized by a 1DFA $\mathscr{A}$ with n states, then $\mathrm{MIN}_{\mathrm{lex}}(L) = \mathrm{MIN}_{\mathrm{dict}}(L \cap \Sigma^m)$, where m ($\leqslant n$) is the length of the shortest directed path from the start state of $\mathscr{A}$ to any accept state; $L \cap \Sigma^m$ has a 1DFA with $\leqslant nm$ states (using the well-known cartesian-product construction), and applying $\mathrm{MIN}_{\mathrm{dict}}$ does

not increase the number of states (as we just saw). This yields a 1DFA of $nm \leqslant n^2$ states for $\mathrm{MIN}_{\mathrm{lex}}(L)$.

From the state-complexity point of view, the generalized dictionary and lexical orders do not lead to problems about exponential lower bounds, so we will not study them any further in this paper.

This completes the proof of Theorem 2.1. □

Theorem 2.2 (Lower bounds). (1) *The upper bounds of Theorem* 2.1(1)–(3) *are reachable even if nondeterministic automata are used.*

More precisely: For every $n \geqslant 3$ *there exists a partially ordered alphabet* $(\Sigma_n, \leqslant)$ *of size* $O(n^3)$, *and a regular language* $L_n \subseteq \Sigma_n^*$ *such that:* L_n *is recognized by a deterministic finite automaton with n states, but any deterministic or nondeterministic finite automaton recognizing* $\mathrm{MIN}(L_n)$ *needs* $(n-1)2^{n-2}+1$ *states, in the case of the generalized* prefix *or* suffix *orders (or* $(n-2)2^{n-3}+2$ *states in the case of the generalized* subsegment *order).*

For the generalized subsequence *order there exists an alphabet* Σ_n *of size* $< n^2$, *and a language* L_n *such that any nondeterministic automaton recognizing* $\mathrm{MIN}(L_n)$ *needs* $(n-2)2^{n-3}+2$ *states (even for the ordinary subsequence order).*

(2) *Exponential lower bounds with a fixed small alphabet: There exists a partially ordered alphabet* $(\Sigma, \leqslant)$ *with just four letters, such that for all* $n \geqslant 3$ *there is a language* $L_n \subseteq \Sigma^*$ *satisfying:* L_n *is recognized by a deterministic finite automaton with n states, but any nondeterministic finite automaton recognizing* $\mathrm{MIN}(L_n)$ *(with respect to the generalized prefix, suffix, or subsegment orders) needs* $\geqslant c^{\sqrt{n}}$ *states (for some constant* $c > 1$*); the Hasse diagram of the partial order* $(\Sigma, \leqslant)$ *has just one edge.*

For the generalized subsequence order, any nondeterministic finite automaton for $\mathrm{MIN}(L_n)$ *needs* $\geqslant c^n$ *states (for some constant* $c > 1$*); the alphabet* Σ *has size* 2 *and need not be ordered at all (ordinary subsequence order).*

The proof of Theorem 2.2 is given in the following section.

Corollary 2.3. *Suppose L is recognized by a* 1*AFA (one-way* alternating *finite automaton) with n states. Then* $\mathrm{MIN}(L)$ *can be recognized by a* 1*AFA with* $\leqslant 2^n+n-2$ *states, in the case of the generalized prefix or suffix orders (*$\leqslant 2^n+n-3$ *states in the case of the generalized subsegment or subsequence orders).*

Moreover, these bounds are optimal for all n (i.e., they can be reached).

This corollary is surprising: Intuitively, the MIN operation is just a combination of nondeterminism and negation (see the results in Examples 3.10 and 3.15 in the next section); so one would expect alternation to handle $\mathrm{MIN}(L)$ using few states, even if L has an n-state AFA.

Proof of Corollary 2.3. See [2] for a definition of alternating finite automata. We will use the following theorem of Leiss, Brzozowski and Kozen: A language L is

recognized by a 1AFA with $\leqslant n$ states *iff* L^{rev} (the reverse of L) is recognized by a 1DFA with $\leqslant 2^n$ states (see [6, 1] for other consequences of this theorem). Moreover, for any partially ordered alphabet $(\Sigma, \leqslant)$ one easily checks that $\mathrm{MIN}(L^{\mathrm{rev}}) = (\mathrm{MIN}(L))^{\mathrm{rev}}$, with respect to the generalized subsegment or subsequence orders; similarly, $\mathrm{MIN}_{\mathrm{pref}}(L^{\mathrm{rev}}) = (\mathrm{MIN}_{\mathrm{suff}}(L))^{\mathrm{rev}}$. Now applying Theorems 2.1 and 2.2, one obtains the upper (and lower) bound $\lceil \log_2((2^n-1)2^{2^n-2}+1)\rceil = 2^n+n-2$ in the prefix and suffix case (and $\lceil \log_2((2^n-2)2^{2^n-3}+2)\rceil = 2^n+n-3$ in the subsegment and subsequence case).

3. Lower bounds, examples

Example 3.1 (*Reachability of the upper bounds for the state-complexity of* $\mathrm{MIN}(L)$ *for the generalized prefix, suffix, and subsegment orders*). Construction of a partially ordered alphabet $(\Sigma_n, \leqslant)$ and a regular language $L_n \subseteq \Sigma_n^*$ such that:

(a) L_n is recognized by a deterministic finite automaton with n states.

(b) Any deterministic, and also any nondeterministic, finite automaton recognizing $\mathrm{MIN}(L_n)$, with respect to the generalized prefix or suffix (or subsegment) orders, requires $\geqslant 1+(n-1)2^{n-2}$ (or $\geqslant 2+(n-2)2^{n-3}$) states. So the upper bounds (Theorem 2.1) are reached, even if one allows nondeterminism.

The alphabet Σ_n in this example has cardinality $O(n^2(n+1)^n)$. In Example 3.9 we modify Example 3.1 in such a way that properties (a) and (b) still hold, but the alphabet has cardinality $O(n^3)$.

Finally, in Examples 3.10 and 3.15, Example 3.9 is further modified to prove Theorem 2.2(2), for the generalized prefix, suffix, and subsegment orders. The subsequence order is studied in Example 3.18.

The idea for our example was inspired from the graph languages of Sakoda and Sipser [9]. Here, in addition, we put a partial order on the graph alphabet.

In the following, n is any integer greater than 2. When f is a partial function $\{1,\dots,n\} \to \{1,\dots,n\}$ then f^{-1} denotes the inverse of f (f^{-1} is not a partial function, in general); $\mathrm{Dom}(f)$ is the domain of f.

Alphabet: A letter of the alphabet Σ_n is of the form $(f^{-1}, (f(x), x), \gamma_k)$ or $(f, (x, f(x)), \gamma_k)$, where f is any partial function $\{1,\dots,n\} \to \{1,\dots,n\}$; $(f(x), x)$ (or $(x, f(x))$) with $x \in \mathrm{Dom}(f)$, is a singled-out edge; γ_k (with $0 \leqslant k \leqslant n$) is the cyclic permutation of $\{1,\dots,n\}$, defined by: $\gamma_k(x) = x+k$, for each $x \in \{1,\dots,n-k\}$, and $\gamma_k(x) = x+k-n$ for $x \in \{n-k+1,\dots,n\}$. So, Σ_n has $< 2n^2(n+1)^n$ elements.

In the definition of Σ_n we used partial functions and their inverses; more generally, we could have used arbitrary binary relations R on $\{1,\dots,n\}$, and get letters of the form $(R, (x, y), \gamma_k)$, with $(x, y) \in R \subseteq \{1,\dots,n\} \times \{1,\dots,n\}$, and γ_k as before.

The definition of the partial order, in general, is: $(R_1, (x_1, y_1), \gamma_j) \leqslant (R_2, (x_2, y_2), \gamma_k)$ iff $R_1 = R_2$, $j = k$, $\gamma_k^{-1}(x_1) \leqslant \gamma_k^{-1}(x_2)$, $\gamma_k^{-1}(y_1) \leqslant \gamma_k^{-1}(y_2)$.

The definition of $(R,(x,y),\gamma_k)$ and the partial order become clearer in the following *graphical representation* (see Fig. 1 for an example).

A letter $(R,(x,y),\gamma_k)$ is represented by drawing the directed bipartite graph of R (with edges pointing from left to right), in which the edge (x,y) is drawn in bold; if $k=0$ (i.e., $\gamma_k=$identity map) the vertices are labeled from 1 to n in each column, *starting at the bottom*; if $k\neq 0$, we label the vertices from $k+1$ $(=\gamma_k(1))$ to n, and continue from 1 to k $(=\gamma_k(n))$, *starting at the bottom*.

Two letters are ordered (according to the above definition) iff they have the same underlying graph $(R_1=R_2)$, the same labeling of the vertices $(j=k)$, and the marked edge (x_1,y_1) of the first letter appears entirely below the marked edge (x_2,y_2) of the second letter (except for possible common vertices).

To define the language L_n, we first define the *graph of a word* $w\in\Sigma_n^+$: it is the graph obtained by putting the graphs of the letters of w in the sequence in which they appear in w, and connecting equally labeled vertices in adjacent letters. See Fig. 2 for a small example. Thus, the graph of w will have $2n|w|$ vertices.

We define $L_n=\{w\in\Sigma_n^+ \mid$ the graph of w has a path which starts at the vertex labeled **1** in the left-most column, which reaches the vertex labeled $\boldsymbol{n}$ in the right-most column, and which consists only of *marked* edges and inter-letter connections; this path should not encounter a vertex labeled $\boldsymbol{n}$ until the right end is reached$\}$.

See Fig. 2. This definition is inspired from the graph languages of [9]. The last sentence in the definition of L_n implies that L_n is prefix-free (with respect to the ordinary prefix order).

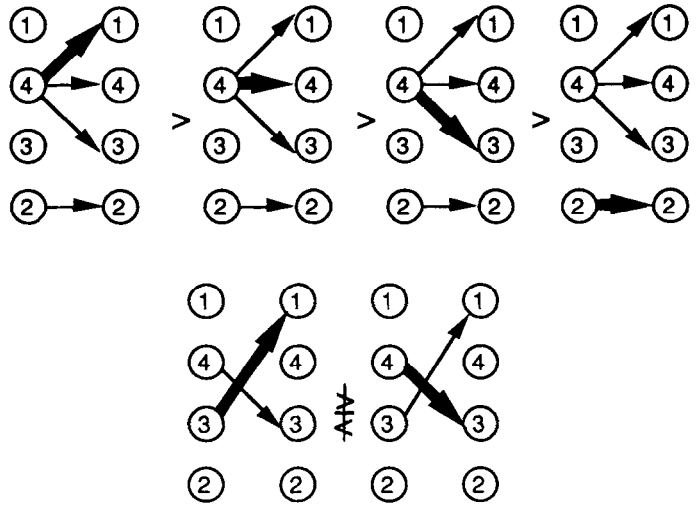

Fig. 1. Some letters in Σ_4 with their order. The first letter is $(f^{-1},(4,1),\gamma_1)$, where f^{-1} is given by $4\to 1$, $4\to 4$, $4\to 3$, $2\to 2$. In the second letter, the marked edge is $(4,4)$. Since we have $\gamma_1^{-1}(4)=3\leqslant\gamma_1^{-1}(1)=4$, the first letter is indeed higher in the order than the second letter. The last two letters are incomparable.

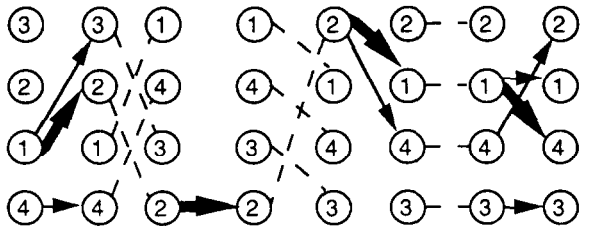

Fig. 2. A word $w \in L_4$, of length 4. Note that $w \notin \mathrm{Min}(L_4)$ (because of the path $1 \to 2 \to 2 \to 4$ in a generalized prefix of w), with respect to the generalized prefix order.

It is very easy to see that L_n has an n-state minimum deterministic automaton (the states are the vertex labels $1, 2, \ldots, n$, with 1 as the start state and n as the sole accept state). Recall also that we use *partial* automata throughout this paper.

The fact that any nondeterministic finite automaton for $\mathrm{MIN}(L_n)$ has $\geqslant 1+(n-1)2^{n-2}$ states (in the case of the generalized prefix or suffix orders) or $\geqslant 2+(n-2)2^{n-3}$ states (in the case of the generalized subsegment order) follows from the Facts 3.2, 3.5 and 3.6. To prove Facts 3.2 and 3.5 (deterministic case) only the letters of the form $(f^{-1}, (f(x), x), \gamma_k)$ play a role; the letters $(f, (x, (f(x)), \gamma_k)$ will be used to prove lower bounds for nondeterministic automata (Fact 3.6).

Fact 3.2. *The deterministic automaton for* $\mathrm{MIN}(L_n)$, *with respect to the generalized prefix order (constructed from the above n-state automaton for* L_n, *in the proof of Theorem* 2.1(1)) *is minimal. The generalized suffix order can be handled similarly.*

Proof. We shall show that (a) every state is reachable from the start state $(1, \emptyset)$; (b) from every state one can reach the accept state $\boldsymbol{f}$; (c) every two states are distinguishable by some input string.

(a) Reachability from $(1, \emptyset)$: For any state (q, S) we have $(q, S) = (1, \emptyset) \cdot (f^{-1}, (1, q), \gamma_q)$, where f is the partial function defined by $f(x) = 1$ if $x \in \{q\} \cup S$ (and $f(x)$ is undefined, otherwise). See Fig. 3(a) for an example.

(b) $\boldsymbol{f}$ is reachable from all states: For any state (q, S) we have $\boldsymbol{f} = (q, S) \cdot (g^{-1}, (q, n), \gamma_q)$, where g is the partial function defined by $g(n) = q$ (and $g(x)$ is undefined for $x \neq n$). See Fig. 3(b) for an example.

(c) Distinguishability of all states: We consider the following cases.

Case 1: To distinguish the state $\boldsymbol{f}$ from any state (q, S), consider the input letter $a = (g^{-1}, (q, n), \gamma_q)$ defined a few lines ago. Then $\boldsymbol{f} \cdot a$ is undefined, but $(q, S) \cdot a = \boldsymbol{f}$.

Case 2: To distinguish two states (q, S), (p, T) where $p \neq q$, consider again the input letter $a = (g^{-1}, (q, n), \gamma_q)$, defined a few lines ago. Then $(p, T) \cdot a$ is undefined, but $(q, S) \cdot a = \boldsymbol{f}$. See Fig. 4 for an example.

Case 3: To distinguish two states (q, S), (q, T) where $S \neq T$, assume there is an element r which belongs to S but not to T. (If instead, there exists $r \in T - S$, the proof is similar.) Consider the input string (of length 2): $ab = (f_1^{-1}, (q, q), \gamma_q)(f_2^{-1}, (q, n), \gamma_q)$,

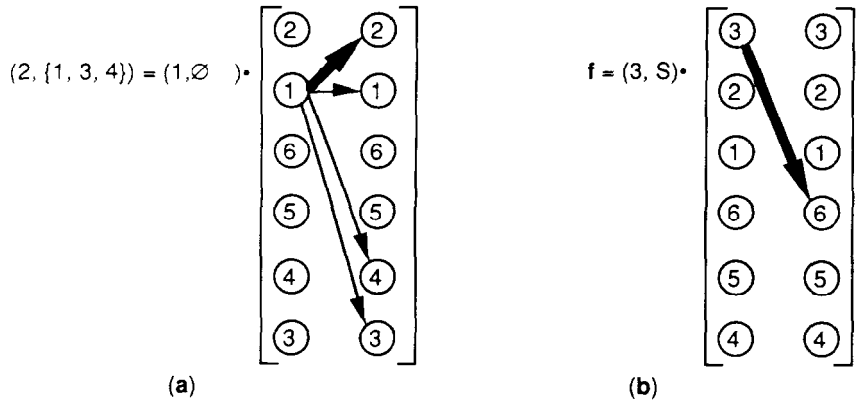

Fig. 3. Here $n=6$. (a) Reachability of $(2,\{1,3,4\})$ from $(1,\emptyset)$. (b) Reachability of f from $(3,S)$, where $S\subseteq\{1,\ldots,n-1\}-\{3\}$.

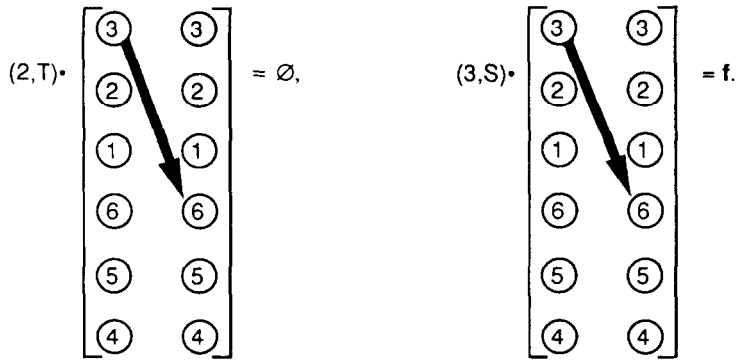

Fig. 4. Distinguishing $(2,T)$ from $(3,S)$; here $n=6$, $q=3$, $p=2$.

where the partial functions f_1 and f_2 are defined as follows: $f_1(q)=q$, $f_1(n)=r$ (this is well defined, since $q\neq n$; indeed, since (q,S) is a state we have $q\neq n$), and $f_1(x)$ is undefined for $x\notin\{q,n\}$; $f_2(n)=q$, and $f_2(x)$ is undefined for $x\neq n$. Then $(q,S)\cdot a$ and, hence, $(q,S)\cdot ab$, is undefined; but $(q,T)\cdot ab=(q,\emptyset)\cdot b=f$. See Fig. 5 for an example.

This proves that the minimum deterministic automaton for $\mathrm{MIN}(L_n)$ is the automaton of the proof of Theorem 2.1(1) and, thus, has $1+(n-1)2^{n-2}$ states.

Remark 3.3. If we had simply defined our alphabet to be $\{(f^{-1},(f(x),x)),(f,(x,f(x)))\mid f$ is a partial function $\{1,\ldots,n\}\to\{1,\ldots,n\}$, and $x\in\mathrm{Dom}\,f\}$ (equivalently, we would just take the subset of Σ_n obtained when γ_k is always replaced by γ_0) then the minimum deterministic automaton for $\mathrm{MIN}(L_n)$ would only need the states $\{f\}\cup\{T\subseteq\{1,\ldots,n-1\}\mid T\neq\emptyset\}$. Indeed, in this case we would replace the state (q,S) of the proof of Theorem 2.1(1) by $T=S\cup\{q\}$; then T uniquely determines (q,S), since

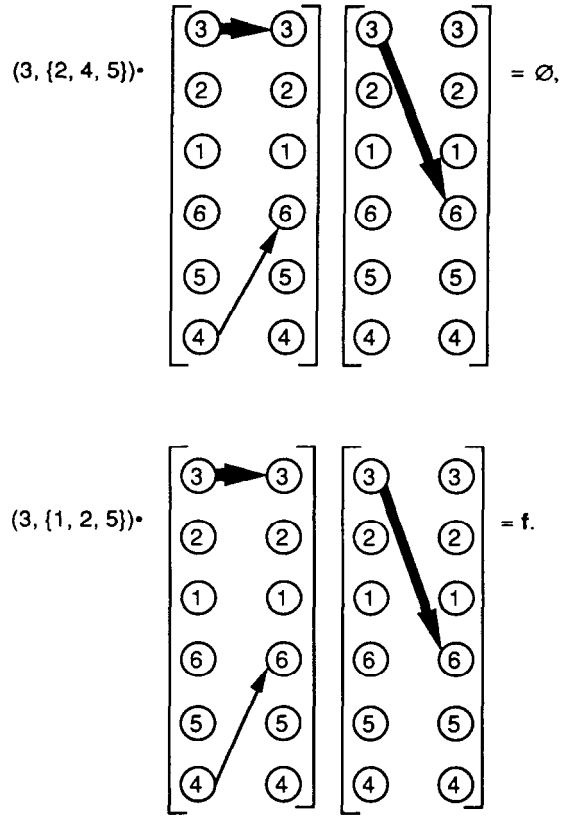

Fig. 5. Distinguishing $(3,\{2,4,5\})$ from $(3,\{1,2,5\})$. Here $n=6$, $q=3$, and $r=4\in\{2,4,5\}-\{1,2,5\}$.

(in the absence of γ_k's): $q=\min T$ and $S=T-\{q\}$. So without the γ_k's, we do not reach the upper bound, although we do obtain an exponential lower bound $1+2^{n-2}$.

Remark 3.4. If one looks at the letters of the alphabet Σ_n which are actually used to prove Fact 3.2, one sees that one only needs $O(n^2 2^n)$ letters. Indeed, the following groups of letters are used:

(a) Letters of the form $(f^{-1},(1,p),\gamma_q)$ where $q\in\{1,\ldots,n-1\}$ and f is the "constant 1" function restricted to domain S, where $q\notin S\subseteq\{1,\ldots,n-1\}$ and where $p\in S$. There are $(n-1)(n-2)2^{n-2}$ such letters.

(b) Letters of the form $(g^{-1},(q,n),\gamma_q)$ as in part (b) of the proof; here $q\neq n$. There are $n-1$ such letters.

(c) Letters of the form $(f^{-1},(q,q),\gamma_q)$, where $q\in\{1,\ldots,n-1\}$ and $f(q)=q$, $f(n)=r$ (for some $r\in\{1,\ldots,n-1\}$, $r\neq q$) and f is undefined elsewhere. There are $(n-1)(n-2)$ such letters. This corresponds to case 3 of part (c) of the above proof (cases 1 and 2 use the same letters as part (b)).

Let us consider now the generalized subsegment order. In reference to the same alphabet and the same language L_n as in Fact 3.2 we have:

Fact 3.5. *The deterministic automaton for* MIN(L_n), *with respect to the generalized subsegment order (constructed in the proof of Theorem* 2.1(2)*from the n-state automaton of* L_n) *is minimal.*

Proof. The proof scheme is similar to the one for Fact 3.2.

(a) Reachability from $(1,\emptyset)$: For any state (q,S) (with $1\neq q\neq n$, $1\in S$, $q\notin S$, $S\subseteq\{1,\ldots,n-1\}$) we have $(q,S)=(1,\emptyset)\cdot(f^{-1},(1,q),\gamma_q)$, where f is the partial function defined by $f(x)=1$ for all $x\in\{q\}\cup S$ (and $f(x)$ is undefined otherwise). This is just like in the proof of Fact 3.2, except that we only consider sets S with $1\in S$.

(b) f is reachable from all states: This is again exactly like in Fact 3.2.

(c) Distinguishability of all states. This is also exactly like in Fact 3.2.

This proves Fact 3.5. □

The previous remark (to the effect that only a subalphabet of $O(n^2 2^n)$ letters is needed) also applies here.

We now prove the lower bound for *nondeterministic* finite automata.

Fact 3.6. *Every* nondeterministic *automaton for* MIN(L_n), *with respect to the generalized prefix order* (or *the generalized suffix or subsegment orders*), *requires as many states as the minimum deterministic finite automaton.*

Proof. The proof is based on the same idea as the new proof of the Sakoda and Sipser theorem given in the Appendix.

For every state (p,S) of the deterministic automaton for MIN(L_n), we choose two words $u_{(p,S)}$ and $v_{(p,S)}$ as follows (they are actually single letters here):

$u_{(p,S)}=(f^{-1},(1,p),\gamma_p)$, where f is the partial function defined by $f(x)=1$ if $x\in S\cup\{p\}$ (and $f(x)$ is undefined otherwise);

$v_{(p,S)}=(g,(p,n),\gamma_p)$, where g is the partial function defined by $g(x)=n$ if $x\notin S$ (and $g(x)$ is undefined if $x\in S$).

We also define $u_{\mathbf{f}}=(f_0^{-1},(1,n),\gamma_n)$, where f_0 is the (total) function defined by $f_0(x)=1$ for all x; and $v_{\mathbf{f}}=\varepsilon$ (the empty string).

One can check the following (for the generalized prefix, suffix, and subsegment orders):

(1) For all states (p,S) of the deterministic automaton: $u_{(p,S)}v_{(p,S)}\in$MIN(L_n); also $u_{\mathbf{f}}v_{\mathbf{f}}\in$MIN($L_n$).

(2) For all states (p,S), (q,T) of the deterministic automaton: If $(p,S)\neq(q,T)$ then $u_{(p,S)}v_{(q,T)}\notin$MIN(L_n) *or* $u_{(q,T)}v_{(p,S)}\notin$MIN(L_n); also $u_{(p,S)}v_{\mathbf{f}}\notin$MIN($L_n$), and $u_{\mathbf{f}}v_{(p,S)}\notin$MIN($L_n$).

Indeed, if $p\neq q$ then $u_{(p,S)}v_{(q,T)}$ and $u_{(q,T)}v_{(p,S)}$ do not belong to L_n; if $p=q$ and $S\neq T$ then either $S\cap\bar{T}\neq\emptyset$ (and then $u_{(p,S)}v_{(q,T)}$ is not minimal in L_n), or $\bar{S}\cap T\neq\emptyset$ (and then $u_{(q,T)}v_{(p,S)}$ is not minimal).

Let $\mathscr{B}=(R, \Sigma_n, \cdot, r_0, \{r_f\})$ be a *nondeterministic* finite automaton recognizing $\mathrm{MIN}(L_n)$ (we may, without loss of generality, assume that $\mathscr{B}$ has only one accept state).

For each state (q, T) of the deterministic automaton, we choose a state $r_{(q,T)} \in R$ such that $r_{(q,T)} \in r_0 \cdot u_{(q,T)}$ and $r_f \in r_{(q,T)} \cdot v_{(q,T)}$ (i.e., $r_{(q,T)}$ can be reached on an accepting computation on input $u_{(q,T)}v_{(q,T)}$ after $u_{(q,T)}$ was read). We choose r_f to be r_f (the accept state of $\mathscr{B}$).

Claim 3.7. *For all states (q, T), (p, S) of the deterministic automaton of* $\mathrm{MIN}(L_n)$ *we have: if $(q, T) \neq (p, S)$ then $r_{(q,T)} \neq r_{(p,S)}$; also $r_{(q,T)} \neq r_f$. (From this claim one immediately concludes that $\mathscr{B}$ has at least as many states as the deterministic automaton.)*

Proof. Suppose, by contradiction, that $r_{(q,T)} = r_{(p,S)}$, with $(q, T) \neq (p, S)$. Then the words $u_{(q,T)}v_{(p,S)}$ and $u_{(p,S)}v_{(q,T)}$ do not both belong to $\mathrm{MIN}(L_n)$, as we saw. But $\mathscr{B}$ will accept both; the following describes an accepting computation of $\mathscr{B}$ on input $u_{(p,S)}v_{(q,T)}$: starting in state r_0, $\mathscr{B}$ reads $u_{(p,S)}$ and reaches the state $r_{(p,S)}$ $(= r_{(q,T)})$; then, by definition of $r_{(q,T)}$, $\mathscr{B}$ can reach the accept state r_f by reading $v_{(q,T)}$. In a similar way one can describe an accepting computation on $u_{(q,T)}v_{(p,S)}$.

That $r_{(q,T)} \neq r_f$ follows immediately from the definition of $r_{(q,T)}$. This proves Fact 3.6. □

Remark 3.8. In the proof of Fact 3.6 we not only use letters of the form $(f^{-1}, (f(x), x), \gamma_k)$, but also of the form $(f, (x, f(x)), \gamma_k)$. The number of letters used is $O(n^2 2^n)$ (see a previous remark; actually, the use of letters $(f, (x, f(x)), \gamma_k)$ doubles the number of letters, but we still have $O(n^2 2^n)$ valid).

Example 3.9 (*Reachability of the upper bounds of the state-complexity of* $\mathrm{MIN}(L)$ *for the generalized prefix, suffix, and subsegment orders, with alphabet of size* $O(n^3)$). Construction of a partially ordered alphabet $(\Gamma_n, \leqslant)$ and a regular language $L'_n \subseteq \Gamma_n^*$ such that properties (a) and (b) of Example 3.1 hold; in addition Γ_n has $O(n^3)$ elements only.

We start from Example 3.1 in which, however, we keep only the required $O(n^2 2^n)$ letters. There are at least two natural ways of reducing the size of an alphabet (assuming that useless letters have been discarded already): One is to encode the larger alphabet into fixed-length strings over a small alphabet (e.g. binary encoding); we will do this in Example 3.15; but because we are working with ordered alphabets, the encoding does not work as well as in other situations, and the lower bounds will not be optimal (although still exponential). In a second method, which we use in the present example, each letter of Σ_n is "*factored* as a product of generators"; Σ_n is then replaced by the (smaller) set Γ_n of generators.

Each element of Σ_n is of the form $(f^{-1}, (f(x), x), \gamma_k)$ or $(f, (x, f(x)), \gamma_k)$, where f is a partial function. One can decompose f as a composition of other partial functions. It

is well known that the set of all partial functions $\{1,\ldots,n\}\to\{1,\ldots,n\}$ can be generated by just four partial functions, e.g., by the four partial functions whose function tables are $\alpha=[2,3,\ldots,n,1]$ (cyclic shift), $\beta=[2,1,3,4,\ldots,n]$ (transposition of 1 and 2), $\gamma=[1,1,3,4,\ldots,n]$ (collapse of 1 and 2 to 1), $\delta=[-,2,3,\ldots,n]$ (identity, left undefined on 1).

We will use the following alphabet:

$$\Gamma_n=\{(f^{-1},(f(x),x),\gamma_k),\ (f,(x,f(x)),\gamma_k)\mid x\in\mathrm{Dom}\,f,\ k\in\{1,\ldots,n\},\ f\in\Phi\},$$

where γ_k denotes the cyclic k-shift ($\gamma_k(x)=x+k$ if $x\leqslant n-k$, and $\gamma_k(x)=x+k-n$ if $x>n-k$), and Φ is the following set of partial functions: $\Phi=\{u_i\mid 1\leqslant i\leqslant n\}\cup\{c\}\cup\{t_i\mid 1\leqslant i\leqslant n-1\}$. These partial functions are defined as follows: $u_i(x)=x$ if $x\neq i$ (and is undefined if $x=i$); $c(x)=1$ if $x\neq n$ (and is undefined if $x=n$); and t_i is the transposition (i,n) (defined by $t_i(i)=n$, $t_i(n)=i$, $t_i(x)=x$ for $i\neq x\neq n$). Note that u_i and t_i are injective, and $u_i=u_i^{-1}$, $t_i=t_i^{-1}$.

Clearly, Γ_n is a subset of Σ_n; we will use the partial order of Σ_n (restricted to Γ_n) as the partial order of Γ_n. The number of elements of Γ_n is $O(n^3)$.

As our language L_n' we pick $L_n\cap\Gamma_n^*$ (where L_n is the language of Example 3.1).

The same n-state automaton which recognizes L_n can also be used for L_n'; we just have to restrict it to the alphabet $\Gamma_n(\subseteq\Sigma_n)$.

To show that any deterministic finite automaton which recognizes $\mathrm{MIN}(L_n')$ needs $(n-1)2^{n-2}+1$ states in the case of the generalized prefix or suffix orders (and $(n-2)2^{n-3}+2$ states for the generalized subsegment order), we proceed as in example 3.1: Facts 3.2 and 3.5 can be proved for L_n' too, but the letters that were used must now be factored over the alphabet Γ_n.

Part (a) of the proof of Facts 3.2 and 3.5 (*reachability of* (q,S) *from the start state* $(1,\emptyset)$): We have $(1,\emptyset)\cdot(c^{-1},(1,q),\gamma_q)=(q,\{1,\ldots,q-1,q+1,\ldots,n-1\})$, and $(q,\{1,\ldots,q-1,q+1,\ldots,n-1\})\cdot\prod_{i\notin S\cup\{q\}}(u_i,(q,q),\gamma_q)=(q,S)$. See Fig. 6 for an example.

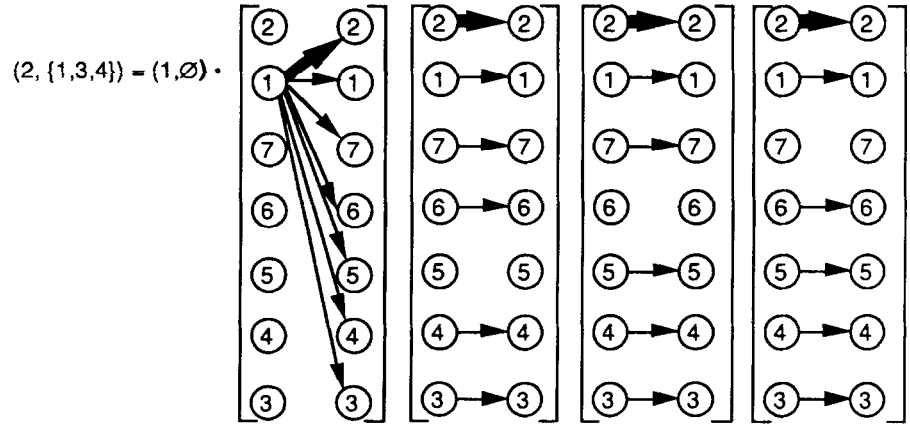

Fig. 6. Reachability of $(2,\{1,3,4\})$ from $(1,\emptyset)$; here $n=7$.

Part (b) (Reachability of f from any (q,S)): We have $(q,S)\cdot\prod_{i\in S}(u_i,(q,q),\gamma_q)=(q,\emptyset)$, and $(q,\emptyset)\cdot(t_q,(q,n),\gamma_q)=f$. See Fig. 7.

Part (c) (Distinguishability of all states): We consider the following cases.

Case 1: To distinguish (q,S) from f one uses the same input as in part (b) above.

Case 2: To distinguish (q,S) from (p,T) when $p\neq q$, one uses again the same input string $w\in\Gamma_n^*$ as in part (b) above. Then $(q,S)\cdot w=f$ and $(p,T)\cdot w=\emptyset$ (since $(p,T)\cdot(\ldots,(q,q),\ldots)=\emptyset$ when $p\neq q$).

Case 3: To distinguish (q,S) from (q,T) when $S\neq T$, assume there exists $r\in S-T$ (the proof is similar if, instead, there exists $r\in T-S$). We have $(q,T)\cdot\prod_{i\notin\{q,r\}}(u_i,(q,q),\gamma_q)=(q,\emptyset)$ since $r\notin T$; also $(q,\emptyset)\cdot(t_r,(q,q),\gamma_q)=(q,\emptyset)$; also, as we just saw in part (b): $(q,\emptyset)\cdot(t_q,(q,n),\gamma_q)=f$. On the other hand, $(q,S)\cdot\prod_{i\notin\{q,r\}}(u_i,(q,q),\gamma_q)=(q,\{r\})$, and $(q,\{r\})\cdot(t_r,(q,q),\gamma_q)=\emptyset$ (undefined). Thus, for the input string $v=\prod_{i\notin\{q,r\}}(u_i,(q,q),\gamma_q)(t_r,(q,q),\gamma_q)(t_q,(q,n),\gamma_q)$ we have $(q,S)\cdot v=f$, but $(q,T)\cdot v=\emptyset$. See Fig. 8.

Fact 3.6 is also proved like Example 3.1; now $u_{(p,S)}$ and $v_{(p,S)}$ are words obtained by factoring the partial functions appearing in the $u_{(p,S)}$ and $v_{(p,S)}$ of Example 3.1. The details are the same as for adapting Facts 3.2 and 3.5: to obtain the new $u_{(p,S)}$, proceed as in part (a); to obtain the new $v_{(p,S)}$, proceed also as in part (a), but use partial functions rather than inverses of partial functions.

Example 3.10 (*Exponential lower bound for the state-complexity of* MIN(L) *for the generalized prefix, suffix, and subsegment orders, with an alphabet of size* O(n)). Construction of a partially ordered alphabet $(\Delta_n,\leqslant)$ and a regular language $L_n''\subseteq\Delta_n^*$ such that:

(a) L_n'' is recognized by a deterministic finite automaton with n states

(b) Any *nondeterministic* finite automaton recognizing MIN(L_n''), with respect to the generalized prefix, suffix, or subsegment orders, requires $\geqslant 2^{n-3}-1$ states.

(c) The alphabet Δ_n has cardinality O(n).

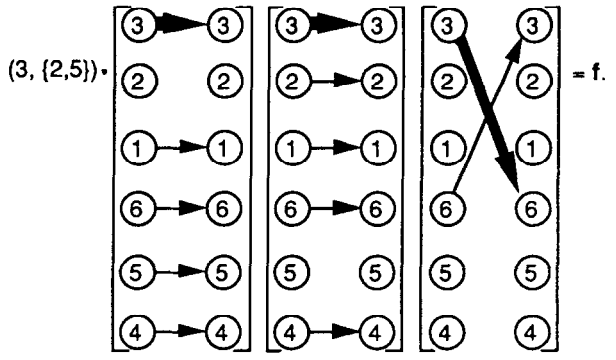

Fig. 7. Reachability of f from $(3,\{2,5\})$; here $n=6$.

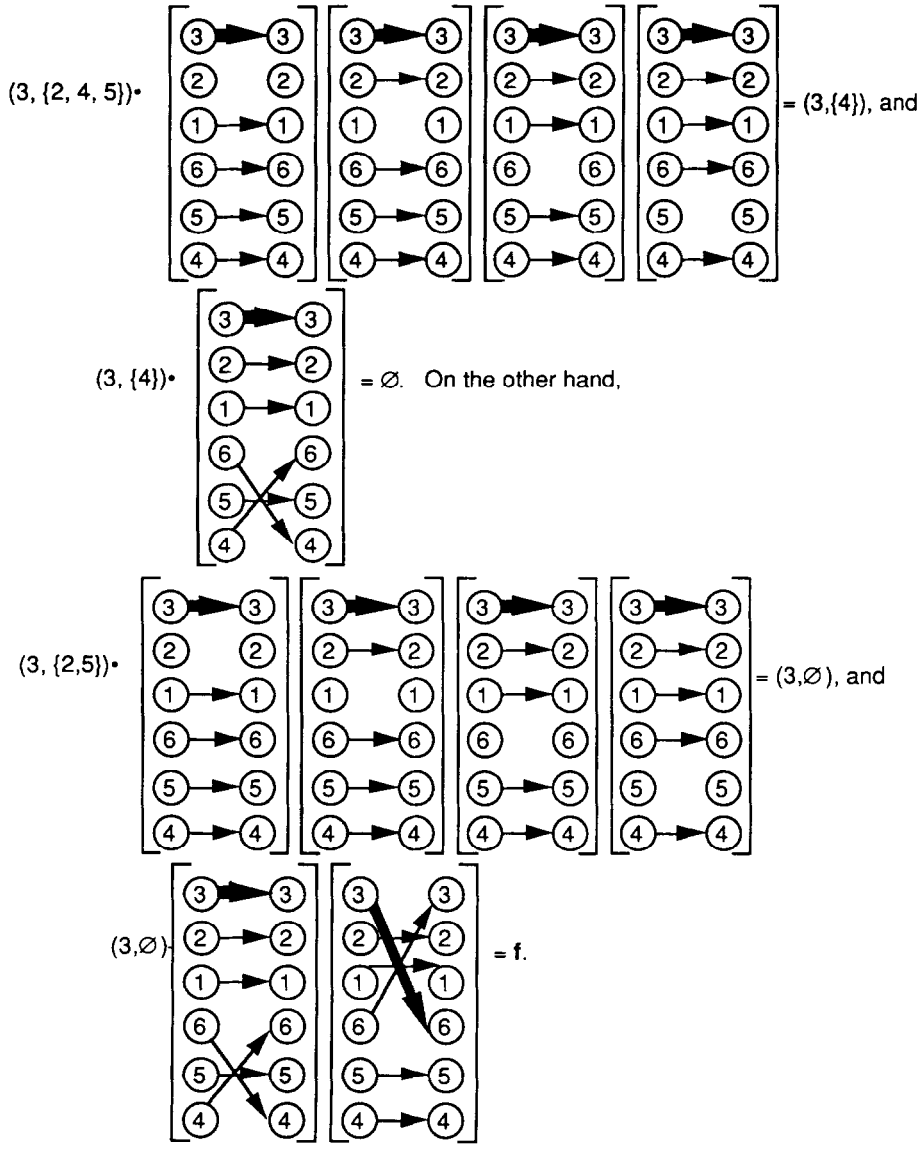

Fig. 8. Distinguishing the states $(3,\{2,4,5\})$ and $(3,\{2,5\})$. Here $r=4$, $n=6$.

This example and the proof of its properties are based on the previous examples, and on a result of Sakoda and Sipser [9].

In [9, p. 281] the following example is introduced: They consider the alphabet $I_n=\{f^{-1}\mid f$ is a partial function from $\{1,\ldots,n\}$ into $\{1,\ldots,n\}\}$, and the language $T_n=\{w\in I_n^*\mid$ in the graph of w there is a path from the extreme left column to the extreme right column$\}$. This was the inspiration of our Example 3.1 (except that in [9]

there is no marked edge and no cyclic permutation of the vertices). They prove the following result (see [9, Theorem 4.1.3 and Remarks on p. 281], and see the Appendix of the present paper for a new proof):

Theorem 3.11 (Sakoda and Sipser [9]). *The language T_n is recognized by a nondeterministic finite automaton with n states, but every nondeterministic finite automaton recognizing $\bar{T}_n$ ($=I_n^* - T_n$, the complement of T_n) must have $\geqslant 2^n$ states.*

The alphabet I_n used here has $(n+1)^n$ letters. But the semigroup of all partial functions from $\{1,\ldots,n\}$ into $\{1,\ldots,n\}$ under composition is generated by just four partial functions $\alpha_n, \beta_n, \gamma_n, \delta_n$ (as we saw at the beginning of Example 3.9); the function tables of $\alpha_n, \beta_n, \gamma_n, \delta_n$ are: $\alpha_n=[2,3,4,\ldots,n,1]$ cyclic shift; $\beta_n=[2,1,3,4,\ldots,n]$ transposition of 1 and 2; $\gamma_n=[1,1,3,4,\ldots,n]$ so $\gamma_n(1)=\gamma_n(2)=1$, $\gamma_n(x)=x$ if $x>2$; $\delta_n=[-,2,3,\ldots,n]$ so $\delta_n(x)=x$ if $x\neq 1$, $\delta_n(1)$ is undefined. So we could replace the alphabet I_n by the alphabet $J_n=\{\alpha_n^{-1},\beta_n^{-1},\gamma_n^{-1},\delta_n^{-1}\}$ (inverses of these four partial functions); we also consider $Y_n=T_n\cap(J_n)^*$ and $\bar{Y}_n=(J_n)^*-Y_n$. The theorem of Sakoda and Sipser still holds for Y_n and now we have a constant-size alphabet ($|J_n|=4$). See our Appendix for a slightly stronger result and a new proof.

To make use of this theorem we need the notion of *reduction* between regular languages.

Definition 3.12. Let $L_1\subseteq\Sigma^*$, $L_2\subseteq\Delta^*$ be languages, where Σ and Δ are finite alphabets. Then $L_1\trianglelefteq L_2$ (L_1 *reduces to* L_2) iff there exist $u,v\in\Delta^*$ and a homomorphism $\varphi:\Sigma^*\to\Delta^*$ such that $L_1=\varphi^{-1}(u^{-1}L_2v^{-1})$ (or equivalently: $x\in L_1$ iff $u\varphi(x)v\in L_2$).

Theorem 3.13. *If $L_1\trianglelefteq L_2$, and if L_2 is recognized by a nondeterministic finite automaton with n states, then L_1 is recognized by some nondeterministic finite automaton with $\leqslant n+1$ states.*

(The proof is straightforward; see e.g. [4] p. 61 for inverse homomorphisms, and pp. 62 and 63 for right-quotients; left-quotients are similar, but one might need several start states in the nondeterministic case, or one *new* start state; the proofs in [4] also work for nondeterministic automata.)

We are now ready to introduce Example 3.10.

To define the alphabet Δ_n we will use partial functions $\{1,\ldots,n-3\}\to\{1,\ldots,n-3\}$ and their generators $\alpha_{n-3}=\alpha$, $\beta_{n-3}=\beta$, $\gamma_{n-3}=\gamma$ and $\delta_{n-3}=\delta$;

$$\Delta_n=\{(f_\mu^{-1},(f_\mu(x),x))\mid \mu\in\{\alpha,\beta,\gamma,\delta\},\ x\in\mathrm{Dom}(f_\mu),\ \text{and } f_\mu \text{ is the partial function } \{1,\ldots,n\}\to\{1,\ldots,n\} \text{ defined as follows: } f_\mu(2)=2, f_\mu(1) \text{ and } f_\mu(n) \text{ are undefined, and } f(x)=\mu(x-2) \text{ for all } x\in\{3,\ldots,n-1\}\}$$

$$\cup\{(g_1^{-1},(g_1(x),x))\mid x\in\mathrm{Dom}(g_1)\}$$

$$\cup\{(g_n,(x,n))\mid 2\leqslant x\leqslant n-1\}.$$

Here g_1 and g_n are partial functions defined by: $g_1(x)=1$ for $2\leqslant x\leqslant n-1$, (and $g_1(1)$ and $g_1(n)$ are undefined); $g_n(x)=n$ for $2\leqslant x\leqslant n-1$ (and $g_n(1)$ and $g_n(n)$ are undefined). See Fig. 9.

One can check that $|\Delta_n|=O(n)$. Also, Δ_n will be considered a subset of Σ_n (Example 3.1), by identifying $(R,(x,y))$ with $(R,(x,y),\gamma_2)$. The language $L_n''\subseteq\Delta_n^*$ and the partial order on Δ_n are defined as in Examples 3.1 and 3.9. Clearly, L_n'' can be recognized by a deterministic finite automaton with n states (just like L_n).

In view of the preceding two theorems, property (b) of Example 3.10 follows from the next claim (which gives an interesting inherent connection between minimization and negation):

Claim 3.14. $\bar{Y}_{n-3}\not\subseteq \mathrm{MIN}(L_n'')$, *where* $\not\subseteq$ *denotes reduction, as defined above*; MIN *is taken with respect to the generalized prefix, suffix, or subsegment orders. For the generalized prefix and suffix orders we actually have* $\bar{Y}_{n-2}\not\subseteq \mathrm{MIN}(L_n'')$.

Proof (*see also Fig.* 9). In the notation of the Definition of *reduction*, we let: $\Sigma=J_{n-3}$; $\Delta=\Delta_n$; u is the letter $(g_1^{-1},(1,2))$; v is the letter $(g_n,(2,n))$; finally, $\varphi:\Sigma^*\to\Delta^*$ is the length-preserving homomophism defined by: for every $\mu^{-1}\in J_{n-3}=\{\alpha^{-1},\beta^{-1},\gamma^{-1},\delta^{-1}\}$ we let $\varphi(\mu^{-1})=(f_\mu^{-1},(2,2))\in\Delta_n$. See Fig. 9 for an example.

Then for every $w\in(J_{n-3})^*$ we have: $w\in\bar{Y}_{n-3}$ iff $u\varphi(w)v\in\mathrm{MIN}(L_n'')$. Indeed, the marked path $1\to2\to2\to\cdots\to2\to2\to n$ in $u\varphi(w)v$ is minimal iff *no* other path in $u\varphi(w)v$

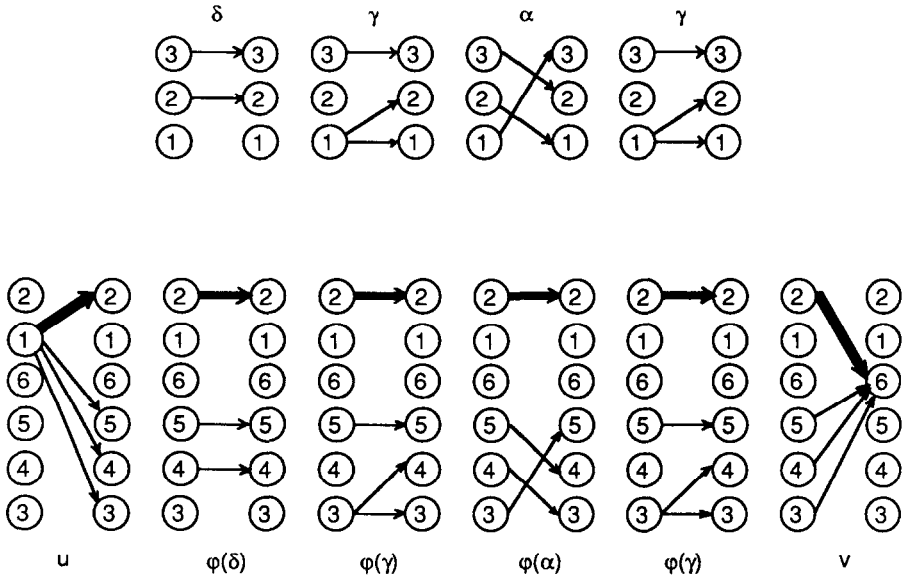

Fig. 9. A word $w=\delta\gamma\alpha\gamma\in(J_3)^*$, and the corresponding word $u\varphi(w)v\in\Delta_6^*$; here $w\notin Y_3$, and $u\varphi(w)v\in\mathrm{MIN}(L_6'')$.

goes all the way through; by the choice of u and v, this holds iff $w \notin Y_{n-3}$. This proves the claim. □

Finally, by Theorem 3.11 any nondeterministic automaton for $\bar{Y}_{n-3}$ needs $\geqslant 2^{n-3}-1$ states; and thus, by Theorem 3.13, this must also be true for $\mathrm{MIN}(L_n'')$.

Note that for the generalized prefix order we could identify vertices 1 and 2; so then $\bar{Y}_{n-2} \trianglelefteq \mathrm{MIN}(L_n'')$.

Example 3.15 (*Exponential lower bounds for the generalized prefix, suffix, and subsegment orders, using a fixed alphabet of size* 4). Construction of a regular language C_n over the partially ordered four-letter alphabet $(\{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\},\leqslant)$, such that:

(a) C_n is recognized by a deterministic finite automaton with $O(n^2)$ states.

(b) Any *nondeterministic* finite automaton recognizing $\mathrm{MIN}(C_n)$, with respect to the generalized prefix, suffix, or subsegment orders, requires $\geqslant 2^{n-3}-1$ states; so the relation between the number of states for C_n and $\mathrm{MIN}(C_n)$ is a function $\geqslant c^{\sqrt{n}}$, for some constant $c>1$.

(c) The partially ordered alphabet $(\{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\},\leqslant)$ does not depend on n, and the Hasse diagram of its partial order contains only one edge (namely, "$\mathbf{0}<\mathbf{1}$").

Example 3.15 will be based on Example 3.10 (where we had an alphabet Δ_n with $O(n)$ letters); we encode the alphabet Δ_n into strings over the alphabet $\{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\}$, as follows:

The letter $(f^{-1},(f(j),j)) \in \Delta_n$ (with $f \in \{f_\alpha, f_\beta, f_\gamma, f_\delta, g_1\}$) is *encoded* into the string $\mathrm{CODE}(f^{-1},(f(j),j)) = f^{\mathrm{code}}\mathbf{0}^{n-f(j)+1}\mathbf{1}^{f(j)}\mathbf{0}^{n-j+1}\mathbf{1}^{j}$, where f^{code} is defined as follows:

$$(f_\alpha)^{\mathrm{code}} = \boldsymbol{aaa},\ (f_\beta)^{\mathrm{code}} = \boldsymbol{aab},\ (f_\gamma)^{\mathrm{code}} = \boldsymbol{aba},\ (f_\delta)^{\mathrm{code}} = \boldsymbol{abb},\ (g_1)^{\mathrm{code}} = \boldsymbol{baa}.$$

The letter $(g_n,(j,n)) \in \Delta_n$ is encoded as $\mathrm{CODE}(g_n,(j,n)) = \boldsymbol{bab}\mathbf{0}^{n-j+1}\mathbf{1}^{j}\mathbf{0}\mathbf{1}^{n}$. A word $w = a_1 \ldots a_k \in \Delta_n^*$ is encoded into $\mathrm{CODE}(w) = \mathrm{CODE}(a_1)\mathrm{CODE}(a_2)\ldots\mathrm{CODE}(a_k)$ (concatenation of the encodings of the letters).

The language C_n is obtained by encoding the language L_n'' (of Example 3.10), as above, word for word, i.e., $C_n = \mathrm{CODE}(L_n'') = \{\mathrm{CODE}(w) \mid w \in L_n''\}$.

Claim 3.16. *C_n is recognized by a deterministic finite automaton with $O(n^2)$ states.*

The proof is elementary, and we omit it.

In order to prove property (b), we will reduce the language $\bar{Y}_{n-3}$ (introduced in Example 3.10) to $\mathrm{MIN}(C_n)$.

Claim 3.17. *$\bar{Y}_{n-3} \trianglelefteq \mathrm{MIN}(C_n)$, where $\trianglelefteq$ denotes the reduction introduced by Sakoda and Sipser.*

Proof. Here (referring to the definition of $\trianglelefteq$) we pick $\Sigma = J_{n\ 3} = \{\alpha^{-1}, \beta^{-1}\gamma^{-1}, \delta^{-1}\}$ (see Example 3.10); $\Delta = \{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\}$; the length-preserving homomorphism ψ is defined

as follows $\psi(\alpha^{-1})=\mathrm{CODE}(f_\alpha^{-1},(2,2))$, $\psi(\beta^{-1})=\mathrm{CODE}(f_\beta^{-1},(2,2))$, $\psi(\gamma^{-1})=\mathrm{CODE}(f_\gamma^{-1},(2,2))$, and $\psi(\delta^{-1})=\mathrm{CODE}(f_\delta^{-1},(2,2))$; finally, $u=\mathrm{CODE}(g_1^{-1},(1,2))$, $v=\mathrm{CODE}(g_n,(2,n))$.

We must show that $w\in\bar{Y}_{n-3}$ iff $u\psi(w)v\in\mathrm{MIN}(C_n)$. We know from Example 3.10 that $w\in\bar{Y}_{n-3}$ if and only if $(g_1^{-1},(1,2))\varphi(w)(g_n,(2,n))\in\mathrm{MIN}(L_n'')$. Also, the function $\mathrm{CODE}:\Delta_n\to\{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\}^*$ is *injective* and *doubly order-preserving*; i.e., $(R_1,(x_1,y_1))\leqslant(R_2,(x_2,y_2))$ in $(\Delta_n,\leqslant)$ *iff* $\mathrm{CODE}(R_1,(x_1,y_1))\leqslant\mathrm{CODE}(R_2,(x_2,y_2))$ with respect to either of the generalized prefix and the subsegment orders on $\{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\}^*$. This follows from the facts that (1) there is no order relation between $\boldsymbol{a}$ and $\boldsymbol{b}$, (2) we have $\mathbf{0}<\mathbf{1}$, and (3) the marked edges of Δ_n are coded in unary notation over $\{\mathbf{0},\mathbf{1},\boldsymbol{a},\boldsymbol{b}\}^*$. From this we conclude that $(g_1^{-1},(1,2))\varphi(w)(g_n,(2,n))\in\mathrm{MIN}(L_n'')$ iff $\mathrm{CODE}(g_1^{-1},(1,2))\varphi(w)(g_n,(2,n)))\in\mathrm{MIN}(C_n)$. This proves Claim 3.17.

From Claim 3.17 and Theorems 3.11 and 3.13 it follows that any nondeterministic finite automaton recognizing $\mathrm{MIN}(C_n)$ must have at least $2^{n-3}-1$ states.

Example 3.18 (*Reachability of the upper bound for the subsequence order*). Construction of an alphabet K_n (of size $\leqslant n^2$) and a regular language $L_n\subseteq K_n^*$ such that

(1) L_n is recognized by a deterministic finite automaton with n states;

(2) every *nondeterministic* finite automaton recognizing $\mathrm{MIN}(L_n)$, with respect to the *ordinary subsequence order*, requires $\geqslant(n-2)2^{n-3}+2$ states; so the upper bound in Theorem 2.1(3) is reached (even if non-determinism is used).

We assume $n>2$.

We take the alphabet $K_n=\{(i,j)\mid i,j\in\{1,\ldots,n\}\text{ and }i\neq j\}$, and we define our language $L_n\subseteq K_n^*$ by $L_n=\{w\in K_n^*\mid$ the first letter in w is of the form $(1,j)$ for some j, the last letter of w is of the form (i,n) for some i, and any two neighboring letters of w are of the form $(i,j)(j,k)$, for some $i,j,k\}$.

One can check easily that, with respect to the ordinary subsequence order, we have: $\mathrm{MIN}(L_n)=\{(1,i_1)(i_1,i_2)(i_2,i_3)\ldots(i_{r-1},i_r)(i_r,n)\in K_n^+\mid r\geqslant 1$, and in the sequence $1,i_1,i_2,i_3,\ldots,i_{r-1},i_r,n$ no integer occurs more than once$\}$.

Graphically, K_n is the complete directed graph on n vertices without loops: L_n is the set of all *walks* from 1 to n in this complete graph; $\mathrm{MIN}(L_n)$ is the set of all *paths* from 1 to n (walks without repeated vertex) in this complete graph.

It is easy to see that L_n can be recognized by a deterministic finite automaton (DFA) with n states. To establish the state-complexity of $\mathrm{MIN}(L_n)$, we first show that the DFA constructed in the proof of Theorem 2.1(3) is minimal; then we show that the same lower bound also holds for all NFAs (*non*deterministic finite automata) recognizing $\mathrm{MIN}(L_n)$.

Fact 3.19. *The DFA for* $\mathrm{MIN}(L_n)$, *constructed in the proof of Theorem* 2.1(3), *is the minimum automaton of* $\mathrm{MIN}(L_n)$.

Proof. Notation $\boldsymbol{n}=\{1,\ldots,n\}$. The *states* of this DFA (see the proof of Theorem 2.1(3)) for our example are $\{f\}\cup\{(1,\emptyset)\}\cup\{(i,T)\in\boldsymbol{n}\times 2^{\boldsymbol{n}}\mid 1\neq i\neq n,\ i\notin T,\ 1\in T,\ n\notin T\}$.

The start state is $(1,\emptyset)$, and the sole accept state is f. The next-state function in this example (with input letter $(j,k)\notin K_n$) becomes: $(i,T)\cdot(j,k)=\emptyset$ if $i\neq j$; and

$$(i,T)\cdot(i,k)=\begin{cases}(k,\{i\}\cup T) & \text{if } n\neq k \text{ and } k\notin T,\\ f & \text{if } n=k,\\ \emptyset & \text{if } k\in T.\end{cases}$$

We must show that in this DFA (1) every state (as defined above) is reachable from $(1,\emptyset)$, (2) from every state one can reach f, (3) every two states are distinguishable.

(1) The state (i,T), with $T=\{j_1,\ldots,j_r\}$ (r distinct states), is reachable from $(1,\emptyset)$ by reading the input string $(1,j_1)(j_1,j_2)(j_2,j_3)\ldots(j_{r-1},j_r)$.

(2) From the state (i,T) one reaches f by reading the input (i,n).

(3) Any state (i,T) is distinguishable from f, because in state f no next state is defined. Two states (i,T), (k,U) with $i\neq k$ are distinguished by applying the input (i,n): $(i,T)\cdot(i,n)=f$, but $(k,U)\cdot(i,n)=\emptyset$. Two states (i,T), (i,U) with $T\neq U$ are distinguished as follows: Assume $p\in T-U$ (the proof would be similar if there exists $p\in U-T$), and consider the input $w=(i,p)(p,n)\in K_n^*$. Then $(i,T)\cdot(i,p)=\emptyset$, since $p\in T$, and so $(i,T)\cdot w=\emptyset$. On the other hand, $(i,U)\cdot(i,p)=(p,\{i\}\cup U)$ since $p\notin U$ (and also $p\neq n$, because $p\in T$, and (i,T) is a state); thus $(i,U)\cdot w=(p,\{i\}\cup U)\cdot(p,n)=f$. This proves Fact 3.19.

Fact 3.20. *Every NFA for* $\mathrm{MIN}(L_n)$ *needs as many states as the minimum DFA for* $\mathrm{MIN}(L_n)$.

Proof. We apply the same idea as in the new proof of the theorem of Sakoda and Sipser (see Appendix). For every state (q,T) of the minimum DFA of $\mathrm{MIN}(L_n)$ (described in the proof of Fact 3.19 above), we choose two words $u_{(q,T)}$, $v_{(q,T)}\in K_n^*$ as follows: Let $T=\{p_1=1,p_2,\ldots,p_k\}$ with $p_i\neq p_j$ for $i\neq j$; then we define $u_{(q,T)}=(1,p_2)(p_2,p_3)\ldots(p_{k-1},p_k)(p_k,q)$. Let $\bar{T}=\boldsymbol{n}-T=\{p_{k+1}=q,p_{k+2},\ldots,p_{n-1},p_n=n\}$ with $p_i\neq p_j$ for $i\neq j$; then we define $v_{(q,T)}=(q,p_{k+2})(p_{k+2},p_{k+3})\ldots(p_{n-2},p_{n-1})(p_{n-1},n)$. (In the definitions of $u_{(q,T)}$ and $v_{(q,T)}$, it does not matter how the elements of T and $\bar{T}$ are ordered, except that in $u_{(q,T)}$ we must start with 1, and in $v_{(q,T)}$ we must start with q and end with n.) As a special case we have $u_{(1,\emptyset)}=\varepsilon$ (the empty word), and $v_{(1,\emptyset)}=(1,2)(2,3)\ldots(n-2,n-1)(n-1,n)$ (since here $\bar{T}=\boldsymbol{n}$). We also define, for the state f: $u_f=(1,2)(2,3)\ldots(n-2,n-1)(n-1,n)$, and $v_f=\varepsilon$.

From these definitions, and the description of $\mathrm{MIN}(L_n)$ one concludes that:

(1) $u_f v_f\in\mathrm{MIN}(L_n)$, and $u_{(q,T)}v_{(q,T)}\in\mathrm{MIN}(L_n)$, for every state (q,T) of the DFA.

(2) If $(p,S)\neq(q,T)$ then $u_{(p,S)}v_{(q,T)}\notin\mathrm{MIN}(L_n)$ or $u_{(q,T)}v_{(p,S)}\notin\mathrm{MIN}(L_n)$; also $u_{(p,S)}v_f\notin\mathrm{MIN}(L_n)$, and $u_f v_{(p,S)}\notin\mathrm{MIN}(L_n)$ (for all states (p,S), (q,T) of the DFA). See the proof of Fact 3.6 of Example 3.1.

Let $\mathscr{A}=(R,\Sigma,\cdot,r_0,\{r_f\})$ be a NFA recognizing $\mathrm{MIN}(L_n)$ (we may assume, without loss of generality, that $\mathscr{A}$ has a single accept state).

For each state (q, T) of the DFA, we choose a state $r_{(q,T)} \in R$ such that $r_{(q,T)} \in r_0 \cdot u_{(q,T)}$ and $r_f \in r_{(q,T)} \cdot v_{(q,T)}$. For $(1, \emptyset)$ we have then $r_{(1,\emptyset)} = r_0$; and for the state f of the DFA we have $r_f = r_f$. We assume now that a fixed choice of $r_{(q,T)}$ has been made for very (q, T).

Claim 3.21. *For all states* (q, T), (p, S) *of the* DFA: *if* $(q, T) \neq (p, S)$ *then* $r_{(q,T)} \neq r_{(p,S)}$; *also* $r_{(q,T)} \neq r_f$. (From this claim one immediately concludes that the NFA $\mathcal{A}$ has at least as many states as the DFA recognizing MIN(L_n).)

For a proof of the claim see Fact 3.6 of Example 3.1.

Remark 3.22. One could easily encode our alphabet K_n over a 2-letter alphabet, so that the encoded language L_n^{coded} has a DFA with $O(n)$ states, whereas every NFA for MIN(L_n^{coded}) requires $> c^n$ states (for some constant $c > 1$).

Appendix: a new proof of a theorem of Sakoda and Sipser about the complementation of nondeterministic finite automata

We give a new (and probably simpler) proof of the following result [9, p. 281, Theorem 4.1.3] (slightly improved here regarding the alphabet size):

Theorem A.1. *For every* $n > 1$ *there exists a language* L_n *over a three-letter alphabet* Σ *such that:*

(1) L_n *is recognized by a NFA with n states (in fact, more strongly, the reverse* L_n^{rev} *of* L_n *is recognized by a DFA with n states, with a single accept state).*

(2) *Any NFA recognizing* $\bar{L}_n (= \Sigma^* - L_n)$ *requires* $\geqslant 2^n$ *states.*

Our language L_n will be almost the same as the language T_n of Sakoda and Sipser; the only modifications are: (1) We want the alphabet to consist of 3 letters (rather than $(n+1)^n$), (2) we want L_n^{rev} to be recognized by a DFA (this is almost true in [9]).

For this, we use the three generators α, β, γ, of the semigroup of all total functions $\{1, \ldots, n\} \to \{1, \ldots, n\}$; $\alpha = [2, 3, 4, \ldots, n-1, n, 1]$ (cyclic shift), $\beta = [2, 1, 3, 4, \ldots, n]$ (transposition of 1 and 2), $\gamma = [1, 1, 3, 4, \ldots, n]$ (i.e., $\gamma(1) = \gamma(2) = 1$, $\gamma(x) = x$ if $x > 2$). Similar to Example 3.10 of this paper, we pick our alphabet to be $\Sigma = \{\alpha^{-1}, \beta^{-1}, \gamma^{-1}\}$ (i.e., the inverses of the three generators). In analogy with the definition of T_n, we pick our language to be $L_n = \{w \in \Sigma^* \mid$ in the graph of w there is a path, starting at vertex **1** in the extreme left column, and ending in vertex **2** in the extreme right column$\}$.

Clearly, L_n^{rev} is recognized by a DFA with n states (in fact, the minimum DFA has n states), with a single accept state. One can also check that the minimum DFA for L_n has 2^n states (see [1, Section 2]).

So far we have closely followed [9]. To show that any NFA recognising $\bar{L}_n$ requires $\geqslant 2^n$ states, we will use a new argument.

For every set $S \subseteq \{1, \ldots, n\}$ we choose two words $u_S, v_S \in \Sigma^*$ as follows.

For u_S: The word u_S is chosen in such a way that S consists of the labels of those vertices in the extreme right column of u_S that are reachable by a path in the graph of u_S, starting at **1** in the extreme left column.

For v_S: The word v_S is chosen in such a way that S consists of the labels of those vertices in the extreme left colume of v_S from which **2** in the extreme right column is *not* reachable.

In other words, the relations on $\{1, \ldots, n\}$ determined by the graphs of u_S and v_S are such that the image of **1** under u_S is S and the inverse image of **2** under v_S is $\bar{S}$.

From these properties of u_S and v_S we immediately conclude (for all $S, Z \subseteq \{1, \ldots, n\}$):

(1) $u_S v_S \in \bar{L}_n$,

(2) if $S \neq Z$ then $u_S v_Z \in L_n$ or $u_Z v_S \in L_n$ (since $S \neq Z$ iff $(S \cap \bar{Z}) \cup (\bar{S} \cap Z) \neq \emptyset$).

Let $\mathcal{A} = (Q, \Sigma, \cdot, p_0, F)$ be a 1NFA recognizing $\bar{L}_n$. For each set S (once we have fixed u_S and v_S) we choose a state $q_S \in Q$ such that $q_S \in p_0 \cdot u_S$ and $q_S \cdot v_S \cap F \neq \emptyset$ (i.e., q_S is a state on an accepting computation on $u_S v_S$, that is reached after u_S was read). We now assume that a fixed choice of q_S has been made for each set S.

Claim A.2. *For all sets $S, Z \subseteq \{1, \ldots, n\}$ we have: if $S \neq Z$ then $q_S \neq q_Z$.*

From the claim, one immediately concludes that the 1NFA $\mathcal{A}$ has $\geqslant 2^n$ states, since there are 2^n subsets in $\{1, \ldots, n\}$.

Proof. Suppose, by contradiction, that $q_S = q_Z$, with $S \neq Z$. Consider the words $u_S v_Z$ and $u_Z v_S$, of which (as we noted above) at least one belongs to L_n; so they should not, both, be accepted by $\mathcal{A}$. Nevertheless, the following describes an accepting computation on input $u_S v_Z$: $\mathcal{A}$ starts in state p_0 and, by reading u_S it can reach the state $q_S = q_Z$; next, by the definition of q_Z, $\mathcal{A}$ can reach an accept state ($\in F$) from q_Z by reading v_Z. In a similar way, $\mathcal{A}$ also accepts the word $u_Z v_S$. So $\mathcal{A}$ accepts both words, which contradicts the fact that at least one of them is not in $\bar{L}_n$.

Acknowledgment

I would like to thank my colleagues David Klarner and Stuart Margolis for pointing out references [3, 5, 7].

References

[1] J.C. Birget, State-complexity of finite-state devices, state compressibility and incompressibility, *Math. Systems Theory* (1993), to appear.

[2] A. Chandra, D. Kozen and L. Stockmeyer, Alternation, *J. ACM* **28** (1981) 114–133.

[3] G. Higman, Ordering by divisibility in abstract algebras, *Proc. London Math. Soc.* **2** (1952) 326–336.
[4] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).
[5] J.B. Kruskal, The theory of well quasi ordering: a frequently discovered concept. *J. Combin. Theory, Ser. A*, **13** (1972) 297–305.
[6] E. Leiss, Succinct representation of regular languages by boolean automata, *Theoret. Comput. Sci.* **13** (1981) 323–330.
[7] M. Lothaire, *Combinatorics on Words* (Addison-Wesley, Reading, MA, 1983).
[8] S. Kundu, The minimal strings in a regular language with respect to a partial order on the alphabet, *Theoret. Comput. Sci.* **83** (1991) 287–300.
[9] W. Sakoda and M. Sipser, Non-determinism and the size of two-way finite automata, *Proc. 10th ACM Symp. on Theory of Computing* (1978) 275–286.