Theoretical Computer Science 410 (2009) 2961–2971

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Nondeterministic state complexity of nested word automata

Yo-Sub Han [a], Kai Salomaa [b,*]

[a] Department of Computer Science, Yonsei University, Seoul 120-749, Republic of Korea
[b] School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

ARTICLE INFO

Keywords:
State complexity
Nondeterminism
Finite automata on nested words
Language operations

ABSTRACT

We study the nondeterministic state complexity of Boolean operations on regular languages of nested words. For union and intersection we obtain matching upper and lower bounds. For complementation of a nondeterministic nested word automaton with n states we establish a lower bound $\Omega(\sqrt{n!})$ that is significantly worse than the exponential lower bound for ordinary nondeterministic finite automata (NFA). We develop techniques to prove lower bounds for the size of nondeterministic nested word automata that extend the known techniques used for NFAs.

1. Introduction

Nested words and finite automata on nested words have been introduced by Alur and Madhusudan [2] as a natural computation model for applications like XML document processing, where data has a dual linear-hierarchical structure. More information on applications of nested word automata can be found e.g. in [3,4]. The symbols of a nested word are ordered in the usual way linearly, and additionally there is a recursively defined matching of occurrences of special call and return symbols. Trees are often used as models of XML, and it can be noted that trees can be viewed as special cases of nested words [3]. Finite automata on nested words are an attractive computational model because the class of recognized languages, the regular nested word languages, retains many of the nice closure and decision properties of ordinary regular languages [2,3]. When viewed as sets of ordinary words the regular nested word languages are, in general, not regular and form a subclass of the deterministic context-free languages recognized by visibly pushdown automata [1,3,4].

Alur and Madhusudan [2,3] have established that a nondeterministic nested word automaton (NNWA) can be determinized and, interestingly, the state space blow-up is worse than in the case of ordinary finite automata. The operational state complexity of deterministic nested word automata (DNWA) has been considered in [14]. Lower bounds for operations such as catenation or reversal are of a different order than the known results for the state complexity of ordinary deterministic finite automata (DFA) [17,18].

The state complexity of basic operations on nondeterministic finite automata (NFA) has been systematically studied by Holzer and Kutrib [9], and more references on descriptional complexity of NFAs can be found in [16]. Here we study the worst-case size of NNWAs for nested word languages obtained by applying Boolean operations to regular nested word languages, and discuss upper bounds for catenation and Kleene star (extended to nested words).

When constructing an NNWA for the union of two nested word languages, the automaton can, in a certain sense, reuse the set of hierarchical states as long as it uses disjoint sets of linear states for the two computations. This makes it more difficult to establish a tight lower bound for union. The NNWA constructed to recognize the intersection of two languages can use the cross products of the sets of, respectively, linear and of hierarchical states.

Complementation is a "hard" operation for nondeterministic automata and can be expected to cause a large state space blow-up. Even in the case of ordinary NFAs it took some effort to establish a tight lower bound for complementation using a

* Corresponding author. Tel.: +1 613 533 6073; fax: +1 613 533 6513.
E-mail addresses: emmous@cs.yonsei.ac.kr (Y.-S. Han), ksalomaa@cs.queensu.ca (K. Salomaa).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.01.004

binary alphabet [12]. Here we show that there exist regular nested word languages L_n recognized by an NNWA with n states such that any NNWA recognizing the complement of L_n needs $\Omega(\sqrt{n!})$ states. This is worse than the exponential lower bound for complementation of NFAs, however, it does not coincide with the upper bound and the precise nondeterministic state complexity of complementation remains open.

For Kleene star of regular nested word languages we establish an upper bound that improves on the one obtained directly from the construction showing closure under this operation [3]. Finding an optimal lower bound for Kleene star remains open.

When considering state complexity of NNWAs, the roles played by linear and hierarchical states, respectively, are different. For example, in worst-case examples for the construction for union, hierarchical states can be reused in both "parts" of the automaton but the same does not hold for linear states. Thus, when we want precise results for nondeterministic state complexity it is appropriate to obtain the bounds separately for linear and hierarchical states. Naturally, it can be argued that a more complete descriptional complexity measure for nondeterministic automata would include the number of transitions [6,16], however, we consider here only state complexity.

In order to prove lower bounds for nondetermistic state complexity we first extend to nested word languages some techniques introduced in [5,7,10,11] for NFAs. As is the case already when considering NFAs, the lower bound techniques cannot be expected to always provide optimal results [8]. It is well known that a minimal NFA need not be unique, and in the case of nested words, even a minimal deterministic automaton need not be unique [1,14]. For example, when considering union, our general lower bound technique provides a bound that is close to the upper bound but in order to get a matching lower bound we need to rely on ad hoc arguments.

2. Preliminaries

We assume that the reader is familiar with formal languages and, in particular, with finite automata and state complexity, see [15,17,18]. Here we very briefly recall the definition of nested words. For more details on nested words, including motivating examples, the reader is referred to [2,3].

Let Σ be a finite alphabet. The *tagged alphabet* corresponding to Σ is $\hat{\Sigma} = \Sigma \cup \langle\Sigma \cup \Sigma\rangle$, where $\langle\Sigma = \{\langle a \mid a \in \Sigma\}$ is the set of *call symbols* and $\Sigma\rangle = \{a\rangle \mid a \in \Sigma\}$ is the set of *return symbols.* Elements of Σ are called *internal symbols.* A *tagged word* over Σ is a sequence of symbols of the tagged alphabet $\hat{\Sigma}$, $w = u_1 \cdots u_m$, $u_i \in \hat{\Sigma}$, $i = 1, \ldots, m$. We define recursively a hierarchical matching relation in a tagged word. For w as above, a call symbol $u_i \in \langle\Sigma$ *matches* a return symbol $u_j \in \Sigma\rangle$, $i < j$, if in the subsequence $u_{i+1} \cdots u_{j-1}$ every call symbol (respectively, return symbol) has a matching return symbol (respectively, call symbol). Symbol occurrences $u_i \in \langle\Sigma$ that do not have a matching return, $1 \leq i \leq m$, are referred to as *pending calls*, and $u_i \in \Sigma\rangle$ that does not have a matching call is a *pending return.* The above conditions define a unique matching relation between call symbol occurrences and return symbol occurrences in any tagged word,

By a *nested word* we mean a tagged word together with the usual linear ordering of symbols and the hierarchical matching relation between occurrences of call and return symbols. The set of nested words over Σ is denoted $\mathrm{NW}(\Sigma)$. A nested word language is any subset of $\mathrm{NW}(\Sigma)$. A nested word is *well-matched* if every call symbol has a matching return and every return symbol has a matching call. An example of a nested word is $ab\rangle a\langle caa\langle dc\rangle ad\rangle ab\rangle a\langle b$. Here all occurrences of a are linear, the call-symbol $\langle c$ (respectively, $\langle d$) matches return symbol $d\rangle$ (respectively, $c\rangle$), both occurences of $b\rangle$ are pending returns and $\langle b$ is a pending call. The word is not well-matched since it has pending calls and/or returns. An example of a well-matched nested word is $a\langle b\langle bab\rangle\langle caa\langle dc\rangle ad\rangle aab\rangle$.

Often it is convenient to view a nested word $u_1 \cdots u_m$ as a directed graph where there is a linear edge from u_i to u_{i+1}, $i = 1, \ldots, m-1$, and additionally each pair of matching call and return symbols is connected by a hierarchical edge. Furthermore, there is an incoming hierarchical edge (with no specified source node) to each pending return and an outgoing hierarchical edge (with no target node) from each pending call.

For $w \in \mathrm{NW}(\Sigma)$ and $b \in \hat{\Sigma}$, $|w|_b$ denotes the number of occurrences of b in the nested word w. For $w \in \mathrm{NW}(\Sigma)$, we denote by $\mathrm{base}(w)$ the *underlying word of* w over the tagged alphabet $\hat{\Sigma}$, that is, $\mathrm{base}(w) \in \hat{\Sigma}^*$ is an ordinary word. Note that $\mathrm{base}(w)$ is a word over the alphabet $\Sigma \cup \langle\Sigma \cup \Sigma\rangle$, that is, certain symbol occurrences are marked as call and return symbols, respectively, but there are no hierarchical edges between the call and return symbols.

Next, we recall the definition of nondeterministic nested word automata from [3]. This definition explicitly distinguishes the linear states that the computation passes following the linear ordering of the symbols and the hierarchical states that are passed along hierarchical edges. The distinction will be useful for obtaining precise bounds for state complexity. A simplified definition has been used in [2].

Definition 2.1. A *nondeterministic nested word automaton,* NNWA, is a tuple $A = (\Sigma, Q, Q_0, Q_f, P, P_0, P_f, \delta_c, \delta_i, \delta_r)$, where Σ is the input alphabet, Q is the finite set of linear states, $Q_0 \subseteq Q$ is the set of initial linear states, $Q_f \subseteq Q$ is the set of final linear states, P is the finite set of hierarchical states, $Q \cap P = \emptyset$, $P_0 \subseteq P$ is the set of initial hierarchical states, $P_f \subseteq P$ is the set of final hierarchical states, $\delta_c : Q \times \langle\Sigma \longrightarrow 2^{Q \times P}$ is the call transition function, $\delta_i : Q \times \Sigma \longrightarrow 2^Q$ is the internal transition function, and $\delta_r : Q \times P \times \Sigma\rangle \longrightarrow 2^Q$ is the return transition function.

Consider a nested word $w = u_1 \cdots u_m$, $u_i \in \hat{\Sigma}$, $i = 1, \ldots, m$ that has k occurrences of call symbols and let A be as in Definition 2.1. A *computation* of A on w consists of a sequence of linear states $q_i \in Q$, $i = 0, \ldots, m$, and a sequence

of hierarchical states $p_j \in P$, $j = 1, \ldots, k$, corresponding to call symbol occurrences in w, such that $q_0 \in Q_0$, and for $i \in \{1, \ldots, m\}$, the following holds:

(i) If $u_i \in \Sigma$ is an internal symbol, then $q_i \in \delta_i(q_{i-1}, u_i)$.
(ii) If $u_i \in \langle\Sigma$ is the jth call symbol occurrence, $1 \leq j \leq k$, then $(q_i, p_j) \in \delta_c(q_{i-1}, u_i)$.
(iii) Assume that $u_i \in \Sigma\rangle$ is a return symbol occurrence. If u_i is matched with the j_ith call symbol occurrence, $1 \leq j_i \leq k$, then $q_i \in \delta_r(q_{i-1}, p_{j_i}, u_i)$. If u_i is a pending return, then $q_i \in \delta_r(q_{i-1}, p_0, u_i)$ for some $p_0 \in P_0$.

Intuitively, A begins a nondeterministic computation in some initial linear state $q_0 \in Q_0$. It reads an internal symbol using the internal transition function similarly as an ordinary NFA. When encountering a call symbol $\langle a$ in a linear state q, A sends along the linear edge a state $q' \in Q$ and along the hierarchical edge a state $p' \in P$ where $(q', p') \in \delta_c(q, \langle a)$ is nondeterministically chosen. When A encounters a return-symbol $a\rangle$ in a linear state state q and receives state $p \in P$ along the hierarchical edge, the computation continues in some linear state of $\delta_r(q, p, a\rangle)$. If $a\rangle$ is a pending return, A uses an arbitrary initial hierarchical state $p_0 \in P_0$ as the second argument for δ_r.

The *frontier* of a computation of A corresponding to a prefix w_1 of the input w is a tuple $(p_1, \ldots, p_r, q)$, where $p_i \in P$, $i = 1, \ldots, r$, $r \geq 0$, are the states sent along pending hierarchical edges and $q \in Q$ is the linear state reached at the end of w_1. Here pending hierarchical edges refer to call symbols such that the current prefix w_1 does not have a matching return. A matching return may, or may not, occur in the part of the input to be processed. The frontier of the computation completely determines how the computation can be continued on the remainder of the input.

The NNWA A *accepts* a nested word w if in some nondeterministic computation it reaches the end of w in a final linear state and all hierarchical states of the computation corresponding to pending calls are final, that is, the frontier at the end of the computation is of the form $(p_1, \ldots, p_r, q)$, $q \in Q_f$, $p_i \in P_f$, $i = 1, \ldots, r$, $r \geq 0$. The nested word language recognized by A is denoted $L(A)$. Two NNWAs are said to be *equivalent* if they recognize the same language. A nested word language is *regular* if it is recognized by an NNWA.

An NNWA is said to be *linearly accepting* if all hierarchical states are final. A linearly accepting NNWA decides whether or not to accept the input based only on the linear state it reaches at the end of the computation. In the natural way we can define a *deterministic nested word automaton,* DNWA, as a special case of an NNWA, see [2,3,14] for details. An extension of the subset construction allows a deterministic simulation of an NNWA. The following result from [3], see also [2], gives an upper bound for the size blow-up of determinizing an NNWA.

Proposition 2.1 ([3]). *A linearly accepting* NNWA *with k linear states and h hierarchical states can be simulated by a* DNWA *with $2^{k \cdot h}$ linear states and 2^{h^2} hierarchical states.*

The following result is proven in [3] for DNWAs (and the number of hierarchical states of the linearly accepting automaton can be slightly reduced if the original automaton has final hierarchical states [14]). Exactly the same construction works for NNWAs.

Proposition 2.2 ([3]). *Every* NNWA *with n linear states and h hierarchical states has an equivalent linearly accepting* NNWA *with $2n$ linear and $2h$ hierarchical states.*

3. Nondeterministic state complexity

We define the *nondeterministic state complexity* of a regular language of nested words L, denoted $\mathrm{nsc}(L)$, as the smallest total number of states (linear and hierarchical states) of any NNWA recognizing L. As discussed in the introduction, the roles played by the (numbers of) linear and hierarchical states, respectively, are different and, often, we formulate explicit bounds both for the numbers of linear and of hierarchical states. The combined value $\mathrm{nsc}(L)$ can be used as a first approximation of the descriptional complexity of an NNWA. A DNWA with n linear states obviously needs at most $n \cdot |\Sigma|$ hierarchical states, where Σ is the input alphabet. For NNWAs we get only a quadratic upper bound and later, in Proposition 3.1, we will see that this bound can be reached in the worst case.

Lemma 3.1. *Let $A = (\Sigma, Q, Q_0, Q_f, P, P_0, P_f, \delta_c, \delta_i, \delta_r)$ be an* NNWA *and denote $|Q| = n$. There exists an equivalent* NNWA *B having n linear and $n^2 \cdot |\Sigma|$ hierarchical states.*

Proof. The NNWA B can use the same set of linear states Q and $Q \times \Sigma \times Q$ as the set of hierarchical states. The automaton B simulates a call transition $(q_1, p_1) \in \delta_c(q, \langle b)$, $q, q_1 \in Q$, $p_1 \in P$, $b \in \Sigma$, by choosing q_1 as the new linear state and sending (q, b, q_1) along the hierarchical edge. At the matching return symbol B can then nondeterministically select p_1 from all choices that could have been made in the current linear computation. The sets of final and initial hierarchical states are chosen appropriately. We leave the details of the construction to the reader. ■

Already in the deterministic case it is known that a state minimal DNWA for a regular nested word language need not be unique. This follows (as explained in the full version of [14]) using a simple modification of a corresponding non-uniqueness property of minimal visibly pushdown automata [1,13]. It is well known that a state minimal NFA need not be unique, see [9,16] for references, and hence this property necessarily holds for NNWAs which are extensions of both NFAs and DNWAs.

We develop techniques that can be used to establish lower bounds for the number of states needed by any NNWA to recognize a given nested word language. The following lemma is an extension of the fooling set techniques for NFAs introduced in [5,7]. These in turn can be viewed as special cases of techniques based on communication complexity [10,11].

Definition 3.1. A set of pairs of nested words $F = \{(x_i, y_i) \mid i = 1, \ldots, m\}$ is said to be a *fooling set* for a nested word language L if:

(F1) $x_i y_i \in L$, $i = 1, \ldots, m$, and
(F2) for any $1 \leq i < j \leq m$, $x_i y_j \notin L$ or $x_j y_i \notin L$.

The set F is a k-fooling set, $k \geq 0$, if each x_i has exactly k pending call symbols.

Lemma 3.2. *Let A be an* NNWA *with a set of linear states Q and a set of hierarchical states P. Assume that $L(A)$ has a k-fooling set $\{(x_i, y_i) \mid i = 1, \ldots, m\}$. Then $|P|^k \cdot |Q| \geq m$.*

Proof. Let $q_i \in Q$ and $p_{i,1}, \ldots, p_{i,k} \in P$ be such that $(p_{i,1}, \ldots, p_{i,k}, q_i)$ is the frontier of an arbitrarily chosen accepting computation of A on $x_i y_i$ after A has read the prefix x_i, $i = 1, \ldots, m$. Since the frontier and the remaining suffix completely determine whether a computation can be completed to an accepting computation, the condition (F2) of Definition 3.1 implies that $(p_{i,1}, \ldots, p_{i,k}, q_i) \neq (p_{j,1}, \ldots, p_{j,k}, q_j)$ when $i \neq j$. Hence $|P|^k \cdot |Q| \geq m$. ■

Note that by choosing $k = 0$, Lemma 3.2 gives the lower bound criterion for NFAs from [5]. As a first application of the technique we show that an NNWA with n linear states may need $\Omega(n^2)$ hierarchical states. The lower bound is within a constant factor of the upper bound of Lemma 3.1.

Proposition 3.1. *Choose $\Sigma = \{a, b\}$ and let $n \in \mathbb{N}$ be arbitrary. There exists $L_n \in \mathrm{NW}(\Sigma)$ such that L_n is recognized by an* NNWA *with $2n$ linear states and any* NNWA *for L_n needs n^2 hierarchical states.*

Proof. Consider a tagged word

$$w = a^{i_0} x_1 a^{i_1} x_2 \cdots x_r a^{i_r}, \qquad x_j \in \{b, \langle b, b\rangle\}, \quad j = 1, \ldots, r, \ r \geq 0, \tag{1}$$

$i_j \geq 0$, $0 \leq j \leq r$. We say that w is *b-alternating* if r is odd, $x_j \in \{\langle b, b\rangle\}$ always when j is odd and $x_j = b$ when j is even, $0 \leq j \leq r$. The *two-sided count* of a symbol occurrence $x_j \in \{b, \langle b, b\rangle\}$, $1 \leq j \leq r$, in the tagged word w is defined as $\mathrm{count}(x_j) = (i_{j-1}, i_j)$. We denote $[n] = \{0, \ldots, n-1\}$ and $[\overline{n}] = \{\overline{0}, \ldots, \overline{n-1}\}$.

We define L_n to consist of all b-alternating well-matched nested words w (as in (1)) such that for each pair of matching call and return symbol occurrences y_1 and y_2, $\mathrm{count}(y_i) \in [n] \times [n]$, $i = 1, 2$, and $\mathrm{count}(y_1) = \mathrm{count}(y_2)$.

The nested word language L_n is recognized by the following NNWA

$$A = (\Sigma, [n] \cup [\overline{n}], \{0\}, \{\overline{0}\}, [n] \times [n], \emptyset, \emptyset, \delta_c, \delta_i, \delta_r),$$

where the transition relations are defined as follows. All transitions not listed below will be undefined. For $i \in [n]$ the call transitions are $\delta_c(i, \langle b) = \{(\bar{j}, (i, j)) \mid j \in [n]\}$. The internal a-transitions are defined by setting for $j \in [n]$, $\delta_i(j, a) = \{j+1\}$ if $j \neq n-1$, $\delta_i(\bar{j}, a) = \{\overline{j-1}\}$ if $j \neq 0$. The only internal b-transition is $\delta_i(\overline{0}, b) = \{0\}$. Finally the return transitions are defined for $i, j, k \in [n]$ by setting

$$\delta_r(i, (j, k), b\rangle) = \begin{cases} \{\overline{k}\} & \text{if } i = j, \\ \emptyset & \text{otherwise.} \end{cases}$$

The only initial linear state is 0, the symbols $\langle b$ and $b\rangle$ change the linear state from an element of $[n]$ to an element of $[\overline{n}]$, the symbol b makes the reverse change, and $\overline{0}$ is the only final linear state. This means that all accepted words must be b-alternating. The NNWA A has no initial or final hierarchical states which guarantees that it can accept only well-matched nested words.

Consider an input w as in (1) and an occurrence $x_s = \langle b$ of a call symbol in w, where $\mathrm{count}(x_s) = (i, j)$. Since w is b-alternating, we know that s has to be odd and this means that any computation of A reaches this occurrence of $\langle b$ in the linear state i. Again since w is b-alternating, we know that $x_{s+1} = b$ and the only internal transition defined for b uses argument $\overline{0}$. This means that in an accepting computation at call symbol x_s, the automaton has to make the nondeterministic choice $(\bar{j}, (i, j))$, where j is the number of a's between x_s and x_{s+1}. Thus the computation sends the pair (i, j) along the hierarchical edge to the return symbol $x_{s'}$ that matches x_s. According to the definition of the return transitions, the computation then verifies that $\mathrm{count}(x_{s'}) = (i, j)$. This means that $L(A) = L_n$.

To establish the lower bound for the number of hierarchical states, for $k \geq 1$, we define the k-fooling set

$$F_k = \{(a^{i_1} \langle b a^{j_1} b a^{i_2} \langle b a^{j_2} b \cdots a^{i_k} \langle b a^{j_k} b, a^{i_k} b\rangle a^{j_k} b a^{i_{k-1}} b\rangle a^{j_{k-1}} b \cdots a^{i_1} b\rangle a^{j_1}) \mid 0 \leq i_r, j_s \leq n-1, \ 1 \leq r, s \leq k\}.$$

Since in words of L_n the two-sided counts of matching call and return symbols must coincide, it is clear that F_k is a fooling set for L_n. Let B be an arbitrary NNWA recognizing L_n, where the sets of linear and hierarchical states are, respectively, Q and P. By Lemma 3.2, $|Q| \cdot |P|^k \geq |F_k| = n^{2k}$ for all $k \geq 1$. Since $|Q|$ is independent of k, this implies $|P| \geq n^2$. ■

3.1. Union and intersection

First we establish upper bounds for the state complexity of union and intersection. In the case of union we note that the hierarchical states of one of the component automata can be recycled in the other component, provided that we introduce new states that are used, respectively, as an initial and final hierarchical state. On the other hand, the definition of NNWAs allows multiple initial states, which means that we do not need to add a "new" initial linear state as in the case of ordinary NFAs [9]. The result for intersection turns out to be analogous to the case of ordinary NFAs, the main difference being that we need to consider separately the sets of linear and hierarchical states.

Lemma 3.3. *Let A_i be an* NNWA *with n_i linear and h_i hierarchical states, $i = 1, 2$.*

(a) *The nested word language $L(A_1) \cup L(A_2)$ can be recognized by an* NNWA *with $n_1 + n_2$ linear and $\max(h_1, h_2) + 2$ hierarchical states.*
(b) *The nested word language $L(A_1) \cap L(A_2)$ can be recognized by an* NNWA *with $n_1 \cdot n_2$ linear and $h_1 \cdot h_2$ hieachical states.*

Proof. (a) Let $A_1 = (\Sigma, Q, Q_0, Q_f, P, P_0, P_f, \delta_c, \delta_i, \delta_r)$ and $A_2 = (\Sigma, Q', Q'_0, Q'_f, P', P'_0, P'_f, \delta'_c, \delta'_i, \delta'_r)$. Without loss of generality $Q \cap Q' = \emptyset$ and $h_1 = |P| \geq |P'| = h_2$. Let g be a bijective mapping $R \longrightarrow P'$, where $R \subseteq P$. We define

$$B = (\Sigma, Q \cup Q', Q_0 \cup Q'_0, Q_f \cup Q'_f, P \cup \{z_1, z_2\}, \{z_1\}, \{z_2\}, \gamma_c, \gamma_i, \gamma_r),$$

where z_1, z_2 are new elements to be used, respectively, as the only initial and the only final hierarchical state of B. The call transitions of B are defined by setting, for $b \in \Sigma$,

$$\gamma_c(q, \langle b) = \begin{cases} \delta_c(q, \langle b) \cup \{(q_1, z_2) \mid (\exists p_1 \in P_f)(q_1, p_1) \in \delta_c(q, \langle b)\} & \text{if } q \in Q, \\ \{(q_1, p_1) \mid (q_1, g(p_1)) \in \delta'_c(q, \langle b), p_1 \in R\} \cup & \\ \{(q_1, z_2) \mid (\exists p_1 \in P'_f)(q_1, p_1) \in \delta'_c(q, \langle b)\} & \text{if } q \in Q'. \end{cases}$$

On elements of Q the call transitions simulate the call transitions of A_1 and additionally for any final hierarchical state introduce an additional nondeterministic choice where the hierarchical state is replaced by z_2. Note that no transitions are defined for z_2 which means that it can be used in accepting computations only at pending call symbols. On elements of Q', the transition relation γ_c similarly simulates the call transitions of A_2.

The internal transition relation of B, γ_i, on states of Q (respectively, of Q') simply simulates internal transitions of A_1 (respectively, of A_2). The return transitions are defined by setting for $b \in \Sigma$,

$$\gamma_r(q, p, b\rangle) = \begin{cases} \delta_r(q, p, b\rangle) & \text{if } q \in Q, p \in P, \\ \bigcup_{p_1 \in P_0} \delta_r(q, p_1, b\rangle) & \text{if } q \in Q, p = z_1, \\ \delta'_r(q, g(p), b\rangle) & \text{if } q \in Q', p \in R, \\ \bigcup_{p_1 \in P'_0} \delta'_r(q, p_1, b\rangle) & \text{if } q \in Q', p = z_1, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Depending on the linear state q given as argument, the return transitions simulate either return transitions of A_1 or A_2, where in both cases the new hierarchical state z_1 is interpreted as an arbitrary initial hierarchical state. Since the sets Q and Q' are disjoint, each computation of B simulates either a computation of A_1 or of A_2, that is, the two types of computations cannot be mixed. This means that B nondeterministically recognizes $L(A_1) \cup L(A_2)$.

(b) An NNWA that simulates the computation of two given NNWAs can use a cross-product construction separately for the linear and for the hierarchical states. The details of the construction are left to the reader. ■

In the following two lemmas we show that the upper bounds can be reached in cases where the number of linear states is a multiple of the number of hierarchical states.

Lemma 3.4. *Let $n_1, n_2, h \geq 2$ be integers such that h divides n_1 and n_2. There exist L_i recognized by an* NNWA *A_i with n_i linear and h hierarchical states, $i = 1, 2$, such that* $\mathrm{nsc}(L_1 \cup L_2) \geq n_1 + n_2 + h + 2$.

Proof. Let $\Sigma = \{a, b, c\}$. Consider $w \in \mathrm{NW}(\Sigma)$ and let z be an occurrence of a call or return symbol in w. For $x \in \Sigma$, we define the notion of (x, h)-count of the symbol z in w, $\mathrm{count}_{x,h,w}(z)$, as the number modulo h of occurrences of the symbol x in the prefix preceding z. We say that a word $w \in \mathrm{NW}(\Sigma)$ is *(x, h)-matched, $x \in \Sigma$*, or *x-matched* for short when h is known, if for every pair of matching calls and returns z, z',

$$\mathrm{count}_{x,h,w}(z) = \mathrm{count}_{x,h,w}(z'). \tag{2}$$

We define the language L_1 to consist of all nested words w such that $\mathrm{base}(w) \in \{a, \langle c, \ c\rangle\}^*$, w is (a, h)-matched, $|w|_a \equiv 0 \pmod{n_1}$, and the following condition holds. If z is a pending call (respectively, a pending return) in w, then $\mathrm{count}_{a,h,w}(z) = 0$ (respectively, $\mathrm{count}_{a,h,w}(z) = 1$).

The language L_2 is defined to consist of all (b, h)-matched and well-matched nested words w such that $\mathrm{base}(w) \in \{b, \langle c, c\rangle\}^*$ and $|w|_b \equiv 0 \pmod{n_2}$.

Since n_1 is a multiple of h, the language L_1 can be recognized by an NNWA A_1 where the n_1 linear states count symbols a modulo n_1 and the h hierarchical states guarantee that the input is (a, h)-matched. The NNWA has one initial hierarchical

state (that corresponds to count one modulo h) and one final hierarchical state (that corresponds to count zero modulo h). Similarly, L_2 is recognized by an NNWA A_2 with n_2 linear and h hierarchical states. The NNWA A_2 has no initial or final hierarchical states.

Let B be an arbitrary NNWA for $L_1 \cup L_2$ with sets of linear and hierarchical states, respectively, Q and P. First using Lemma 3.2 we show that $|Q| \geq n_1 + n_2$ and $|P| \geq h$, and after that using a more detailed analysis we show that B, in fact, needs at least two additional states.

We use a k-fooling set $F_1^{(k)} \cup F_2^{(k)}$, $k \geq 0$, for $L_1 \cup L_2$. To keep the notations simple, below we give the definition only for $k = 2$. Denote

$$\begin{aligned} F_1^{(2)} &= \{(a^{r_1}\langle ca^{r_2}\langle ca^s,\ a^{u_2}c\rangle a^{u_1}c\rangle a^v) \mid 1 \leq r_i, u_i \leq h,\ i = 1, 2,\ 1 \leq s, v \leq n_1, \\ &\qquad r_2 + s + u_2 + u_1 \equiv s + u_2 \equiv 0 (\mathrm{mod}\ h),\ r_1 + r_2 + s + u_2 + u_1 + v \equiv 0 (\mathrm{mod}\ n_1)\}, \\ F_2^{(2)} &= \{(b^{r_1}\langle cb^{r_2}\langle cb^s,\ b^{u_2}c\rangle b^{u_1}c\rangle b^v) \mid 1 \leq r_i, u_i \leq h,\ i = 1, 2,\ 1 \leq s, v \leq n_2, \\ &\qquad r_2 + s + u_2 + u_1 \equiv s + u_2 \equiv 0 (\mathrm{mod}\ h),\ r_1 + r_2 + s + u_2 + u_1 + v \equiv 0 (\mathrm{mod}\ n_2)\}. \end{aligned}$$

Note that $|F_1^{(2)}| = n_1 \cdot h^2$: the values $1 \leq r_i \leq h$, $1 \leq i \leq 2$, and $1 \leq s \leq n_1$ can be chosen arbitrarily and they determine uniquely the values of u_1, u_2 and v. Similarly, $|F_2^{(2)}| = n_2 \cdot h^2$. It is easy to verify that $F_1^{(2)} \cup F_2^{(2)}$ is a 2-fooling set for $L_1 \cup L_2$: the first components of the pairs of $F_1^{(2)}$ (respectively, of $F_2^{(2)}$) have distinct pairs of a-counts (respectively, b-counts) for the two pending calls and these must be matched by equal a-counts (respectively, b-counts) for the pending returns in the second components of the pairs.

Using exactly the same idea as above we can define a k-fooling set $F_1^{(k)} \cup F_2^{(k)}$ of cardinality $(n_1 + n_2) \cdot h^k$ for an arbitrary $k \geq 0$.[1] Note that with $k = 0$ the fooling set is

$$F_1^{(0)} \cup F_2^{(0)} = \{(a^{n_1}, a^{n_1}), (a, a^{n_1-1}), \ldots, (a^{n_1-1}, a), (b^{n_2}, b^{n_2}), (b, b^{n_2-1}), \ldots, (b^{n_2-1}, b)\}$$

Lemma 3.2 gives now $|Q| \cdot |P|^k \geq (n_1 + n_2) \cdot h^k$, and since this has to hold for arbitrarily large values of k, the only possibility is that $|P| \geq h$. By choosing $k = 0$ we get $|Q| \geq n_1 + n_2$.

Now in order to complete the proof of Lemma 3.4 it is sufficient to show that $|Q| = n_1 + n_2$, $|P| = h + 1$ and $|Q| = n_1 + n_2 + 1$, $|P| = h$ are both impossible. Note that a situation where $|Q| = n_1 + n_2$, $|P| = h$ can be viewed as an instance of the above cases where we have one useless linear or hierarchical state, respectively.

Below we have to use a somewhat tedious case analysis. In the following, let $\gamma_c, \gamma_i, \gamma_r$ denote the transition relations of B.

(I) Case $|Q| = n_1 + n_2$, $|P| = h + 1$: By considering accepting computations of B on words of a^+ (respectively, b^+) it is easy to verify that the set of linear states Q has to be a disjoint union $Q_1 \cup Q_2$, $|Q_i| = n_i$, where internal a-transitions on states of Q_1 define a cycle of length n_1 and a-transitions are undefined on states of Q_2. Similarly, internal b-transitions are defined only on states in Q_2 and form a cycle of length n_2. If these conditions are not met, it is immediate that B cannot accept the correct subset $a^+ \cup b^+$ or B needs more than $n_1 + n_2$ linear states. For example, if both a and b transitions are be defined for some state, this state cannot occur in a cycle and B needs more than $n_1 + n_2$ linear states.

We consider accepting computations of B on words xy, where the pairs (x, y) form the set $F_2^{(1)}$. These computations use as set of linear states $Q_2' \subseteq Q_2$ and a set of hierarchical states P_2. Similarly as in the proof of Lemma 3.2 we see that $|Q_2'| \cdot |P_2| \geq |F_2^{(1)}| = n_2 \cdot h$, and since we know that $|Q_2| = n_2$ this gives $|P_2| \geq h$. Furthermore, we note that all states of Q_2 are reachable from an initial state and reach a final state with a linear word in b^*. Thus, if some of the hierarchical states of P_2 would be initial or final, B would necessarily accept nested words with one or more b's and pending call or return symbols, which is a contradiction.

On the other hand, there exists an initial hierarchical state p_{ini} that is used in an accepting computation on $ac\rangle a^{n_1-1}$ and a final hierarchical state p_{acc} used in a computation on $\langle ca^{n_1}$. In order to show that $|P| \geq h + 2$, it is sufficient to show that $p_{\mathrm{ini}} \neq p_{\mathrm{acc}}$.

Denote $Q_1 = \{q_0, q_1, \ldots, q_{n_1-1}\}$ where internal a-transitions cycle the states in this order and q_0 is an initial and final linear state. In the accepting computation on $\langle ca^{n_1}$ the first call transition is $(q_0, p_{\mathrm{acc}}) \in \gamma_c(q_0, \langle c)$. (If Q_1 would have other initial linear states besides q_0, it is immediate that B would accept words where the total number of a's is not divisible by n_1.) Similarly we verify that the return transition used in an accepting computation on $ac\rangle a^{n_1-1}$ must be $q_1 \in \gamma_r(q_1, p_{\mathrm{ini}})$. Now the condition $p_{\mathrm{ini}} = p_{\mathrm{acc}}$ means that B would accept $\langle cac\rangle a^{n_1-1}$ that is not in L_1 because the (a, h)-counts of the call and return symbols are unequal. (Recall that $h \geq 2$.)

(II) Case $|Q| = n_1 + n_2 + 1$, $|P| = h$: Similarly as above, by considering computations of B on words of a^+ and b^+ we observe that, assuming all states are useful, the set of linear states Q can be written as a disjoint union of Q_1 and Q_2, $|Q_i| \geq n_i$, $i = 1, 2$, where states of Q_1 (respectively, Q_2) can be used only in computations on words of a^+ (respectively, b^+). There are two possibilities to consider.

[1] A similar definition for arbitrary values of k is given in the proof of Lemma 3.5 when dealing with intersection.

(IIa) First, assume that $|Q_2| = n_2$, $|Q_1| = n_1 + 1$. Now the states of Q_2 must form a b-cycle, and exactly as in (I) above we verify that computations on words corresponding to the pairs of the set $F_2^{(1)}$ must use h distinct hierarchical states, none of which can be initial or final. Since B must contain initial and final hierarchical states and $|P| = h$, we are done. (Note that now it could be more complicated to show that the initial and final hierarchical states are distinct, but we do not need this property since we only need to show that $|P| = h$ is impossible.)

(IIb) Second, consider the possibility $|Q_2| = n_2 + 1$, $|Q_1| = n_1$. As in (I) above we see that the linear a-transitions on Q_1 have to form a cycle of length n_1, and computations on nested words corresponding to pairs of the set $F_1^{(1)}$ must use h distinct hierarchical states, one of which is initial and another one of which is final.

Due to the additional state in Q_2, the b-transitions need not form a simple cycle, however, again assuming all states are useful,

$$\text{no state of } Q_2 \text{ can be reached by words } w_1, w_2 \text{ where } |w_1|_b \not\equiv |w_2|_b \pmod{n_2}. \tag{3}$$

This means that some set of states $Q_2' = \{q_0', q_1', \ldots, q_{n_2-1}'\} \subseteq Q_2$ forms a cycle with b-transitions where q_0' is initial and final. The set Q_2' need not be uniquely determined. We simply choose Q_2' to consist of states that occur in an arbitrary, but fixed, b-cycle of length n_2.

For $0 \leq j \leq h-1$, let D_j be an accepting computation of B on the nested word $b^j\langle cc\rangle b^{n_2-j}$. Let $q_j \in Q_2$ be the linear state that occurs after D_j has processed the prefix b^j, $j = 0, \ldots, h-1$. By (3), we know that the states q_j, $j = 0, \ldots, h-1$, are all distinct and at least $h-1$ of them are in Q_2'. (Recall that Q_2 has only one state not in Q_2'.) Below we consider the case where there exists $j_0 \in \{0, \ldots, h-1\}$ such that $q_{j_0} \notin Q_2'$ and we denote $H_0 = \{0 \leq k \leq h-1 \mid k \neq j_0\}$. The other possibility is that all the states q_j, $0 \leq j \leq h-1$, would be in Q_2', and in this case the following argument is similar but simpler.[2]

Now (3) implies that in a computation D_j, $j \in H_0$, the linear state q_j after reading each of the prefixes b^j, $b^j\langle c$ and $b^j\langle cc\rangle$ must be equal to $q_j' \in Q_2'$.

Let p_j be the hierarchical state used in the computation D_j, $j \in H_0$, and assume that $p_{j_1} = p_{j_2}$ for some $j_1 < j_2$, $j_1, j_2 \in H_0$. In particular, this means that $(q_{j_1}', p_{j_1}) \in \gamma_c(q_{j_1}', \langle c)$ and $q_{j_2}' \in \gamma_r(q_{j_2}', p_{j_1}, c\rangle)$. Thus, B has an accepting computation on $b^{j_1}\langle cb^{j_2-j_1}c\rangle b^{n_2-j_2}$. This is impossible because the (b, h)-counts of the matching symbols $\langle c$ and $c\rangle$ are unequal.

We have seen that each of the $h-1$ computations D_j, $j \in H_0$, must use a distinct hierarchical state. Using the fact that the b-transitions on Q_2' form a cycle, in the same way as in (I) above we see that none of these $h-1$ hierarchical states can be initial or final. On the other hand, above at the begin of (IIb) we have seen that computations corresponding to the pairs of $F_1^{(1)}$ need an initial and a final hierarchical state that are distinct. This means that the number of hierarchical states has to be at least $h+1$. ■

Note that in the proof of Lemma 3.4 the lower bound technique of Lemma 3.2 gives directly that any NNWA recognizing $L_1 \cup L_2$ needs at least $n_1 + n_2$ linear and h hierarchical states, and much additional effort was needed to show that, in fact, two more states are needed. In the proof we have shown that the total number of states is at least $n_1 + n_2 + h + 2$. We believe that any NNWA for $L_1 \cup L_2$ with a minimal total number of states, in fact, has at least $h+2$ hierarchical states but do not attempt to prove this claim.

Lemma 3.5. *Let n_i, h_i be positive integers such that h_i divides n_i, $i = 1, 2$. There exist L_i recognized by an* NNWA *A_i with n_i linear and h_i hierarchical states, $i = 1, 2$, such that any* NNWA *for $L_1 \cap L_2$ needs $n_1 \cdot n_2$ linear and $h_1 \cdot h_2$ hierarchical states.*

Proof. Let $\Sigma = \{a, b, c\}$. We define L_1 to consist of all well-matched nested words w such that $\text{base}(w) \in \{a, b, \langle c, c\rangle\}^*$, w is (a, h_1)-matched (as defined in (2)) and $|w|_a \equiv 0 \pmod{n_1}$. The language L_2 is defined to consist of all well-matched nested words w such that $\text{base}(w) \in \{a, b, \langle c, c\rangle\}^*$, w is (b, h_2)-matched and $|w|_b \equiv 0 \pmod{n_2}$. Similarly as in the construction for union, using the fact that n_i is a multiple of h_i it is easy to construct an NNWA with n_i linear and h_i hierarchical states for the language L_i, $i = 1, 2$. Note that since L_i consists of well-matched words, A_i does not need to have any initial or final hierarchical states.

Denote $L = L_1 \cap L_2$. For each $k \geq 0$ we define a k-fooling set $F^{(k)}$ for L. Let

$$\begin{aligned} F^{(k)} = \Big\{ & (a^{r_1}b^{s_1}\langle ca^{r_2}b^{s_2}\cdots a^{r_k}b^{s_k}\langle ca^{x_1}b^{y_1}, a^{u_k}b^{v_k}c\rangle a^{u_{k-1}}b^{v_{k-1}}\cdots a^{u_1}b^{v_1}c\rangle a^{x_2}b^{y_2}) \mid \\ & 0 \leq r_i, u_i < h_1,\ 0 \leq s_i, v_i < h_2,\ 0 \leq x_1, x_2 < n_1,\ 0 \leq y_1, y_2 < n_2, \\ & \sum_{i=1}^{k}(r_i + u_i) + x_1 + x_2 \equiv 0 \pmod{n_1},\ \sum_{i=1}^{k}(s_i + v_i) + y_1 + y_2 \equiv 0 \pmod{n_2}, \end{aligned}$$

[2] In particular, if $h \neq n_2$ we can always select h nested words $b^j\langle cc\rangle b^{n_2-j}$ where j ranges over h consecutive values and some accepting computations on these words correspond to the simpler case.

$$(\forall j \in \{2, \ldots, k+1\})\left[\sum_{i=j}^{k}(r_i + u_i) + u_{j-1} + x_1 \equiv 0 \pmod{h_1}\ \&\right.$$
$$\left.\sum_{i=j}^{k}(s_i + v_i) + v_{j-1} + y_1 \equiv 0 \pmod{h_2}\right]\Bigg\} \tag{4}$$

The cardinality of $F^{(k)}$ is $(n_1 \cdot n_2)(h_1 \cdot h_2)^k$. When determining the elements of $F^{(k)}$, the values of x_1, y_1 and $r_i, s_i, i = 1, \ldots, k$, can be chosen arbitrarily in the given ranges and after this the remaining values are completely determined. Note that the condition (4) for a particular j says that, in the nested word that is the catenation of the two components of the pair, the (a, h_1)- and (b, h_2)-counts of the $(j-1)$th call symbol and the matching return symbol must coincide. The equation with value $j = k+1$ corresponds to the pair consisting of the last call and the first return symbol.

The above argument shows that, for any w_1 that occurs as a first component of a pair of $F^{(k)}$, there is a unique nested word w_2 among the second components of the pairs of $F^{(k)}$ such that $w_1 \cdot w_2 \in L$. This means that $F^{(k)}$ is, indeed, a fooling set for L.

Let B be an arbitrary NNWA for L with a set of linear states Q and set of hierarchical states P. Lemma 3.2 gives that $|Q| \cdot |P|^k \geq |F^{(k)}|$, $k \geq 0$. Since the inequality holds for all $k \geq 0$, we conclude that $|Q| \geq n_1 \cdot n_2$ and $|P| \geq h_1 \cdot h_2$. ■

As a consequence of Lemmas 3.3–3.5 we have:

Theorem 3.1. *Let A_i be an* NNWA *with n_i linear and h_i hierarchical states, where h_i divides n_i, $i = 1, 2$. The tight worst-case nondeterministic state complexity of $L(A_1) \cap L(A_2)$ is $n_1 \cdot n_2 + h_1 \cdot h_2$. If $h_1 = h_2 = h$, the worst-case state complexity of $L(A_1) \cup L(A_2)$ is precisely $n_1 + n_2 + h + 2$.*

3.2. Complementation

We give a lower bound for the nondeterministic state complexity of complementation that is significantly worse than the exponential lower bound for complementation of ordinary NFAs. For $L \subseteq \mathrm{NW}(\Sigma)$, denote $\overline{L} = \mathrm{NW}(\Sigma) - L$.

Lemma 3.6. *Let $n \in \mathbb{N}$. There exists a nested word language L recognized by an* NNWA *with $O(2^n)$ states such that*

$$\mathrm{nsc}(\overline{L}) \geq \sqrt{(2^n)!}$$

Proof. Let $\Sigma = \{0, 1, c, \#, \$\}$. For $w \in \{0, 1\}^*$, $\mathrm{num}(w)$ denotes the binary number represented by the word w. (Note that w may contain leading zeros.) We define for $n \geq 1$,

$$\begin{aligned} L_n = \{&\langle \$u_0 c u_1 c \cdots c u_r \# v_0 c v_1 \cdots c v_s \$\rangle \mid u_i, v_j \in \{0, 1\}^+, 0 \leq i \leq r, \\ &0 \leq j \leq s,\ r, s, \geq 0, (\exists\ 0 \leq i \leq \min(r, 2^n - 1))\ [\mathrm{num}(v_{\mathrm{num}(u_i)}) \neq i, \\ &u_i, v_{\mathrm{num}(u_i)} \in \{0, 1\}^n,\ \mathrm{num}(u_i) \leq s]\}. \end{aligned}$$

The language L_n is recognized by an NNWA A as follows. The below discussion assumes that the input is in $\langle \$(\{0, 1\}^+ c)^+ \{0, 1\}^+ \# (\{0, 1\}^+ c)^+ \{0, 1\}^+ \$\rangle$. If this is not the case, it is easy (without exceeding the upper bound $O(2^n)$ for the number of states) to guarantee that all computations of A reject.

At the call symbol $\langle \$$, A guesses a number $i \in \{0, 1, \ldots, 2^n - 1\}$ and continues the linear computation in a state q_i and sends a state p_i along the hierarchical edge, where both states encode the number i. Using the linear state as a counter, A "passes by" i subwords of the form $\{0, 1\}^+ c$. After this A reads and stores in the linear state the following subword $z \in \{0, 1\}^n$, and checks that it is followed by a c. The automaton then "passes by" subwords in $\{0, 1\}^+ c$ until it encounters the "middle marker" #.

After the marker #, using the linear state z as a counter A "passes by" $\mathrm{num}(z)$ subwords in $\{0, 1\}^+ c$. Then it reads the following subword $z' \in \{0, 1\}^n$ and verifies that it is followed by a c. The linear state "remembers" z' and A "passes by" subwords in $\{0, 1\}^+ c$ until it reaches the return symbol $\$\rangle$. Here the return transition relation checks that $\mathrm{num}(z') \neq i$ where i is the number encoded by the hierarchical state p_i sent from the first call symbol. In this way A is able to nondeterministically verify that the input is in L_n.

The linear computation has to remember binary words of length up to n and uses these as counters. This can be done with $O(2^n)$ linear states. The number of hierarchical states can be chosen to be exactly 2^n.

Next we establish the lower bound for the nondeterministic state complexity of the complement of L_n. Let B be an arbitrary NNWA recognizing $\overline{L_n}$ and let Q (respectively, P) be the set of linear (respectively, hierarchical) states of B. We denote by $\mathcal{B}$ the set of all bijections $\{0, 1, \ldots, 2^n - 1\} \to \{0, 1\}^n$. For $f, g \in \mathcal{B}$, we denote

$$x_f = \langle \$ f(0) c f(1) c \cdots c f(2^n - 1) \#,\ \text{and},\ y_g = g(0) c g(1) c \cdots c g(2^n - 1) \$\rangle$$

We note that for any $f, g \in \mathcal{B}$,

$$x_f y_g \notin L_n \ \text{ iff } (\forall\ 0 \leq i \leq 2^n - 1)\ \ \mathrm{num}(g(\mathrm{num}(f(i)))) = i \text{ iff } \hat{g} = f^{-1}, \tag{5}$$

where $\hat{g} : \{0, 1\}^n \to \{0, 1, \ldots, 2^n - 1\}$ is defined by setting $\hat{g}(w) = \mathrm{num}(g(\mathrm{num}(w)))$, $w \in \{0, 1\}^n$. Now (5) implies that $\{(x_f, y_g) \mid f, g \in \mathcal{B}, \hat{g} = f^{-1}\}$ is a 1-fooling set for $\overline{L_n}$. By Lemma 3.2 it follows that $|P| \cdot |Q| \geq |\mathcal{B}| = (2^n)!$, and consequently $\mathrm{nsc}(\overline{L_n}) \geq \sqrt{(2^n)!}$. ■

The lower bound for the nondeterministic state complexity of complementation of regular nested words languages can be stated as follows.

Theorem 3.2. *For arbitrarily large $n \geq 1$, there exist regular nested word languages L'_n such that*

$$\mathrm{nsc}(L'_n) \in O(n) \text{ and } \mathrm{nsc}(\overline{L'_n}) \in \Omega(\sqrt{n!}).$$

The result shows that the worst-case blow-up of nondeterministic state complexity of complementation of regular nested word languages is much bigger than the exponential blow-up of complementation of NFAs [9,12,17]. However, the lower bound of Theorem 3.2 does not match the upper bound $O(2^{n^2})$ implied by Proposition 2.1 and the deterministic state complexity of complementation [14]. The precise nondeterministic state complexity of complementation of regular nested word languages remains open.

3.3. Upper bounds for catenation and Kleene star

The regular nested word languages are closed under the operations of reversal, catenation and Kleene star, as extended to nested words, see [3]. Since NNWAs allow multiple initial states, the worst-case nondeterministic state complexity of reversal is obviously the identity function. The following upper bound for the nondeterministic state complexity of catenation is immediate and we omit the straightforward proof.

Lemma 3.7. *Let A_i be an* NNWA *with n_i linear and h_i hierarchical states, $i = 1, 2$. The nested word language $L(A_1) \cdot L(A_2)$ can be recognized by an* NNWA *with $n_1 + n_2$ linear and $h_1 + h_2$ hierarchical states.*

We believe that the upper bound of Lemma 3.7 is tight but do not have a proof for this claim. Consider NNWAs A_i with n_i linear and h_i hierarchical states, $i = 1, 2$. It is easy to construct examples where any NNWA recognizing the catenation of $L(A_1)$ and $L(A_2)$ needs $n_1 + n_2$ linear states. Furthermore, due to the fact that words in $L(A_1)$ can have pending calls, it seems that an NNWA recognizing $L(A_1) \cdot L(A_2)$ could not, in general, "reuse" the hierarchical states of A_1 in computations simulating A_2 on a suffix of the input. However, using Lemma 3.2 or techniques similar to its proof, we seem to get only a lower bound of $\max(h_1, h_2)$ hierarchical states. In all the examples we have managed to come up with, a k-fooling set for $L(A_1) \cdot L(A_2)$ seems to have only, roughly, $n_1 \cdot h_1^k + n_2 \cdot h_2^k$ elements. Using ad hoc arguments it is possible to show that the number of hierarchical states, in general, needs to be greater than $\max(h_1, h_2)$ but we do not have a construction giving the lower bound $h_1 + h_2$.

It can be noted that if the languages $L(A_i)$, $i = 1, 2$, are well-matched, then in the construction proving Lemma 3.7 we can reuse the hierarchical states and, thus, Lemma 3.2 gives a tight lower bound for the catenation of well-matched languages.

The Kleene star L^* of a nested word language L is defined in the natural way, see [3]. The proof establishing closure under Kleene star (Theorem 6 of [3]) uses a weakly-hierarchical normal form for nested word automata. Due to Proposition 2.2 and ([3] Theorem 2) it is known that an arbitrary DNWA A with n linear states and input alphabet Σ can be transformed into an equivalent weakly-hierarchical automaton B with $2n|\Sigma|$ linear states and the same set of hierarchical states.[3] However, if the construction of the proof of ([3] Theorem 2) is modified for NNWAs, the automaton would need to remember both the symbol labeling the innermost pending call and the nondeterministic choice made at that symbol. When applying this construction to NNWAs, the number of states of the weakly hierarchical automaton would, in the worst case, depend quadratically on the number of linear states of the original NNWA. Furthermore, the construction showing closure under star doubles the number of states. Thus, a construction directly following the proof of ([3] Theorem 6) gives an upper bound of $8|\Sigma|n^2$ for $\mathrm{nsc}(L^*)$ when $\mathrm{nsc}(L) = n$.

Below using a construction based directly on the linearly accepting normal form for NNWAs we give an improved upper bound for the nondeterministic state complexity of Kleene star.

Proposition 3.2. *Let A be an* NNWA *with n linear and h hierarchical states. The language $L(A)^*$ can be recognized by an* NNWA *B with $4n$ linear and $4h$ hierarchical states.*

Proof. By Proposition 2.2, A has an equivalent linearly accepting NNWA $C = (\Sigma, Q, Q_0, Q_f, P, P_0, P, \delta_c, \delta_i, \delta_r)$ where $|Q| = 2n$ and $|P| = 2h$. (Note that all hierarchical states of C are final.) For a set X we denote $X_{(i)} = \{x_{(i)} \mid x \in X\}$, $i = 1, 2$. Based on C we define the linearly accepting automaton

$$B = (\Sigma, Q_{(1)} \cup Q_{(2)}, [Q_0]_{(1)}, [Q_f]_{(1)} \cup [Q_f]_{(2)}, P_{(1)} \cup P_{(2)}, [P_0]_{(1)}, P_{(1)} \cup P_{(2)}, \gamma_c, \gamma_i, \gamma_r),$$

where the transition relations are defined below.

For $\langle b \in \langle \Sigma, q \in Q$ and $j \in \{1, 2\}$ we set

$$\gamma_c(q_{(j)}, \langle b) = \{(q'_{(2)}, p_{(j)}) \mid (q', p) \in \delta_c(q, \langle b)\} \cup Y_1, \quad \text{where} \tag{6}$$

$$Y_1 = \begin{cases} [Q_0]_{(1)} \times P_{(1)} & \text{if } \delta_c(q, \langle b) \text{ contains an element with first component in } Q_f, \\ \emptyset & \text{otherwise.} \end{cases}$$

[3] A weakly hierarchical automaton sends along a hierarchical edge always the current linear state, for a detailed definition see [3].

For $b \in \Sigma$, $q \in Q$ and $j \in \{1, 2\}$ the internal transitions are defined by setting

$$\gamma_i(q_{(j)}, b) = \{q'_{(j)} \mid q' \in \delta_i(q, b)\} \cup Y_2, \quad Y_2 = \begin{cases} [Q_0]_{(1)} & \text{if } \delta_i(q, b) \cap Q_f \neq \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$$

Finally, for $b\rangle \in \Sigma\rangle$, $q \in Q$, $p \in P$ and $j = 1, 2$, we define the return transitions by

$$\gamma_r(q_{(1)}, p_{(j)}, b\rangle) = \{q'_{(1)} \mid q' \in \bigcup_{p' \in P_0} \delta_r(q, p', b\rangle) \cup Y_3, \tag{7}$$

$$\gamma_r(q_{(2)}, p_{(j)}, b\rangle) = \{q'_{(j)} \mid q' \in \delta_r(q, p, b\rangle)\} \cup Y_4, \tag{8}$$

$$Y_3 = \begin{cases} [Q_0]_{(1)} & \text{if } \bigcup_{p' \in P_0} \delta_r(q, p', b\rangle) \cap Q_f \neq \emptyset, \\ \emptyset & \text{otherwise,} \end{cases} \qquad Y_4 = \begin{cases} [Q_0]_{(1)} & \text{if } \delta_r(q, p, b\rangle) \cap Q_f \neq \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$$

The automaton B simulates a computation of C, and always when the latter enters a final linear state, using the choices defined by the sets Y_i, $i = 1, \ldots, 4$, B can nondeterministically begin a new computation in an initial linear state. The crucial point of the construction is that, after beginning a new computation, at each return symbol B has to be able to recognize whether or not the matching call symbol occurred before the last nondeterministic guess where B began to simulate a new computation of C. In the affirmative case, B treats the symbol as a pending return.

The linear states of $Q_{(1)}$ are used to indicate that the currently simulated computation of C has no pending calls. Note that a nondeterministic choice beginning a new computation always enters a state of $[Q_0]_{(1)}$. When encountering a return symbol in a state of $Q_{(1)}$, according to rules (7), B interprets incoming hierarchical states as arbitrary initial hierarchical states.

When a linear state of $Q_{(1)}$ encounters a call symbol, according to rules (6) with $j = 1$, the new linear state will be in $Q_{(2)}$ and B sends along the hierarchical edge a state of $P_{(1)}$. A state of $P_{(1)}$ is used to mark an "outermost" hierarchical edge in the currently simulated computation of C. According to rules (6) with $j = 2$, B sends a state of $P_{(2)}$ along a hierarchical edge if the current computation has earlier pending calls. When a return transition receives along the hierarchical edge a state of $P_{(1)}$, B knows that the current computation again has no pending calls and the new linear state, according to (8) with $j = 1$, will be an element of $Q_{(1)}$.

Note that the definition of the set Y_1 in (6) allows as the second component any element of $P_{(1)}$. A nondeterministic choice corresponding to Y_1 means that B begins to simulate a new computation of C and a hierarchical state p produced here will be ignored at the matching return symbol (if it exists). Since all hierarchical states are final, here p can be chosen to be arbitrary.

Since C is linearly accepting and, as described above, B is able to keep track of whether a hierarchical state received as second argument for a return transition originates from the currently simulated computation or should be interpreted as an initial hierarchical state, it is clear that B recognizes the language $L(C)^*$. The NNWA B has $4n$ linear and $4h$ hierarchical states. ■

We do not have a matching lower bound for Proposition 3.2 and the precise nondeterministic state complexity of Kleene star remains open.

Acknowledgements

The first author's research supported in part by the IT R&D program of MKE/IITA 2008-S-024-01 and the KRCF research grant. The second author's research supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] R. Alur, V. Kumar, P. Madhusudan, M. Viswanathan, Congruences for visibly pushdown languages, in: Proceedings of 32nd ICALP, in: Lect. Notes Comput. Sci., vol. 3580, Springer, 2005, pp. 1102–1114.
[2] R. Alur, P. Madhusudan, Adding nesting structure to words, in: O.H. Ibarra, Z. Dang (Eds.), Developments in Language Theory, DLT 2006, in: Lect. Notes Comput. Sci., vol. 4036, Springer, 2006, pp. 1–13.
[3] R. Alur, P. Madhusudan, Adding nesting structure to words. Full version of [2]. Currently available at www.cis.upenn.edu/~alur/Stoc04Dlt06.pdf.
[4] M. Arenas, P. Barceló, L. Libkin, Regular languages of nested words: Fixed points, automata and synchronization, in: Proceedings of 34th ICALP, in: Lect. Notes Comput. Sci., vol. 4596, Springer, 2007, pp. 888–900.
[5] J.C. Birget, Intersection and union of regular languages and state complexity, Inform. Process. Lett. 43 (1992) 185–190.
[6] M. Domaratzki, K. Salomaa, Lower bounds for the transition complexity of NFAs, J. Comput. System. Sci. 74 (2008) 1116–1130.
[7] I. Glaister, J. Shallit, A lower bound technique for the size of nondeterministic finite automata, Inform. Process. Lett. 59 (1996) 75–77.
[8] H. Gruber, M. Holzer, Inapproximability of nondeterministic state and transition complexity assuming P ≠ NP, in: T. Harju, J. Karhumäki, A. Lepistö (Eds.), Developments in Language Theory, DLT 2007, in: Lect. Notes Comput. Sci., vol. 4588, Springer-Verlag, 2007, pp. 205–216.
[9] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, Internat. J. Found. Comput. Sci. 14 (2003) 1087–1102.
[10] J. Hromkovič, Communication Complexity and Parallel Computing, Springer-Verlag, 1997.
[11] J. Hromkovič, S. Seibert, J. Karhumäki, H. Klauck, G. Schnitger, Communication complexity method for measuring nondeterminism in finite automata, Inform. and Comput. 172 (2002) 202–217.
[12] G. Jirásková, State complexity of some operations on binary regular languages, Theoret. Comput. Sci. 330 (2005) 287–298.
[13] V. Kumar, P. Madhusudan, M. Viswanathan, Minimization, learning, and conformance testing of Boolean programs, in: C. Baier, H. Hermanns (Eds.), CONCUR 2006, in: Lect. Notes Comput. Sci., vol. 4137, Springer, 2006, pp. 203–217.

[14] X. Piao, K. Salomaa, Operational state complexity of nested word automata, Descriptional Complexity of Formal Systems, DCFS 2008, Charlottetown, Canada, July 16–18, 2008, pp. 194–206 (Full version accepted for publication in Theoretical Computer Science).
[15] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. I, Springer-Verlag, 1997.
[16] K. Salomaa, Descriptional complexity of nondeterministic finite automata, in: T. Harju, J. Karhumäki, A. Lepistö (Eds.), Developments in Language Theory, DLT 2007, in: Lect. Notes Comput. Sci., vol. 4588, Springer, 2007, pp. 31–35.
[17] S. Yu, Regular languages, in [15], pp. 41–110.
[18] S. Yu, State complexity: Recent results and open problems, Fund. Inform. 64 (2005) 471–480.