CS–2008–05

State-Complexity Hierarchies of Uniform Languages of Alphabet-Size Length

Janusz Brzozowski and Stavros Konstantinidis

Technical Report 05

David R. Cheriton School of Computer Science
University of Waterloo

March 11, 2008
Revised April 20, 2008

State-Complexity Hierarchies of Uniform Languages of Alphabet-Size Length *

Janusz Brzozowski
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1
brzozo@uwaterloo.ca
and
Stavros Konstantinidis
Department of Mathematics and Computing Science
Saint Mary's University
Halifax, NS, Canada B3H 3C3
s.konstantinidis@smu.ca

April 20, 2008

Abstract

We study the state complexity of a class of simple languages. If A is an alphabet of k letters, a *k-language* is a nonempty set of words of length k, that is, a uniform language of length k. By a new construction, we show that the maximal state complexity of a k-language is $(k^{k-1}-1)/(k-1)+2^k+1$, and every k-language of this complexity is also a uniform language of length k of the maximal state complexity previously known. We then prove that, for every i between minimal and maximal complexities, there is a language of complexity i: for each i we exhibit such a language. We introduce "pi automata" accepting languages whose words are permutations of the alphabet; the complexities of these languages form a complete hierarchy between k^2-k+3 and 2^k+1. We start with an automaton with k^2-k+3 states and show that states can be added one at a time, until the automaton has 2^k+1 states. We construct another class of automata, based on k-ary trees, whose languages define a complete hierarchy of complexities between 2^k+1 and the maximal complexity. Here, we start with an automaton with the maximal number of states and remove states one at a time, until an automaton with 2^k+1 states is reached.

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grants no. OGP000871 and R220259.

1 Introduction

State complexity has received considerable attention recently [2]–[6]. In particular, the problem of finding a tight upper bound on the state complexity of a uniform language of length n has been considered in [2, 4]. We study a special class of uniform languages, namely, the nonempty languages over an alphabet A of cardinality k in which all the words are of length k. Such languages, called *k-languages,* have two advantages. First, the special form of a k-language makes it easier to reason about its state complexity. Second, in spite of their simplicity, k-languages exhibit a complete hierarchy of state complexities.

The paper is organized as follows. The next section contains our basic terminology and notation. Section 3 defines k-languages and shows a tight upper bound of $(k^{k-1}-1)/(k-1)+2^k+1$ on the state complexity of these languages; this bound coincides with the maximal possible state complexity of languages of length k as shown in [2, 4]. Sections 4–7 demonstrate constructively that there are k-languages of state complexity i for every i between the minimum and maximum possible state complexities. In particular, Section 4 does this for all i between complexities $k+2$ and k^2-k+3. Section 5 defines Pi automata, which are used in Section 6 to establish all state complexities between k^2-k+3 and 2^k+1. Finally, Section 7 shows a construction for automata with complexities between 2^k+1 and $(k^{k-1}-1)/(k-1)+2^k+1$. Section 8 concludes the paper.

2 Terminology and notation

The cardinality of a set S is denoted by $|S|$. If A is an alphabet, then A^* denotes the free monoid generated by A. The empty word is 1, and the length of a word $w \in A^*$ is $|w|$. If $u, v, w \in A^*$ and $w = uv$, then u is a prefix of w and v is a suffix of w. If $w = uxv$, then x is a factor of w. A language over an alphabet A is any subset of A^*. If L is a language, and $w \in \Sigma^*$, *the (left) quotient* of L by w is

$$w^{-1}L = \{x \mid wx \in L\}. \tag{1}$$

A *(deterministic finite) automaton* is a tuple $\mathcal{A} = (A, Q, q_0, \tau, F)$, where A is a finite *alphabet,* Q is a finite *set of states,* $q_0 \in Q$ is the *initial state,* $\tau : Q \times A \to Q$ is the *transition function,* and $F \subseteq Q$ is the *set of final states.* Two states p and q of $\mathcal{A}$ are *distinguishable,* if there exists a word $w \in \Sigma^*$ such that $\tau(p, w) \in F$ and $\tau(q, w) \notin F$, or *vice versa.*

For a regular language L, the number of distinct quotients is finite [1]. We define the *quotient automaton* of L as $\mathcal{A} = (A, Q, q_0, \tau, F)$, where $Q = \{w^{-1}L \mid w \in A^*\}$, $q_0 = 1^{-1}L = L$, $\tau(w^{-1}L, a) = (wa)^{-1}L$, and $F = \{w^{-1}L \mid 1 \in w^{-1}L\}$. This automaton is minimal.

The *state complexity,* $\mathbf{c}(L)$, of a language L is the number of states in the minimal deterministic automaton accepting L.

From now on we assume that the alphabet A has $k > 0$ letters.

3 k-languages and complexity bounds

A language L is *uniform (of length n)* if all the words in L are of the same length (n).

Proposition 1 *Let L_n be a uniform language of length n over A. Then*

1. *If $|w| > n$, then $w^{-1}L_n = \emptyset$.*
2. *If $|u|, |v| \leq n$ and $|u| \neq |v|$, then $u^{-1}L_n \cap v^{-1}L_n = \emptyset$.*

Proof: This is obvious. □

By Proposition 1, two words of different lengths lead to distinguishable states in the quotient automaton of L_n. Let the *level* of a state q (other than the rejecting sink state ∞ corresponding to the empty quotient) be the length of any word leading to q; thus 0 is the level of the start state, and k is the level of the single final state. The level of ∞ is $k + 1$.

The problem of finding a tight upper bound on the state complexity of a uniform language of length n has been considered in [2, 4], where it has been shown that this bound is

$$C_{k,n} = \frac{k^r - 1}{k - 1} + \sum_{j=0}^{n-r}(2^{k^j} - 1) + 1, \tag{2}$$

where $r = \min\{m \mid k^m \geq 2^{k^{n-m}} - 1\}$.

We study a special class of uniform languages called *k-languages.* These are nonempty languages over an alphabet of cardinality k in which all the words are of length k. We denote by $\mathbf{K}_k$ the set of all k-languages.

Theorem 1 *The following bounds hold for k-languages, $k \geq 1$:*

1. *If $L \in \mathbf{K}_1$, then $\mathbf{c}(L) = 3$.*
2. *If $L \in \mathbf{K}_k$, then $k + 2 \leq \mathbf{c}(L)$, and the bound is reachable.*
3. *If $L \in \mathbf{K}_2$, then $\mathbf{c}(L) \leq 5$, and the bound is reachable.*
4. *For $k \geq 3$, if $L \in \mathbf{K}_k$, the bound below is reachable.*

$$\mathbf{c}(L) \leq B_k = \frac{k^{k-1} - 1}{k - 1} + 2^k + 1, \tag{3}$$

Proof:

1. Let $A = \{a\}$; there is only one language $L = \{a\}$ in $\mathbf{K}_1$, and $\mathbf{c}(L) = 3$.

2. By definition, a k-language L is nonempty. Suppose $w = a_1 \cdots a_k \in L$. Then $(a_1 \cdots a_i)^{-1}L$ is nonempty for all $i = 0, \ldots, k$. By Proposition 1, all these $k+1$ quotients are distinct, and there is also an empty quotient with respect to any word of length $> k$. Hence $k+2 \leq \mathbf{c}(L)$. The language $L_{min}(k) = \{a^k\}$, meets the bound, for any $a \in A$.

3. Let $A = \{a, b\}$; if $L \in \mathbf{K}_2$, L is a nonempty subset of $\{aa, ab, ba, bb\}$. Any such L has one quotient by the empty word 1, and at most two by words of length 1. If it is nonempty, the quotient by any word of length 2 is 1, and there is also the empty quotient. Thus there are at most 5 distinct quotients. One verifies that $\{aa, bb\}$ has complexity 5.

4. Let $A = \{a_1, \ldots, a_k\}$. We have $w^{-1}L = \emptyset$, if $|w| > k$, and $w^{-1}L = 1$, if $w \in L$ (and hence $|w| = k$). Next, $w^{-1}L \subseteq A$ if $|w| = k - 1$; since A has cardinality k, there are at most $2^k - 1$ such quotients which are nonempty. For any i, $0 \leq i \leq k-2$, there are at most k^i distinct quotients, since there are k^i words of length i. Altogether, we have an upper bound of

$$1 + 1 + (2^k - 1) + (1 + k + \cdots + k^{k-2}) = \frac{k^{k-1} - 1}{k - 1} + 2^k + 1.$$

Now consider the language $L_{max}(k)$ defined by the automaton

$$\mathcal{A}_{max}(k) = (A, Q, q_0, \tau, F), \text{where}$$

- $Q = R \cup P \cup \{f\} \cup \{\infty\}$,
- $R = \{w \in A^* \mid |w| \leq k-2\}$,
- $P = \{S \mid S \subseteq A, S \neq \emptyset\}$,
- $f \notin R \cup P$ is the only final state,
- $\infty \notin R \cup P \cup \{f\}$ is the rejecting sink state,
- $q_0 = 1$,
- $F = \{f\}$, and
-

$$\tau(q,a) = \begin{cases} qa & \text{for all } q = w \in R,\ |w| \leq k-3, \\ \text{is defined below} & \text{if } q = w \in R,\ |w| = k-2, \\ f & \text{for all } q = S \in P,\ a \in S, \\ \infty & \text{otherwise.} \end{cases}$$

It remains to define τ for states of level $k-2$. There are $s = k^{k-2}$ words of length $k-2$, and $r = (2^k)^k - 1$ ordered k-tuples of states chosen from the set of 2^k subsets of A, such that at least one component of each k-tuple is nonempty. We claim that there are at least as many k-tuples as there are words of length $k-2$; since $2^k > k$, we have $(2^k)^k > (2^k)^{k-2} > k^{k-2}$. Hence $(2^k)^k \geq k^{k-2} + 1$, and

$$r = (2^k)^k - 1 \geq k^{k-2} = s.$$

Enumerate the k-tuples $t_1, \ldots, t_r$ in such a way that all the $2^k - 1$ different subsets of A appear in the first k^{k-2} k-tuples; this is possible as $k \cdot k^{k-2} \geq 2^k - 1$, for all $k \geq 3$. Each k-tuple t_i has the form $t_i = (S_{i_1}, \ldots, S_{i_k})$, where S_{i_j} is a subset of A for $j = 1, \ldots, k$. Order the words of length $k-2$ in some way, say $u_1, \ldots, u_s$, and assign to u_i the k-tuple t_i. Since $r \geq s$, each word of length $k-2$ is assigned a unique k-tuple. If $S_{i_j} \neq \emptyset$, define $\tau(u_i, a_j) = S_{i_j}$; otherwise, $\tau(u_i, a_j) = \infty$. It follows that

$$u_i{}^{-1} L_{max}(k) = \bigcup_{j=1}^{k} a_j S_{i_j}.$$

Since each u_i has a distinct k-tuple, these quotients are distinct.

Notice that the states in R form a full k-ary tree in which the leaf nodes correspond to the k^{k-2} distinct quotients. We claim that *all* the states in the tree correspond to distinct quotients. Suppose $|u| < k-2$ and $u^{-1}L_{max}(k) = v^{-1}L_{max}(k)$, for some $v \neq u$. By Proposition 1, we must have $|u| = |v|$. Let w be any word such that $|uw| = k-2$; because we have a full k-ary tree, uw is a different state from vw. Since all the states in level $k-2$ are distinguishable, so are $u^{-1}L_{max}(k)$ and $v^{-1}L_{max}(k)$, and we have reached a contradiction. Consequently, $L_{max}(k)$ meets the upper bound. □

When the length n of the words in a uniform language is the same as the size k of the alphabet, expression (2) becomes

$$C_{k,k} = \frac{k^r - 1}{k-1} + \sum_{j=0}^{k-r} (2^{k^j} - 1) + 1, \tag{4}$$

where $r = \min\{m \mid k^m \geq 2^{k^{k-m}} - 1\}$. We now show that the bound found using our construction coincides with $C_{k,k}$:

Proposition 2 *The bound B_k of Theorem 1 coincides with the bound $C_{k,k}$.*

Proof: One verifies that, if $r = k-1$, then $B_k = C_{k,k}$. Next we show that r must be equal to $k-1$. We use the fact that $2^k > k+1$ for all $k > 1$. First we prove that $r \leq k-1$. For this it is sufficient to show that $1 + k^{k-1} \geq 2^{k^{k-(k-1)}} = 2^k$. The case of $k = 3$ holds. Assume that the statement is true for some $k \geq 3$. Then,

$$1 + (k+1)^k > 1 + k^k \geq 1 + 3k^{k-1} = 1 + k^{k-1} + 2k^{k-1} \geq 2^k + 2 \cdot 2^{k-1} = 2^{k+1}.$$

We prove that $r \geq k-1$ by contradiction. Assume that $r \leq k-2$. As $2^k > k+1$, we have $(2^k)^k > (k+1)^k$, and hence $2^{k^2} > k^k + 1$ and $2^{k^{k-r}} > k^k + 1$, using the assumption $k - r \geq 2$. But $1 + k^r < 1 + k^k < 2^{k^{k-r}}$, contradicting the definition of r. Hence, $r \geq k-1$. Altogether, $r = k-1$. □

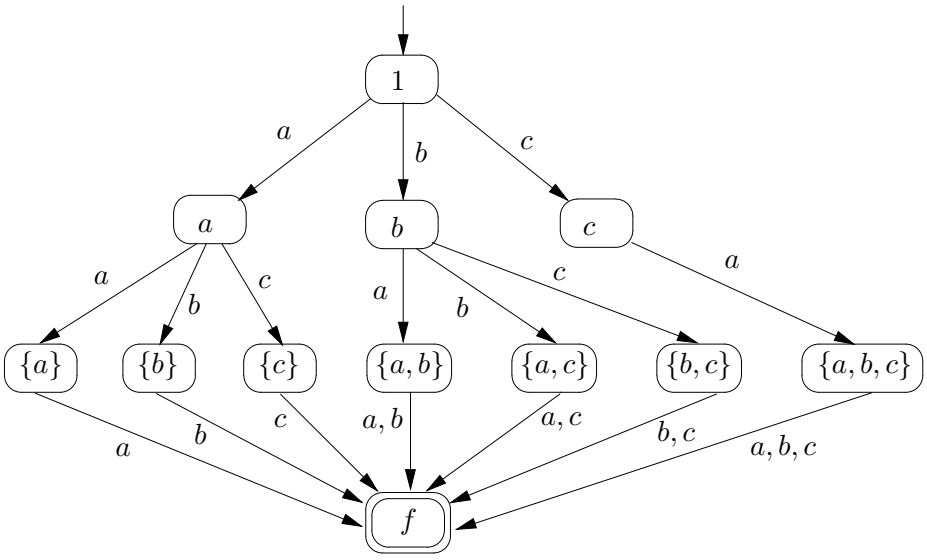

Figure 1: Automaton $\mathcal{A}_{max}(3)$, state ∞ not shown.

Example 1 *Consider the automaton $\mathcal{A}_{max}(3)$ of Fig. 1. The states corresponding to words of length $\leq k-2 = 1$ are 1, a, b, and c, and form a ternary tree. The states at level $k-1 = 2$ are the nonempty subsets of $\{a, b, c\}$, and f accepts $\{1\}$. State ∞ and transitions to it are not shown.*

We assign the triple $(\{a\}, \{b\}, \{c\})$ to state a, $(\{a, b\}, \{a, c\}, \{b, c\})$ to state b, and $(\{a, b, c\}, \emptyset, \emptyset)$ to state c. Of course, there are many ways to assign such triples, since there are $(2^3)^3 - 1 = 511$ possible triples. The state complexity of $\mathcal{A}_{max}(3)$ meets the upper bound of 13 for $k = 3$. □

4 From k-complexity to k^2-complexity

It is our aim to show that for every i, such that $k+2 \leq i \leq B_k$, there exists a language of state complexity i. We do this in several steps. We first study languages with state complexities between $k+2 = 1(k-1)+3$ and $k(k-1)+3$, which are referred to as *k-complexity* and *k^2-complexity,* respectively. Note that, for $k = 1$, k-complexity is both the upper and the lower bound, since the complexity of the only 1-language is $k + 2 = 3$. Also, k^2-complexity coincides with k-complexity for $k = 1$.

Proposition 3 *Let $k \geq 1$ and $L_{powers}(k) = \bigcup_{a \in A}\{a^k\}$; then $\mathbf{c}(L_{powers}(k)) = k^2 - k + 3$.*

Proof: One verifies that the quotients of $L_{powers}(k)$ with respect to words in the set $\{a^i \mid a \in A, 0 \leq i \leq k-1\}$ are all nonempty and pairwise distinct, and each contains a word of length > 0; this gives $1 + k(k-1)$ states. Moreover, $(a^k)^{-1}L_{powers}(k) = 1$, for all $a \in A$, and $w^{-1}L_{powers}(k) = \emptyset$ for all other words. Hence the proposition follows. □

Proposition 4 *The following properties hold for $\mathbf{K}_k$:*

1. *For each i such that $1 \leq i \leq k$ there is a language L_i in $\mathbf{K}_k$ with complexity $i(k-1)+3$.*
2. *For each $i \leq k-1$ and j such that $i(k-1)+3 \leq j \leq (i+1)(k-1)+3$ there is a language L_j in $\mathbf{K}_k$ with complexity j.*

Proof:

1. Let $A = \{a_1, \ldots, a_k\}$. The language $\{a_1^k, \ldots, a_i^k\}$ has complexity $i(k-1)+3$. The proof is similar to that of Proposition 3.
2. The language $\{a_1^k, \ldots, a_{i+1}^k\}$ has complexity $(i+1)(k-1)+3$. To reduce the number of states by 1 we can use $\{a_1^{k-1}, a_2^{k-1}\}A \cup \{a_3^k, \ldots, a_{i+1}^k\}$. To reduce the number of states by 2, use $\{a_1^{k-2}, a_2^{k-2}\}A^2 \cup \{a_3^k, \ldots, a_{i+1}^k\}$. This approach works until we get $\{a_1, a_2\}A^{k-1} \cup \{a_3^k, \ldots, a_{i+1}^k\}$, which results in a reduction of $k-1$ states and the language has complexity $i(k-1)+3$, as required. □

Corollary 1 *For each i such that $k+2 \leq i \leq k^2 - k + 3$ there is a language with complexity i.*

5 Pi automata

We now introduce a class of automata, all of which have the same form, and differ only in the state set and the transition function. These automata permit us to show that for every i, such that $k^2 - k + 3 \leq i \leq 2^k + 1$, there exists a language of state complexity i.

Definition 1 *Let $k \geq 1$, and let $A = \{a_1, a_2, \ldots, a_k\}$ be an alphabet. A* pi automaton *is an automaton $\mathcal{A}_\pi(k) = (A, Q_\pi, \emptyset, \tau_\pi, \{A\})$, where*

- *$Q_\pi = R_\pi \cup \{\infty\}$ is the set of states,*
- *∞ is a rejecting sink state,*
- *R_π is a subset of 2^A which contains $\emptyset$ and A,*
- *for any $q \in Q_\pi$, $a \in A$,*

$$\tau_\pi(q, a) = \begin{cases} q \cup \{a\} & \text{if } q \in R_\pi,\ a \notin q, \text{ and } q \cup \{a\} \in R_\pi, \\ \infty & \text{otherwise.} \end{cases}$$

Moreover, $\mathcal{A}_\pi(k)$ satisfies the following conditions:

- *For every state q other than $\emptyset$ and ∞ there is a* predecessor *state p such that $\tau(p, a) = q$, for some $a \in q \setminus p$.*
- *For every state q other than A and ∞ there is a* successor *state s such that $\tau(q, a) = s$, for some $a \in s \setminus q$.*

Proposition 5 *Every pi automaton $\mathcal{A}_\pi(k)$ accepts a uniform language of length k in which all the words are permutations of the alphabet A. Moreover, $\mathcal{A}_\pi(k)$ is minimal.*

Proof: Since every state q, other that $\emptyset$ and ∞, has a predecessor p such that $|p| = |q| - 1$, it follows that there is a path from the initial state to q spelling some word u. The empty path spelling 1 takes state $\emptyset$ to itself. Thus, for any state $q \in R_\pi$ there is a path from the initial state to q. Moreover, if there is a transition from a state p to a state q, then $q = p \cup \{a\}$, for some $a \notin p$. Hence any word u from the initial state to any state $q \in R_\pi$ is a permutation of the letters of q. For $q = A$, every word taking the initial state to A is a permutation of all the letters of the alphabet A, as required.

Dually, since every state $q \in R_\pi \setminus \{A\}$ has a successor s such that $|s| = |q| + 1$, and $\tau(q, s) = q \cup \{a\}$, for some $a \notin q$, there is a path from q to A spelling a word v

which is a permutation of the letters of $A \setminus q$. The empty path from A to A spells the word 1 over the empty alphabet. Thus every state in R_π is distinguishable from every other state in R_π. Also, every state in R_π is distinguishable from state ∞, which accepts no words. Therefore $\mathcal{A}_\pi(k)$ is minimal. □

A language is called a *permutation language* if it is accepted by a pi automaton.

Example 2 *Figure 2 shows three pi automata, where we omit the rejecting state ∞ and all the transitions to it. Also, we do not show the letter causing a transition from a state p to a state q, since that letter is the only letter which is in q but not in p.*

In Fig. 2 (a), we show the only pi automaton over the 1-letter alphabet $\{a\}$; it accepts the language $L = \{a\}$. The automaton in Fig. 2 (b) is a pi automaton over the 2-letter alphabet $\{a, b\}$ and it accepts $L = \{ab, ba\}$. The language accepted by the pi automaton of Fig. 2 (c) is $L = \{abcd, bacd\}$. □

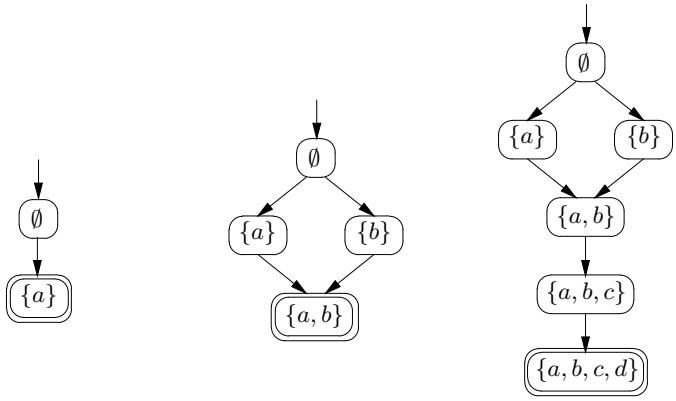

Figure 2: Pi automata.

6 From k^2-complexity to 2^k-complexity

We now consider the hierarchy of languages with complexities between $k^2 - k + 3$ and $2^k + 1$; we call the latter 2^k-*complexity.*

6.1 The automaton $\mathcal{A}_0(k)$

Let $A = \{a_1, a_2, \ldots, a_k\}$, and define the *circular order* to be that of the word $x_k = a_1a_2\cdots a_ka_1a_2\cdots a_{k-1}$. For $i = 0, \ldots, k$, let C_i be the set of all subsets S of A of cardinality i, such that the letters of S can be arranged to form a factor of length i of x_k. Such subsets are called *circular;* otherwise, they are *noncircular*. In case $S = \emptyset$, we consider it to be circular, and the corresponding factor is 1.

For example, if $A = \{a, b, c, d\}$, then $x_4 = abcdabc$, word ac is in circular order, but is not circular, while cd and dab are. Hence $\{a, c\}$ is noncircular, whereas $\{c, d\}$ and $\{a, b, d\}$ are circular.

We now define a pi automaton that uses circular sets as states. Let $\mathcal{A}_0(k) = (A, Q_0, \emptyset, \tau_0, \{A\})$, where $Q_0 = R_0 \cup \{\infty\}$, $R_0 = \bigcup_{i=0}^{k} C_i$, and for any $q \in Q_0$, $a \in A$,

$$\tau_0(q, a) = \begin{cases} q \cup \{a\} & \text{if } q \in R_0,\ a \notin q, \text{ and } q \cup \{a\} \in R_0, \\ \infty & \text{otherwise.} \end{cases}$$

The language $L_{cir}(k)$ is defined to be the language accepted by $\mathcal{A}_0(k)$.

Example 3 *For $k \leq 3$, all subsets of A are circular. The automata $\mathcal{A}_0(1)$ and $\mathcal{A}_0(2)$ are shown in Figs. 2 (a) and (b), respectively. The language $L_{cir}(3) = \{abc, acb, bac, bca, cab, cba\}$ consists of all the permutations of $\{a, b, c\}$.*

For $k = 4$, let $A = \{a, b, c, d\}$. There are 14 circular subsets; we list them in groups of the same cardinality: $C_0 = \{\emptyset\}$, $C_1 = \{\{a\}, \{b\}, \{c\}, \{d\}\}$, $C_2 = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}$, $C_3 = \{\{a, b, c\}, \{b, c, d\}, \{c, d, a\}, \{d, a, b\}\}$, and $C_4 = \{\{a, b, c, d\}\}$. There are two noncircular subsets $\{a, c\}$ and $\{b, d\}$.

The automaton $\mathcal{A}_0(4)$ is shown in Fig. 3 (a), where we represent states by words instead of sets of letters to make their circularity explicit.

For $k = 5$, there are 10 noncircular subsets of $A = \{a, b, c, d, e\}$: $\{a, c\}$, $\{b, d\}$, $\{c, e\}$, $\{d, a\}$, $\{e, b\}$, $\{a, b, d\}$, $\{b, c, e\}$,$\{c, d, a\}$, $\{d, e, b\}$, $\{e, a, c\}$. □

Proposition 6 *The language $L_{cir}(k)$ of $\mathcal{A}_0(k)$ has complexity $k^2 - k + 3$.*

Proof: There is one circular set of cardinality 0 and one of cardinality k. For each cardinality i, $1 \leq i \leq k - 1$, there are k circular sets, for a total of $2 + (k - 1)k$. Adding state ∞, $\mathcal{A}_0(k)$ has $k^2 - k + 3$ states. It is minimal by Proposition 5. □

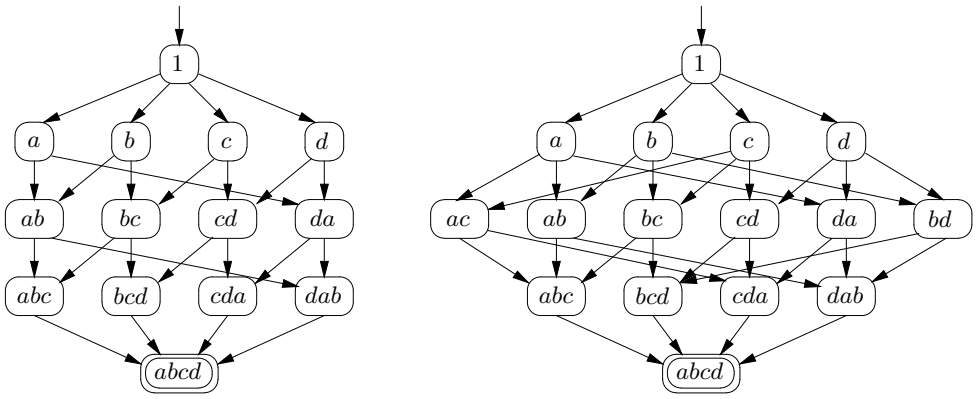

Figure 3: Automata: (a) $\mathcal{A}_0(4)$; (b) $\mathcal{A}_1(4) = \mathcal{A}_{all}(4)$.

6.2 The automaton $\mathcal{A}_{all}(k)$

Next, we introduce a pi automaton $\mathcal{A}_{all}(k)$ with $2^k + 1$ states; the state set of this automaton consists of all the 2^k subsets of A plus the sink state ∞. Let $\mathcal{A}_{all}(k) = (A, Q_{all}, \emptyset, \tau_{all}, \{A\})$, where $Q_{all} = R_{all} \cup \{\infty\}$, $R_{all} = 2^A$, and for any $q \in Q_{all}$, $a \in A$,

$$\tau_{all}(q, a) = \begin{cases} q \cup \{a\} & \text{if } q \in R_{all} \text{ and } a \notin q, \\ \infty & \text{otherwise.} \end{cases}$$

The language $L_{all}(k)$ is defined to be the language accepted by $\mathcal{A}_{all}(k)$.

Proposition 7 *The language $L_{all}(k)$ consists of all the words which are permutations of A, and has complexity $2^k + 1$.*

Proof: $\mathcal{A}_{all}(k)$ has $2^k + 1$ states and is minimal by Proposition 5. □

The automaton of Fig. 3 (b) is $\mathcal{A}_{all}(4)$.

6.3 A hierarchy of pi automata

Let $q \subseteq A$. The *predecessor distance* $d_p(q)$ of q is the minimum number of letters that have to be removed from q to obtain a circular set. Similarly, the *successor distance* $d_s(q)$ of q is the minimum number of letters that have to be added to q to obtain a circular set. The *distance* $d(q)$ of q is defined as the maximum of the predecessor and successor distances. Thus all circular sets have distance 0. For

$A = \{a, b, c, d, e, f, g, h\}$, the set $\{c, f, h\}$ has predecessor distance 2 and successor distance 3.

We now define a class of automata inductively. The basis is the automaton $\mathcal{A}_0(k)$ above. Given an automaton $\mathcal{A}_i(k) = (A, Q_i, \emptyset, \tau_i, \{A\})$, define $\mathcal{A}_{i+1}(k) = (A, Q_{i+1}, \emptyset, \tau_{i+1}, \{A\})$, where $Q_{i+1} = Q_i \cup R_{i+1}$, $R_{i+1} = \{q \mid d(q) = i + 1\}$, and for any $q \in Q_{i+1}$, $a \in A$,

$$\tau_i(q, a) = \begin{cases} q \cup \{a\} & \text{if } q \neq \infty,\ a \notin q, \text{ and } q \cup \{a\} \in Q_{i+1}, \\ \infty & \text{otherwise.} \end{cases}$$

Proposition 8 *For every $k \geq 1$, there exists an integer n_k, $0 \leq n_k < k$ such that $\mathcal{A}_{n_k}(k) = \mathcal{A}_{n_k+1}(k) = \mathcal{A}_{all}(k)$.*

Proof: Automaton $\mathcal{A}_0(k)$ contains all the states of distance 0. If $\mathcal{A}_i(k)$ contains all the states of distance $\leq d$, then $\mathcal{A}_{i+1}(k)$ contains all the states of distance $\leq d + 1$. Since the distance is certainly less than k, we eventually include all the states, and obtain automaton $\mathcal{A}_{all}(k)$. □

6.4 From $\mathcal{A}_i(k)$ to $\mathcal{A}_{i+1}(k)$

Recall that $x_k = a_1 a_2 \cdots a_k a_1 a_2 \cdots a_{k-1}$ is the word that defines the circular order. Suppose we have a set q that is not circular, and suppose the predecessor and successor distances of q are $d_p(q)$ and $d_s(q)$, respectively.

After we add the smallest number, $d_s(q)$, of letters to q so that it becomes circular, we obtain a circular word v which is a factor of x_k. We can think of q as being represented by a word $v_\#$, which is v with each missing letter replaced by #. All such words $v_\#$ have the same length, if the #s represent the smallest number of missing letters. By definition, it is possible to remove $d_p(q)$ letters from q, and hence also from $v_\#$, to obtain another circular word u, which is a factor of v.

Example 4 *If $A = \{a, \ldots, i\}$ and $q = \{b, e, g, h\}$, then $d_p(q) = 2$, $d_s(q) = 3$, $d(q) = 3$, and q is in $\mathcal{A}_3(9)$. Here we can choose $v_\# = e\#gh\#\#b$, and $v = efghiab$ or $v'_\# = b\#\#e\#gh$ and $v' = bcdefgh$. To get a circular word by removing letters, we must remove b and e to obtain the word $u = gh$.*

Consider the predecessors of q. We have $p_1 = \{e, g, h\}$, $p_2 = \{b, g, h\}$, $p_3 = \{b, e, h\}$, and $p_4 = \{b, e, g\}$. Their predecessor and successor distances are $d_p(p_1) = 1$, $d_s(p_1) = 1$, $d_p(p_2) = 1$, $d_s(p_2) = 2$, $d_p(p_3) = 2$, $d_s(p_3) = 4$, and $d_p(p_4) = 2$, $d_s(p_4) = 3$. Hence $d(p_1) = 1$, $d(p_2) = 2$, $d(p_3) = 4$, and $d(p_4) = 3$, and only p_1 and p_2 are in $\mathcal{A}_2(9)$.

Now consider the successors of q. We have $s_1 = \{a, b, e, g, h\}$, $s_2 = \{b, c, e, g, h\}$, $s_3 = \{b, d, e, g, h\}$, $s_4 = \{b, e, f, g, h\}$, and $s_5 = \{b, e, g, h, i\}$. Their predecessor and

successor distances are $d_p(s_1) = 3$, $d_s(s_1) = 2$, $d_p(s_2) = 3$, $d_s(s_2) = 2$, $d_p(s_3) = 3$, $d_s(s_3) = 2$, $d_p(s_4) = 1$, $d_s(s_4) = 2$, *and* $d_p(s_5) = 2$, $d_s(s_5) = 2$. *Hence* $d(s_1) = 3$, $d(s_2) = 3$, $d(s_3) = 3$, $d(s_4) = 2$, *and* $d(s_5) = 2$, *and only* s_4 *and* s_5 *are in* $\mathcal{A}_2(9)$.

If we want to add q *to pi automaton* $\mathcal{A}_2(9)$, *then we can use* $\tau(p_1, b) = q$ *or* $\tau(p_2, e) = q$, *and* $\tau(q, f) = s_4$ *or* $\tau(q, i) = s_5$.

The set $q = \{a, b, e, f\}$ *over the alphabet* $\{a, \ldots, i\}$ *has* $d_p(q) = d_s(q) = d(q) = 2$, *and hence belongs to* $\mathcal{A}_2(9)$. *However, no predecessor of* q *belongs to* $\mathcal{A}_1(9)$. *This shows that it is not possible to add states of distance* $i + 1$ *to* $\mathcal{A}_i(k)$ *in an arbitrary order to obtain* $\mathcal{A}_{i+1}(k)$. □

Lemma 1 *Suppose* $q \subseteq A$, *and* $d_p(q), d_s(q) \geq 1$. *Then there exists a predecessor* p *of* q *such that* $d_p(p) = d_p(q) - 1$ *and* $d_s(p) \leq d_s(q)$. *Dually, there exists a successor* s *of* q, *such that* $d_s(s) = d_s(q) - 1$ *and* $d_p(s) \leq d_p(q)$.

Proof: If $d_p(q), d_s(q) \geq 1$, then $v_\#$ of q has the form $c_1\#^i c_2$ or $c_1\#^i y_\#\#^j c_2$, where $|c_1|, |c_2| \geq 1$, c_1 and c_2 are circular, $i, j \geq 1$, and $y_\#$ is a factor of $v_\#$. We call c_1 and c_2 the *end blocks* of $v_\#$. Construct a predecessor p of q in one of the following ways:

1. If $|c_1| \leq |c_2|$, remove the first letter of c_1.
2. If $|c_1| > |c_2|$, remove the last letter of c_2.

Suppose $|c_1| \leq |c_2|$. If $v_\# = c_1\#^i c_2$, then $|c_1|$ is the predecessor distance of q. If we remove the first letter a of $c_1 = ac_1'$ to obtain a predecessor p, then $d_p(p) = |c_1'|$. Thus the predecessor distance decreases by 1. In the second case, let $|y_\#|_A$ be the number of letters in $y_\#$. If $v_\# = c_1\#^i y_\#\#^j c_2$ and $|y_\#|_A \leq |c_2|$, then the predecessor distance of q is $|c_1| + |y_\#|_A$. If $v_\# = c_1\#^i y_\#\#^j c_2$ and $|y_\#|_A > |c_2|$, then the predecessor distance of q is $|c_1| + |c_2|$. In either case, it necessary to remove c_1 to define the predecessor distance. If we remove the first letter a of $c_1 = ac_1'$, then the predecessor again distance decreases by 1.

A similar argument works if $|c_1| > |c_2|$, and we remove the last letter of c_2 to get a predecessor p.

In both of these cases, the number of #s that have to be replaced by letters cannot possibly increase, since the letter removed from an end block need not be replaced to get a circular word. Hence $d_s(p) \leq d_s(q)$.

Let $q' = A \setminus q$; then $d_p(q) = d_s(q')$, and $d_s(q) = d_p(q')$. For suppose that, by adding a set r to q, we obtain a circular set $q \cup r$. Since the complement of a circular set is circular, we know that $A \setminus (q \cup r)$ is also circular. Now adding r to q corresponds to subtracting r from q'. Thus, by removing r from q' we get a circular set, and $d_p(q') = |r| = d_s(q)$. A similar argument works for the second claim.

Consequently, to find a successor s of q satisfying $d_s(s) = d_s(q) - 1$ and $d_p(s) \leq d_p(q)$, find a predecessor p' of q' as above. Then let $s = A \setminus p'$. □

Recall that $Q_i(k) = Q_i$ is the set of states of pi automaton $\mathcal{A}_i(k)$, for $0 \leq i \leq n_k$. For any state $q \subseteq A$, let the *sum* of q be $\sigma(q) = d_p(q) + d_s(q)$. For a fixed $i \geq 0$, define $P_j = \{q \subseteq A \mid d(q) = i + 1, \text{ and } \sigma(q) = j\}$, where $j > 0$. Let $S_{i,j} = \bigcup_{h=1}^{j} P_h$, and let $Q_{i,j} = Q_i \cup S_{i,j}$.

Lemma 2 *If $i \geq 0$, $j \geq 1$, and $q \in Q_{i,j}$, then q has a predecessor p and successor s, such that $p, q \in Q_{i,j-1}$.*

Proof: There are no states with $j = 1$, for then one of the distances $d_p(q), d_s(q)$ would have to be 0, and q would be circular. If $j = 2$, then we must have $i = 0$, and $d_p(q) = 1, d_s(q) = 1$; clearly, each such state has a circular predecessor p and a circular successor s, which are both in Q_0. Consequently, $Q_{1,2} = Q_1$. Now suppose that the lemma holds for $j \geq 2$, and consider a state $q \in P_{j+1}$; then $\sigma(q) = j + 1$. By Lemma 1, q has a predecessor p such that $d_p(p) = d_p(q) - 1$ and $d_s(p) \leq d_s(q)$; hence $\sigma(p) \leq \sigma(q) - 1$. Similarly q has a successor s with $\sigma(s) \leq \sigma(q) - 1$. □

Lemma 2 gives us the order in which nodes have to be added to $\mathcal{A}_i(k)$ to get $\mathcal{A}_{i+1}(k)$.

Example 5 *First, we illustrate the use of complements in finding successors. Let $A = \{a, \ldots, i\}$. Consider the complement of $q = \{b, e, g, h\}$, namely, $q' = \{a, c, d, f, i\}$, and find its representation as $v'_{\#} = ia\#cd\#f$. We have $d_p(q) = 2 = d_s(q')$, and $d_s(q) = 3 = d_p(q')$. To find a successor s of q with the smallest distance, we find the predecessor p' of q'. We must delete f from the end block f of q', obtaining $p' = \{a, c, d, i\}$, $d_p(p') = 2$ and $d_s(p') = 1$. Hence we have the successor $s = \{b, e, f, g, h\}$ of q with $d_p(s) = d_s(p') = 1$ and $d_s(s) = d_p(p') = 2$.*

Now we illustrate how states can be added with the aid of Lemma 2. As we have seen in Example 4, the set $q = \{a, b, e, f\}$ over the alphabet $\{a, \ldots, i\}$ belongs to $\mathcal{A}_2(9)$, but has no predecessor in $\mathcal{A}_1(9)$. Also, $d(q) = d_p(q) = d_s(q) = 2$, and $\sigma(q) = 4$. Thus q is in $Q_{1,4}$. We must consider a predecessor and a successor of q as obtained in the proof of Lemma 1. One choice is $p = \{b, e, f\}$ and $s = \{a, b, c, e, f\}$ both with distance 2 and sum 3. Hence these states are in $Q_{1,3}$, and we must continue. We find a predecessor $p_p = \{e, f\}$, which is circular, and a successor $p_s = \{b, d, e, f\}$, which has sum 2. Thus $p_p \in Q_0$ and $p_s \in Q_{1,2}$. Similarly, for $s = \{a, b, c, e, f\}$, we find a predecessor $s_p = \{a, b, c, e\} \in Q_{1,2}$ and successor $s_s = \{a, b, c, d, e, f\} \in Q_0$.

Figure 4 summarizes the order of adding the states to Q_1. Predecessors are shown on the left and successors on the right. States in Q_0 and $Q_{1,2}$ are in Q_1. Thus we first add $\{b, e, f\}$ and $\{a, b, c, e, f\}$, and then $\{a, b, e, f\}$.

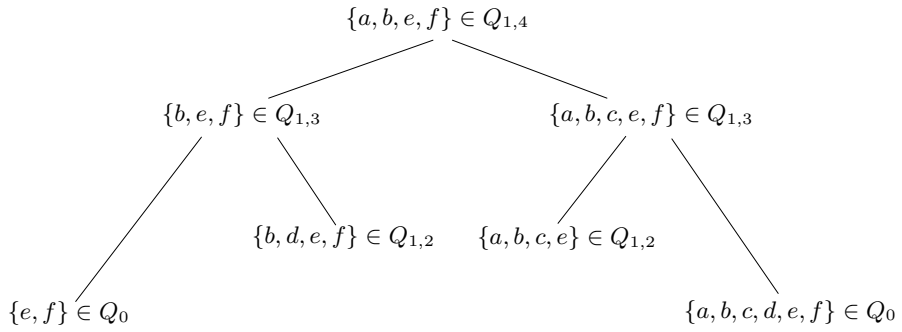

Figure 4: Adding states to Q_i.

Theorem 2 *Let $k \geq 1$. For every i such that $k^2 - k + 3 \leq i \leq 2^k + 1$, there exists a permutation k-language of state complexity i.*

Proof: There is nothing to prove for $k = 1$, so assume that $k > 1$. Automaton $\mathcal{A}_0(k)$ has $k^2 - k + 3$ states, by Proposition 6. By adding states to any minimal pi automaton $\mathcal{A}_i$ using Lemmas 1 and 2, we obtain another pi automaton, $\mathcal{A}_{i+1}$, which is still minimal. By Proposition 8, we eventually reach $\mathcal{A}_{all}(k)$, which has $2^k + 1$ states, by Proposition 7. Hence the claim holds. □

6.5 The automaton $\mathcal{A}_1(k)$

A state is *near-circular* if its distance is 1. Recall that, if either the predecessor or successor distance a state is 0, then the state is circular. Thus if the distance is 1, both predecessor and successor distances must be 1. We now define the *near-circular* language $L_{nrcir}(k)$ as the language accepted by the automaton $\mathcal{A}_1(k) = (A, Q_1, \emptyset, \tau_1, A)$.

Proposition 9 *The number of near-circular states in $\mathcal{A}_1(k)$ is 0 for $k \leq 3$, 2 for $k = 4$, and $2k^2 - 8k$ for $k > 4$.*

Proof: When $k \leq 3$, all states are circular. If $k = 4$, there are two near-circular states, as shown in Fig. 3 (b).

For $k > 4$, states of cardinality 0, 1, $k - 1$ and k are all circular. It is convenient now to represent states by words in circular order instead of sets. Suppose that $x = a_1 a_2 \cdots a_{i-1} a_i$ is in circular order and is a near-circular state of cardinality i, $1 < i < k - 1$.

Since x must become circular after some letter is added, there must exist $b \in A$ such that $w = a_1 a_2 \cdots a_j b a_{j+1} \cdots a_{i-1} a_i$ is circular. There must be a gap in the

circular order between a_i and a_1; for $a_{j+1}\cdots a_i a_1 \cdots a_j$ would be circular, otherwise. Also, b cannot be the first nor the last letter of w, because then x would be circular. Moreover, x must have a circular subset of cardinality $i-1$. This can only happen in two ways: either $a_1a_2\cdots a_{i-1}$ is circular, or $a_2a_3\cdots a_i$ is circular. Thus there are two circular words, $w' = a_1a_2\cdots a_{i-1}ba_i$ and $w'' = a_1ba_2\cdots a_{i-1}a_i$, beginning with a_1, after b is inserted.

If $i = 2$, the two ways of inserting b coincide. For $i = k-2$, after b is inserted, only one letter, say c is missing in w. Thus $ca_1a_2\cdots a_jba_{j+1}\cdots a_{i-1}a_i$ is a circular permutation of $a_1a_2\cdots a_jba_{j+1}\cdots a_{i-1}a_ic$. Consequently, for each circular $w = a_1a_2\cdots a_jba_{j+1}\cdots a_{i-1}a_i$, $u = a_{j+1}\cdots a_{i-1}a_ica_1a_2\cdots a_j$ is also circular. Hence, $a_1a_2\cdots a_ja_{j+1}\cdots a_{i-1}a_i$ and $a_{j+1}\cdots a_{i-1}a_ia_1a_2\cdots a_j$ are both near-circular and represent the same state. Thus, if we count two near-circular states beginning with each letter as above, we count each state twice. Therefore the total number of near-circular states of cardinality 2 and $k-2$ is only k, and not $2k$.

For any i such that $2 < i < k-2$, the two words w' and w'' above represent distinct states, and so the number of near-circular states of cardinality i is $2k$. There are $k-5$ lengths i such that $2 < i < k-2$.

We can now find the total number of near-circular states as follows: There are k such states for length 2 and $k-2$, and $2k$ such states for the $k-5$ other lengths. Thus the total is $2k + (k-5)2k = 2k^2 - 8k$. □

Proposition 10 *For $k \leq 3$, $L_{nrcir}(k) = L_{cir}(k)$. The state complexity of $L_{nrcir}(k)$ is 17 for $k = 4$, and $3k^2 - 9k + 3$ for $k > 4$.*

Proof: The automaton $\mathcal{A}_1(k)$ is minimal, since it is a pi automaton. The state complexity of $L_{nrcir}(k)$ is the number of near-circular states plus the state complexity of $L_{cir}(k)$, that is, $\mathbf{c}(L_{nrcir}(4)) = 17$, and, for $k > 4$, $\mathbf{c}(L_{nrcir}(k)) = 2k^2 - 8k + k(k-1) + 3 = 3k^2 - 9k + 3$. □

Example 6 *Automaton $\mathcal{A}_1(4)$ is shown in Fig. 3 (b). Automaton $\mathcal{A}_1(5)$ has $33 = 2^5 + 1$ states, and $\mathcal{A}_1(5) = \mathcal{A}_2(5) = \mathcal{A}_{all}(5)$. For $k = 6$, let $A = \{a, b, c, d, e, f\}$. There are 32 circular states, and 24 near-circular states:*

$$
\begin{aligned}
N_2 &= \{\{a,c\},\{a,e\},\{b,d\},\{b,f\},\{c,e\},\{d,f\}\} \\
N_3 &= \{\{a,b,d\},\{a,b,e\},\{a,c,d\},\{a,c,f\},\{a,d,e\},\{a,d,f\},\{b,c,e\}, \\
&\quad \{b,c,f\},\{b,d,e\},\{b,e,f\},\{c,d,f\},\{c,e,f\}\} \\
N_4 &= \{\{a,b,c,e\},\{a,b,d,f\},\{a,c,d,e\},\{a,c,e,f\},\{b,c,d,f\},\{b,d,e,f\}
\end{aligned}
$$

Thus $\mathcal{A}_1(6)$ has a total of 57 states. On the other hand, $\mathcal{A}_{all}(6)$ has 65 states; hence $\mathcal{A}_2(6) \neq \mathcal{A}_1(6)$. This leaves 8 states to be accounted for. □

6.6 The automata $\mathcal{A}_j(k)$

We do not know a formula for the cardinality of $\mathcal{A}_j(k)$, for $j > 1$. However, we have calculated some values for small k, as shown in Table 1. The dashes indicate that the automata are not needed, since the complexity of $\mathcal{A}_{all}$ has already been reached. The numbers in boldface indicate the last automaton needed to reach $\mathcal{A}_{all}$. For example, for $k = 6$, automaton $\mathcal{A}_2(6)$ has 65 states – as many as $\mathcal{A}_{all}(6)$; hence $\mathcal{A}_j(6) = \mathcal{A}_2(6)$ for all $j \geq 2$.

Table 1: Complexities for small values of k.

k	$k+2$	$\mathcal{A}_0$	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_{all}$	$\mathbf{c}_{max}$
1	**3**	–	–	–	–	–	3	3
2	4	**5**	–	–	–	–	5	5
3	5	**9**	–	–	–	–	9	13
4	6	15	**17**	–	–	–	17	38
5	7	23	**33**	–	–	–	33	189
6	8	33	57	**65**	–	–	65	1,620
7	9	45	87	**129**	–	–	129	19,737
8	10	59	123	231	**257**	–	257	299,850
9	11	75	165	363	507	**513**	513	5,381,353

7 From 2^k-complexity to maximal complexity

Before attacking the main problem of establishing a complete hierarchy between 2^k-complexity and maximal complexity, we prove three lemmas for later use. Subsection 7.1 may be omitted on first reading.

7.1 Three technical lemmas

Lemma 3 *For all integers $k \geq 2$ and $h \geq 1$,*

$$2h < \frac{k^{h+1} - 1}{k - 1}.$$

Proof: The statement follows easily by induction on h, using the fact that $k^{h+2} - 1 = k^{h+2} - k^{h+1} + k^{h+1} - 1 = k^{h+1}(k-1) + k^{h+1} - 1$ in the induction step from h to $h+1$. □

Lemma 4 *For all integers $k \geq 5$,*

$$k^{k-2} - 1 \geq \frac{k^{k-2} - 1}{k-1} + 2^k - 1.$$

Proof: One verifies that the claim holds for $k = 5$, so assume that $k \geq 6$. Let $m = (k^{k-2}-1)/(k-1)$, and suppose, for the sake of contradiction, that the lemma does not hold. Then $k^{k-2}-1 < m+2^k-1$, $(k-1)m < m+2^k-1$, and $(k-2)m < 2^k-1$. One verifies that, for $k \geq 6$, we always have $2^k - 1 \leq k^{k-3}$, and hence, $(k-2)m < k^{k-3}$. From the definition of m, we then have $(k-2)(k^{k-2}-1) < (k-1)k^{k-3}$. This is equivalent to $k^{k-1}+k^{k-3}+2 < 3k^{k-2}+k$. Since $k^{k-1} \geq 3k^{k-2}$ for $k \geq 3$, this means that $k^{k-3}+2 < k$, which is impossible for $k > 3$. Hence the lemma holds. □

Lemma 5 *For all integers $k > 0$, $k^k \geq (k+1)^{k-1}$.*

Proof: The lemma holds for $k = 1$, so assume that $k \geq 2$. The claim is equivalent to

$$k^k(k+1) \geq (k+1)^k = \sum_{i=0}^{k} \left(\begin{array}{c} k \\ i \end{array} \right) k^i,$$

which, in turn, is equivalent to

$$k^{k+1} \geq \sum_{i=0}^{k-1} \left(\begin{array}{c} k \\ i \end{array} \right) k^i.$$

So it is sufficient to show that

$$\left(\begin{array}{c} k \\ i \end{array} \right) \leq k^{k-i}$$

for $i = 0, \ldots, k-1$. This holds trivially for $i \in \{0, k-1\}$. So assume $0 < i < k-1$. As

$$\left(\begin{array}{c} k \\ i \end{array} \right) = \frac{(i+1)\cdots(k)}{(k-i)!} = \frac{(i+1)\cdots(i+k-i)}{(k-i)!},$$

we need to show $(k-i)!k^{k-i} \geq (i+1)\cdots(i+k-i)$, or

$$\left(\prod_{r=1}^{k-i} r \right) k^{k-i} = \prod_{r=1}^{k-i} (rk^{k-i}) \geq \prod_{r=1}^{k-i} (i+r).$$

For this, it is sufficient to show that $rk^{k-i} \geq i+r$ for all $r = 1, \ldots, k-i$. The smallest value of rk^{k-i} occurs when $r = 1$, and $i = k-2$, and that value is $1 \cdot k^{k-(k-2)} = k^2$. The largest value of $i+r$ occurs when $r = k-i$, and that value is k. Since $k^2 \geq k$ for $k \geq 2$, the claim follows. □

7.2 Targeted tree structures

We require some notation involving trees. In a tree T, the *level* of any node is the distance of the node from the root. Thus the level of the root is 0, and the height of T is the largest level of the nodes in T. We write $N_T[j,l]$ to denote the j-th node (counting from left to right) at level l of T. We denote by $T[j,l,h]$ the subtree of T whose root is $N_T[j,l]$ and whose remaining nodes are all the descendants of $N_T[j,l]$ in T of level at most h.

If T is a full k-ary tree, then $T[j,l,h]$ has height $h-l$, k^{h-l} leaves, and

$$1 + k + \cdots + k^{h-l} = \frac{k^{h-l+1} - 1}{k-1}$$

nodes. For example, consider the tree T that results if we remove states f and ∞ from the automaton of Fig. 1. The tree $T[2,1,2]$ is the subtree of T whose root is the second node at level 1, $N_T[2,1] = b$, and whose other nodes are $\{a,b\}$, $\{a,c\}$, and $\{b,c\}$.

Definition 2 *A* targeted tree structure *is a directed graph (T,t) consisting of a nonempty tree T of some height h and a node t not in T such that*

1. *Every node of T at level h has at least one edge going to t.*
2. *For every node in T, there is a path from that node to t.*

The level *of node t is $h+1$, and the* height *of (T,t) is $h+1$.*

The automaton in Fig. 1 is a targeted tree structure (T,f) of height 3.

Lemma 6 *Let $k \geq 2$ and $h \geq 1$, let T be a full k-ary tree of height h, let $m = (k^{h+1}-1)/(k-1)$ be the number of nodes in T, let (T,t) be a targeted tree structure, and let n be any integer such that $0 < n \leq m-(h+1)$. Then it is possible to remove n nodes other than the root from T, in such a way that the resulting graph is a targeted tree structure (T',t) of height $h+1$.*

Proof: When $k \geq 2$ and $h \geq 1$, by Lemma 3, we have $m-2h > 0$. Thus $m-(h+1) \geq m-2h > 0$, and n always exists.

We use induction on the height h. For $h = 1$, the tree consists of the root and k leaves. As $n \leq k-1$, we can remove any n of the leaves and the resulting graph is a targeted tree structure of height 2.

Assume the claim holds for $h \geq 1$. Let (T,t) be a targeted tree structure of height $h+2$, and let n be an integer satisfying $0 < n \leq (k^{h+2}-1)/(k-1)-(h+2)$. Then

$$n \leq \frac{k(k^{h+1}-1)+k-1}{k-1} - (h+2) = km - (h+1) = (k-1)m + m - (h+1).$$

Then we can write $n = pm + r$, for some $p \leq k-1$ and $0 \leq r < m$, such that $r \leq m-(h+1)$ if $p = k-1$.

Note that T consists of the root and the k subtrees $T[j, 1, h+1]$, for $j = 1, \ldots, k$, and each such subtree has m nodes. Also, each $(T[j, 1, h+1], t)$, is a targeted tree structure with T of height $h+1$. First, we remove the p subtrees $T[j, 1, h+1]$ for $j = 1, \ldots, p$, and the resulting graph is still a targeted tree structure. In doing so we remove pm nodes from (T, t).

If $r \leq m-(h+1)$, we can remove r nodes from $T[p+1, 1, h+1]$ using the induction hypothesis, and we are done. In particular, this holds when $p = k-1$.

If $r > m-(h+1)$, we must have $n = pm+r$, where $p \leq k-2$, and $1 \leq r \leq m-1$, where $1 \leq r$ because $m-(h+1) > 0$. Thus we can write $r = m-1-(h-s)$, where $h-s \geq 0$, that is, $r = m-(h+1)+s$, where $1 \leq s \leq h$. Now, since $m-(h+1)$ satisfies the condition of the lemma, the induction hypothesis applies, and we can remove $m-(h+1)$ nodes from $T[p+1, 1, h+1]$.

We still need to remove s nodes to make the induction step go through. We claim that $s \leq m-(h+1)$. We know that $s \leq h$, so it is sufficient to show that $h \leq m-(h+1)$. This last inequality is equivalent to $2h < m$, which holds by Lemma 3. Thus we can remove s nodes from $T[p+2, 1, h+1]$ using the induction hypothesis, and the lemma holds. □

7.3 The automaton $\mathcal{A}'_{max}(k)$

For $k \geq 3$, we now define an automaton $\mathcal{A}'_{max}(k)$ over the alphabet $A = \{a_1, \ldots, a_k\}$, which accepts a language of maximal complexity. We will show that it is possible to remove some states from $\mathcal{A}'_{max}(k)$ one at a time, in such a way that the resulting automaton is always minimal. This will establish a complexity hierarchy from 2^k+1 to the maximal complexity.

First we give an informal description of $\mathcal{A}'_{max}(k)$. The automaton is a particular instance of the automaton $\mathcal{A}_{max}(k)$ defined in Theorem 1; here we choose a specific way of assigning k-tuples of level-$(k-1)$ states to the k^{k-2} states at level $k-2$. We start with a full k-ary tree T_{k-1} of height $k-1$ in which each node is labeled by the word of A^* that takes the root to that node. Then we consider T_{k-1} as an automaton where the transition from a node (state) labeled w with $|w| < k-1$ under input a is the node (state) wa, and there are no other transitions for the time being.

For $l \geq 0$, we order the set A^l of all k^l words of length l in lexicographical order: $1, 2, \ldots, k^l$. If $w \in A^l$, let $\nu(w)$ be the position of w in this order; then w is the label of node $N_{T_{k-1}}[\nu(w), l]$ in the tree. In T_{k-1} there are k^{k-2} nodes at level $k-2$ and k^{k-1} nodes at level $k-1$. Since we are dealing with the case $k \geq 3$, we

have $k^{k-1} > 2^k - 1$, as is easily verified. We define tree T'_{k-1} by deleting all nodes $N_{T_{k-1}}[j, k-1]$ such that $j > 2^k - 1$, and also interpret this tree as an automaton as above. Note that, since $k^{k-1} > 2^k - 1$, all the nodes at level $k-1$ in T'_{k-1} belong to the subtree $T'_{k-1}[1, 1, k-1]$.

We can write $2^k - 1 = ck + d$, where $0 \leq d < k$. If $d = 0$, then each of the first c nodes at level $k-2$ has k successors, and all the nodes at level $k-1$ of T'_{k-1} can be reached from the first c nodes at level $k-2$. If $d > 0$, then we need the first $c+1$ nodes at level $k-2$ to reach all the nodes at level $k-1$ of T'_{k-1}. Also, the $(c+1)$st node w has only $d < k$ successors. For each letter a of the alphabet such that $\nu(wa) > 2^k - 1$, we introduce a transition from w to $a_1^{k-1} = N_{T'_{k-1}}[1, k-1]$, the first node at level $k-1$.

In general, there are additional nodes at level $k-2$ that are not used for the purpose of reaching all the nodes at level $k-1$. The transition from such a node i under input a_1 is to node $N_{T'_{k-1}}[1, k-1]$. Let K be the set of the first k nodes at level $k-1$, and let $B = K \setminus \{N_{T'_{k-1}}[1, k-1]\}$. For node i and letters $a_2, \ldots, a_k$, we choose a $(k-1)$-tuple of elements from B, as is explained below. This general structure is shown in Fig. 5 for $k = 4$; the figure is discussed in detail in Example 7 below. The transitions from nodes at level $k-1$ to state t are like those in the proof of Theorem 1, and are defined below.

Definition 3 *Let $k \geq 3$, and $2^k - 1 = ck + d$, where $0 \leq d < k$. Also, define*

- $A = \{a_1, \ldots, a_k\}$,
- $q_0 = 1$ *(the empty word),*
- $Q_{\leq k-2} = \{w \in A^* \mid |w| \leq k-2\}$,
- $Q_{k-2,\leq c} = \{w \in A^{k-2} \mid \nu(w) \leq c\}$,
- $w_{k-2,c+1} \in A^{k-2}$ *is the label of* $N_{T'_{k-1}}[c+1, k-2]$,
- $$S = \begin{cases} \{w \in A^{k-2} \mid \nu(w) > c\} & \text{if } d = 0, \\ \{w \in A^{k-2} \mid \nu(w) > c+1\} & \text{otherwise,} \end{cases}$$
- $Q_{k-1} = \{w \in A^{k-1} \mid \nu(w) \leq 2^k - 1\}$,
- $B = \{N_{T'_{k-1}}[2, k-1], \ldots, N_{T'_{k-1}}[k, k-1]\}$,
- t *and* ∞ *are two distinct elements not in* A^*,

- $Q = Q_{\leq k-2} \cup Q_{k-1} \cup \{t, \infty\}$.
- $\psi : S \to B^{k-1} \setminus \{(N_{T'_{k-1}}[2, k-1], \ldots, N_{T'_{k-1}}[k, k-1])\}$ *is a mapping assigning to each* $w \in S$ *a distinct* $(k-1)$*-tuple of elements of* B*. This is possible because* $(k-1)^{k-1} \geq k^{k-2}$ *by Lemma 5, and hence* $(k-1)^{k-1} > k^{k-2} - ck$*. Let* $\psi_i(w)$ *be the* i*th component of the* $(k-1)$*-tuple of* w*.*
- $\varphi : Q_{k-1} \to (2^A) \setminus \{\emptyset\}$ *is a mapping assigning to each* $w \in Q_{k-1}$ *a distinct nonempty subset of* A*. This is possible, since* Q_{k-1} *has exactly* $2^k - 1$ *elements.*

Then our automaton is $\mathcal{A}'_{max}(k) = (A, Q, q_0, \tau, \{t\})$*, where*

$$\tau(q,a) = \begin{cases} qa & \text{if } q \in Q_{\leq k-2},\ qa \in Q_{\leq k-2} \cup Q_{k-1}, \\ N_{T'_{k-1}}[1, k-1] & \text{if } d > 0,\ q = w_{k-2,c+1},\ qa \notin Q_{k-1}, \\ N_{T'_{k-1}}[1, k-1] & \text{if } q \in S,\ a = a_1, \\ \psi_{i-1}(q) & \text{if } q \in S,\ i > 1,\ a = a_i, \\ t & \text{if } q \in Q_{k-1},\ \text{and } a \in \varphi(q), \\ \infty & \text{otherwise.} \end{cases}$$

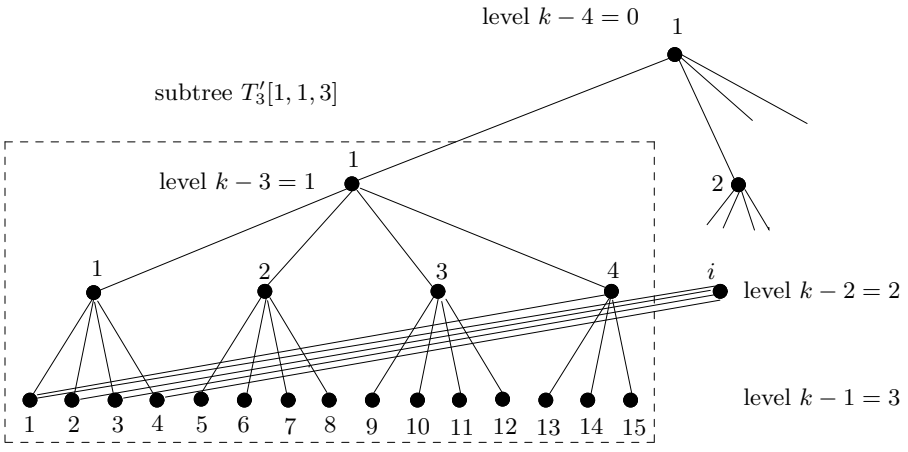

Figure 5: Illustrating the construction of automaton $\mathcal{A}'_{max}(4)$.

Example 7 *In Fig. 5,* $k = 4$*,* $Q_{\leq k-2}$ *is the set of nodes of the full* k*-ary tree of height* 2*. Next,* $Q_{k-1} = \{w \in A^3 \mid \nu(w) \leq 15\}$ *is the set of nodes at level* 3 *of the* k*-ary tree* T'_3 *of height* 3*, after nodes of level* 3 *with positions higher than* 15 *have been deleted from the full* k*-ary tree* T_3 *of height* 3*. Since* $15 = 3 \cdot 4 + 3$*, we have* $c = 3$ *and*

$d = 3$. The first rule of τ defines the tree T'_3. Since $d > 0$, the node $w_{k-2,c+1} = w_{2,4}$ at level $k - 2 = 2$ is needed to cover all the nodes in $Q_{k-1} = Q_3$. The second rule states that all the unused letters of the alphabet (here only a_4) should be sent from $w_{2,4}$ to $N_{T'_3}[1, k-1] = N_{T'_3}[1, 3]$. In this case, all the $2^k - 1 = 15$ nodes at level $k - 1 = 3$ can be reached from the leaves of the subtree $T'_3[1, 1, k-1] = T'_3[1, 1, 3]$ of T'_3 with root $N_{T'_3}[1, 1]$.

The nodes in S, left over at level $k - 2 = 2$, like the typical node i in Fig. 5, have one edge connected to node $N_{T'_3}[1, k-1] = N_{T'_3}[1, 3]$, as stated by the third rule of τ. The $k - 1 = 3$ additional edges are chosen according to the mapping ψ. For example, for node i we can assign the $(k-1)$-tuple (3-tuple) $(3, 2, 4)$, as follows: $\tau(i, b) = 3$, $\tau(i, c) = 2$, and $\tau(i, d) = 4$. The edges from nodes at level $k - 1 = 3$ to the target state t of T'_3 (not shown) are omitted in Fig. 5; they are chosen according to the mapping φ. □

7.4 Hierarchy between $2^k + 1$ and maximal complexity

In $\mathcal{A}'_{max}(k)$ we are able to remove any of the nodes $N_{T'_{k-1}}[i, k-1]$, for $i = ck + 1, \ldots, 2^k - 1 = ck + d$, without disconnecting the automaton. Also, we can remove any $k - 1$ leaves from every subtree $T'_{k-1}[j, k-2, k-1]$, for $j = 2, \ldots, c$, without disconnecting the automaton. Moreover, each tree $T'_{k-1}[j, 1, k-2]$, for $j = 2, \ldots, k$ can be totally removed, or we can remove up to $(k^{k-2} - 1)/(k-1) - (k-2)$ nodes from that tree according to Lemma 6 without disconnecting the automaton. We note that when we remove nodes, the remaining ones are all distinguishable and, therefore, the resulting automaton is minimal. We never remove the special nodes $N_{T'_{k-1}}[1, k-1], \ldots, N_{T'_{k-1}}[k, k-1]$.

Theorem 3 *For $k \geq 3$ and each i such that $2^k + 1 \leq i \leq \frac{k^{k-1}-1}{k-1} + 2^k + 1$ there is a language of state complexity i.*

Proof: First we prove that the automaton $\mathcal{A}'_{max}$ of Definition 3 is minimal. By construction, every state in Q is reachable from the initial state q_0. Every two states at level $k - 1$ are distinguishable, because each accepts a different nonempty subset of A.

If $d = 0$, every two nodes in $Q_{k-2,\leq c}$ are distinguishable, because the transition under input a_1 takes each state to a different state at level $k - 1$. Every two nodes in S are distinguishable from each other because each has a distinct $(k-1)$-tuple of nodes from B, so that at least one input leads to two different states at level $k - 1$. Any state in S has a transition to $N_{T'_{k-1}}[1, k-1]$ under a_1, and states in $Q_{k-2,\leq c} \setminus \{N_{T'_{k-1}}[1, k-2]\}$ do not have such transitions. Finally, state $N_{T'_{k-1}}[1, k-2]$ is different

from any state in S, because the $(k-1)$-tuple $(N_{T'_{k-1}}[2,k-1],\ldots,N_{T'_{k-1}}[k,k-1])$ is not assigned to any state in S. If $d>0$, the arguments above also work, if we replace $Q_{k-2,\leq c}$ by $Q_{k-2,\leq c}\cup\{w_{k-2,c+1}\}$. Note that the removal of any node from level $k-2$ does not affect the distinguishability of the remaining nodes at this level.

For any two states p and q at any level $l\leq k-3$, every word of length $k-l-2$ leads to distinguishable states at level $k-2$. Hence p and q are also distinguishable, and automaton $A'_{max}(k)$ is of maximal state complexity, according to Theorem 1.

Our challenge is to remove states of $A'_{max}(k)$ in such a way that all the remaining states are reachable from q_0 and can reach the final state t and remain distinguishable. Since $A'_{max}(k)$ has $(k^{k-1}-1)/(k-1)+2^k+1$ states, and we wish to reach an automaton with 2^k+1 states, we need to be able to remove n states, where

$$n\leq B'_k=(k^{k-1}-1)/(k-1).$$

The case of $k=3$ can be resolved easily by using the automaton of Fig. 1. For example, one can remove nodes $\{b\}$, $\{c\}$, $\{a,c\}$, and $\{b,c\}$ in any order, and the resulting automaton is always minimal. From now on we assume that $k\geq 4$.

Let $m=(k^{k-2}-1)/(k-1)$; each subtree $T'_{k-1}[i,1,k-2]$, for $i=2,\ldots,k$, is of height $k-3$ and has m nodes. Also, $(T'_{k-1}[i,1,k-2],N_{T'_{k-1}}[1,k-1])$ is a targeted tree structure. By Lemma 6 with $h=k-3$, we can remove any n nodes from this structure and the result is still a targeted tree structure, as long as $n\leq m-(k-2)$. Also, the entire subtree can be removed without affecting the minimality of the resulting automaton.

Suppose now that we wish to remove $n\leq B'_k$ states from $A'_{max}(k)$, and $n=pm+r$, where $0\leq p\leq k$, and $0\leq r<m$. We distinguish three cases:

Case 1 $p\leq k-2$**:** Remove the p trees $T'_{k-1}[i,1,k-2]$, for $i=2,\ldots,p+1$. If $r\leq m-(k-2)$, remove r nodes from $T'_{k-1}[p+2,1,k-2]$ using Lemma 6. If $r>m-(k-2)$, then $r=m-(k-2)+s$, for some s such that $1\leq s<k-2$, since $r<m$. Remove $m-(k-2)$ nodes from $T'_{k-1}[p+2,1,k-2]$ using Lemma 6. Finally, remove s leaves from $T'_{k-1}[2,k-2,k-1]$; this is possible, since $s<k-2$, and there are k leaves.

Case 2 $p=k$**:** First, remove the tree $T'_{k-1}[1,1,k-1]$ except for the branch

$$N_{T'_{k-1}}[1,1],\ldots,N_{T'_{k-1}}[1,k-2]$$

and the k leaves

$$N_{T'_{k-1}}[1,k-1],\ldots,N_{T'_{k-1}}[k,k-1].$$

As the tree $T'_{k-1}[1,1,k-1]$ has $m+2^k-1$ nodes, we have removed $m+2^k-1-(k-2+k)=m+2^k-2k+1$ nodes. Next remove the trees $T'_{k-1}[i,1,k-2]$,

for $i = 2, \ldots, k-1$, with a total of $m(k-2)$ nodes. By now we have removed $m + 2^k - 2k + 1 + m(k-2) = m(k-1) + 2^k - 2k + 1$ nodes, and we still need to remove $km + r - (m(k-1) + 2^k - 2k + 1) = m + r - 2^k + 2k - 1$.

Recall that we must have $n \leq B'_k$; hence $km + r \leq (k^{k-1} - 1)/(k-1)$. The last inequality reduces to $r \leq 1$. To use Lemma 6, we need $m + r - 2^k + 2k - 1 \leq m - (k-2)$, or $2^k - 2k + 1 - r \geq k - 2$. Since $2^k - 2k > k - 2$, for $k \geq 3$, we have the required inequality, and we can remove $m + r - 2^k + 2k - 1$ nodes from $T'_{k-1}[1, k, k-2]$ using Lemma 6.

Case 3 $p = k - 1$**:** Here $n = k^{k-2} - 1 + r$ and Lemma 4 implies that, if $k > 4$, then $n \geq m + 2^k - 1 + r$. This, in turn, implies that $n > m + 2^k - 1 - (2k-2) = m + 2^k - 2k + 1$. In fact the same holds when $k = 4$. So, as in the previous case, we remove $m + 2^k - 2k + 1$ nodes from the tree $T'_{k-1}[1, 1, k-1]$ except for the $(2k-2)$ special nodes.

We still need to remove

$$n' = (pm + r) - (m + 2^k - 2k + 1) = (k-2)m + r - (2^k - 2k + 1)$$

nodes. One verifies that $k - 1 < 2^k - 2k + 1$. Also $(k-1)m > (k-2)m + r$, since $r < m$. Hence $n' < (k-1)m - (k-1)$. Since $n' < (k-1)m$, we can write $n' = p'm + r'$ with $p' \leq k-2$ and $0 \leq r' < m$. Now remove the trees $T'_{k-1}[j, 1, k-2]$, for $j = 2, \ldots, p'+1$ for a total of $p'm$ nodes. Then, if $r' \leq m - (k-2)$, remove r' nodes from $T'_{k-1}[p'+2, 1, k-2]$ using Lemma 6.

Otherwise, we have $m - (k-2) < r' < m$; hence we can write $r' = m - (k-2) + r''$, with $1 \leq r'' < k-2$. Now it cannot be that $p' = k-2$, for then $n' = (k-2)m + r'$ and $n' < (k-1)m - (k-1) = (k-2)m + m - (k-1)$, and $r' < m - (k-1)$. Hence also $r' \leq m - (k-2)$, which is a contradiction. Therefore, we have $p' \leq k-3$. So we can remove $m - (k-2)$ nodes from $T'_{k-1}[p'+2, 1, k-2]$ and r'' nodes from $T'_{k-1}[p'+3, 1, k-2]$. □

8 Conclusions

We have studied k-languages, which are uniform languages of length k over an alphabet of k letters. For these languages the minimal state complexity is $k+2$ and the maximal one is $(k^{k-1} - 1)/(k-1) + 2^k + 1$. We have shown that for every i between the minimal and maximal complexities there is a k-language of complexity i. In proving this result, we have introduced two new types of automata: pi automata accepting languages which are permutations of the alphabet, and targeted tree structures. It is hoped that these ideas will help to improve our understanding of the state complexity of more general finite languages.

References

[1] J. A. Brzozowski, "Derivatives of regular expressions", *J. Assoc. Comp. Mach.* vol. 11, no. 4, pp. 481–494, October 1964.

[2] C. Câmpeanu, "How many states can a minimal DFA have that accepts words of length less than or equal to l?" In J. Dassow, M. Hoeberechts, H. Jürgensen, and D. Wotschke, eds., Pre-Proceedings of *Descriptional Complexity of Formal Systems*, London, ON, Canada, Aug. 21–24, 2002. (The journal version appeared as [4].)

[3] C. Câmpeanu, K. Culik, K. Salomaa, S. Yu, "State complexity of basic operations on finite languages", Proc. *Fourth Int. Workshop on Implementing Automata WIA'99,* Potsdam, Germany, July 17–19, LNCS 2214, pp. 60–70, 2001.

[4] C. Câmpeanu, W. H. Ho, "The Maximum state complexity for finite languages", *J. Automata, Languages and Combinatorics,* vol. 9, no. 2/3, pp. 189–202, 2004.

[5] A. Salomaa, K. Salomaa, S. Yu, "State complexity of combined operations", *Theoretical Computer Science*, vol. 383, no. 2–3, pp. 140–152, 2007.

[6] S. Yu, Q. Zhuang, K. Salomaa, "The state complexities of some basic operations on regular languages", *Theoretical Computer Science*, vol. 125, no. 2, pp. 315–328, 1994.