DCFS 2008 (Draft)

Descriptional Complexity of Formal Systems
Deadline for submissions: April ~~7~~ 27, 2008(strict)
Final versions: June 25, 2008

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

Janusz Brzozowski[(A)] Stavros Konstantinidis[(B)]

[(A)]David R. Cheriton School of Computer Science – University of Waterloo
Waterloo, ON – Canada N2L 3G1
brzozo@uwaterloo.ca

[(B)]Department of Mathematics and Computing Science – Saint Mary's University
Halifax, NS – Canada B3H 3C3
s.konstantinidis@smu.ca

Abstract. We study the state complexity of certain simple languages. If A is an alphabet of k letters, then a k-language is a nonempty set of words of length k, that is, a uniform language of length k. We show that the minimal state complexity of a k-language is $k+2$ and the maximal, $(k^{k-1}-1)/(k-1)+2^k+1$. We prove constructively that, for every i between the minimal and maximal bounds, there is a language of state complexity i. We introduce a class of automata accepting sets of words that are permutations of A; the complexities of these languages form a complete hierarchy between k^2-k+3 and 2^k+1. The languages of another class of automata, based on k-ary trees, define a complete hierarchy of complexities between 2^k+1 and $(k^{k-1}-1)/(k-1)+2^k+1$. This provides new examples of uniform languages of maximal complexity.

Keywords: automata, bounds, hierarchies, languages, permutations, state complexity, trees, uniform

1 Introduction

State complexity has received considerable attention recently [3]–[7]. 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 [3, 5]. We study a special class of uniform languages, namely, the nonempty languages over an alphabet of cardinality k in which all the words have length k. Such k-languages, have two advantages. First, the special form makes it easier to reason about state complexity. Second, in spite of their simplicity, k-languages exhibit a complete hierarchy of state complexities.

In the remainder of the paper, Section 2 introduces our notation. Section 3 defines k-languages and shows a tight upper bound on the state complexity of these languages; this bound coincides with the bound for uniform languages of length k as shown in [3, 5]. Sections 4–6 demonstrate constructively that there are k-languages of state complexity i for every i between the minimum and maximum bounds. Section 7 concludes the paper. Proofs that are omitted in this paper can be found in [2].

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

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

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_n is uniform of length n if all the words in L_n are of length n. By Proposition 1, for a uniform L_n, two words of different lengths lead to distinguishable states in the quotient automaton of L_n.

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

(I) *If $|w| > n$, then $w^{-1}L_n = \emptyset$.*

(II) *If $|u|, |v| \leq n$ and $|u| \neq |v|$, then $u^{-1}L_n \cap v^{-1}L_n = \emptyset$.*

A tight upper bound $C_{k,n}$ on the state complexity of a uniform language of length n has been found in [3, 5]:

$$C_{k,n} = \frac{k^{r_0} - 1}{k - 1} + \sum_{j=0}^{n-r_0} (2^{k^j} - 1) + 1, \tag{1}$$

where k is the alphabet size and $r_0 = \min\{m \mid k^m \geq 2^{k^{n-m}} - 1\}$.

We study a special class of nonempty uniform languages, the k-languages, over an alphabet of cardinality k, with words of length k. Let $\mathbf{K}_k$ be the set of all k-languages.

Theorem 2. *For $k \geq 1$, the following bounds hold for k-languages:*

(I) *If $L \in \mathbf{K}_1$, then $\mathbf{c}(L) = 3$.*

(II) *If $L \in \mathbf{K}_k$, then $k + 2 \leq \mathbf{c}(L)$, and the bound is reachable.*

(III) *If $L \in \mathbf{K}_2$, then $\mathbf{c}(L) \leq 5$, and the bound is reachable.*

(IV) *For $k \geq 3$, if $L \in \mathbf{K}_k$, the bound below is reachable.*

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

Proof: The first three parts are easy to verify. For the last claim, 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 $|w| = k$. 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 $1 + 1 + (2^k - 1) + (1 + k + \cdots + k^{k-2}) = (k^{k-1} - 1)/(k-1) + 2^k + 1$.

Now consider the automaton $\mathcal{A}_{max}(k) = (A, Q, q_0, \tau, F)$, 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,
- $q_0 = 1$ (the empty word),
- $F = \{f\}$, and
- For all $a \in A$,

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

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 $(2^k)^k > k^{k-2}$; 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 $S_{i_j} \subseteq A$ 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 $\tau(u_i, a_j) = S_{i_j}$, for all $a_j \in A$; 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. See Fig. 1 for an example.

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. □

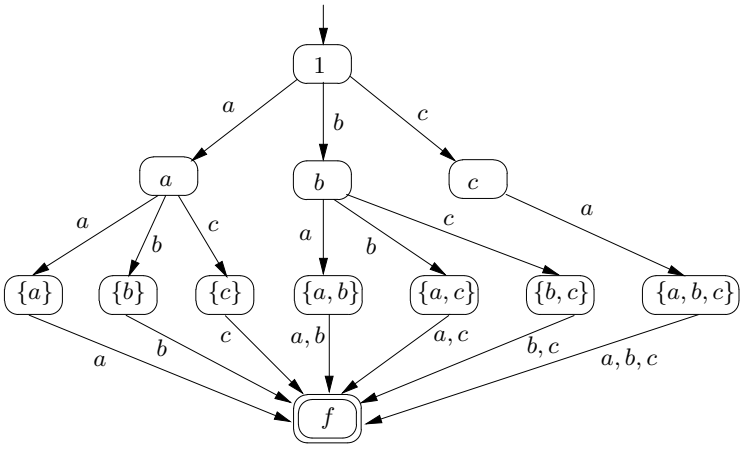

Figure 1. Automaton $\mathcal{A}_{max}(3)$, state ∞ not shown.

When the length n is the same as the size k of the alphabet, expression (1) becomes

$$C_{k,k} = \frac{k^{r_0} - 1}{k - 1} + \sum_{j=0}^{k-r_0} (2^{k^j} - 1) + 1, \tag{3}$$

where $r_0 = \min\{m \mid k^m \geq 2^{k^{k-m}} - 1\}$. The construction of $A_{max}(k)$ in Theorem 1 constitutes a new example of uniform languages with maximal complexity, and this leads to the form B_k of the bound $C_{k,k}$, as we now confirm:

Proposition 3. *Bound B_k coincides with $C_{k,k}$; hence, every k-language of maximal state complexity is also a uniform language of length k of maximal state complexity.*

4 From k-complexity to k^2-complexity

We show, in several steps, that for each i, $k + 2 \leq i \leq B_k$, there is a language of complexity i. We call $k+2$ and k^2-k+3, k-complexity and k^2-complexity, respectively.

Proposition 4. *Let $k \geq 1$ and $L_{pwr}(k) = \bigcup_{a \in A}\{a^k\}$; then $\mathbf{c}(L_{pwr}(k)) = k^2 - k + 3$.*

Proof: The quotients of $L_{pwr}(k)$ by words in $\{a^i \mid a \in A, 0 \leq i \leq k-1\}$ are all nonempty and distinct, and each has a word of length > 0, for $1 + k(k-1)$ states. Also, $(a^k)^{-1}L_{pwr}(k) = 1$, for all $a \in A$, and $w^{-1}L_{pwr}(k) = \emptyset$ for all other words. □

Proposition 5. *The following properties hold for $\mathbf{K}_k$: For each i, $1 \leq i \leq k$, there is a language $L_i \in \mathbf{K}_k$ with complexity $i(k-1)+3$. For each j, $i(k-1)+3 \leq j \leq (i+1)(k-1)+3$, there is a language L_j in $\mathbf{K}_k$ with complexity j.*

Corollary 6. *For each i, $k + 2 \leq i \leq k^2 - k + 3$, there is a language of complexity i.*

5 From k^2-complexity to 2^k-complexity

We now consider i, where $k^2 - k + 3 \leq i \leq 2^k + 1$; we refer to $2^k + 1$ as 2^k-complexity.

5.1 Pi automata

We introduce a class of automata, all of which have the same form, and differ only in the state set and the transition function, for the range $k^2 - k + 3 \leq i \leq 2^k + 1$.

Definition 7. Let $k \geq 1$, and let $A = \{a_1, \ldots, a_k\}$ 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 8. *Each pi automaton $\mathcal{A}_\pi(k)$ accepts a k-language of words that 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 $\tau(q, a) = s = q \cup \{a\}$, 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.

5.2 A hierarchy of pi automata

Let $A = \{a_1, \ldots, a_k\}$, and define the circular order to be that of the word $x_k = a_1 a_2 \cdots a_k a_1 a_2 \cdots a_{k-1}$. 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. We consider $S = \emptyset$ to be circular, with the corresponding factor 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 with 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 $q \in Q_0$, $a \in A$,

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

Let $L_{cir}(k)$ be the language accepted by $\mathcal{A}_0(k)$. In automaton $\mathcal{A}_0(4)$ of Fig. 2 (a) we represent states by words instead of sets of letters to make their circularity explicit.

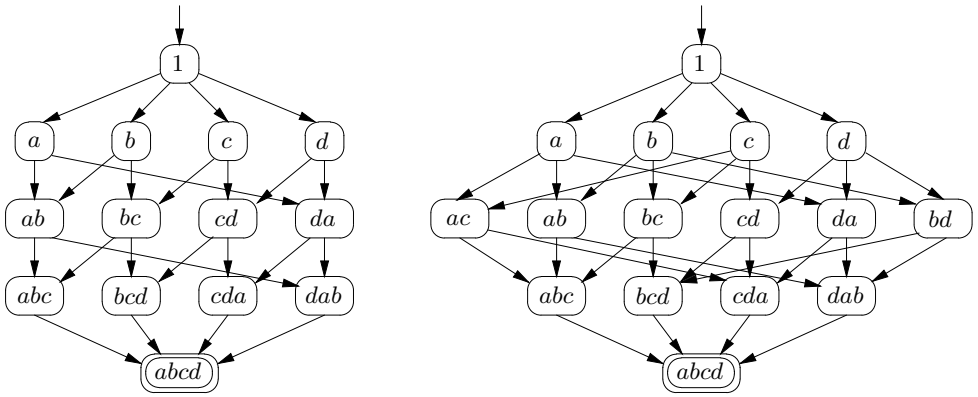

Figure 2. Automata: (a) $\mathcal{A}_0(4)$; (b) $\mathcal{A}_1(4) = \mathcal{A}_{all}(4)$.

Proposition 9. *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 8. □

Next, we introduce a pi automaton $\mathcal{A}_{all}(k)$ with $2^k + 1$ states; the state set of $\mathcal{A}_{all}(k)$ consists of all 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 $q \in Q_{all}$, $a \in A$,

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

Let $L_{all}(k)$ be the language accepted by $\mathcal{A}_{all}(k)$. The automaton of Fig. 2 (b) is $\mathcal{A}_{all}(4)$.

Proposition 10. *The language $L_{all}(k)$ consists of all the words which are permutations of A, and has complexity $2^k + 1$.*

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 $A = \{a, \ldots, h\}$, 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 11. *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 $\mathcal{A}_0(k)$ has all the states of distance 0. If $\mathcal{A}_i(k)$ has all the states of distance $\leq d$, then $\mathcal{A}_{i+1}(k)$ has all the states of distance $\leq d+1$. Since the distance is certainly less than k, we eventually include all the states, and obtain $\mathcal{A}_{all}(k)$. □

Lemma 12. *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 $d_p(s) \leq d_p(q)$.*

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

Lemma 13 permits us to add states of distance $i+1$ to pi automaton $\mathcal{A}_i(k)$ in increasing order of j, the sum of a state q, to eventually obtain $\mathcal{A}_{i+1}(k)$.

Theorem 14. *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 9. By adding one state to any minimal pi automaton as guided by Lemma 13, we obtain another pi automaton, which is still minimal. By Proposition 11, we eventually reach $\mathcal{A}_{all}$, which has $2^k + 1$ states, by Proposition 10. Hence the claim holds. □

6 From 2^k-complexity to maximal complexity

6.1 Targeted tree structures

A tree is a directed acyclic graph $T = (N, E)$, where N is a nonempty set of nodes, and E is a set of edges. One node n_0 is of indegree 0; this is the root of the tree, and every node in the tree is reachable from n_0 by exactly one path. 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 the number of nodes is

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

Definition 15. 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

(I) Every node of T at level h has at least one edge going to t.

(II) 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 $h + 1$.

For example, consider the tree T that results if we remove states f and ∞ from the automaton of Fig. 1. Then $T[2, 1, 2]$ 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 $\{b, c\}$. The automaton in Fig. 1 is a targeted tree structure (T, f) of height 3.

Lemma 16. *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 $h + 1$.*

6.2 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.

The automaton $\mathcal{A}'_{max}(k) = (A, Q, 1, \tau, \{t\})$ is a particular instance of the automaton $\mathcal{A}_{max}(k)$ defined in Theorem 2; 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 then add state t as a target for the targeted tree structure (T'_{k-1}, t).

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 distinct $(k-1)$-tuple of elements from B. This structure is shown in Fig. 3 for $k = 4$; the figure is discussed in Example 17 below. The transitions from nodes at level $k-1$ to state t are like those in the proof of Theorem 2.

Example 17. In Fig. 3, $k = 4$, $A = \{a, b, c, d\}$, and $Q_{\leq 2}$ is the set of nodes of the full 4-ary tree of height 2. Next, $Q_3 = \{w \in A^3 \mid \nu(w) \leq 15\}$ is the set of nodes at level 3 of the 4-ary tree T'_3 of height 3, after nodes of level 3 with positions higher than 15 have been deleted from the full 4-ary tree T_3 of height 3. Since $15 = 3 \cdot 4 + 3$, we have $c = 3$ and $d = 3$. If $q \in Q_{\leq 2}$, $a \in A$, and $qa \in Q_{\leq 2} \cup Q_3$, then $\tau(q, a) = qa$. This defines the tree T'_3.

From the first three nodes at level 2 we can reach the first 12 states at level 3. Since $d > 0$, node $w_{2,4}$ at level 2 is needed to reach the rest of the nodes in Q_3. All the unused letters of the alphabet (here only a_4) are sent from $w_{2,4}$ to $N_{T'_3}[1, 3]$. All the $2^k - 1 = 15$ nodes at level $k-1 = 3$ can be reached from the leaves of the subtree $T'_3[1, 1, 3]$ of T'_3 with root $N_{T_3}[1, 1]$.

The nodes left over at level $k-2 = 2$, like the typical node i in Fig. 3, have one edge connected to node $N_{T'_3}[1, k-1] = N_{T'_3}[1, 3]$. For the remaining edges, each such node is assigned a distinct $(k-1)$-tuple from B. 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. 3; they are chosen as in the automaton of Theorem 2.

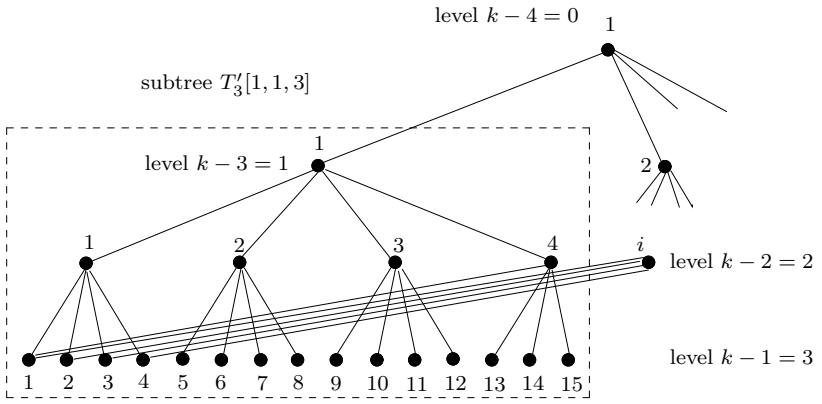

Figure 3. Illustrating the construction of automaton $\mathcal{A}'_{max}(4)$.

6.3 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 16 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 $N_{T'_{k-1}}[1,k-1],\ldots,N_{T'_{k-1}}[k,k-1]$. Using this approach we prove the main result of this section with the aid of the following lemma:

Lemma 18. *For all integers $k\geq 5$, $k^{k-2}-1 \geq (k^{k-2}-1)(k-1)+2^k-1$.*

Theorem 19. *For $k\geq 3$ and each i such that $2^k+1 \leq i \leq (k^{k-1}-1)/(k-1)+2^k+1$ there is a k-language of state complexity i.*

Proof: First we prove that $\mathcal{A}'_{max}(k)=(A,Q,1,\tau,\{t\})$ 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 $Q_{k-2,\leq c}=\{w\in A^{k-2} \mid \nu(w)\leq c\}$ are distinguishable, because the transition under input a_1 takes each state to a different state at level $k-1$. Every two nodes in

$$S=\begin{cases}\{w\in A^{k-2}\mid \nu(w)>c\} & \text{if } d=0,\\ \{w\in A^{k-2}\mid \nu(w)>c+1\} & \text{otherwise,}\end{cases}$$

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 $Q_{k-2,\leq c}\cup\{w_{k-2,c+1}\}$, where $w_{k-2,c+1}\in A^{k-2}$ is the label of $N_{T'_{k-1}}[c+1,k-2]$.

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 2.

Our challenge is to remove states of $A'_{max}(k)$ in such a way that all the remaining states are reachable from q_0, 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 16 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 16. 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 16. 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 16, 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 16.

Case 3 $p=k-1$**:** Here $n=k^{k-2}-1+r$ and Lemma 18 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 16. 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 $T'_{k-1}[p'+3, 1, k-2]$. □

7 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. ACM, 11(4), 481–494, 1964.

[2] J. A. Brzozowski, S. Konstantinidis. State-complexity hierarchies of uniform languages of alphabet-size length. Technical Report CS-2008-05, David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada, 26 pp., March 2008. http://www.cs.uwaterloo.ca/research/tr/2008/

[3] 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, D. Wotschke (eds.), Pre-Proc. Descriptional Complexity of Formal Systems, London, ON, Canada, 2002. (Journal version is [5].)

[4] C. Câmpeanu, K. Culik, K. Salomaa, S. Yu. State complexity of basic operations on finite languages, In O. Boldt, H. Jürgensen (eds.), Proc. Fourth Int. Workshop on Implementing Automata, Lect. Notes Comput. Sci., 2214, 60–70. Springer-Verlag, 2001.

[5] C. Câmpeanu, W. H. Ho. The maximum state complexity for finite languages. J. Automata, Languages and Combinatorics, 9(2/3),189–202, 2004.

[6] A. Salomaa, K. Salomaa, S. Yu. State complexity of combined operations. Theoretical Computer Science, 383, (2/3), 140–152, 2007.

[7] S. Yu, Q. Zhuang, K. Salomaa. The state complexities of some basic operations on regular languages. Theoretical Computer Science, 125(2), 315–328, 1994.