Theoretical Computer Science 410 (2009) 2431–2441

Contents lists available at ScienceDirect

Theoretical Computer Science

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

State complexity of unique rational operations

Narad Rampersad [a], Nicolae Santean [a,*], Jeffrey Shallit [a], Bala Ravikumar [b]

[a] School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada

[b] Department of Computer Science, Sonoma State University, 1801 East Cotati Avenue, Rohnert Park, CA 94928, USA

ARTICLE INFO

Keywords:
Unique union
Unique concatenation
Unique star
State complexity

ABSTRACT

For each basic language operation we define its "unique" counterpart as being the operation that results in a language whose words can be obtained uniquely through the given operation. These unique operations can arguably be viewed as combined basic operations, placing this work in the popular area of state complexity of combined operations on regular languages. We study the state complexity of unique rational operations and we provide upper bounds and empirical results meant to cast light into this matter. Equally important, we hope to have provided a generic methodology for estimating their state complexity.

1. Introduction

Finite automata (FAs) are ubiquitous objects in computer science theory as much as in computer applications. They model finite state systems, from a door lock to the entire Universe – in some views – and check the syntax of regular languages. Computers are deterministic finite automata (DFAs), and the English lexicon can be spell-checked by FAs. Recently, automata have found new practical applications, such as in natural language processing [17], communications [3] and software engineering [11] — applications increasingly demanding in terms of computing resources. In this context, the study of state complexity of operations on FAs and their languages has become a topic of paramount importance.

From the formal languages point of view, FAs are yet another tool for defining the family of regular (or rational, as known in certain literature) languages, along with regular expressions and right linear grammars. They arise from the perpetual mathematical effort of expressing infinite objects by finite means. In this paper we pursue a new direction in their study, namely, that of analyzing the succinctness of expressing a language obtained by certain *unique* language operations, in terms of the descriptional complexity of the languages involved.

Similar directions have been pursued in automata theory before, e.g., for *basic* language operations [29,5,6,25] and *combined* operations [9,24,28] on regular languages. In the present paper, we make a leap from the current trend, by addressing the succinctness of some special operations; namely, we address those operations derived from the basic ones, that reach a result in a unique manner: an object obtained in two (or more) ways by applying the given operation is excluded from the result. This work is distinct from another recent examination of concatenation uniqueness in [7], where the operation is defined in an "all-or-nothing" manner. Our definition of unique concatenation was briefly mentioned in [15], where pebble automata were used to infer the regularity of this operation. We go beyond this matter, by rigorously studying all rational operations, with a focus on their state complexity. An extended version of this paper, with more details, experiments, and complete proofs, can be found in [21].

* Corresponding address: Department of Computer and Information Sciences, Indiana University South Bend, 1700 Mishawaka Ave., South Bend, IN 46634, USA.

E-mail addresses: nrampersad@cs.uwaterloo.ca (N. Rampersad), nsantean@iusb.edu (N. Santean), shallit@cs.uwaterloo.ca (J. Shallit), ravi@cs.sonoma.edu (B. Ravikumar).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.02.035

2. Definitions and notation

Let Σ be an alphabet, i.e., a non-empty, finite set of symbols (letters). By Σ^* we denote the set of all finite words (strings of symbols) over Σ, and by ε, the empty word (a word having zero symbols). The operation of concatenation (juxtaposition) of two words u and v is denoted by $u \cdot v$, or simply uv. For $w \in \Sigma^*$, we denote by w^R the word obtained by reversing the order of symbols in w. A non-deterministic finite automaton over Σ, NFA for short, is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, $\delta : Q \times \Sigma \to 2^Q$ is a next-state function, q_0 is an initial state and $F \subseteq Q$ is a set of final states. δ is extended over $Q \times \Sigma^*$ in the usual way. M is deterministic (DFA) if $\delta : Q \times \Sigma \to Q$. We consider *complete* DFAs, that is, those whose transition function is a total function. The size of M is the total number of its states. When we want to emphasize the number n of states of M, we say that M is an n-state DFA. The language of M, denoted by $L(M)$, belongs to the family of *regular languages* and consists of those words accepted by M in the usual way. For a background on finite automata and regular languages we refer the reader to [27].

Definition 1. Let L, R be languages over Σ.

(i) The *unique concatenation* of L and R is the set

$$L \circ R = \{w \mid w = uv,\ \ u \in L,\ v \in R,\ \text{and such factorization is unique}\}.$$

(ii) The *unique star* of L is the set

$$L^\circ = \{\varepsilon\} \cup \{w \mid w = u_1 \cdots u_n,\ \ n \in \mathbb{N},\ \ u_i \in L \setminus \{\varepsilon\}\ \ \forall 1 \le i \le n, \text{and such factorization is unique}\}.$$

(iii) The *unique union* of L and R is the set $L \mathbin{\mathring{\cup}} R = (L \setminus R) \cup (R \setminus L)$.

We could have defined L° such that the factorization in the above definition involves ε as well. In this case, if L contained ε, then L° would be empty. Moreover, the connection with unambiguous regular expressions (Lemma 6) could not be made. For these reasons we adopt the above definition.

Notation-wise, we denote $L \diamond R = LR \setminus (L \circ R)$, $L^\diamond = L^* \setminus L^\circ$, and $L \mathbin{\mathring{\cup}} R = L \cap R$, and we refer to these operations as *poly concatenation*, *poly star* and *poly union*. Note that $\varepsilon \notin L^\diamond$. We also consider *unique square* and *poly square*, given by $L^{\circ 2} = L \circ L$ and $L^{\diamond 2} = L^2 \setminus L^{\circ 2}$.

The reversal operation is compatible with the unique operations and with their "poly" counterparts. Indeed, for $L_1, L_2 \subseteq \Sigma^*$, one can verify that

$$(L_1 \circ L_2)^R = L_2^R \circ L_1^R, \quad (L_1 \mathbin{\mathring{\cup}} L_2)^R = L_1^R \mathbin{\mathring{\cup}} L_2^R, \quad (L_1^\circ)^R = (L_1^R)^\circ,$$
$$(L_1 \diamond L_2)^R = L_2^R \diamond L_1^R, \quad (L_1 \mathbin{\mathring{\cup}} L_2)^R = L_1^R \mathbin{\mathring{\cup}} L_2^R, \quad (L_1^\diamond)^R = (L_1^R)^\diamond.$$

The next examples show that unique and poly concatenation are not associative:

(1) For $L_1 = \{b, ba^2\}$, $L_2 = \{a^3, a^4\}$ and $L_3 = \{ab, a^2b, a^3b\}$ we have $(L_1 \circ L_2) \circ L_3 = \{ba^4b, ba^9b\}$, and $L_1 \circ (L_2 \circ L_3) = \{ba^4b, ba^6b, ba^7b, ba^9b\}$.
(2) For $L_1 = \{b, ba\}$, $L_2 = \{a^3, a^4\}$, $L_3 = \{ab, a^2b, a^3b\}$ we have $(L_1 \diamond L_2) \diamond L_3 = \emptyset$ and $L_1 \diamond (L_2 \diamond L_3) = \{ba^6b\}$.

Consequently, unique concatenation and unique star are unrelated: if $L = \{a, b, b^2\}$ then $ab^2 \in (L \circ L) \circ L$; however $ab^2 \notin L^\circ$. Various connections among unique operations can be drawn. For example,

Lemma 2. *If L is an arbitrary language then*

$$(L^* \setminus \{\varepsilon\})^{\diamond 2} = L^\diamond, \quad \textit{and} \quad (L^* \setminus \{\varepsilon\})^{\circ 2} = L^\circ \setminus \{\varepsilon\}.$$

Definition 3. By a unique regular expression, or *unireg expression* for short, we understand the well-formed, parenthesized formulas with the following operators: all symbols $a \in \Sigma$ and ε are nullary, $^\circ$ is unary, and $\circ$, $\oplus$ are binary. The operator $\oplus$ is the expression counterpart of $\mathring{\cup}$.

Since $\{a\} \circ \{b\} = a \circ b = ab$, for any symbols $a, b \in \Sigma$ we denote the unique concatenation of symbols by juxtaposition, as for the usual concatenation. The language $\mathcal{L}(e)$, denoted by unireg expression e, is defined recursively as in the case of regular expressions [13]. However, for reasons that will become apparent later, *we consider only fully-parenthesized expressions*.

3. Properties

In the following, we denote by $\sqcup\!\sqcup$ the shuffle operation on words or languages. We also use two new symbols, $1, 2 \notin \Sigma$, and we denote by h_{12} the homomorphism that deletes these symbols in words, $h_{12} : (\Sigma \cup \{1, 2\})^* \to \Sigma^*$.

Lemma 4. *If L and R are regular languages then $L \circ R$, L°, and $L \mathbin{\mathring{\cup}} R$ are regular languages.*

Proof. It is clear that $L \overset{\circ}{\cup} R$ is regular. For the other two, it suffices to observe that the languages

$$L \diamond R = h_{12}\Big((L1R \sqcup 2) \cap (L2R \sqcup 1) \cap \big(\Sigma^*(1\Sigma^+2 + 2\Sigma^+1)\Sigma^*\big)\Big) \quad \text{and}$$
$$L^{\diamond} = h_{12}\Big(\big((L'1)^* \sqcup 2^*\big) \cap \big((L'2)^* \sqcup 1^*\big) \cap \big(\Delta^*(1\Sigma^+2 + 2\Sigma^+1)\Delta^*\big)\Big),$$

where $L' = L \setminus \{\varepsilon\}$ and $\Delta = \Sigma \cup \{1, 2\}$, are both regular, their definition involving operations under which regular languages are closed. Then, since $L \circ R = LR \setminus (L \diamond R)$ and $L^{\circ} = L^* \setminus L^{\diamond}$, the conclusion follows. Note that the expressions for $L \diamond R$ and $L^{\diamond}$ can be simplified using "sequential insertion" [16]. □

Let R be a regular expression over Σ, containing r occurrences of symbols in Σ (multiple occurrences are counted separately). Let $\Sigma' = \{a_1, \ldots, a_r\}$ denote an alphabet of r new symbols, and consider $h_R : \Sigma'^* \to \Sigma^*$ the homomorphism that maps a_i to the symbol in Σ representing the ith occurrence of a symbol in R. By R_h we denote the regular expression obtained from R by replacing each ith occurrence of a symbol in Σ with the corresponding $a_i \in \Sigma'$. For example, if $R = (a + ab)^*b^*$, then $R_h = (a_1 + a_2a_3)^*a_4^*$, and $h_R(a_1) = h_R(a_2) = a$, $h_R(a_3) = h_R(a_4) = b$.

Definition 5. With the above notation, a regular expression R is *unambiguous* if and only if the restriction of h_R to $\mathcal{L}(R_h)$ is injective. *(This definition is equivalent to that given in [4].)*

According to this definition, the above expression $R = (a + ab)^*b^*$ is not unambiguous, since $a_1a_4, a_2a_3 \in L(R_h)$ and $h_R(a_1a_4) = h_R(a_2a_3) = ab$. An unambiguous regular expression "matches" any word in at most one way.

Let R be a fully-parenthesized regular expression over Σ and denote by $\tilde{R}$ the unireg expression obtained from R by replacing its regular operations with their unique counterparts. The following result can be proven by induction on the number of regular operations involved:

Lemma 6. *If R is unambiguous then $\mathcal{L}(R) = \mathcal{L}(\tilde{R})$.*

The converse of this lemma does not hold. Indeed, if $R = a + (a + a)$, then $\tilde{R} = a \oplus (a \oplus a)$ and $\mathcal{L}(R) = \mathcal{L}(\tilde{R}) = \{a\}$. However, R is obviously ambiguous.

From Lemmas 4, 6, and from the fact that any regular language is represented by an unambiguous regular expression [4], we infer a fundamental fact, that

Corollary 7. *Unireg expressions define the family of regular languages.*

The question whether context-free languages are closed under unique operations appears naturally at this point, and is answered in the following.

Proposition 8. *The following families are not closed under unique union: DCF, CF, and linear CF.*

Proof. It is clear that $\Sigma^* \overset{\circ}{\cup} L = \overline{L}$. However, it is well-known that the families CF and linear CF are not closed under complement.

For the DCF family, let $L_1 = \{a^ib^jc^k \mid i \neq j\}$ and $L_2 = \{a^ib^jc^k \mid j \neq k\}$. Clearly L_1 and L_2 are deterministic CFLs. Then $L_1 \overset{\circ}{\cup} L_2 = \{a^ib^jc^k \mid i = j \neq k \text{ or } i \neq j = k\}$.

We claim that $L_1 \overset{\circ}{\cup} L_2$ is not context-free. Suppose it is, and let n be the constant of Ogden's lemma. Let $z = a^nb^nc^{n+n!}$ and mark the b's. Let $z = uvwxy$ be a decomposition satisfying the conditions of Ogden's lemma. We have the next cases:

- $v = a^p$ and $x = b^q$. If $p = q$, then $uv^{n!/p}wx^{n!/q}y = a^{n+n!}b^{n+n!}c^{n+n!} \notin L_1 \overset{\circ}{\cup} L_2$. If $p \neq q$, then clearly $uv^2wx^2y \notin L_1 \overset{\circ}{\cup} L_2$.
- $v = b^p$ and $x = c^q$. Then clearly $uv^2wx^2y \notin L_1 \overset{\circ}{\cup} L_2$.
- $vwx = b^p$. Then clearly $uv^2wx^2y \notin L_1 \overset{\circ}{\cup} L_2$.

Thus the decomposition $z = uvxwy$ fails to satisfy the conditions of Ogden's lemma, a contradiction; thus, $L_1 \overset{\circ}{\cup} L_2$ is not context-free. □

Proposition 9. *The following families are not closed under unique concatenation: DCF, CF and linear CF.*

Proof. Consider the following CFL: $L_1 = \{a^n \mid n \geq 1\} \cup \{a^nb^n \mid n \geq 1\}$ and $L_2 = \{c^n \mid n \geq 1\} \cup \{b^nc^n \mid n \geq 1\}$. It is easy to see that

$$\begin{aligned} L_1 \circ L_2 &= \{a^nc^m \mid m, n \geq 1\} \cup \{a^nb^{n+m}c^m \mid m, n \geq 1\} \\ &\quad \cup \{a^nb^mc^m \mid m \neq n;\ m, n \geq 1\} \cup \{a^nb^nc^m \mid m \neq n;\ m, n \geq 1\}, \end{aligned}$$

that is, the only words in L_1L_2 that are written as a concatenation in more than one way are $a^nb^nc^n$, $n \geq 1$. We prove that $L_1 \circ L_2$ is not context-free by Ogden's lemma.

Assume by contradiction that it is, and let N be the constant of Ogden's lemma. Take the word $z = a^Nb^Nc^{N+N!} \in L_1 \circ L_2$. By Ogden's lemma, if we mark the a's, there exists a factorization $z = uvwxy$ such that vx has at least one marked symbol, vwx

has at most N marked symbols and $uv^iwx^iy \in L_1 \circ L_2$ for all $i \geq 0$. One can observe that such factorization must necessarily have $v = a^t$ and $x = b^t, 0 < t \leq N$. But then, for $i = 1 + N!/t$ we have $uv^iwx^iy = a^{N+N!}b^{N+N!}c^{N+N!}$ contradicting that such word must not be in $L_1 \circ L_2$.

Since L_1 and L_2 are both deterministic and linear CF languages, the conclusion follows. □

Proposition 10. *CF and linear CF families are not closed under unique star.*

Proof. Let L_1 and L_2 be as in the previous proposition. Clearly, $L_1 \cup L_2$ is context-free. It can be shown that $(L_1 \cup L_2)^\circ$ is not context-free by showing that $(L_1 \cup L_2)^\circ \cap a^*b^*c^*$ is not context-free, as before. From this, it follows that CF family is not closed under unique star. Since $L_1 \cup L_2$ is a linear CF language, it follows that linear CF family is not closed under unique star. □

4. State complexity

Before dealing with the state complexity of unique operations, we first prove a result concerning unambiguous computations in NFAs. The idea behind the following construction will be useful in proving upper bounds for the state complexities of unique concatenation and unique star.

Lemma 11. *Let A be an NFA of size m, and let L be the language of those words in Σ^* that are accepted unambiguously by A. Then L is regular and its state complexity is at most $3^m - 2^m + 1$.*

Proof. We construct a DFA whose states are vectors with m components, showing the number of paths from the initial state to each state: 0, 1 or 2 (2 stands for "two or more paths"). The final states are those vectors that denote exactly one path to one final state of the initial NFA.

Formally, let $A = (Q_A, \Sigma, \delta_A, q_A, F_A)$, $|Q_A| = m$. We construct a DFA $B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ for L of size $3^m - 2^m + 1$ as follows:

(1) $Q_B = V_B - V_B' \cup \{s\}$, where $V_B = \{0, 1, 2\}^m$ and V_B' is the subset of V_B consisting of those vectors that do not have the value 1 in any component. Clearly, $|Q_B| = 3^m - 2^m + 1$. State s has the role of a sink state.
(2) $q_B = (1, 0, \ldots, 0)$, F_B is the set of all vectors v such that the sum of all components of v corresponding to the final states of A is precisely 1.
(3) For all $a \in \Sigma$, we denote by M_a the incidence matrix of A with respect to the symbol a ($M_a[i, j]$ is 1 if there is a transition from q_i to q_j labeled with a, and 0 otherwise). Then, for all $v \in V_B - V_B'$,

$$\delta_B(v, a) = \begin{cases} vM_a, & \text{if } vM_a \notin V_B'; \\ s, & \text{if } vM_a \in V_B'; \end{cases}$$

and $\delta_B(s, a) = s$. Here the matrix multiplication is done as usual, but with $\oplus$ and $\otimes$ as component-wise addition and multiplication, described as follows: for $a, b \in \{0, 1, 2\}$, we define $a \oplus b = \min(a + b, 2)$ and $a \otimes b = \min(a \cdot b, 2)$. (See [18] for an early use of these operations.) □

Notice that the construction can be modified to recognize the language of those words that are accepted ambiguously by the NFA: just make the appropriate states final. Since this symmetry holds for most constructions proposed throughout the paper, we state, a priori, the following fact:

Theorem 12. *The state complexity results on unique operations, based on this construction and described in the following, hold for poly operations as well.*

While we have not proven that the bound given in Lemma 11 is tight, we can give an exponential lower bound, as follows. For $k \geq 0$, define a language

$$L_k = (0 + 1)^*0(0 + 1)^k1(0 + 1)^*.$$

The languages L_k (or variations thereof) have been used by several authors [22,12,14,8] to prove lower bounds for nondeterministic state complexity. The language L_k consists of all words containing *at least one occurrence* of a word in $0(0+1)^k1$. Now consider the language $UL_k = (0 \oplus 1)^\circ \circ 0 \circ (0 \oplus 1)^{\circ k} \circ 1 \circ (0 \oplus 1)^\circ$, obtained from the regular expression for L_k by replacing the ordinary operations with the unique ones. The language UL_k consists of all words containing *exactly one occurrence* of a word in $0(0 + 1)^k1$.

Lemma 13. *Any NFA accepting UL_k has at least 2^k states.*

Proof. For every word $x \in \{0, 1\}^k$, define a pair $(0x, 1x)$. Note that $0x1x$ is in UL_k, since there is exactly one instance where a 0 is followed by a 1 k positions later. However, for any 2 distinct words x and y, at least one of the words $0x1y$ or $0y1x$ must contain *two* occurrences of a subword in $0(0 + 1)^k1$ (since x and y must mismatch in at least one place). Thus, at least one of $0x1y$ or $0y1x$ is not in L. Since there are 2^k pairs, it follows from a result of Birget [1] that any NFA for UL_k has at least 2^k states. □

From this, we easily deduce the following results:

Proposition 14. *There exists an NFA M_k with $O(k)$ states such that any DFA, NFA, or regular expression for the set of words accepted unambiguously by M_k has size at least 2^k.*

Proof. The language L_k is accepted by an NFA M_k with $O(k)$ states. The set of words accepted unambiguously by M_k is exactly UL_k. The result now follows from Lemma 13. □

Proposition 15. *There exists a regular language generated by a unireg expression of size $O(k)$ such that any equivalent DFA, NFA, or regular expression has size at least 2^k.* (The language UL_k used in Lemma 13 has the desired properties.)

4.1. Unique union

For the unique union, we observe that given two DFAs A and B, of size m and n respectively, we can easily construct a DFA of size mn for $L(A) \mathbin{\mathring{\cup}} L(B)$ by performing the cross-product of A and B and by setting as final states those state pairs that have exactly one component final. We prove that this upper bound of mn is tight.

Theorem 16. *For $m, n \geq 3$, let L_1 and L_2 be accepted by DFAs with m and n states respectively. The state complexity of $L_1 \mathbin{\mathring{\cup}} L_2$ is mn.*

Proof. For $m, n \geq 3$, we use the languages $A = \{w \in \{0,1\}^* \mid |w|_0 \equiv m-1 \bmod m\}$ and $B = \{w \in \{0,1\}^* \mid |w|_1 \equiv n-1 \bmod n\}$, that were used by Maslov [19] to prove a similar result for the ordinary union. Clearly, A is accepted by an m-state DFA, B is accepted by an n-state DFA, and $C = A \mathbin{\mathring{\cup}} B$ is accepted by an mn-state DFA. We show that mn states are necessary.

For integers i, i' and j, j', $0 \leq i, i' \leq m-1$ and $0 \leq j, j' \leq n-1$, let $x = 0^i 1^j$ and $y = 0^{i'} 1^{j'}$ be distinct words. To complete the proof, it is enough to show that x and y are inequivalent with respect to the Myhill–Nerode equivalence relation. We leave the details to the reader. □

4.2. Unique concatenation

We now approach the more difficult problem of determining the state complexities of unique concatenation and unique star.

Theorem 17. *The state complexity of unique concatenation for regular languages is at most $m3^n - k3^{n-1}$, where m and n are the sizes of the input DFAs, and k is the number of final states of the first DFA.*

Proof (*Sketch*). Let $A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ be the input DFAs, of size m and n respectively. For proving the upper bound we construct a DFA $C = (Q_C, \Sigma, \delta_C, q_C, F_C)$ for $L(A) \circ L(B)$, of size $m3^n - k3^{n-1}$:

(1) $Q_C = Q_A \times V_B - F_A \times V'_B$, where $V_B = \{0,1,2\}^n$ and V'_B is a subset of V_B, of those vectors that have 0 in their first component. Clearly, $|Q_C| = m3^n - k3^{n-1}$.
(2) $q_C = \langle q_A, (0,\ldots,0)\rangle$ if $q_A \notin F_A$ and $q_C = \langle q_A, (1,0,\ldots,0)\rangle$ otherwise; F_C is the set of those states $\langle q, v\rangle$ such that the sum of all components of v corresponding to the final states of B is precisely 1.
(3) For $a \in \Sigma$ we denote by M_a the incidence matrix of B with respect to the symbol a. Then $\delta_C(\langle q, v\rangle, a) = \langle \delta_A(q,a), v'\rangle$, where $v' = vM_a$ if $\delta_A(q,a) \notin F_A$ and $v' = vM_a + (1,0,\ldots,0)$ otherwise. The matrix operations are done as usual, with the component-wise $\oplus$ and $\otimes$ as described in the proof of Lemma 11.

The idea of this construction is to find the "multiplicity" of ambiguous computations of the NFA for $L(A)L(B)$, as inspired by the proof of Lemma 11. □

Considering the DFA N_n in Fig. 3, we have found that this DFA is a state complexity worst-case for unique square, proving that the upper bound is reached:

Proposition 18. *For $n \geq 3$, the state complexity of $L(N_n)^{\circ 2}$ is $n3^n - 3^{n-1}$, thus this is a sharp upper bound for unique square (when $k = 1$).*

Proof (*Sketch – [21] Provides a Complete Proof*). We show that the construction in the proof of Theorem 17 leads to a minimal DFA, by proving its total reachability and non-mergibility. Consider that the states of N_n are numbered (and named) from 0 to $n-1$, and recall that the corresponding DFA (as constructed in Theorem 17) has states of the form $\langle i, (x_1,\ldots x_n)\rangle$, where $x_j \in \{0,1,2\}$ and the component-wise operations of the vectors $(x_1,\ldots x_n)$ are $x \oplus y = \min(x+y, 2)$ and $x \otimes y = \min(x \cdot y, 2)$.

The following facts about state transitions are useful:

$$\langle 0, (x_1,\ldots,x_n)\rangle \xrightarrow{a} \langle 0, (x_1 \oplus x_n, 0, x_2, \ldots, x_{n-1})\rangle,$$
$$\langle i, (x_1,\ldots,x_n)\rangle \xrightarrow{a} \langle i+1 \bmod n, (x_1 \oplus x_n, 0, x_2, \ldots, x_{n-1})\rangle, \quad \forall i \neq 0,$$
$$\langle j, (x_1,\ldots,x_n)\rangle \xrightarrow{b} \langle j+1 \bmod n, (x_n, x_1, \ldots, x_{n-1})\rangle, \quad \forall j \neq n-2,$$

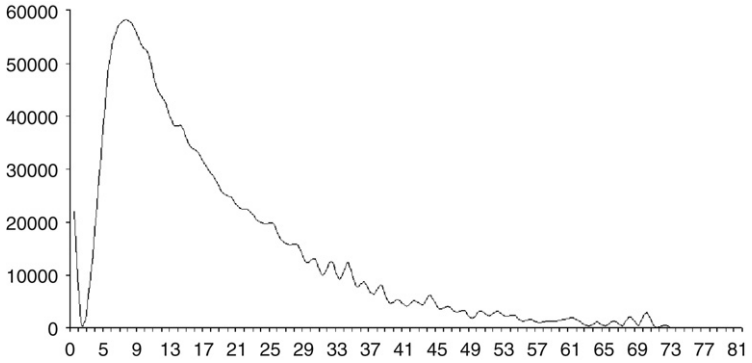

Fig. 1. Histogram for unique concatenation over 3-state minimal DFAs.

$$
\begin{aligned}
&\langle n-2,(x_1,\ldots,x_n)\rangle \xrightarrow{b} \langle n-1,(x_n\oplus 1,x_1,\ldots,x_{n-1})\rangle,\\
&\langle 0,(x_1,\ldots,x_n)\rangle \xrightarrow{b^n} \langle 0,(x_1,x_2\oplus 1,x_3,\ldots,x_n)\rangle,\\
&\langle 1,(x_1,\ldots,x_n)\rangle \xrightarrow{b^n} \langle 1,(x_1,x_2,x_3\oplus 1,x_4,\ldots,x_n)\rangle,\\
&\qquad\ldots\\
&\langle j,(x_1,\ldots,x_n)\rangle \xrightarrow{b^n} \langle j,(x_1,x_2,\ldots,x_{j+2}\oplus 1,\ldots,x_n)\rangle, \quad \forall j\neq n-1,\\
&\qquad\ldots\\
&\langle n-2,(x_1,\ldots,x_n)\rangle \xrightarrow{b^n} \langle n-2,(x_1,x_2,\ldots,x_n\oplus 1)\rangle,\\
&\langle n-1,(x_1,\ldots,x_n)\rangle \xrightarrow{b^n} \langle n-1,(x_1\oplus 1,x_2,\ldots,x_n)\rangle.
\end{aligned}
$$

Reachability. Let $\langle i,(x_1,\ldots,x_n)\rangle$ be an arbitrary state, $i<n-1$. From the initial state $\langle 0,(0,\ldots,0)\rangle$ we first reach $\langle 0,(x_{i+1},\ldots,x_n,x_1,\ldots,x_i)\rangle$ by reading the word $b^{nx_{i+1}}ab^{nx_i}ab^{nx_{i-1}}\cdots ab^{nx_1}ab^{nx_n}ab^{nx_{n-1}}\cdots ab^{nx_{i+2}}$. If we further apply the word b^i we reach $\langle i,(x_1,\ldots x_n)\rangle$. For reaching $\langle n-1,(x_1,\ldots,x_n)\rangle$, with $x_1>0$ (recall that x_1 cannot be 0 in this case), we first reach $\langle 0,(x_n,x_1-1,\ldots,x_{n-1})\rangle$, then we apply b^{n-2} reaching $\langle n-2,(x_2,\ldots,x_n,x_1-1)\rangle$, and then we apply b one more time.

Non-mergibility. We now show that no two distinct states $\langle i,(x_1,\ldots,x_n)\rangle$ and $\langle j,(y_1,\ldots,y_n)\rangle$ are mergible, by providing a word that maps one of these states into a final state and the other into a non-final state. Incidentally, it becomes apparent that our DFA has no sink state.

If $i\neq j$, we choose the word $a^{n-i}b^na^{n-2}$. From $\langle i,(x_1,\ldots,x_n)\rangle$ we reach a final state, namely $\langle n-2,(1\oplus\sum_{j=1}^n x_j,0,\ldots,0,1)\rangle$. For the other state we reach $\langle j+n-i \mod n,(\sum(\cdots),0,\ldots,1,\ldots,0)\rangle$, which is not final.

If $i=j$, there must be a position k such that $x_k\neq y_k$, since the states are distinct. Without loss of generality we may assume that $x_k<y_k$ (otherwise we flip the states). We distinguish the following subcases:

I. $x_k=1$ or $y_k=1$. For $x_k=1$ we use the word b^{n-k} (recall that $x_k\neq y_k$), and for $y_k=1$ we flip the states.

II. $x_k=0$, $y_k=2$. Here we distinguish two situations. If $k=i+2$ (thus, $i\leq n-2$) we choose the word $b^{2(n-1)-i}$, which maps $\langle i,(x_1,\ldots,x_n)\rangle$ into the final state $\langle n-2,(x_{k+1},\ldots x_n,x_1,\ldots,1=x_k\oplus 1)\rangle$. The same word maps $\langle i,(y_1,\ldots y_n)\rangle$ into a non-final state, for $y_k\oplus 1=2$. If $k\neq i+2$, we first apply the word $b^{n-k+2}a^{n-2}ba$, which maps $\langle i,(x_1,\ldots,x_n)\rangle$ into $\langle t,(x_k,0,z,0,\ldots,0)\rangle$, for some $t\in\{0,\ldots,n-1\}$ and $z\in\{0,1,2\}$. From here, there exists a word $w=a^r$ that continues the computation up to $\langle *,(1=x_k\oplus 1,0,\ldots 0,z,0,\ldots,0)\rangle$, and after that, the word b^{n-1} leads to the state $\langle *,(0,\ldots 0,z,0,\ldots,0,1)\rangle$. Thus, the word $b^{n-k+2}a^{n-2}bawb^{n-1}$ maps $\langle i,(x_1,\ldots,x_n)\rangle$ into a final state, however, this is not true for $\langle i,(y_1,\ldots,y_n)\rangle$. □

We can also prove the following exponential lower bound for the non-deterministic state complexity of unique concatenation.

Proposition 19. *There exists a pair of NFAs M_1 and M_2 with $O(k)$ states combined, such that $L(M_1)L(M_2)$ is accepted by an $O(k)$ state NFA, but any NFA accepting $L(M_1)\circ L(M_2)$ has at least 2^k states.*

Proof. Take $L(M_1)=(0+1)^*0(0+1)^k1$ and $L(M_2)=(0+1)^*$. Then $L(M_1)L(M_2)$ is accepted by an $O(k)$ state NFA, but any NFA accepting $L(M_1)\circ L(M_2)=UL_k$ has at least 2^k states (see Lemma 13 and def. of UL_k used in Lemma 13). □

On the experimental side, we generated all minimal DFAs with 3 states and performed unique concatenation on all pairs. There are 1028 distinct DFAs, leading to 1056,784 operations. Fig. 1 provides a histogram of our results: the x-axis represents the size of the output DFAs, and the y-axis plots the number of cases that resulted in DFAs of that size. For two DFAs of size m and n, the theoretical upper bound is $m3^n-k3^{n-1}$ (k is the number of final states in the first DFA). The largest DFAs obtained from this experiment are of size 72, and are the result of operations where the first DFA has precisely one final state. Thus the bound is reached for $m=n=3$ and $k=1$. Notice that small DFAs have a higher incidence rate, hinting at the fact that *the worst-case scenarios are very sparse*.

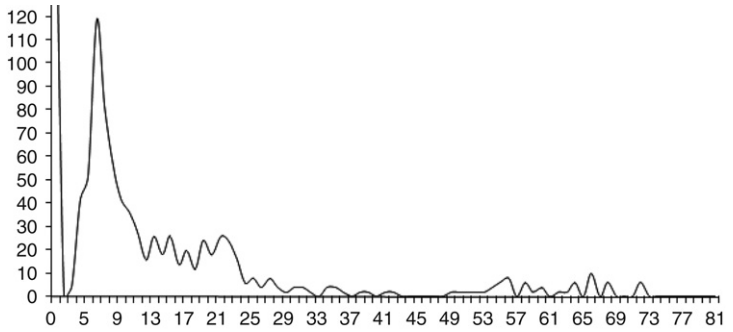

Fig. 2. Histogram for unique square over 3-state minimal DFAs.

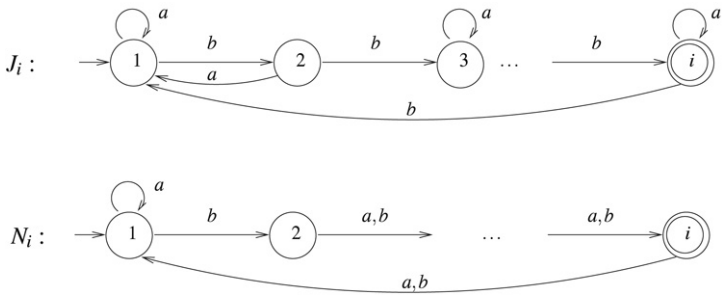

Fig. 3. Parameterized automata J_i and N_i.

Then, we investigated whether the unique square has a smaller state complexity. For this we performed this operation for all minimal DFAs of size 3 and 4. The results for 3-state DFAs are shown in Fig. 2. We found 6 minimal DFAs of size 3 (with one final state) whose unique square reaches the upper bound of 72.

Consider the two parameterized minimal DFAs, J_i and N_i, with $i \geq 3$, shown in Fig. 3. Our experiments show that the upper bound is reached for any of the following combinations: $L(J_i)^{\circ 2}$, $L(N_i)^{\circ 2}$, $L(J_i) \circ L(J_j)$, $L(N_i) \circ L(N_j)$, $L(J_i) \circ L(N_j)$, with i, j arbitrary integers greater than 2. It is interesting to notice that J_i is given in [20] as example for reaching the upper bound for the normal concatenation, hence it may provide an example where worst-case is achieved for both concatenation and unique concatenation.

4.3. Unique star

Using a technique similar to that for unique concatenation we can derive an upper bound for the unique star:

Theorem 20. *If $L \setminus \{\varepsilon\}$ is accepted by a DFA A of size m and with k final states, then a DFA for L° has at most $3^{m-1} + (k+2)3^{m-k-1} - (2^{m-1} + 2^{m-k-1} - 2)$.*

Proof. Let $A = (Q_A = \{1, 2, \ldots, m\}, \Sigma, \delta_A, 1, F_A)$ be a DFA for L, of size m, and $F_A = \{m-k+1, \ldots, m\}$. By M_a we denote the adjacency matrix of A with respect to the symbol $a \in \Sigma$, and the matrix operations are performed with the usual $\oplus$ and $\otimes$ component-wise operations. We define a DFA $B = (Q_B, \Sigma, \delta_B, 0, F_B)$ for L° as follows:

(1) $Q_B = V \cup \{0\}$, where 0 is the initial state of B and V is the set of all vectors with m components holding values in $\{0, 1, 2\}$. The vector entries are indexed from 1 to m.
(2) The transition function is defined as follows:
   (a) $\delta_B(0, a) = v_a$, where $v_a[\delta_A(1, a)] = 1$, $v_a[1] = 1$ if $\delta_A(1, a) \in F_A$, and $v_a[i] = 0$ for all other indices i.
   (b) Denote $S_k[v]$ to be the value $v[m-k+1] \oplus \cdots \oplus v[m]$. For all $j \in \{1, \ldots, m\}$ and $a \in \Sigma$ we set $\delta_B(v, a) = v' + v''$, where $v' = vM_a$ and $v'' = (S_k(v'), 0, 0, \ldots, 0)$.
(3) $F_B = \{v \in V \mid S_k(v) = 1\} \cup \{0\}$.

We use vectors to store the number of computations in A, from the initial state to every state: 0, 1, or 2 standing for more than one computation. If a vector v is reached during the computation of B, the value $S_k(v)$ gives the number of different computations in A reaching final states. This number has been added to the first component of v, meaning that reaching a final state in A implies reaching its initial state as well, for we aim at accepting words in L^*. If a word w "reaches" a state-vector v in B, then $v[i]$ gives the number (0, 1, or 2) of distinct paths in A, labeled with w, from the initial state of A to its state i, when A is modified to accept L^* in the standard way. By setting as final states in B all those vectors that denote exactly one successful such path, we force B to accept exactly the words in L°.

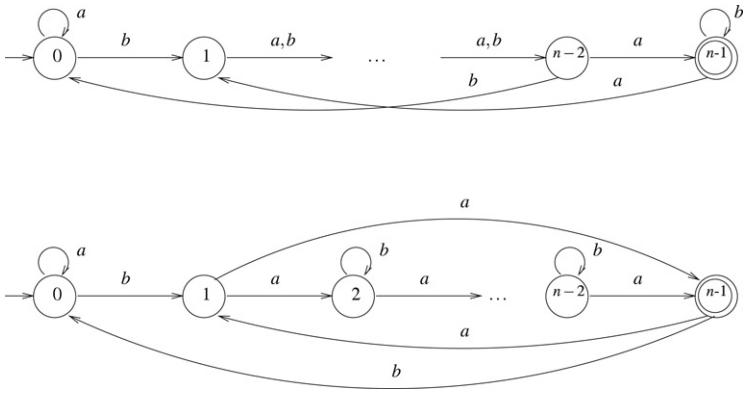

Fig. 4. Worst-case candidates for unique star.

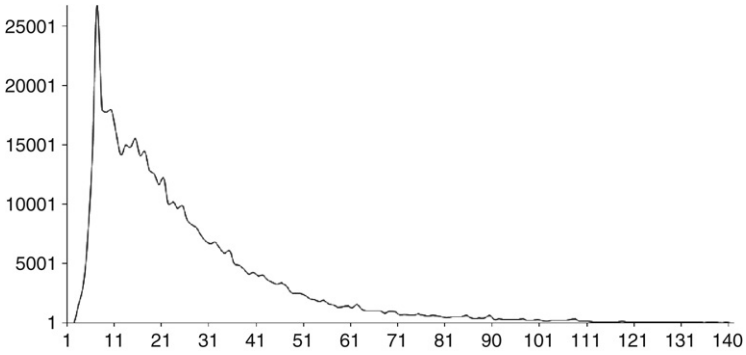

Fig. 5. Histogram for unique star over 5-state minimal DFAs with $k = 1$.

We now make two crucial observations: (a) for a reachable state $v \in V$ we must have $v[1] \geq S_k(v)$, and (b) any reachable state $v \in V$ containing only values 0 and 2 is mergible into (or, is equivalent to) the sink state. It now remains to compute how many states can possibly be reached in B:

1. There is an initial state 0 and eventually a sink state, amounting for $\underline{2 \text{ states}}$.
2. At most $3^{m-k} - 1$ vectors v with $S_k(v) = 0$ can be reached (the null vector cannot be reached). From these vectors, we subtract those having only 0's and 2's, for they will eventually be merged together within a sink state when B is minimized. There are $2^{m-k} - 1$ such vectors, without counting the null vector. Thus, we have altogether $\underline{3^{m-k} - 2^{m-k} \text{ states}}$ in this case.
3. At most $\underline{2k \cdot 3^{m-k-1} \text{ states}}$ v with $S_k(v) = 1$ can be reached. Observe that once $S_k(v) = 1$ we cannot have $v[1] = 0$ since $S_k(v)$ has been added to $v[1]$ during an eventual transition. Thus, $v[1]$ can take two values (1 and 2), then the portion of the vector $v[2, \ldots, m-k]$ gives 3^{m-k-1} combinations, and there are at most k combinations of $v[m-k+1, \ldots, m]$ that ensure $S_k(v) = 1$.
4. Finally, at most $3^{m-k-1}(3^k - k - 1)$ states v with $S_k(v) = 2$ can be reached. Indeed, we have at most $3^k - k - 1$ combinations in $v[m-k+1, \ldots, m]$ that ensure $S_k(v) = 2$. Then $v[1]$ must be 2 (since $S_k(v)$ has been added to it), and there are 3^{m-k-1} combinations for $v[2, \ldots, m-k]$. However, some of these vectors are mergible into the sink state: those with only 0's and 2's. There are exactly $2^{m-k-1}(2^k - 1)$ such vectors v, since: $v[1] = 2$, there are $(2^k - 1)$ combinations in $v[m-k+1, \ldots, m]$ (this portion cannot be all 0's), and there are 2^{m-k-1} combinations of 0's and 2's in $v[2, \ldots, m-k]$. Combining these numbers, we obtain $\underline{3^{m-k-1}(3^k - k - 1) - 2^{m-k-1}(2^k - 1) \text{ states}}$ in this case.

It remains to add up the figures underlined in the above cases 1–4. □

The upper bound in Theorem 20 has been reached for $k = 1$ and $m = 2, \ldots, 8$ by the generic examples in Fig. 4 (which are good candidates for the worst-case in general), and we conjecture that this upper bound is sharp in both n and k. In Fig. 5 we plotted the histogram for all minimal DFAs with 5 states and one non-initial final state.

The case when $\varepsilon \in L$, and we are given a DFA for L, is proven similarly, and may lead to a slightly different upper bound. In fact, we can immediately derive an upper bound by noticing that a DFA for $L \setminus \{\varepsilon\}$ has one state more than the DFA for L (thus, we just replace m by $m + 1$ in the above result). Nevertheless, a proof as in Theorem 20 may improve such upper bound, and it merely involves a different state-indexing scheme. We leave this exercise to the reader.

5. Decision problems

We consider two decision problems involving unireg expressions, and start with the membership problem:

Theorem 21. *The membership problem for unireg expressions is in P.*

Proof. Let R be a unireg expression over Σ, and w be a string in Σ^*. If $w = \varepsilon$, we can determine efficiently the membership, by consulting the parse-tree for R. If $w \neq \varepsilon$, we proceed as follows. Let $w = a_1 a_2 \dots a_k$ and let R' be the regular expression obtained from R by replacing the unique operations with the corresponding regular operations. We use the algorithm of Glushkov to obtain an ε-free NFA M such that the set of strings accepted by M is the same as the set of strings generated by R' (with the possible exception of ε). It is known [4] that Glushkov's algorithm preserves the degree of ambiguity of representation, that is, the number of accepting computations in M for an input word w equals the number of ways in which R' generates w. Then, $L(R)$ consists of those words that are accepted by M in an unique computation or, we say, unambiguously. Thus, it now suffices to detect whether our word w is accepted unambiguously by M. We consider that the states in M are numbered from 0 to $m-1$, and denote $\{T_a\}_{a\in\Sigma}$ the set of incidence matrices of M. Let $S = [s(0), s(1), \dots, s(m-1)]$ where $s(i) = 1$ if i is the start state, 0 otherwise. Similarly, let $F = [f(0), f(1), \dots, f(m-1)]$ where $f(j) = 1$ if j is an accepting state of M, 0 otherwise. Then $ST_{a_1}T_{a_2}\dots T_{a_k}F$ is the number of accepting paths for the string w in M. By computing the above matrix chain product, we can determine the number of accepting paths for w. If this number is 1, then w is accepted; otherwise it is rejected. This algorithm runs in time polynomial in $|R|+|w|$, where $|R|$ is the number of symbols in R. □

Theorem 22. *The non-emptiness problem for unireg expressions is in PSPACE.*

Proof. Let R be a unireg expression over Σ and R' be the regular expression obtained from R by replacing the unique operations with the standard regular operations. Let M be the NFA obtained by applying Glushkov's algorithm to R'. Then $L(R)$ is non-empty if and only if there exists a word w accepted unambiguously by M. By Savitch's theorem [26], it suffices to give a non-deterministic polynomial space algorithm to test for the existence of such word w.

For $a \in \Sigma$, let B_a denote the adjacency matrix of M with respect to the input a. By Lemma 11, if there is a word w accepted unambiguously by M, there is such a w of length at most 3^n, where n is the number of states of M. We thus non-deterministically guess such a word $w = w_1 w_2 \dots w_r$, $r \leq 3^n$, symbol by symbol, and we compute the matrix product $B_{w_1}B_{w_2}\cdots B_{w_r} = B$, reusing space after each matrix multiplication. Here the matrix multiplication is again done with $\oplus$ and $\otimes$ as component-wise operations. We maintain an $O(n)$ bit counter to keep track of the length of the guessed string w. We verify that M accepts w unambiguously by looking at the row of B corresponding to the start state of M and summing up the entries in the columns corresponding to the final states of M. This quantity is exactly 1 if and only if M accepts w unambiguously.

The transformation of R to R' and then to M can be done in polynomial space, and the non-deterministic algorithm described above uses only polynomial space. It follows that the non-emptiness problem can be solved in polynomial space. □

6. Application: 2-DFAs with a pebble

It is well-known that if M is a 2-DFA with a pebble [2], $L(M)$ is regular. Here we revisit the following question, studied in [10]: *What is the worst-case blow-up in the number of states when a 2-DFA with a pebble is converted into a 1-DFA?* Let $f(n)$ denote this function. Formally, f is defined by the following two conditions: (1) there is an n-state 2-DFA with a pebble such that the minimum equivalent 1-DFA has $f(n)$ states, and (2) for any n-state 2-DFA with a pebble, there is an equivalent 1-DFA with at most $f(n)$ states. A lower-bound on $f(n)$ can be obtained from the results of the previous section. Indeed, the connection between the state complexity for converting 2-DFA with a pebble to a 1-DFA and the state complexity of unique concatenation is provided by the following proposition.

Proposition 23. *Let A and B be two DFAs, with m and n states. There exists a 2-DFA C with a pebble such that $L(C) = L(A) \circ L(B)$ and C has $2(m+n)+2$ states.*

Proof (*Sketch*). The state set of C is given by $Q_C = Q_A \cup Q_B \cup Q'_A \cup Q'_B$, where $Q'_A = \{q' \mid q \in Q_A\}$ and $Q'_B = \{q' \mid q \in Q_B\} \cup \{r, r'\}$. On an input string \$w#, C starts with the reading head on the left end-marker, in its start state s_C — which is, by definition, s_A. The head moves to the right, and C simulates A until an accepting state is reached. At this point, C places a pebble on the current tape square, enters s_B and moves to the right simulating B, until the right end-marker is reached. If at this point an accepting state of B is not reached, then C proceeds as in Step 1, else it proceeds as in Step 2, detailed as follows:

Step 1: C enters the state r and makes a right-to-left sweep until it reaches the left end-marker, and enters the state s_A. Then it simulates A as usual, with the difference that when it reaches the pebble, it picks it up and continues the computation to the right till another final state of A is reached. Then it drops the pebble and continues with the simulation of B, as in the initial phase.

Step 2: C enters the state r' and makes a right-to-left sweep until it reaches the left end-marker and enters state s'_A. While in a state of the form $q'_a \in Q'_A$, C simulates A, but uses the primed states and keeps moving to the right. More precisely, if $\delta_A(q_a, b) = q_d$, then C, on input b and in state q'_a, changes its current state to q'_d and moves to the right. It continues

this phase until the square with a pebble is detected. Here, it picks up the pebble and moves to the right continuing the simulation of A using the primed states. At this point there are two cases to consider. (a) As the simulation of A continues, the right end-marker is reached without ever reaching an accepting state of A. In this case, C accepts the input and it halts. (b) An accepting state of A is reached before the right end-marker is reached. When an accepting state of A is reached for the first time, the pebble is dropped on the square that stores the last symbol that caused A to reach the accepting state, C enters the state s'_B and it starts the simulation of B using the primed states. The simulation continues until the right end-marker is reached. At this point, if an accepting state of B is reached, C rejects the input and halts. If an accepting state is not reached, then C enters the state r' and repeats Step 2.

It is clear that C accepts $L(A) \circ L(B)$ and the proof is complete. □

This result, together with Theorem 17 or Proposition 18, provides an exponential lower bound for $f(n)$, whereas the lower bound given in [10] is doubly-exponential.

7. Conclusion and further work

In this paper we studied unique rational operations and their state complexity. We drew connections between unireg expressions and unambiguous regular expressions, and we studied the closure of DCF, CF and linear CF languages to unique operations. We obtained a sharp bound of the state complexity for unique union, comparable with that of "plain" union. For unique concatenation we gave a state complexity upper bound that we strongly believe to be sharp, for we provided generic (parameterized) examples that reached the upper bound in all our extensive experiments. For the unique square, we provided sharp upper bounds and a generic worst-case example, in the laborious proof of Proposition 18. Both bounds are significantly higher than those for plain concatenation. For the non-deterministic state complexity of unique concatenation we provided an exponential lower bound. In Theorem 20 we provided a curious upper bound for the unique star, that we believe yet again to be sharp, based on empirical results. Finally, we studied the complexity of the membership and non-emptiness problem for unireg expressions, and we drew a connection between 2-DFAs with a pebble and unique concatenation. This work is in progress, and we feel that much more needs to be done. A list of open questions and further directions can be found in the extended paper [21].

Acknowledgements

This study benefitted greatly from our extensive experiments that could have not been possible without the use of Grail+, carefully developed and maintained by Derick Wood, Sheng Yu, and its project members for over 20 years.

References

[1] J.-C. Birget, Intersection and union of regular languages and state complexity, Information Processing Letters 43 (1992) 185–190.
[2] M. Blum, C. Hewitt, Automata on a 2-dimensional tape, in: Proc. 8th IEEE Symp. Switching and Automata Theory, 1967, pp. 155–160.
[3] G.V. Bochmann, Submodule construction and supervisory control: A generalization, in: Proc. CIAA 2001, in: LNCS, vol. 2494, 2001.
[4] R. Book, S. Even, S. Greibach, G. Ott, Ambiguity in graphs and expressions, IEEE Transactions on Computers C-20 (2) (1971) 149–153.
[5] C. Campeanu, K. Culik, K. Salomaa, S. Yu, State complexity of basic operations on finite languages, in: Proc. WIA 1999, in: LNCS, vol. 2214, 1999.
[6] C. Campeanu, K. Salomaa, S. Yu, Tight lower bound for the state complexity of shuffle of regular langauges, Journal of Automata, Languages and Combinatorics 7 (2002) 303–310.
[7] M. Daley, M. Domaratzki, K. Salomaa, On the operational orthogonality of languages, in: Proc. TALE 2007, vol. 44, TUCS General Publication, June 2007.
[8] K. Ellul, B. Krawetz, J. Shallit, M. Wang, Regular expressions: New results and open problems, Journal of Automata, Languages and Combinatorics 10 (2005) 407–437.
[9] Y. Gao, K. Salomaa, S. Yu, State complexity of catenation and reversal combined with star, in: Proc. DCFS 2006, vol. 2494, 2006.
[10] N. Globerman, D. Harel, Complexity results for two-way and multi-pebble automata and their logics, Theoretical Computer Science 169 (2) (1996) 161–184.
[11] D. Harel, M. Politi, Modeling Reactive Systems with Statecharts, McGraw-Hill, 1998.
[12] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, International Journal of Foundations of Computer Science 14 (6) (2003) 1087–1102.
[13] J. Hopcroft, R. Motwani, J. Ullman, Introduction to Automata Theory, Languages, and Computation, 3rd edition, Addison-Wesley Publishing Co, Boston, MA, 2006.
[14] J. Hromkovič, J. Karhumäki, H. Klauck, G. Schnitger, Communication complexity method for measuring nondeterminism in finite automata, Information and Computation 172 (2002) 202–217.
[15] T. Jiang, B. Ravikumar, A note on the space complexity of some decision problems for finite automata, Information Processing Letters 40 (1) (1991) 25–31.
[16] L. Kari, On language equations with invertible operations, Theoretical Computer Science 132 (2) (1994) 129–150.
[17] L. Karttunen, Applications of finite-state transducers in natural languages processing, in: Proc. CIAA 2000, in: LNCS, vol. 2088, 2000.
[18] A. Mandel, I. Simon, On finite semigroups of matrices, Theoretical Computer Science 5 (1977) 101–111.
[19] A.N. Maslov, Estimates of the number of states of finite automata, Doklady AkademiiNauk SSSR 194 (1970) 1266–1268 (in Russian). English translation in Soviet Mathematics Doklady 11 (1970) 1373–1375.
[20] N. Rampersad, The state complexity of L^2 and L^k, Information Processing Letters 98 (2006) 231–234.
[21] N. Rampersad, B. Ravikumar, N. Santean, J. Shallit, A study on unique rational operations, Tech. Rep. TR-20071222-1, Department of Computer and Information Sciences, Indiana University South Bend, also available at: http://www.cs.iusb.edu/technical_reports.html, December, 2007.
[22] B. Ravikumar, O. Ibarra, Relating the type of ambiguity of finite automata to the succinctness of their representation, SIAM Journal on Computing 18 (1989) 1263–1282.
[23] G. Rozenberg, A. Salomaa, Handbook of Formal Languages, Springer-Verlag, Berlin, Heidelbergm, New York, 1997.
[24] A. Salomaa, K. Salomaa, S. Yu, State complexity of combined operations, Theoretical Computer Science 383 (2007) 140–152.

[25] A. Salomaa, D. Wood, S. Yu, On the state complexity of reversals of regular languages, Theoretical Computer Science 320 (2004) 293–313.
[26] W. Savitch, Relationship between nondeterministic and deterministic tape complexities, Journal of Computer and System Sciences 4 (1970) 177–192.
[27] S. Yu, Regular Languages, In [23] Ch.1 (1997) 41–110.
[28] S. Yu, State complexity: Recent results and open problems, Fundamenta Informaticae 64 (2005) 471–480.
[29] S. Yu, Q. Zhuang, K. Salomaa, The state complexities of some basic operations on regular languages, Theoretical Computer Science 125 (1994) 315–328.