Acta Informatica (2016) 53:67–85
DOI 10.1007/s00236-015-0245-y

ORIGINAL ARTICLE

State complexity of deletion and bipolar deletion

Yo-Sub Han[1] · Sang-Ki Ko[1] · Kai Salomaa[2]

Received: 16 December 2014 / Accepted: 30 May 2015 / Published online: 16 June 2015

Abstract It is well known that the language obtained by deleting an arbitrary language from a regular language is regular. We give an upper bound for the state complexity of deleting an arbitrary language from a regular language and a matching lower bound. We show that the state complexity of deletion is $n \cdot 2^{n-1}$ [respectively, $(n + \frac{1}{2}) \cdot 2^n - 2$] when using complete (respectively, incomplete) deterministic finite automata. We show that the state complexity of bipolar deletion has an upper bound n^n [respectively $(n + 1)^n - 1$] when using complete (respectively, incomplete) deterministic finite automata. In both cases we give almost matching lower bounds.

1 Introduction

The descriptional complexity of finite automata has been studied for over half a century [26, 27, 29, 30] and there has been much renewed interest since the early 90s [13, 15, 24, 36]. Operational state complexity investigates the size of a DFA (deterministic finite automaton)

An earlier version of the paper was presented at the 18th International Conference Developments in Language Theory (Ekaterinburg, Russia, August 26–29, 2014) and an extended abstract appeared in the proceedings of the conference. Han and Ko were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562), the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512) and Yonsei University future-leading research initiative of 2014, and Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

✉ Kai Salomaa
ksalomaa@cs.queensu.ca

Yo-Sub Han
emmous@cs.yonsei.ac.kr

Sang-Ki Ko
narame7@cs.yonsei.ac.kr

1 Department of Computer Science, Yonsei University, 50, Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Korea

2 School of Computing, Queen's University, Kingston, ON K7L 2N8, Canada

needed to recognize the language obtained by applying a regularity preserving operation to given DFAs. The precise worst case state complexity of many basic language operations has been established; see e.g. [2,5,8,14,17,23,25,27,31,33,37]. Also there has been much work on the state complexity of combinations of basic language operations [3,7,9,10,18,32]. More references can be found in a recent review [11].

While most of the literature uses complete DFAs to measure the state complexity of a regular language, also the state complexity based on incomplete DFAs has been considered. Câmpeanu et al. [1] give the state complexity of shuffle in terms of incomplete DFAs while the precise state complexity of shuffle in terms of complete DFAs remains still open. For a given regular language the sizes of the minimal complete and the minimal incomplete DFA differ by at most one state, however, there can be a more significant difference in the state complexity functions when the measure is based on complete and incomplete DFAs, respectively.

Deletion is one of the basic operations in formal language theory [20,21]. The deletion of a string v from a string u is defined as the operation that erases a contiguous substring v from u. We denote the result of deleting a language L_2 from a language L_1 by $L_1 \rightsquigarrow L_2$. (A formal definition of the deletion operation is given in Sect. 2.)

Deletion is the simplest and most natural generalization of the left/right quotient [21]. It is known that for L_1 recognized by a DFA with n states and an arbitrary language L_2, the worst case state complexity of the left-quotient $L_2 \backslash L_1$ is $2^n - 1$ and the state complexity of the right-quotient L_1 / L_2 is n [36]. Recently, the state complexity of insertion which, using the terminology of [19], is the left inverse of deletion, was investigated in [12].

It is well known that $L_1 \rightsquigarrow L_2$ is always regular for a regular language L_1 and an arbitrary language L_2 [21]. However, in spite of deletion being a fundamental language operation its precise state complexity has not been studied in the literature. When L_1 is recognized by a DFA with n states, the proof of Theorem 1 of [21] yields an upper bound 2^{2n} for the size of the DFA needed to recognize $L_1 \rightsquigarrow L_2$. The proof works for an arbitrary language L_2 and is not, in general, effective.

More general types of deletion operations, called *deletion along trajectories* have been considered by Domaratzki [4] and Kari and Sosik [22]. The operation is inspired by the notion of shuffle along trajectories originally considered by Mateescu et al. [28]. In the context of deletion along trajectories, $T \subseteq \{i, d\}^*$ is a set of trajectories where i stands for "insert" and d stands for "delete". Consider $t \in T$ and $u, v \in \Sigma^*$, $i, d \notin \Sigma$. Assuming the length of u equals the length of t, we denote by $u_{t,d}$ (respectively, $u_{t,i}$) the string obtained by concatenating the symbols of u that correspond to occurrences of d (respectively, occurrences of i) in the string t. Now assuming $v = u_{t,d}$, the result of deleting string v from u along the trajectory t is the string $u_{t,i}$, and it is denoted $u \rightsquigarrow_t v$. For example, $abcab \rightsquigarrow_{diidd} aab = bc$. If the length of u does not equal the length of t or $v \neq u_{t,d}$, the result $u \rightsquigarrow_t v$ is undefined. Note that, for given strings u, v and t, $u \rightsquigarrow_t v$ consists of a unique string or is undefined.

Deletion along trajectories is extended in the usual way for languages and sets of trajectories and the result of deleting a language L_2 from a language L_1 along a set of trajectories $T \subseteq \{i, d\}^*$ is denoted $L_1 \rightsquigarrow_T L_2$. Deletion along a regular set of trajectories preserves regularity [4], that is, for regular languages L_1, L_2 and T, also $L_1 \rightsquigarrow_T L_2$ is regular. The ordinary deletion operation is defined by the set of trajectories $i^*d^*i^*$, and the set of trajectories $d^*i^*d^*$ defines the *bipolar deletion* operation [4,19,21].

The construction used in the proof of Lemma 3.1 of [4] yields an upper bound 2^{3mn} for the state complexity of both $L_1 \rightsquigarrow_{i^*d^*i^*} L_2$ and $L_1 \rightsquigarrow_{d^*i^*d^*} L_2$ when L_1 (respectively, L_2) is recognized by a DFA of size m (respectively, n). Naturally, Lemma 3.1 of [4] deals with deletion along arbitrary regular sets of trajectories and the result cannot be expected to yield a good bound in the very special cases we are considering here.

As our main result we give a tight state complexity bound for the language obtained from a regular language by deleting an arbitrary language. We show that if L_1 is recognized by a complete DFA with n states and L_2 is an arbitrary language, the complete DFA for the language $L_1 \rightsquigarrow L_2$ needs $n \cdot 2^{n-1}$ states in the worst case. The corresponding state complexity function based on incomplete DFAs is shown to be $(n + \frac{1}{2}) \cdot 2^n - 2$. While the upper bounds hold for arbitrary languages L_2 (that need not be even recursively enumerable) we show that matching lower bound constructions can be found where L_2 consists of a single string of length one. We give conditions based on L_2 and the DFA for L_1 that are necessary for the state complexity of deletion to reach the worst case bound.

At first sight it might appear that the state complexity bound for the bipolar deletion of L_2 from L_1, $L_1 \rightsquigarrow_{d^*i^*d^*} L_2$, would need to depend on the sizes of DFAs for both L_1 and L_2. However, using the terminology of Domaratzki and Salomaa [6], $d^*i^*d^*$ is an i-regular set of trajectories and it follows that $L_1 \rightsquigarrow_{d^*i^*d^*} L_2$ is regular always when L_1 is regular. The proof of Theorem 6 of [6] gives an upper bound, albeit a very large upper bound, for the size of a DFA for $L_1 \rightsquigarrow_{d^*i^*d^*} L_2$ as a function of the size of a DFA for L_1.

Assume L_1 is recognized by a complete DFA A. We show that $L_1 \rightsquigarrow_{d^*i^*d^*} L_2$ can be recognized by a DFA B that, roughly speaking, simulates the computation of A starting from all possible states, each computation "remembers" the state it originated from, and the final states are assigned in a suitable way. This gives an upper bound n^n [respectively, $(n+1)^n - 1$] for the state complexity of bipolar deletion when using complete (respectively, incomplete) DFAs.

The above described operation of the DFA constructed to recognize $L_1 \rightsquigarrow_{d^*i^*d^*} L_2$ has close resemblance to the monoid of transformations of the DFA recognizing L_1. Holzer and König [16] have shown that the syntactic monoid of an n-state regular language has size n^n in the worst case. Based on an n-state DFA whose monoid of transformations contains all functions on the set of states given by Holzer and König [16], we establish an almost matching lower bound for the state complexity of bipolar deletion. Both in the case of complete and incomplete DFAs the lower bound differs from the upper bound only by an additive term $n - 1$ where n is the state complexity of L_1. Properties of the monoid of transformations of a DFA have been used in the study of state complexity of operations on regular languages also by Krawetz et al. [23] and Salomaa et al. [33].

2 Preliminaries

We assume that the reader is familiar with the basics of finite automata and formal languages and recall here just some definitions and notation. For a general introduction to the topic the reader may consult the monographs [34,35] or the survey [36]. More information on state complexity of operations can be found in the survey [11]. The operations of deletion and bipolar deletion are special cases of deletion along trajectories [4,22].

In the following Σ always stands for a finite alphabet and the set of strings over Σ is Σ^*. A language is a subset of Σ^* and the empty string is denoted by ε. The cardinality of a finite set S is denoted $|S|$ and the set of functions $S \to S$ is denoted S^S.

The set of strings obtained from $u \in \Sigma^*$ by deleting a string $v \in \Sigma^*$ is

$$u \rightsquigarrow v = \left\{ w \in \Sigma^* \mid \left(\exists u_1, u_2 \in \Sigma^*\right) \quad w = u_1 u_2 \quad \text{and} \quad u = u_1 v u_2 \right\}.$$

For example, $bababa \rightsquigarrow aba = \{bba, bab\}$. The *deletion operation* is extended in the natural way to languages $L_1, L_2 \subseteq \Sigma^*$ by setting

$$L_1 \rightsquigarrow L_2 = \bigcup_{u \in L_1, v \in L_2} u \rightsquigarrow v.$$

The *bipolar deletion* [4,19,21] of string $v \in \Sigma^*$ from string $u \in \Sigma^*$ is defined as

$$u \rightsquigarrow_{\text{bi}} v = \left\{ w \in \Sigma^* \mid \left(\exists v_1, v_2 \in \Sigma^*\right) \quad v = v_1 v_2 \quad \text{and} \quad u = v_1 w v_2 \right\}.$$

For example, $bababa \rightsquigarrow_{\text{bi}} ba = \{abab, baba\}$. Again the bipolar deletion operation is extended in the natural way to an operation on languages.

An *incomplete deterministic finite automaton* (incomplete DFA) is a five-tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, δ is a partial function $Q \times \Sigma \to Q$, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is a set of final (or accepting) states.

The transition function δ is extended as a partial function $Q \times \Sigma^* \to Q$ by setting, for $q \in Q$, $a \in \Sigma$ and $w \in \Sigma^*$, $\delta(q, \varepsilon) = q$ and $\delta(q, w \cdot a) = \delta(\delta(q, w), a)$. The language recognized by A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. A language is regular if it is recognized by some DFA.

For $q \in Q$, $P \subseteq Q$, $b \in \Sigma$ and $L \subseteq \Sigma^*$ we also denote

$$\delta(P, b) = \{\delta(p, b) \mid p \in P\} \quad \text{and} \quad \delta(q, L) = \{\delta(q, w) \mid w \in L\}.$$

A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is said to be *complete* if δ is a total function $Q \times \Sigma \to Q$. We will use both complete and incomplete DFAs and, when not explicitly mentioned, by a DFA we mean an incomplete DFA. Naturally, a complete DFA is just a special case of an incomplete DFA.

A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is *minimal* if each state $q \in Q$ is reachable from the initial state q_0 [that is, $\delta(q_0, w) = q$ for some string w] and no two states $q_1, q_2 \in Q$, $q_1 \neq q_2$, are equivalent. States $q_1, q_2 \in Q$ are said to be equivalent if

$$\left(\forall w \in \Sigma^*\right) \ \delta(q_1, w) \in F \quad \text{iff} \quad \delta(q_2, w) \in F.$$

States q_1 and q_2 are *distinguishable* using a string w if $\delta(q_1, w) \in F$ and $\delta(q_2, w) \notin F$, or vice versa.

The minimal (complete or incomplete) DFA for a given regular language L is unique and the sizes of the minimal complete and incomplete DFAs for L differ by at most one state. The minimal complete DFA may have a dead state (or sink state). In the minimal incomplete DFA the dead state can always be omitted.

The *state complexity* of L, $\text{sc}(L)$, is the size of the minimal complete DFA recognizing L. Similarly, the *incomplete state complexity* of L, $\text{isc}(L)$, is the size of the minimal incomplete DFA recognizing L. For each regular language L either $\text{sc}(L) = \text{isc}(L)+1$ or $\text{sc}(L) = \text{isc}(L)$.

The *syntactic congruence* of a regular language L over an alphabet Σ, $\sim_L \subseteq \Sigma^* \times \Sigma^*$, is defined by setting, for $w_1, w_2 \in \Sigma^*$,

$$w_1 \sim_L w_2 \quad \text{iff} \left(\forall u, v \in \Sigma^*\right) \ u w_1 v \in L \Leftrightarrow u w_2 v \in L.$$

The *syntactic monoid* of L is the quotient monoid $\Sigma^* / \sim_L$. If A is the minimal complete DFA for L, the syntactic monoid of L is isomorphic to the monoid of transformations of A, that is, the set of functions on the states of A induced by the transitions of A with function composition as the monoid operation.

3 Upper bound for deletion

It is known that the result of deleting an arbitrary language from a regular language is regular [21]. Hence in the lemmas establishing the upper bound for deletion (for complete or incomplete DFAs) we do not need to assume that the deleted language L_2 is regular. However, in the case of an arbitrary L_2 finding a DFA for the language $L_1 \rightsquigarrow L_2$ is not, in general, effective.

First we give an upper bound construction for complete DFAs.

Lemma 1 *Consider $L_1, L_2 \subseteq \Sigma^*$, where L_1 is recognized by a complete DFA with n states. Then*

$$\text{sc}(L_1 \rightsquigarrow L_2) \leq n \cdot 2^{n-1}.$$

Proof Let $A = (Q, \Sigma, \delta, q_0, F_A)$ be a complete DFA for L_1, where $|Q| = n$. To recognize the language $L_1 \rightsquigarrow L_2$ we define a DFA

$$B = (P, \Sigma, \gamma, p_0, F_B),$$

where $P = \{(r, R) \mid r \in Q, R \subseteq Q, \delta(r, L_2) \subseteq R\}$, $p_0 = (q_0, \delta(q_0, L_2))$ and

$$F_B = \{(r, R) \mid r \in Q, R \subseteq Q, \delta(r, L_2) \subseteq R \quad \text{and} \quad R \cap F_A \neq \emptyset\}.$$

It remains to define the transitions of γ. For $(r, R) \in P$ and $b \in \Sigma$ we set

$$\gamma((r, R), b) = (\delta(r, b), \quad \delta(R, b) \cup \delta(\delta(r, b), L_2)). \tag{1}$$

The transition relation always adds the elements of $\delta(\delta(r, b), L_2)$ to the second component and, consequently, the state $\gamma((r, R), b)$, as defined above, is an element of P.

The intuitive idea of the construction is as follows. In order to recognize the language $L_1 \rightsquigarrow L_2$, the DFA B must check that the input string w can be completed to a string of L_1 by inserting a string $u \in L_2$ in some position, that is, for some decomposition $w = w_1w_2$ we have $w_1uw_2 \in L_1$. Since we do not know at which position the string $u \in L_2$ is to be inserted and B has to be deterministic, roughly speaking, B has to keep track of all computations of A on strings where a string of L_2 was deleted from some earlier position.

The first component of the states of B simply simulates the computation of A, i.e., it keeps track of the state of A, assuming that up to the current position in the input a string of L_2 was not yet deleted. The second component of the states of B keeps track of all states that A could be in assuming that at some point in the preceding computation a string of L_2 was deleted from the input.

We need to verify that the transitions of B [as defined in (1)] preserve these properties. For the first component it is clear that the simulation works as claimed. To verify the claim for the second component, assume that the input is ubv, $u, v \in \Sigma^*$, $b \in \Sigma$ and after reading the prefix u the DFA B has reached a state (r, R). In the following discussion b refers to the particular symbol occurrence just after the prefix u. After reading the symbol b, the second component of the state of B will be $\delta(R, b) \cup \delta(\delta(r, b), L_2))$, where R consists of states that A could be in, assuming a string of L_2 was deleted somewhere before the symbol b and the states of $\delta(R, b)$ are then the states A could be in after reading b assuming a string of L_2 was deleted before symbol occurrence b. On the other hand, $r = \delta(q_0, u)$, i.e., r is the state A reaches after reading the prefix u, where no deletion has occurred, and hence $\delta(\delta(r, b), L_2)$ consists of exactly all states A can be in the simulated computation, assuming a string of L_2 was deleted directly after the symbol occurrence b. This means that the transition relation γ

correctly preserves the property that the second component of the state of B consists of all states that A could be in assuming a string of L_2 was deleted some time previously.

The choice of final states guarantees that B accepts exactly the strings obtained from strings of $L(A)$ by deleting a string of L_2 at some position.

We still need to verify that the number of states of B is as claimed. If $L_2 = \emptyset$, then $L_1 \rightsquigarrow L_2 = \emptyset$ and $L_1 \rightsquigarrow L_2$ has a DFA of size one. Hence in what follows we can assume that $L_2 \neq \emptyset$.

Since A is a complete DFA and $L_2 \neq \emptyset$, for each $r \in Q$ we have $|\delta(r, L_2)| \geq 1$. This means that for a given $r \in Q$, there exist at most $2^{|Q|-1}$ sets R such that (r, R) is a state of B. Thus, the number of states of B is at most $|Q| \cdot 2^{|Q|-1}$. □

Next we consider the case of incomplete DFAs. The upper bound construction uses similar ideas as the above proof of Lemma 1, and we just need to modify the construction to allow the possibility of undefined transitions.

Lemma 2 *Let $L_1, L_2 \subseteq \Sigma^*$, where L_1 is recognized by an incomplete DFA A with n states. Then*

$$\operatorname{isc}(L_1 \rightsquigarrow L_2) \leq (n+1) \cdot 2^n - \left(2^{n-1} + 2\right).$$

Proof Let $A = (Q, \Sigma, \delta, q_0, F_A)$ be an incomplete DFA for L_1, $|Q| = n$. We define the completion of δ as a function $\delta' : (Q \cup \{\text{dead}\}) \times \Sigma \to Q \cup \{\text{dead}\}$ by setting for $r \in Q \cup \{\text{dead}\}$ and $b \in \Sigma$,

$$\delta'(r, b) = \begin{cases} \delta(r, b), & \text{if } r \in Q \text{ and } \delta(r, b) \text{ is defined;} \\ \text{dead}, & \text{otherwise.} \end{cases}$$

To recognize the language $L_1 \rightsquigarrow L_2$ we define a DFA

$$B = (P, \Sigma, \gamma, p_0, F_B),$$

where $P = (Q \cup \{\text{dead}\}) \times 2^Q - \{(\text{dead}, \emptyset), (\text{dead}, Q)\}$, $p_0 = (q_0, \delta(q_0, L_2))$ and

$$F_B = \{(r, R) \mid r \in Q \cup \{\text{dead}\}, R \subseteq Q \text{ and } R \cap F_A \neq \emptyset\}.$$

(Note that $|P| = (n+1) \cdot 2^n - 2$. However, as will be seen below at least 2^{n-1} elements of P will be unreachable as states of B.)

The transitions of γ are defined by setting, for $(r, R) \in P$ and $b \in \Sigma$,

$$\gamma((r, R), b) = \begin{cases} \left(\delta'(r, b), \delta(R, b) \cup \delta\left(\delta(r, b), L_2\right)\right), & \text{if } r \in Q \text{ and } \left(\delta'(r, b) \neq \text{dead} \right. \\ & \left. \text{ or } \delta(R, b) \cup \delta\left(\delta(r, b), L_2\right) \neq \emptyset\right); \\ (\text{dead}, \delta(R, b)), & \text{if } r = \text{dead and } \delta(R, b) \neq \emptyset; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

As in the proof of the previous lemma, the idea is that the first component simulates the computation of the original DFA A, assuming a string of L_2 has so far not been deleted (and using the new state "dead" to indicate that the simulated computation of A has failed), and the second component keeps track of the set of all possible states that A can be in, assuming a string of L_2 was deleted somewhere previously. The latter set can now be empty because A is incomplete. We leave the details of verifying that B recognizes $L(A) \rightsquigarrow L_2$ to the reader.

Below we explain why the states $(\text{dead}, \emptyset)$ and (dead, Q) can be omitted from the state set of B, and that, furthermore at least 2^{n-1} elements of P must be unreachable as states of B.

The state (dead, $\emptyset$) would be a sink state of the DFA B and, according to the definition of γ, it is never entered. Also, we note that when the computation, for the first time, reaches a state where the first component is "dead" this has to occur on an alphabet symbol that has at least one undefined transition in A. This means that when the computation initially reaches a state with first component "dead", the cardinality of the second component is at most $|Q|-1$, and after that point the transitions of γ do not add new states to the second component because if the deletion of the string of L_2 did not occur previously, the computation has already failed. Thus, the state (dead, Q) is always unreachable.

To verify the unreachability of 2^{n-1} further states, without loss of generality, we can assume that for some $q_1 \in Q$ and $w_1 \in L_2$, $\delta(q_1, w_1)$ is defined. Note that in the opposite case, no string of L_2 can occur as a substring of a string of $L(A) = L_1$ and, hence, $L_1 \rightsquigarrow L_2 = \emptyset$. Now all transitions of B that enter a state with the first component q_1 add the element $\delta(q_1, w_1)$ to the second components. This means that all elements (q_1, R), $\delta(q_1, w_1) \notin R$, are unreachable. □

In the above proof we noted that 2^{n-1} states of the constructed DFA B are unreachable for each state q of A such that $\delta(q, w)$ is defined for some $w \in L_2$. Thus, the worst case state complexity blow-up can occur only when transitions spelling out a string in L_2 originate only from one state of A and, slightly more precisely, we get the following necessary condition for languages that can reach the worst case state complexity of the deletion operation.

Corollary 1 *Let $A = (Q, \Sigma, \delta, q_0, F)$ be an incomplete DFA with n states, $L_1 = L(A)$ and L_2 is an arbitrary language. Then a necessary condition for* isc$(L_1 \rightsquigarrow L_2)$ *to reach the upper bound $(n+1)\cdot 2^n - (2^{n-1}+2)$ given by* Lemma 2 *is that*

$$(\exists q \in Q) \quad [|\delta(q, L_2)| = 1 \text{ and } (\forall p \in Q, p \neq q)\ \delta(p, L_2) = \emptyset].$$

4 Tight lower bound for deletion

As our main result we show here that the bounds given in the previous section are optimal. We begin with the case of incomplete DFAs which is, perhaps, more interesting. The result of Corollary 1 can be used as a guideline for selecting worst-case languages.

4.1 Lower bound for incomplete DFAs

We show that the state complexity upper bound for incomplete DFAs from Lemma 2 can be reached by DFAs defined over a five-letter alphabet. Based on the observations made in Corollary 1, as the language of deleted strings, we use a singleton set $\{c\}$ where, furthermore, a c-transition is defined only for one state of the DFA recognizing L_1. The conditions of Corollary 1 do not force L_2 to be a singleton set, however, the conditions indicate that a construction may be simpler to achieve using a singleton set.

Lemma 3 *Let $\Sigma = \{a, b, c, d, e\}$. For every $n \geq 4$ there exists a regular language $L_1 \subseteq \Sigma^*$ recognized by an incomplete DFA with n states such that*

$$\text{isc}\,(L_1 \rightsquigarrow \{c\}) = (n+1)\cdot 2^n - \left(2^{n-1}+2\right).$$

Proof Choose $A = (Q, \Sigma, \delta, 1, \{1\})$, where $Q = \{0, 1, \ldots, n-1\}$ and δ is defined by setting

– $\delta(i, a) = i+1, 0 \leq i \leq n-2, \delta(n-1, a) = 0;$

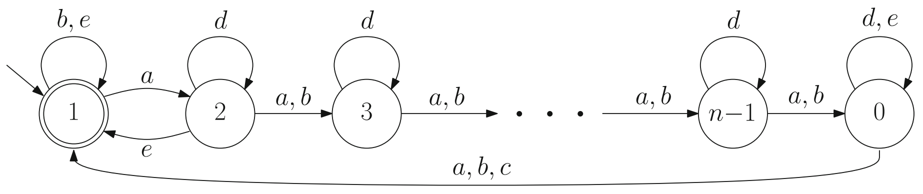

Fig. 1 The incomplete DFA A used in the proof of Lemma 3

- $\delta(0, b) = \delta(1, b) = 1, \delta(i, b) = i + 1, 2 \le i \le n - 2, \delta(n - 1, b) = 0$;
- $\delta(0, c) = 1$;
- $\delta(i, d) = i$ if $i = 0$ or $2 \le i \le n - 1$;
- $\delta(0, e) = 0, \delta(1, e) = \delta(2, e) = 1$.

All transitions not listed above are undefined. Note, in particular, that the only defined c-transition goes from state 0 to state 1. The DFA A is depicted in Fig. 1.

Let $B = (P, \Sigma, \gamma, p_0, F_B)$ be the DFA recognizing $L_1 \rightsquigarrow_T \{c\}$ that is constructed as in the proof of Lemma 2. Furthermore, let $B' = (P', \Sigma, \gamma, p_0, F_B)$ be the DFA obtained from B by omitting the unreachable states, as explained at the end of the proof of Lemma 2. With A as above and the deleted language being $L_2 = \{c\}$, we have

$$P' = \{(r, R) \mid r \in Q \cup \{\text{dead}\}, R \subseteq Q, (r, R) \neq (\text{dead}, \emptyset), (\text{dead}, Q) \\ \text{and } r = 0 \text{ implies } 1 \in R\}$$

In the following we show that all states of B' are reachable and pairwise inequivalent. The argument below uses often, without separate mention, the observation that since the deleted language consists only of the singleton set $\{c\}$ and in A the only c transition originates from state 0, the only transitions of B' that can add a "new" state to the second component are transitions that take the first component of the state to 0.

Claim 1 All states of P' are reachable from $(1, \emptyset)$.

We prove the claim using induction on the cardinality of R, $|R| = k$, where (r, R) is an arbitrary state of P'.

(i) For $k = 0$, we note that $\gamma((1, \emptyset), a^{i-1}) = (i, \emptyset)$, $1 \le i \le n - 1$. (Note that the pair $(0, \emptyset)$ is not in P'.)

(ii) Next consider $k = 1$. We note that $\gamma((1, \emptyset), a^{n-1}) = (0, \{1\})$ and $\gamma((0, \{1\}), a) = (1, \{2\})$. From state $(1, \{2\})$, the DFA B can reach any state $(i, \{j\})$, with $i \neq 0, i \neq j$, by first shifting the singleton set in the second component by b-transitions (that keep state 1 stationary) and then shifting both components together using a-transitions. Note that this process does not need to shift the first component into state 0, and hence none of the transitions adds a state to the second component. (Pairs $(0, \{j\})$, $j \neq 1$, are not in P'.)
To establish reachability of states of the form $(i, \{i\})$, $1 \le i \le n - 1$, we note that $\gamma((1, \{2\}), e) = (1, \{1\})$ and $\gamma((1, \{1\}), a^{i-1}) = (i, i)$, for $1 \le i \le n - 1$.
We still need to consider the case where the first component is "dead". We know that the state $(3, \{2\})$ is reachable.[1] Now $\gamma((3, \{2\}), e) = (\text{dead}, \{1\})$ and from the latter state we can reach all states $(\text{dead}, \{i\})$ simply using the cyclic a-transitions.

(iii) Inductively, assume now that the claim holds for the value $1 \le k < n$, that is, all states $(r, R) \in P'$, where $|R| \le k$ are reachable. Consider a state $(s, S) \in P'$, where

[1] Here we need that $n \ge 4$.

$|S| = k + 1$. Denote $S = \{i_1, i_2, \ldots, i_{k+1}\}$, $0 \le i_1 < i_2 < \cdots < i_{k+1}$. We consider three cases depending on what is s.

(a) $s = 0$: Since $(0, S) \in P'$, we know that $1 \in S$. Define $S' = \{r - 1 \mid r \in S - \{1\}\}$. Now $|S'| = k$ and by the inductive assumption the state $(n - 1, S')$ is reachable. The application of an a-transition to $(n - 1, S')$ adds the element 1 to the second component and, hence, $\gamma((n - 1, S'), a) = (s, S)$.

(b) $1 \le s \le n-1$: Choose $1 \le j \le k+1$ such that $i_j \le s < i_{j+1}$, where addition in the subindices is done modulo $k + 1$. We want that i_{j+1} is the next element "cyclically" $>s$, and, if $s < i_1$ or $s \ge i_{k+1}$, we choose $j = k + 1$ because in both cases the element cyclically following s is i_1.
By the inductive assumption the state

$$\mathbf{u} = \left(n - 1, \left\{i_{j+2} - i_{j+1}, \ldots, i_{k+1} - i_{j+1}, i_1 - i_{j+1}, \ldots, i_j - i_{j+1}\right\}\right)$$

is reachable. In the definition of the second components of the state $\mathbf{u}$ (and in the below argument) subtraction and addition are done modulo n. The idea is that the elements in the second component of $\mathbf{u}$ are obtained from S by omitting the component i_{j+1} and the remaining components are shifted downwards i_{j+1} steps.
Now $\gamma(\mathbf{u}, a^2) = \mathbf{v_1}$, where

$$\mathbf{v_1} = \left(1, \{2, i_{j+2} - \left(i_{j+1} - 2\right), \ldots, i_{k+1} - \left(i_{j+1} - 2\right), i_1 - \left(i_{j+1} - 2\right), \ldots, i_j - \left(i_{j+1} - 2\right)\}\right)$$

Note that the application of the first a-transition to state $\mathbf{u}$ adds an element 1 to the second component, and the second a-transition just cyclically shifts all elements. When the DFA B' in state $\mathbf{v_1}$ reads the string $b^{i_{j+1}-2-(s-1)}$ the first component remains as 1 and the second component becomes

$$V_1 = \left\{i_{j+1} - (s - 1), i_{j+2} - (s - 1), \ldots, i_{k+1} - (s - 1), i_1 - (s - 1), \ldots, i_j - (s - 1)\right\}.$$

Note that since $i_j \le s < i_{j+1}$, in the above computation from the state $\mathbf{v_1}$ on string $b^{i_{j+1}-2-(s-1)}$ none of the elements of the second component tries to exit the state 1, i.e., the sequence of b's just shifts the elements of the second component of $\mathbf{v_1}$ to become V_1.
From the resulting state $(1, V_1)$ we get the state (s, S) by reading a^{s-1}. The a-transitions just cyclically shift all elements and, as $1 \le s \le n-1$, the first component does not have a transition entering the state 0 which could be the only transition that would add an element to the second component.

(c) The remaining case is where $s =$ dead. Note that now we can assume that $k \le n-2$, because the state (dead, Q) is not in P' (as we know from the construction in the proof of Lemma 2 that (dead, Q) is not reachable) and, hence, if $k = n - 1$ there is nothing to prove.
We want to show that the state (dead, $\{i_1, i_2, \ldots, i_{k+1}\}$) is reachable. Since $k + 1 < n$, there exist two elements that are cyclically not consecutive, that is, there exists $1 \le j \le k + 1$ such that $i_{j+1} - i_j \ge 2$. When $j = k + 1$ this is interpreted to mean that $i_1 - i_{k+1} \pmod n$ is at least two.
By (iii)(b) we know that the state

$$\mathbf{w} = \left(1, \left\{i_1 - i_j, i_2 - i_j, \ldots, i_{k+1} - i_j\right\}\right)$$

is reachable. Note that when proving (iii)(b) we did not need the reachability of states where the first component is "dead" and, hence, we can assume that the inductive proof of (iii)(b) is completed first.
Recall that the transitions of A on symbol d are identity, except the d-transition on state 1 is undefined. By the choice of j we know that the second component of $\mathbf{w}$ does not contain the element 1 and hence

$$\gamma(\mathbf{w}, d) = \big(\text{dead}, \big\{i_1 - i_j, i_2 - i_j, \ldots, i_{k+1} - i_j\big\}\big).$$

From the above state we get the state (dead, $\{i_1, i_2, \ldots, i_{k+1}\}$) simply by reading i_j symbols a.

This concludes the proof of *Claim 1*.

Claim 2 All states of B' are pairwise inequivalent.
For the proof of Claim 2 we consider three cases.

(i) Consider two states (r, R_1), $(r, R_2) \in P'$, where $R_1 \neq R_2$, $r \in Q \cup \{\text{dead}\}$, $R_1, R_2 \subseteq Q$. Without loss of generality we can choose $t \in R_1 - R_2$, the other case being completely symmetric.

(a) First we assume that $r \not\equiv t - 1 \pmod n$. This includes also the case where $r = \text{dead}$. Let $k \in \{0, 1, \ldots, n-1\}$ be the smallest nonnegative integer congruent to $n - t + 1$ modulo n. Now the state $\gamma((r, R_1), a^k)$ is final because $t \in R_1$ and a^k shifts t to the final state 1 (of A).
We verify that the state $\gamma((r, R_2), a^k)$ is nonfinal. Since $t \notin R_2$, the computation on a^k does not shift elements of the second component R_2 to a set which contains 1 which is the only final state of A. Also since $r \not\equiv t-1 \pmod n$, the computation does not shift the first component to 0 which would result in an accepting state of B' (by adding 1 to the second component). If at an intermediate stage of the computation the first component is shifted to 0 and 1 is added to the second component, this element 1 cannot cycle back to itself.

(b) The second possibility is that $r \equiv t - 1 \pmod n$. Now let $k \in \{0, 1, \ldots, n-1\}$ be the smallest nonnegative integer congruent to $n - t$ modulo n. We verify that $\gamma((r, R_1), a^k ea) = (\text{dead}, R_1')$, where $1 \in R_1'$ and $\gamma((r, R_2), a^k ea) = (\text{dead}, R_2')$, where $1 \notin R_2'$. We note that above the string a^k shifts the state r (which cyclically precedes t) to state $n-1$ and then application of an undefined e-transition changes the first component to dead. The computation on a^k shifts $t \in R_1$ to 0 and then reading the string ea this becomes 1. On the other, the second component R_2' of the state in the computation originating from (r, R_2) does not have element 1 because $t \notin R_2$.

(ii) Consider two states (r, R_1) and (dead, R_2), where $r \in Q$ and $R_1, R_2 \subseteq Q$. Let $k \in \{0, 1, \ldots, n-1\}$ be the smallest nonnegative integer congruent to $n - r$ modulo n. Then $\gamma((r, R_1), a^k c) = (1, R_1')$ and $\gamma((\text{dead}, R_2), a^k c) = (\text{dead}, R_2')$, where

$$R_i' = \begin{cases} \emptyset & \text{if } r \notin R_i, \\ \{1\} & \text{if } r \in R_i, \end{cases} \quad i = 1, 2.$$

To verify the above we note that the transitions on a^k cycle the first component of (r, R_1) to state 0 and then the following c shifts the first component to state 1. The transition into 0 in the first component adds 1 to the second component but this will be deleted because c-transition on 1 is undefined. Since the only defined c-transition is from 0 to

1, the second component R'_i, $1 \le i \le 2$, will be empty unless $0 \in \delta(R_i, a^k)$, that is, unless $r \in R_i$.

Now in the case $r \notin R_2$, the state (dead, R'_2) is the (undefined) sink state of B' and clearly it is not equivalent with any other state. Thus, in order to complete the argument it is sufficient to separate, respectively, the states $(1, \{1\})$ and (dead, $\{1\}$) and the states $(1, \emptyset)$ and (dead, $\{1\}$). The second pair are not equivalent because $(1, \emptyset)$ is nonfinal and (dead, $\{1\}$) is final. To separate the first pair, we note that

$$\gamma\left((1, \{1\}), a^n d\right) = \gamma\left((1, \{1, 2\}), d\right) = (\text{dead}, \{2\})$$

and $\gamma(\text{dead}, \{1\}), a^n) = (\text{dead}, \{1\})$ and $\gamma((\text{dead}, \{1\}), d)$ is undefined.

(iii) The last case is to consider a pair of states (r_1, R_1) and (r_2, R_2), $0 \le r_1 < r_2 \le n-1$, $R_1, R_2 \subseteq Q$. Let $k \in \{0, 1, \ldots, n-1\}$ be the smallest nonnegative integer congruent to $n - r_2$ modulo n. Now $\gamma((r_2, R_2), a^k c) = (1, R'_2)$ and $\gamma((r_1, R_1), a^k c) = (\text{dead}, R'_1)$. [As in (ii) above it is seen that R'_i, $i = 1, 2$, is always $\emptyset$ or $\{1\}$ but this is not needed.] The states $(1, R'_2)$ and (dead, R'_1) are inequivalent by the case (ii) above.

Any two distinct states of B' fit one of the cases (i)–(iii) above. This concludes the proof of Claim 2 and the proof of the lemma. □

As a result of Lemma 3 we conclude that the upper bound for the size of an incomplete DFA for the language $L_1 \rightsquigarrow L_2$ given in Lemma 2 is tight.

Theorem 1 *For languages $L_1, L_2 \subseteq \Sigma^*$, where L_1 is regular,*

$$\text{isc}\,(L_1 \rightsquigarrow L_2) \le (\text{isc}(L_1) + 1) \cdot 2^{\text{isc}(L_1)} - \left(2^{\text{isc}(L_1)-1} + 2\right).$$

For every $n \ge 4$ there exists a language L_1 over a five-letter alphabet recognized by an incomplete DFA with n states and a singleton language L_2 such that in the above inequality we have an equality.

4.2 Lower bound for complete DFAs

From the construction of the complete DFA B for $L_1 \rightsquigarrow L_2$ in Lemma 1 we know that the possible states of B are pairs (r, R), where r (respectively, R) is a state (respectively, a set of states) of the DFA A recognizing L_1 and R has the property that it contains all states that are reachable from r on a string of L_2 (and possibly other states). Hence a possible worst case construction should use a singleton language L_2 or a language L_2 such that for any given state r of A, all strings of L_2 take r to the same state. In the proof of Lemma 4 we choose L_2 to be a singleton language consisting of a string of length one.

Lemma 4 *Let $\Sigma = \{a, b, c, d, e\}$. For every $n \ge 3$ there exists a complete DFA A over Σ with n states such that*

$$\text{sc}\,(L(A) \rightsquigarrow \{c\}) = n \cdot 2^{n-1}.$$

Proof Choose $A = (Q, \Sigma, \delta, 0, \{0\})$, where $Q = \{0, 1, \ldots, n-1\}$ and the transitions of δ are defined by setting

- $\delta(i, a) = i + 1$ for $0 \le i \le n-2$, $\delta(n-1, a) = 0$;
- $\delta(0, b) = 0$, $\delta(i, b) = i + 1$ for $1 \le i \le n-2$, $\delta(n-1, b) = 0$;
- $\delta(0, c) = 1$, $\delta(i, c) = i$ for $1 \le i \le n-1$;
- $\delta(0, d) = 0$, $\delta(1, d) = 1$, $\delta(i, d) = i + 1$ for $2 \le i \le n-2$, $\delta(n-1, d) = 0$;
- $\delta(0, e) = \delta(1, e) = 1$, $\delta(i, e) = i + 1$ for $2 \le i \le n-2$, $\delta(n-1, e) = 0$.

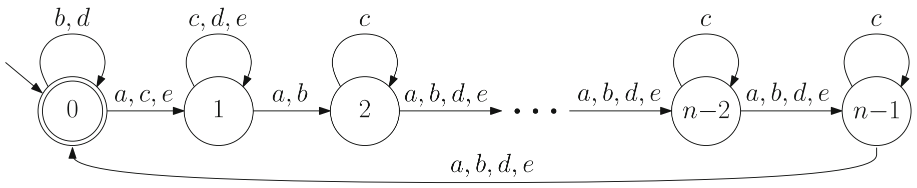

Fig. 2 The complete DFA A used in the proof of Lemma 4

The DFA A is depicted in Fig. 2.

Let $B = (P, \Sigma, \gamma, p_0, F_B)$ be the complete DFA recognizing the language $L(A) \rightsquigarrow \{c\}$ that is constructed as in the proof of Lemma 1. Since the deleted language $\{c\}$ consists of only one string, B has $n \cdot 2^{n-1}$ states and in order to prove the lemma it is sufficient to show that all states of B are reachable from the initial state $p_0 = (0, \{1\})$ and all states of B are pairwise inequivalent.

Claim 1 All states $(0, R) \in P$ where $R \supseteq \delta(0, \{c\}) = \{1\}$ are reachable from $(0, \{1\})$.

We prove the claim by induction on the cardinality of R. In the base case $|R| = 1$ there is nothing to prove because $R = \{1\}$ is the only set satisfying the required condition on R.

Inductively, now assume that the claim holds for all sets R of cardinality $1 \le k < n$, that is, all states $(0, R)$ where $1 \in R$ and $|R| \le k$ are reachable. Now consider a state of P,

$$\mathbf{u} = (0, \{1, i_1, \dots i_k\}), \quad 1 < i_1 < i_2 < \cdots < i_{k-1} \quad \text{and} \quad (i_{k-1} < i_k \text{ or } i_k = 0).$$

Above the elements $i_1, \dots, i_k$ are listed in increasing order, except that 0 is considered the largest element. By the inductive assumption the state

$$\mathbf{u}' = (0, \{1, i_2 - i_1 + 1, i_3 - i_1 + 1, \dots, i_k - i_1 + 1\})$$

is reachable. Here the arithmetic operations are done modulo n. If $i_k = 0$, above $i_k - i_1 + 1$ stands for $n - i_1 + 1$. Now

$$\gamma\left(\mathbf{u}', b\right) = (0, \{1, 2, i_2 - i_1 + 2, i_3 - i_1 + 2, \dots, i_k - i_1 + 2\}),$$

because the transition where the first component enters 0 adds 1 to the second component and otherwise the transition on b cycles "upwards" the states of $\{1, i_2 - i_1 + 1, i_3 - i_1 + 1, \dots, i_k - i_1 + 1\}$. (Note that this set does not contain the element 0.) Next applying to the state $\gamma(\mathbf{u}', b)$ $i_1 - 2$ times a transition on d we reach $\mathbf{u}$. Note that transitions on d "cycle upwards" the states in $\{2, 3, \dots, n-1\}$, and keep the states 0 and 1 stationary, and do not add (in the transition relation of B), new elements to the second component of the state. This concludes the proof of Claim 1.

Next we show that all states (r, R), $R \supseteq \delta(r, \{c\})$, are reachable. We need to consider only cases where $r \neq 0$ (because the case $r = 0$ was handled in Claim 1 above) and, hence, $\delta(r, \{c\}) = \{r\}$. Consider an arbitrary state

$$\mathbf{v} = \left(i_j, \left\{i_1, i_2, \dots, i_{j-1}, i_j, i_{j+1}, \dots, i_k\right\}\right), \quad 0 \le i_1 < i_2 < \cdots < i_k \le n-1.$$

For technical reasons we need to use a slightly different argument depending on whether the difference between i_{j+1} and i_j, as well as, between i_j and i_{j-1} is exactly one. We divide the following argument into three cases.

(i) Case where $i_j \neq i_{j-1} + 1$: This is the case where the set in the second component of $\mathbf{v}$ does not contain the element preceding i_j. By Claim 1 the state

$$\mathbf{v}' = (0, \{1, i_1 - i_j + 1, i_2 - i_j + 1, \dots, i_{j-1} - i_j + 1, i_{j+1} - i_j + 1, \dots, i_k - i_j + 1\})$$

is reachable. In the preceding line all quantities are computed modulo n. Now

$$\gamma(\mathbf{v}', c) = \left(1, \left\{1, i_1 - i_j + 1, i_2 - i_j + 1, \ldots, i_{j-1} - i_j + 1, i_{j+1} - i_j + 1, \ldots, i_k - i_j + 1\right\}\right).$$

Note that, because $i_j \neq i_{j-1}+1$, the sequence $i_x - i_j + 1, x = 1, \ldots, j-1, j+1, \ldots, k$, does not contain the element 0, and hence a transition on c is the identity on these elements. In the DFA A the a-transitions just cycle through the states and hence applying $i_j - 1$ times the a-transition to state $\gamma(\mathbf{v}', c)$ we get the state $\mathbf{v}$.

(ii) Case where $i_j = i_{j-1} + 1$ and $i_{j+1} = i_j + 1$: This is the case where the set in the second component of $\mathbf{v}$ contains both the element preceding i_j and the element following i_j. By Claim 1 the state

$$\mathbf{v_1} = \left(0, \left\{i_1 - i_j, i_2 - i_j, \ldots, i_k - i_j\right\}\right)$$

is reachable. Note that the second component contains the element 1 and hence $\mathbf{v_1}$ is a legal state of B. Now $\gamma(\mathbf{v_1}, a^{i_j}) = \mathbf{v}$.

(iii) Case where $i_j = i_{j-1} + 1$ and $i_{j+1} \neq i_j + 1$: This corresponds to the situation where the second component of $\mathbf{v}$ contains the element preceding i_j but does not contain the element following i_j. Here we cannot use a-transitions alone, because a state where the first component is 0 must contain 1 in the the second component. By Claim 1 the state

$$\mathbf{v_2} = \left(0, \left\{1, i_1 - i_j, i_2 - i_j, \ldots, i_{j-1} - i_j, i_{j+1} - i_j, \ldots, i_k - i_j\right\}\right)$$

is reachable and

$$\gamma\left(\mathbf{v_2}, e\right) = \left(1, \left\{1, i_1 - i_j + 1, i_2 - i_j + 1, \ldots, i_k - i_j + 1\right\}\right).$$

Here we need the fact that $i_{j+1} \neq i_j + 1$ and hence an e-transition adds one to the state $i_{j+1} - i_j$. (Note also that $i_{j-1} - i_j = n - 1$ and, consequently, beginning with a c-transition as in case (i) above would not work, because the second component at the end would not contain i_{j-1}.)
Applying $i_j - 1$ a-transitions to the state $\gamma(\mathbf{v_2}, e)$ we get $\mathbf{v}$.

We have shown that all states of B are reachable and it remains to show that they are all pairwise inequivalent.

First consider two states $(r_1, R_1), (r_2, R_2) \in P$, where $R_1 \neq R_2$. Without loss of generality we can find $s \in R_1 - R_2$ since the other possibility is completely symmetric. If $s = 0$ then (r_1, R_1) is a final state and (r_2, R_2) is a nonfinal state of B. Thus it is sufficient to consider cases $s \in \{1, 2, \ldots, n-1\}$. Now $\gamma((r_1, R_1), a^{n-s}) \in F_B$ because the string a^{n-s} takes the state $s \in R_1$ to the element 0. We show that $\gamma((r_2, R_2), a^{n-s}) \notin F_B$. Since $s \notin R_2$, the string a^{n-s} does not take any element of R_2 to the element 0 (which is the only final state of A). We note that $r_2 \neq s$ because from the definition of legal states of B we know that $\delta(r_2, c)$ must be an element of R_2 [and $\delta(s, c) = s$ when $1 \leq s \leq n-1$]. Also, if during the computation on a^{n-s} starting from (r_2, R_2), the transitions of γ add the element 1 to the second component when the first component becomes 0, then the added element 1 cannot cycle through all states to reach the final state 0 because after adding the element 1 there remains at most $n - s - 1$ input symbols and $s \geq 1$.

Second, consider two states $(r_1, R_1), (r_2, R_2) \in P$, where $r_1 \neq r_2$. Due to symmetry between r_1 and r_2 we can assume that $r_2 \neq 0$.

(i) Case where $r_1 = 0$ and $r_2 \neq n - 1$: Since $r_2 \neq r_1$, we have $r_2 \neq 0$ and we note that $\gamma((0, R_1), b) = (0, R_1')$ where $1 \in R_1'$ (because the self-loop on state 0 adds the element

1 to the second component) and $\gamma((r_2, R_2), b) = (r_2+1, R'_2)$. Here $1 \notin R'_2$ because no transition of A labeled by b reaches the state 1 and also since $r_2 + 1 \neq 0$ the transition cannot add the element 1 to the second component. Since the second components are distinct sets we know that the states $(0, R'_1)$ and $(r_2 + 1, R'_2)$ are distinguishable.

(ii) Case where $r_1 = 0$ and $r_2 = n - 1$: We note that $\gamma((0, R_1), a) = (1, R'_1)$ and $\gamma((n - 1, R_2), a) = (0, R'_2)$. The states $(1, R'_1)$ and $(0, R'_2)$ are inequivalent by case (i) above.

(iii) Case where $r_1 \neq 0$ and $r_2 \neq 0$: By cycling with $n - r_1$ a-transitions we get states $(0, R'_1)$ and $(n - r_1 + r_2, R'_2)$. If $r_1 - r_2 \neq 1$, this case was covered in (i) above and, if $r_1 - r_2 = 1$, this case was covered in (ii) above.

Above (i)–(iii) cover all cases where $r_1 \neq r_2$ and $r_2 \neq 0$. This concludes the proof of the lemma. □

Now by combining Lemmas 1 and 4 we get a tight state complexity bound for deletion also in the case where L_1 and the resulting language are given by a complete DFA.

Theorem 2 *For languages $L_1, L_2 \subseteq \Sigma^*$, where L_1 is regular,*

$$\mathrm{sc}\,(L_1 \rightsquigarrow L_2) \leq \mathrm{sc}(L_1) \cdot 2^{\mathrm{sc}(L_1)-1}.$$

For every $n \geq 3$ there exists a regular language L_1 over a five-letter alphabet with $\mathrm{sc}(L_1) = n$ and a singleton language L_2 such that in the above inequality we have equality.

5 State complexity of bipolar deletion

It is known that bipolar deletion preserves regularity [4,22]. Furthermore, since the set of trajectories $d^*i^*d^*$ defining bipolar deletion is i-regular (as defined in [6]), the bipolar deletion of an arbitrary language from a regular language is regular.

If L_1 has a DFA with n states and L_2 has a DFA with m states, from [4] we know that the language $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$ is recognized by a DFA with $2^{3\cdot m\cdot n}$ states. On the other hand, the proof of Theorem 6 of [6] represents $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$ as a union of n^4 regular languages, each of which is represented by a concatenation of three languages each having a DFA of size n. The known state complexity bounds for concatenation and union [36,37] then imply an upper bound for the size of a DFA needed to recognize $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$ that depends only on n. However, the latter bound is very large.

Here using a direct construction we give an improved upper bound for the state complexity of bipolar deletion that depends only on the size of the DFA for the first language. After that we show that the bound is almost tight.

Lemma 5 *Suppose L_1 has a complete DFA with n states and L_2 is an arbitrary language. Then $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$ is recognized by a complete DFA with n^n states.*

Proof Let $L_1 = L(A)$, where $A = (Q, \Sigma, \delta, q_1, F)$ and $Q = \{q_1, q_2, \ldots, q_n\}$. For $q_i \in Q$, $1 \leq i \leq n$, denote by A_{q_i} the DFA obtained from A by changing the initial state to be q_i.

For $q_i \in Q$, $1 \leq i \leq n$, we define

$$F_{q_i,L_2} = \left\{p \in Q \mid \left(\exists u_1, u_2 \in \Sigma^*\right) \delta(q_1, u_1) = q_i, u_2 \in L(A_p), u_1u_2 \in L_2\right\}.$$

The set F_{q_i,L_2} consists of states p such that for some string $u_1 \cdot u_2$ of L_2 the prefix u_1 takes the initial state q_1 of A to q_i and when the computation of A is continued on the suffix u_2 in state p, it ends in a final state. Thus, a computation of A originating in q_i and ending in state p must spell out a string of $L(A) \rightsquigarrow_{\mathrm{bi}} L_2$.

Thus, we can construct a DFA B for $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$ as follows. On input w, B begins the computation in all states of A and a computation begun in state q_i is successful if it ends in a state of F_{q_i,L_2}.

Define $B = (Q^n, \Sigma, \gamma, (q_1, \dots, q_n), F_B)$, where the transitions of γ for $p_i \in Q$, $i = 1, \dots, n$, and $b \in \Sigma$ are defined by setting

$$\gamma\left((p_1, \dots, p_n), b\right) = \left(\delta(p_1, b), \dots, \delta(p_n, b)\right),$$

and the set of final states is defined as

$$F_B = \left\{(p_1, \dots, p_n) \in Q^n \mid (\exists 1 \le i \le n)\, p_i \in F_{q_i,L_2}\right\}.$$

The computation of B begins in the initial state $(q_1, \dots, q_n)$ and simulates the computation of A on each component. The states are ordered n-tuples so the ith component "remembers" that the corresponding computation began in state q_i. Thus, on input string w, the ith component ending in a state of F_{q_i,L_2} implies that there exist strings u_1 and u_2 such that $u_1u_2 \in L_2$ and $u_1wu_2 \in L_1$, and thus $L(B) \subseteq L_1 \rightsquigarrow_{\mathrm{bi}} L_2$.

Conversely, consider a string w such that, for some strings u_1 and u_2 with $u_1u_2 \in L_2$ we have $u_1wu_2 \in L_1$. Suppose $\delta(q_1, u_1) = q_j$, $1 \le j \le n$. Now

$$\delta\left(q_j, w\right) \in F_{q_j,L_2},$$

because $\delta(\delta(q_j, w), u_2) \in F$. This implies that B accepts w and we have shown that $L_1 \rightsquigarrow_{\mathrm{bi}} L_2 \subseteq L(B)$. □

The bound in Lemma 5 is stated in terms of complete DFAs. The analogous construction implies an upper bound of $(n+1)^n - 1$ for an incomplete DFA B recognizing $L(A) \rightsquigarrow_{\mathrm{bi}} L_2$ when A is an incomplete DFA with n states. If Q is the set of states of A, as the states of B we can use elements of $(Q \cup \{\text{dead}\})^n$, where the added element "dead" indicates that the simulated computation of A is undefined. The tuple (dead, ..., dead) corresponds to the dead state of B and can be omitted.

Corollary 2 *Suppose L_1 has an incomplete DFA with n states and let L_2 be an arbitrary language. Then*

$$\mathrm{isc}\,(L_1 \rightsquigarrow_{\mathrm{bi}} L_2) \le (n+1)^n - 1.$$

The construction used in the proof of Lemma 5 indicates that the state complexity of $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$ is maximized when the transition monoid of the DFA A recognizing L_1 consists of all mappings of Q^Q. Hence a worst case construction for the state complexity of bipolar deletion can be based on the DFAs shown by Holzer and König [16] to have maximal syntactic monoid size.

Lemma 6 *Let $\Sigma = \{a, b, c\}$. For every $n \ge 2$ there exists a complete DFA A with n states over alphabet Σ and a singleton language L_2 such that*

$$\mathrm{sc}\,(L(A) \rightsquigarrow_{\mathrm{bi}} L_2) = n^n - n + 1.$$

Proof Choose $A = (Q, \Sigma, \delta, 1, \{1\})$, where $Q = \{1, \dots, n\}$ and δ is defined by setting

- $\delta(i, a) = i + 1$, $1 \le i \le n - 1$, $\delta(n, a) = 1$;
- $\delta(1, b) = 2$, $\delta(2, b) = 1$, $\delta(i, b) = i$, $3 \le i \le n$;
- $\delta(i, c) = i$, $1 \le i \le n - 1$, $\delta(n, c) = 1$.

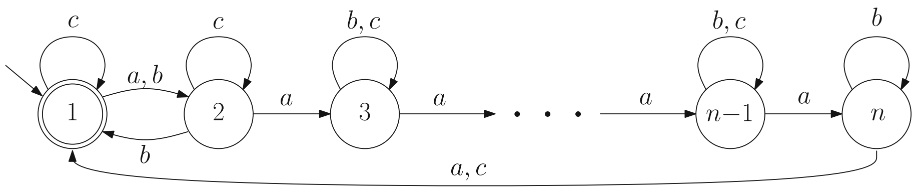

Fig. 3 The complete DFA A with n states and transition monoid of size n^n [16] used in the proof of Lemma 6

The DFA A is depicted in Fig. 3.

It is known from Theorem 3 of [16] that the transition monoid of A (and syntactic monoid of $L(A)$) has n^n elements.[2] Our DFA A has a different final state than the DFA used in [16] but this does not change the size of the transition monoid.

We choose $L_2 = \{a^n\}$. Let $B = (Q^n, \Sigma, \gamma, (1, \ldots, n), F_B)$ be the DFA recognizing the language $L(A) \rightsquigarrow_{\text{bi}} L_2$ constructed as in the proof of Lemma 5. Using the notations of the proof we note that

$$F_{i,L_2} = \{i\}, \quad 1 \le i \le n.$$

This follows by observing that, for $i \neq 1$, the only prefix of a^n that takes the initial state q_1 to q_i is a^{i-1} and then the only string that completes a^{i-1} to a string of L_2 is a^{n-i+1}. In the case $i = 1$, the prefix can be either ε or a^n. Thus,

$$F_B = \left\{(i_1, \ldots, i_n) \mid (\exists 1 \le j \le n)\, i_j = j\right\}.$$

Since the transition monoid of A consists of all functions of Q^Q, all states of Q^n are reachable from the initial state $(1, \ldots, n)$.

Denote $P = \{(i, i, \ldots, i) \mid 1 \le i \le n\}$. Since the transitions of γ apply the transition relation of A componentwise, it is clear that all states reachable from a state of P are in P and, consequently, all states of P are pairwise equivalent.

To complete the proof it is sufficient to show that any two states of Q^n at least one of which is outside P are distinguishable. Consider two distinct states $(i_1, \ldots, i_n)$ and $(j_1, \ldots, j_n)$, where $i_z \neq j_z$, $z \in \{1, \ldots, n\}$, and $(j_1, \ldots, j_n) \notin P$. Since the elements $j_1, \ldots, j_n$ do not all coincide, we can choose $t \in \{1, \ldots, z-1, z+1, \ldots, n\}$ such that $j_z \neq j_t$.

We want to define a function $f \in Q^Q$ such that $f(i_z) = z$, $f(j_z) = t$ and, additionally,

$$(\forall 1 \le x \le n)\, x \neq z \text{ implies } f(j_x) \neq x. \tag{2}$$

We verify that the condition (2) can be satisfied by a function f such that $f(i_z) = z$ and $f(j_z) = t$. If i_z appears in the sequence $j_1, \ldots, j_n$, it must have a subindex different from z, and hence the function value $f(i_z) = z$ is compatible with the condition (2). Similarly, if j_z appears in the sequence $j_1, \ldots, j_{z-1}, j_{z+1}, \ldots, j_n$ it has an index different from t (since $j_z \neq j_t$) and hence also the function value $f(j_z) = t$ is compatible with the condition (2), and the function f with the required values can be defined.

Since the transition monoid of A consists of all functions of Q^Q, there exists a string w such that the transitions of A on w define the function f. Since $f(i_z) = z$, $\gamma((i_1, \ldots, i_n), w) \in F_B$. On the other hand, for all $1 \le x \le n$, $f(j_x) \neq x$, which means that $\gamma((j_1, \ldots, j_n), w) \notin F_B$.
□

[2] Theorem 3 of [16] uses a different DFA in the case $n = 2$ (in order to have a binary alphabet). However, the transition monoid of A is complete also in the case $n = 2$.

By Lemmas 5 and 6 we have the following.

Theorem 3 *For a regular language L_1 and an arbitrary language L_2 we have*

$$\mathrm{sc}\,(L_1 \rightsquigarrow_{\mathrm{bi}} L_2) \le (\mathrm{sc}(L_1))^{\mathrm{sc}(L_1)}\,.$$

For every $n \ge 2$ there exists a regular language L_1 defined over a three letter alphabet with $\mathrm{sc}(L_1) = n$ and a singleton language L_2 such that

$$\mathrm{sc}\,(L_1 \rightsquigarrow_{\mathrm{bi}} L_2) \ge (\mathrm{sc}(L_1))^{\mathrm{sc}(L_1)} - \mathrm{sc}(L_1) + 1.$$

The upper and lower bound of Theorem 3 are very close but differ by an additive term of $\mathrm{sc}(L_1) - 1$. The proof of Lemma 6 uses $L_2 = \{a^n\}$ which yields a transparent definition of the set of final states for the DFA B constructed for $L_1 \rightsquigarrow_{\mathrm{bi}} L_2$. This choice causes also that B has n equivalent states. It is possible that e.g. choosing L_2 to be a non-singleton language could yield a better bound but such choices would likely result in more complicated arguments for correctness. We leave the precise state complexity of bipolar deletion as an open problem.

To give a lower bound for bipolar deletion based on incomplete DFAs, we use the same DFA as in the proof of Lemma 6 except we add transitions on d that guarantee that the transition semigroup of the DFA consists of all partial functions on $\{1, \ldots, n\}$.

Lemma 7 *Let $\Sigma = \{a, b, c, d\}$. For every $n \ge 2$ there exists an incomplete DFA A' with n states over the alphabet Σ and a singleton language L_2 such that*

$$\mathrm{isc}\left(L\left(A'\right) \rightsquigarrow_{\mathrm{bi}} L_2\right) = (n+1)^n - n.$$

Proof Choose $A' = (Q, \Sigma, \delta, 1, \{1\})$, where $Q = \{1, \ldots, n\}$ and δ is defined by setting

- $\delta(i, a) = i + 1$, $1 \le i \le n - 1$, $\delta(n, a) = 1$;
- $\delta(1, b) = 2$, $\delta(2, b) = 1$, $\delta(i, b) = i$, $3 \le i \le n$;
- $\delta(i, c) = i$, $1 \le i \le n - 1$, $\delta(n, c) = 1$.
- $\delta(i, d) = i$, $1 \le i \le n - 1$, and $\delta(n, d)$ is undefined.

The DFA A' is obtained from the DFA A used in the proof of Lemma 6 by adding the transitions on symbol d. Since the transition monoid of A consists of all functions on Q, it is clear that the transition semigroup of A' consists of all partial functions on Q.

Again we choose $L_2 = \{a^n\}$. Let B be the incomplete DFA recognizing $L(A') \rightsquigarrow_{\mathrm{bi}} L_2$ constructed as in Corollary 2. The state set of B is

$$(Q \cup \{\mathrm{dead}\})^n - \{(\mathrm{dead}, \ldots, \mathrm{dead})\}\,.$$

Using the fact that the transition semigroup of A' consists of all partial functions on Q, exactly as in the proof of Lemma 6, it is verified that all states of B are reachable and any two states that are not both of the form $(i, i, \ldots, i)$, $1 \le i \le n$, are distinguishable. This means that the minimal incomplete DFA for $L(A') \rightsquigarrow_{\mathrm{bi}} L_2$ needs at least $(n + 1)^n - n$ states. □

By Corollary 2 and Lemma 7 we have:

Theorem 4 *For a regular language L_1 and an arbitrary language L_2 we have*

$$\mathrm{isc}\,(L_1 \rightsquigarrow_{\mathrm{bi}} L_2) \le (\mathrm{isc}(L_1) + 1)^{\mathrm{isc}(L_1)} - 1.$$

For every $n \ge 2$ there exists a regular language L_1 defined over a four letter alphabet with $\mathrm{isc}(L_1) = n$ and a singleton language L_2 such that

$$\mathrm{isc}\,(L_1 \rightsquigarrow_{\mathrm{bi}} L_2) \ge (\mathrm{isc}(L_1) + 1)^{\mathrm{isc}(L_1)} - \mathrm{isc}(L_1).$$

6 Conclusion

We have established tight state complexity bounds for the deletion of an arbitrary language L_2 from a regular language L_1 both in the case where L_1 and $L_1 \leadsto L_2$ are represented by incomplete DFAs and when they are represented by complete DFAs. Furthermore, in the lower bound construction the deleted language can be chosen to be a singleton set consisting of a string of length one. This result is in some sense the strongest possible because deleting the empty string from L_1 yields just L_1.

Roughly speaking, in the upper bound constructions given in Sect. 3, the DFA B for $L_1 \leadsto L_2$ is based only on the DFA A for L_1 and B depends on L_2 only by way of the transitions the strings of L_2 define on A. This causes that the transitions in parts of B that, respectively, simulate the original DFA A and the computation of A after a string was deleted are closely related and, perhaps partly because of this reason, the lower bound constructions that match the upper bound (respectively, for incomplete and for complete DFAs) are fairly involved. For the constructions we used a five-letter alphabet. The alphabet size could likely be reduced, but this would lead to considerably more complicated proofs of correctness. Furthermore, it does not seem clear whether the general upper bound can be reached using a binary alphabet. Note that for a unary alphabet, deletion coincides with right-quotient and the state complexity is known to be n [36].

We have given almost matching upper and lower bounds for the state complexity of the bipolar deletion of an arbitrary language L_2 from a regular language L_1 both based on complete and incomplete DFAs. The upper and lower bounds differ only by an additive term $n-1$, where n is the size of the DFA for L_1. The lower bound construction based on complete (respectively, incomplete) DFAs uses a three letter (respectively, a four letter) alphabet. The DFA recognizing $L_1 \leadsto_{\text{bi}} L_2$ simulates the transitions of the DFA for L_1 starting from all possible states as an ordered tuple and, thus, the worst-case state complexity of $L_1 \leadsto_{\text{bi}} L_2$ appears to be closely connected to the size of the syntactic monoid of L_1. When the alphabet has at least three letters the syntactic monoid size of an n-state DFA can be n^n, but for binary alphabets estimating the syntactic monoid size is considerably more involved [16]. We leave the precise state complexity of bipolar deletion over binary alphabets as an open problem. Naturally for unary alphabets bipolar deletion again coincides with right-quotient.

References

1. Câmpeanu, C., Salomaa, K., Yu, S.: Tight lower bound for the state complexity of shuffle of regular languages. J. Autom. Lang. Comb. **7**, 303–310 (2002)
2. Cho, D.-J., Han, Y.-S., Ko, S.-K., Salomaa, K.: State complexity ofinversion operations. In: Proceedings of the DCFS 2014. Lecture Notes in Computer Science, vol. 8614. Springer, Berlin, pp. 102–113 (2014)
3. Cui, B., Gao, Y., Kari, L., Yu, S.: State complexity of combined operations with two basic operations. Theor. Comput. Sci. **437**, 98–107 (2012)
4. Domaratzki, M.: Deletion along trajectories. Theor. Comput. Sci. **320**, 293–313 (2004)
5. Domaratzki, M., Okhotin, A.: State complexity of power. Theor. Comput. Sci. **410**, 2377–2392 (2009)
6. Domaratzki, M., Salomaa, K.: Decidability of trajectory-based equations. Theor. Comput. Sci. **345**, 304–330 (2005)
7. Eom, H.-S., Han, Y.-S.: State complexity of combined oeprations for suffix-free regular languages. Theor. Comput. Sci. **510**, 87–93 (2013)
8. Eom, H.-S., Han, Y.-S., Jiráskova, G.: State complexity of basicoperations on non-returning regular languages. In: Proceedings of the DCFS 2013. Lecture Notes in Computer Science, vol. 8031. Springer, Berlin, pp. 54–65 (2013)
9. Eom, H.-S., Han, Y.-S., Salomaa, K.: State complexity of k-unionand k-intersection for prefix-free regular languages. In: Proceedings of the DCFS 2013. Lecture Notes in Computer Science, vol. 8031. Springer, Berlin, pp. 78–89 (2013)

10. Gao, Y., Kari, L.: State complexity of star of union and square of union on k regular languages. Theor. Comput. Sci. **499**, 38–50 (2013)
11. Gao, Y., Moreira, N., Reis, R., Yu, S.: A Review on State Complexity of Individual Operations. Technical report DCC-2011-8, Faculdade de Ciencias, Universidade do Porto. www.dcc.fc.up.pt/dcc/Pubs/TReports/TR11/dcc-2011-08 **(to appear in Computer Science Review)**
12. Gao, Y., Piao, X.: State Complexity of Insertion (manuscript in preparation) (2014)
13. Gao, Y., Yu, S.: State complexity and approximation. Int. J. Found. Comput. Sci. **23**(5), 1085–1098 (2012)
14. Han, Y.-S., Salomaa, K.: State complexity of basic operations on suffix-free regular languages. Theor. Comput. Sci. **410**, 2537–2548 (2009)
15. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata—a survey. Inf. Comput. **209**, 456–470 (2011)
16. Holzer, M., König, B.: On deterministic finite automata and syntactic monoid size. Theor. Comput. Sci. **327**, 319–347 (2004)
17. Jirásek, J., Jiráskova, G., Szabari, A.: State complexity of concatenation and complementation. Int. J. Found. Comput. Sci. **16**, 511–529 (2005)
18. Jiráskova, G., Shallit, J.: The state complexity of star-complement-star. In: Yen, H.-C., Ibarra, O.H. (eds.) Developments in Language Theory. Lecture Notes in Computer Science, vol. 7410. Springer, Berlin, pp. 380–391 (2012)
19. Kari, L.: On language equations with invertible operations. Theor. Comput. Sci. **132**, 129–150 (1994)
20. Kari, L.: On Insertion and Deletion in Formal Languages. Ph.D. thesis, University of Turku (1991)
21. Kari, L.: Deletion operations: closure properties. Int. J. Comput. Math. **52**, 23–42 (1994)
22. Kari, L., Sosik, P.: Aspects of shuffle and deletion on trajectories. Theor. Comput. Sci. **332**, 47–61 (2005)
23. Krawetz, B., Lawrence, J., Shallit, J.: State complexity and the monoid of transformations of a finite set. Int. J. Found. Comput. Sci. **13**, 547–563 (2005)
24. Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. Bull. EATCS **111**, 70–86 (2013)
25. Lavado, G.J., Pighizzini, G., Seki, S.: Operational state complexity under Parikh equivalence. In: Proceedings of the DCFS '14. Lecture Notes in Computer Science, vol. 8614. Springer, berlin pp. 294–305 (2014)
26. Lupanov, O.B.: A comparison of two types of finite sources. Probl. Kibern. **9**, 328–335 (1963)
27. Maslov, A.N.: Estimates on the number of states of finite automata. Sov. Math. Dokl. **11**, 1373–1375 (1970)
28. Mateescu, A., Rozenberg, G., Salomaa, A.: Shuffle on trajectories: syntactic constraints. Theor. Comput. Sci. **197**, 1–56 (1998)
29. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars and formal systems. In: Proceedings of the SWAT (FOCS). IEEE Computer Society, Northridge, California, pp. 188–191 (1971)
30. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Trans. Comput. **C–20**, 1211–1214 (1971)
31. Rampersad, N.: The state complexity of L^2 and L^k. Inf. Proc. Lett. **98**, 231–234 (2006)
32. Salomaa, A., Salomaa, K., Yu, S.: State complexity of combined operations. Theor. Comput. Sci. **383**, 140–152 (2007)
33. Salomaa, A., Wood, D., Yu, S.: On the state complexity of reversals of regular languages. Theor. Comput. Sci. **320**, 315–329 (2004)
34. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press, Cambridge (2009)
35. Wood, D.: Theory of Computation. Wiley, New York (1987)
36. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. I, pp. 41–110. Springer, Berlin (1997)
37. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theor. Comput. Sci. **125**, 315–328 (1994)