

# A General Approach to State Complexity of Operations: Formalization and Limitations<sup>\*</sup>

Sylvie Davies

Department of Pure Mathematics  
University of Waterloo, Waterloo, Ontario, Canada  
sldavies@uwaterloo.ca

**Abstract.** The state complexity of the result of a regular operation is often positively correlated with the number of distinct transformations induced by letters in the minimal deterministic finite automaton of the input languages. That is, more transformations in the inputs means higher state complexity in the output. When this correlation holds, the state complexity of a unary operation can be maximized using languages in which there is one letter corresponding to each possible transformation; for operations of higher arity, we can use  $m$ -tuples of languages in which there is one letter corresponding to each possible  $m$ -tuple of transformations. In this way, a small set of languages can be used as witnesses for many common regular operations, eliminating the need to search for witnesses – though at the expense of using very large alphabets. We formalize this approach and examine its limitations. We define a class of “uniform” operations for which this approach works; the class is closed under composition and includes common operations such as star, concatenation, reversal, union, and complement. Our main result is that the worst-case state complexity of a uniform operation can be determined by considering a finite set of witnesses, and this set depends only on the arity of the operation and the state complexities of the inputs.

## 1 Introduction

Given a regular operation, how do we determine its (deterministic) state complexity? There is probably no universal method for solving these problems. Nonetheless, for many operations there is a common approach we can take. While this approach is not new, it does not seem to be universally known to researchers. The key idea first appeared in a 1978 paper of Sakoda and Sipser [19], but it has seldom been used in the context of state complexity. In cases where it was used, authors typically did not acknowledge the full power and generality of the approach. This paper attempts to give a formal, general account of the approach and its uses in state complexity.

We will refer to the approach in question as the “one letter per action” (OLPA) approach. We give an informal description of the OLPA approach below.

---

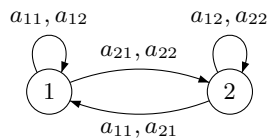
<sup>\*</sup> This work was supported by the Natural Sciences and Engineering Research Council of Canada grant No. OGP0000871.

The root of the approach is to think about regular operations in terms of how they affect deterministic finite automata (DFAs). Typically, a regular operation takes some DFAs as input and modifies or combines them in some way to produce a new DFA. Many of these DFA constructions have the following properties:

- If a new letter is added to the input DFAs, then the state complexity of the output DFA will not decrease.
- If a new letter is added to the input DFAs, and in each DFA this letter acts the same as an existing letter, then the state complexity of the output DFA will stay the same.

If we know an operation has these two properties, this suggests a way to maximize the state complexity of the operation: keep adding new letters to the input DFAs, until the point where we have letters corresponding to all possible actions across all the input DFAs. The first property ensures that adding letters can only increase or maintain the state complexity of the output, while the second property ensures that the state complexity of the output will eventually reach a maximum. In this way, we can obtain witnesses for the worst-case state complexity of the operation.

Figure 1 shows the result of applying this construction to a two-state DFA with unspecified initial and final states, to be used as input for a unary operation. The DFA has one letter for each of the four functions from  $\{1, 2\}$  to itself. To illustrate the construction for a three-state DFA, we would need  $3^3 = 27$  letters!



**Fig. 1.** Two-state “one letter per action” DFA, with initial and final states unspecified.

Since the state complexity of the output is not affected only by the actions of letters in the inputs, but also the initial states and final state sets of the inputs, we construct these witnesses for *each configuration* of initial and final states. Then for each configuration, it remains to solve the combinatorial problem of counting reachable and pairwise distinguishable states in the output DFA, under the assumption that every possible action in the input DFAs is available.

Of course, these combinatorial problems can sometimes be quite hard, and this method produces witnesses over extremely large alphabets. Nonetheless, we believe the OLPA approach is useful to know for several reasons. For one, it gives a way to compute the exact worst-case state complexity of certain operations for inputs with small state complexity. Naively, computing this value would require checking all possible input DFAs under the state complexity threshold. However, if our operation has the aforementioned two properties, we can just check the small set of witness DFAs previously described. This check is quite slow because of the large alphabets of the witnesses, but the computation is often feasible for a

handful of small values, and often these small values are enough to start making conjectures about the general behaviour of the state complexity function.

Second, using “OLPA witnesses” with one letter for each action of the input DFAs can simplify proofs and make the ideas behind them more clear. State complexity proofs using witnesses over optimal or near-optimal alphabets are often rather technical. Consider the following proof that for  $n \geq 2$ , the state complexity of the reversal operation is  $2^n$ :

*Let  $\mathcal{D} = (Q, \Sigma, \delta, q_0, F)$  be a DFA with  $n$  states. Suppose for each function  $t: Q \rightarrow Q$ , there is a letter  $a_t \in \Sigma$  such that  $\delta(q, a_t) = t(q)$ . Let  $\mathcal{R} = (Q, \Sigma, \Delta, F, \{q_0\})$  be an NFA for the reverse of  $\mathcal{L}(\mathcal{D})$ , where  $\Delta(q, a_t) = \{p \in Q : \delta(p, a_t) = q\}$ . For each  $S \subseteq Q$ , let  $s$  be a function that maps  $S$  into  $F$  and  $Q \setminus S$  into  $Q \setminus F$ . Then  $\Delta(F, a_s) = \{p \in Q : \delta(p, a_s) \in F\} = S$ , so all  $2^n$  subsets of  $Q$  are reachable. Distinct sets  $S, T \subseteq Q$  are distinguished as follows: choose an element  $q$  that (without loss of generality) is in  $S$  but not  $T$ , and choose a function  $s$  that maps  $q_0$  to  $q$  and  $Q \setminus \{q_0\}$  into  $Q \setminus \{q\}$ . Then  $\Delta(S, a_s)$  contains the final state  $q_0$  and  $\Delta(T, a_s)$  does not, so the sets are distinguished.  $\square$*

Because we are free to choose letters that do exactly what we want, this proof is very simple. It is also rather illuminating, since we can immediately extract a sufficient condition for attaining the worst-case state complexity from the proof: it suffices that there are letters which induce, for each  $S \subseteq Q$ , a function that maps  $S$  into  $F$  and  $Q \setminus S$  outside of  $F$ , and for each  $q \in Q$ , a function that maps  $q_0$  to  $q$  and no other elements to  $q$ . Furthermore, if one notices that letters can be replaced by words throughout the proof (that is, the hypothesis “there is a letter  $a_t \in \Sigma$  such that  $\delta(q, a_t) = t(q)$ ” can be replaced by “there is a word  $w_t \in \Sigma^*$  such that  $\delta(q, w_t) = t(q)$ ”), one recovers the result of Salomaa, Wood and Yu [20] that the state complexity is maximized if the transition monoid of  $\mathcal{D}$  contains all functions from  $Q$  to itself (in addition to strengthening the aforementioned sufficient condition from letters to words).

For contrast, consider the proofs given by Jirásková and Šebej [14] that the worst-case state complexity can be attained over ternary and binary alphabets. The ternary proof is about as short as the proof above, but somewhat more terse, asking the reader to compute transitions under the word  $bc^{i_2-i_1-1}a^{i_1}$ . It also does not offer the same insight into general conditions for attaining the worst-case complexity. Meanwhile, the binary proof is long and involves a complicated multi-case induction argument. Of course, it is useful and desirable to have proofs that use witnesses with small alphabets, and we do not bring up these proofs to criticize them or suggest they should be replaced. We just believe that proofs using the OLPA approach can sometimes be simpler and more illuminating.

Our third reason for studying the OLPA approach is that we believe it could lead to a better general understanding of state complexity of operations, and the conditions that lead to maximal blow-ups in complexity. The fact that this approach exists and applies to many of the operations studied in the state complexity literature perhaps suggests something about the nature of state complexity. Suppose we think of each letter in an alphabet as an “instruction”, and a sequence of instructions as a “program”; then a language of state complexity  $n$  can

be viewed as a collection of programs for a “computer” with  $n$  possible memory states. It seems many operations attain their worst-case state complexity when provided with languages that have an optimal “economy of description” for programs – one instruction (letter) for each possible program (action). Is the key to finding witnesses over small alphabets to maximize this “economy of description” with respect to the alphabet size? Can the effectiveness of Brzozowski’s “universal witnesses” [3], which have small alphabets and maximize the complexity of several operations simultaneously, be explained in this framework?

We will also see there are operations for which the OLPA approach does not work. Our main example is the operation  $\frac{1}{2}L = \{x \in \Sigma^* : xy \in L, |x| = |y|\}$ , which belongs to a general class of operations called proportional removals [7]. If the OLPA approach fails for an operation, what does this imply about the nature and behaviour of this operation? Are the state complexity problems for these operations “harder” to solve in general? Studying operations for which the OLPA approach fails could be a fruitful line of research.

The purpose of this paper is to initiate a formal and general study of the OLPA approach. We define a class of regular operations, called “uniform operations”, for which the OLPA approach provably works. The class of uniform operations includes common operations such as reversal, star, power, concatenation, and all boolean operations (union, intersection, complement, etc.) but also includes more esoteric operations like cyclic shift [15] and shuffles on trajectories [11,17]. The class is also closed under composition, and thus includes combined operations like “star-complement-star” [13].

We prove that for uniform operations, the worst-case state complexity can be determined by considering just a finite set of witnesses. For an  $m$ -ary operation, if the state complexity of the  $j$ -th input is at most  $n_j$ , then the worst-case state complexity of the output can be determined using  $2(n_1 + \dots + n_m)$  different witness languages, and by testing  $2^m n_1 \dots n_m$  different combinations of these witnesses. Additionally, the same set of witnesses can be used for all uniform operations of a particular arity.

In the main sections of this paper, we will first state our definitions and prove our results in the special case of unary operations, before moving on to the general case. While this is ultimately redundant, focusing on the unary case simplifies the notation and makes the definitions and results easier to digest. The reader may find it useful to skip over the discussions of the general case on the first reading, and come back after they fully understand how the OLPA approach is formalized in the unary case.

To close the introduction, we give a history of the ideas behind the OLPA approach. As mentioned, the key insight dates back to a 1978 paper of Sakoda and Sipser [19]. They constructed languages wherein the alphabet letters were directed graphs representing behaviours of non-deterministic finite automata and two-way deterministic finite automata, with one letter for each possible behaviour. They used these languages to prove results on the complexity of conversions between different models of finite automata.

Perhaps the closest ancestor of our work is a 1990 paper of Ravikumar [18], who treated the “Sakoda-Sipser technique” as a “systematic method to prove lower bounds on the size complexity of finite automata”, and applied it to five different problems, two of which were operational state complexity problems! This is the first work we are aware of to present the OLPA approach as a general problem-solving method. Unfortunately, the field of state complexity was not very well-developed at the time, and Ravikumar seemingly did not realize the full generality and applicability of the approach in operational state complexity. Furthermore, Ravikumar’s use of the OLPA approach was less refined than the version we present; Ravikumar used  $n$ -state OLPA witnesses each with an alphabet of size  $n^n$  as inputs to an  $n$ -ary operation, which is not guaranteed to maximize the state complexity of the operation. Our version of the approach would use an alphabet of size  $n^n \cdots n^n = (n^n)^n$ .

In 1992, building off Ravikumar’s work, Birget [1] used this “unrefined” version of the OLPA approach to prove lower bounds on the state complexity of intersection and union.

In 1994, Yu, Zhuang and Salomaa [22] published their seminal paper on the state complexities of basic operations. Notably, even though Yu, Zhuang and Salomaa cited Ravikumar’s work, they did *not* use or mention the “Sakoda-Sipser technique” anywhere in their paper, instead using various ad-hoc methods to prove lower bounds. The technique then seemingly faded into obscurity for a while. It reappeared as the “full automata technique” in a 2006 paper of Yan [21], who credited Sakoda and Sipser for the idea, and applied it to nondeterministic finite automata and automata on infinite words ( $\omega$ -automata). Yan’s paper is frequently cited in the field of  $\omega$ -automata, so the idea seems to have gained some currency there.

In the field of deterministic state complexity, the OLPA approach has made occasional past appearances. Jirásková and Okhotin [15] and later Domaratzki and Okhotin [10] used the OLPA approach to compute the exact worst-case complexity of the cyclic shift and power operations for small values. Brzozowski, Jirásková, Liu, Rajasekaran, and Szykula [4] used the OLPA approach to obtain reachability results for the state complexity of shuffle. Cho, Han, Ko and Salomaa [6] used an OLPA-like construction to establish lower bounds on the state complexity of some “inversion” operations. Interestingly, their construction includes unnecessary extra letters; perhaps these letters were added to somehow make the proof easier.

Outside the context of descriptive complexity, in 2002, Domaratzki, Kisman and Shallit [9] used OLPA automata to enumerate the languages accepted by  $n$ -state automata.

In 2018, Caron, Hamel-De le court, Luque and Patrou [5] independently obtained many of the results in this paper using a different formalism. OLPA witnesses are called “monsters” in their work, and uniform operations are called “depictable operations”. Their paper was submitted to arXiv just ten days after the first version of this paper was submitted.

## 2 Preliminaries

### 2.1 Relations and Functions

A *binary relation*  $\rho$  between  $X$  and  $Y$  is a subset of  $X \times Y$ . If  $\rho \subseteq X \times Y$  and  $\tau \subseteq Y \times Z$ , the *composition* of  $\rho$  and  $\tau$  is the relation

$$\rho\tau = \{(x, z) \in X \times Z : \text{there exists } y \in Y \text{ such that } (x, y) \in \rho \text{ and } (y, z) \in \tau\}.$$

For  $x \in X$  and  $\rho \subseteq X \times Y$ , the *image* of  $x$  under  $\rho$  is the set  $x\rho = \{y \in Y : (x, y) \in \rho\}$ . For  $x \notin X$  we define  $x\rho = \emptyset$ . The *converse* of a binary relation  $\rho \subseteq X \times Y$  is the relation  $\rho^{-1} = \{(y, x) : (x, y) \in \rho\} \subseteq Y \times X$ . The set  $y\rho^{-1} = \{x \in X : (x, y) \in \rho\}$  is called the *preimage* of  $y$  under  $\rho$ .

A *function*  $f: X \rightarrow Y$  is a binary relation  $f \subseteq X \times Y$  such that  $|xf| = 1$  for all  $x \in X$ . Following our notation for binary relations, we write functions to the *right* of their arguments. Composition of functions is defined by composing the corresponding relations. Thus the order of composition is *left-to-right*; in a composition  $fg$ , first  $f$  is applied and then  $g$ . A *transformation* of a set  $X$  is a function  $t: X \rightarrow X$ , that is, a function from  $X$  into itself.

### 2.2 Languages and Automata

A *finite automaton* (FA) is a tuple  $\mathcal{A} = (Q, \Sigma, T, I, F)$  where  $Q$  is a finite set of *states*,  $\Sigma$  is a finite set of *letters* called an *alphabet*,  $T \subseteq Q \times \Sigma \times Q$  is a set of *transitions*,  $I \subseteq Q$  is a set of *initial states*, and  $F \subseteq Q$  is a set of *final states*. The triple  $(Q, I, F)$  is called the *state configuration* of the automaton.

We now define a binary relation  $T_w \subseteq Q \times Q$  for each  $w \in \Sigma^*$ . Define  $T_\varepsilon = \{(q, q) : q \in Q\}$ ; in terms of maps, this is the identity map on  $Q$ . For  $a \in \Sigma$ , define  $T_a = \{(p, q) \in Q \times Q : (p, a, q) \in T\}$ . For  $w = a_1 \cdots a_k$  with  $a_1, \dots, a_k \in \Sigma$ , define  $T_w = T_{a_1} \cdots T_{a_k}$ . The relation  $T_w$  is called the *relation induced by  $w$*  or the *action of  $w$* . If  $T_w$  is a transformation of the state set  $Q$ , it may also be called the *transformation induced by  $w$* . The set  $\{T_w : w \in \Sigma^*\}$  is a monoid under composition, called the *transition monoid* of  $\mathcal{A}$ .

If  $\mathcal{A} = (Q, \Sigma, T, I, F)$  is a finite automaton such that  $|I| = 1$  and  $T_a$  is a function for each  $a \in \Sigma$ , we say  $\mathcal{A}$  is *deterministic*. We abbreviate “deterministic finite automaton” to DFA. As a result of this definition, that all DFAs we consider in this paper are *complete* DFAs (that is, they have exactly one transition defined for each state-letter pair), and a finite automaton with an empty state set or with no initial state is not considered a DFA.

Let  $\mathcal{A} = (Q, \Sigma, T, I, F)$  be an FA. A word  $w \in \Sigma^*$  is *accepted* by  $\mathcal{A}$  if we have  $IT_w \cap F \neq \emptyset$ . If  $\mathcal{A}$  is a DFA with  $I = \{i\}$ , this condition becomes  $iT_w \in F$ . The *language* of  $\mathcal{A}$ , denoted  $\mathcal{L}(\mathcal{A})$ , is the set of all words it accepts. If  $L$  is the language of  $\mathcal{A}$ , we also say that  $\mathcal{A}$  *accepts  $L$*  and that  $\mathcal{A}$  *is an FA for  $L$* . Languages of FAs are called *regular languages*.

A *regular operation* of arity  $m$  is a function that takes  $m$  regular languages as input and produces a regular language. A *DFA operation* of arity  $m$  is a function

that takes  $m$  DFAs as input and produces a DFA. We say a regular operation  $\Phi$  is *equivalent* to a DFA operation  $\Psi$  if both operations have the same arity  $m$ , and for all  $m$ -tuples of DFAs  $(\mathcal{D}_1, \dots, \mathcal{D}_m)$ , we have  $\mathcal{L}((\mathcal{D}_1, \dots, \mathcal{D}_m)\Psi) = (\mathcal{L}(\mathcal{D}_1), \dots, \mathcal{L}(\mathcal{D}_m))\Phi$ . In other words, they are equivalent if the DFA operation  $\Psi$  is an “implementation” of the regular operation  $\Phi$ , in the sense that we can compute the output of  $\Phi$  by taking arbitrary DFAs for the input languages, feeding them to  $\Psi$ , and taking the language of the output DFA.

In this paper we consider only DFA operations  $\Psi$  that are *alphabet-preserving* in the following sense:

- An input  $(\mathcal{D}_1, \dots, \mathcal{D}_m)$  is only valid if all the DFAs have the same alphabet.
- If  $\Sigma$  is the common alphabet of the input DFAs, then  $\Sigma$  will be the alphabet of the output DFA.

Furthermore, we consider only regular operations that are equivalent to an alphabet-preserving DFA operation.

### 2.3 State Complexity

A DFA for a regular language  $L$  is *minimal* if it has the minimal number of states amongst all DFAs that accept  $L$ . The *state complexity* of a regular language is the number of states in a minimal DFA accepting the language. The state complexity of  $L$  is denoted  $\text{sc}(L)$ .

The notion of state complexity extends to regular operations. Let  $\Phi$  be a unary regular operation. The *state complexity of the operation  $\Phi$*  is the following function which takes a positive integer as input:

$$n \mapsto \max\{\text{sc}(L\Phi) : \text{sc}(L) \leq n\}.$$

That is, the state complexity of  $\Phi$  is the worst-case state complexity of the output  $L\Phi$ , expressed as a function of the maximal allowed state complexity of the input  $L$ . Note that  $\max\{\text{sc}(L\Phi) : \text{sc}(L) \leq n\}$  might not exist for all  $n$ ; in such cases, the output of the function is  $\infty$ .

This idea generalizes to operations of higher arity. Let  $\Phi$  be an  $m$ -ary regular operation. The state complexity of  $\Phi$  is the following function which takes an  $m$ -tuple of positive integers as input:

$$(n_1, \dots, n_m) \mapsto \max\{\text{sc}((L_1, \dots, L_m)\Phi) : \text{sc}(L_i) \leq n_i, 1 \leq i \leq m\}.$$

The output is either a positive integer, or  $\infty$  if the maximum does not exist.

### 2.4 Morphisms

Let  $\Sigma$  and  $\Gamma$  be alphabets. A *morphism* is a function  $\varphi: \Sigma^* \rightarrow \Gamma^*$  such that  $(xy)\varphi = (x\varphi)(y\varphi)$ ; in other words, a morphism is just a monoid homomorphism between two free monoids. To define a morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$ , it is sufficient to specify its values on letters from  $\Sigma$ ; the values on letters completely determine the values on words.

If  $L \subseteq \Gamma^*$  is regular, then  $L\varphi^{-1}$  is regular. To see this, let  $\varphi: \Sigma^* \rightarrow \Gamma^*$  be a morphism and let  $\mathcal{B} = (Q, \Gamma, T, i, F)$  be a DFA. We can construct a DFA for  $\mathcal{L}(\mathcal{B})\varphi^{-1}$  as follows: let  $\mathcal{B}\varphi^{-1} = (Q, \Sigma, T', i, F)$ , where  $T' = \{(q, a, qT_{a\varphi}) : q \in Q, a \in \Sigma\}$ . Then it is easily verified that  $\mathcal{L}(\mathcal{B}\varphi^{-1}) = \mathcal{L}(\mathcal{B})\varphi^{-1}$ . We call  $\mathcal{B}\varphi^{-1}$  the *inverse morphism DFA* of  $\mathcal{B}$  with respect to  $\varphi$ .

Note that  $\mathcal{B}\varphi^{-1}$  has the same state configuration as  $\mathcal{B}$ . This will be useful for multiple reasons, but in particular it implies the following result for regular languages  $L$  and  $K$ :

**Lemma 1.** *If  $L = K\varphi^{-1}$ , then  $\text{sc}(L) \leq \text{sc}(K)$ .*

### 3 Transformation Languages

In this section, we formally define the witness languages that are used in the OLPA approach.

Fix a set  $Q$  and let  $\Sigma$  be a set of transformations of  $Q$ . For  $i \in Q$  and  $F \subseteq Q$ , the *transformation language*  $\Sigma(i, F)$  is the language of the DFA  $(Q, \Sigma, T, i, F)$ , where  $T = \{(q, t, qt) : q \in Q, t \in \Sigma\}$ . This DFA is called the *standard DFA* for the transformation language.

The set of all transformations of a set  $Q$  is called the *full transformation monoid* on  $Q$ , and is denoted  $\mathcal{T}_Q$ . The *full transformation languages* of the form  $\mathcal{T}_Q(i, F)$  play an important role in the theory behind the OLPA approach.

Notice that the language  $\mathcal{T}_Q(i, F)$  has alphabet  $\mathcal{T}_Q$ , and the standard DFA for  $\mathcal{T}_Q(i, F)$  has transitions  $\{(q, t, qt) : q \in Q, t \in \mathcal{T}_Q\}$ . This DFA has *one letter per transformation* of the state set  $Q$ , that is, one letter per possible action on the DFA's states. Full transformation languages are the languages used as witnesses when applying the OLPA approach to unary operations.

Let  $L$  be a regular language over  $\Sigma$  recognized by a DFA  $\mathcal{D} = (Q, \Sigma, T, i, F)$ . The *standard transformation morphism* of  $L$  (with respect to  $\mathcal{D}$ ), denoted by  $\varphi_L: \Sigma^* \rightarrow \mathcal{T}_Q^*$ , is defined by  $a\varphi_L = T_a$ . The following fact is easily verified:

**Proposition 2.**  $L = \mathcal{T}_Q(i, F)\varphi_L^{-1}$ .

Full transformation languages do not suffice as OLPA witnesses for operations of arity greater than one. When applying the OLPA approach to operations of arity  $m$ , we want to use an  $m$ -tuple  $(\mathcal{D}_1, \dots, \mathcal{D}_m)$  of DFAs (where  $\mathcal{D}_j = (Q_j, \Sigma, T_j, i_j, F_j)$  for  $1 \leq j \leq m$ ) with the following property: for each  $m$ -tuple of transformations  $(t_1: Q_1 \rightarrow Q_1, \dots, t_m: Q_m \rightarrow Q_m)$ , there exists a letter  $a \in \Sigma$  such that  $a$  induces transformation  $t_j$  in  $\mathcal{D}_j$  for  $1 \leq j \leq m$ . That is, we have one letter for every possible combination of actions across all the input DFAs.

For this purpose, we define *transformation tuple languages*. Let  $Q_1, \dots, Q_m$  be finite sets and let  $\Sigma$  be a subset of  $\mathbb{T} = \mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m}$ . For  $j$  with  $1 \leq j \leq m$ ,  $i \in Q_j$ , and  $F \subseteq Q_j$ , the *transformation tuple language*  $\Sigma_j(i, F)$  is the language of the DFA  $(Q_j, \Sigma, T, i, F)$  where  $T = \{(q, (t_1, \dots, t_m), qt_j) : q \in Q_j, (t_1, \dots, t_m) \in \Sigma\}$ . This DFA is called the *standard DFA* of the transformation tuple language. The *full transformation tuple languages* of the form  $\mathbb{T}_j(i, F)$  are used as OLPA witnesses in the case of  $m$ -ary operations.



There is a generalization of Proposition 2 for full transformation tuple languages. Let  $(L_1, \dots, L_m)$  be an  $m$ -tuple of regular languages over  $\Sigma$ , where  $L_j$  is recognized by the DFA  $\mathcal{D}_j = (Q_j, \Sigma, T_j, i_j, F_j)$  for  $1 \leq j \leq m$ . The *standard transformation tuple morphism* of  $(L_1, \dots, L_m)$  (with respect to  $(\mathcal{D}_1, \dots, \mathcal{D}_m)$ ), denoted by  $\varphi_{(L_1, \dots, L_m)}: \Sigma^* \rightarrow (\mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m})^*$ , is defined by  $a\varphi_{(L_1, \dots, L_m)} = ((T_1)_a, \dots, (T_m)_a)$ . As shorthand, let  $\mathbb{T} = \mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m}$  and let  $\varphi = \varphi_{(L_1, \dots, L_m)}$ .

**Proposition 3.** *We have  $(L_1, \dots, L_m) = (\mathbb{T}_1(i_1, F_1)\varphi^{-1}, \dots, \mathbb{T}_m(i_m, F_m)\varphi^{-1})$ .*

*Proof.* It suffices to show for all  $w \in \Sigma^*$  that  $w \in L_j \iff w\varphi \in \mathbb{T}_j(i_j, F_j)$ . Fix  $j$  and let  $(Q_j, \mathbb{T}, T, i_j, F_j)$  be the standard DFA of  $\mathbb{T}_j(i_j, F_j)$ . Then we have

$$w \in L_j \iff i_j(T_j)_w \in F_j \iff i_j T_{w\varphi} \in F_j \iff w\varphi \in \mathbb{T}_j(i_j, F_j),$$

as required.

The second two-way implication may not be obvious. To see that it holds, first note that if  $w$  is empty, then  $(T_j)_w$  and  $T_{w\varphi}$  are both the identity map on  $Q_j$ . Otherwise, suppose  $w = a_1 \dots a_k$  with  $a_1, \dots, a_k \in \Sigma$ . We may write  $w\varphi = (a_1\varphi) \dots (a_k\varphi)$ , and thus  $T_{w\varphi} = T_{a_1\varphi} \dots T_{a_k\varphi}$ . By definition, we have  $a_i\varphi = ((T_1)_{a_i}, \dots, (T_m)_{a_i})$  for  $1 \leq i \leq k$ . This  $m$ -tuple of transformations is a “letter” of the alphabet  $\mathbb{T} = \mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m}$ . Then  $T_{a_i\varphi}$  is the transformation of  $Q_j$  induced by the “letter”  $a_i\varphi$ . By definition, this induced transformation is the map  $q \mapsto q(T_j)_{a_i}$  for  $q \in Q_j$ . Thus  $T_{a_i\varphi} = (T_j)_{a_i}$  for  $1 \leq i \leq k$ . It follows that

$$T_{w\varphi} = T_{a_1\varphi} \dots T_{a_k\varphi} = (T_j)_{a_1} \dots (T_j)_{a_k} = (T_j)_w.$$

Hence the implication holds.  $\square$

## 4 Uniform Regular Operations

Our goal in this section is to define a large class of operations for which the OLPA approach works. The approach does not work for all regular operations; it is easy to come up with rather contrived examples of operations for which OLPA fails. Consider an operation which sends languages with one letter per action to the empty language, and acts as the identity on all other languages. There are a few ways we could implement this operation as a DFA operation:

- If the input DFA has one letter per action, output a DFA with no final states. Otherwise, output the input DFA.
- If the input DFA has one letter per action, output a DFA in which the initial state is non-final and all the actions send the initial state to a sink state. Otherwise, output the input DFA.

The problem with this operation is that its behaviour is not “uniform” across all languages; it detects particular languages and has special behaviour for them. In the first case, the operation does not behave uniformly on states: for most

DFAs it preserves the final state set, but for DFAs with one letter per action it can change the final state set. In the second case, the operation is not uniform on states *or* on actions: for most DFAs it preserves the state configuration and actions, but for DFAs with one letter per action, it can change whether the initial state is final, and also replace the actions by completely different actions.

We now attempt to formally define this idea of “uniformity” for unary operations. Let  $\Psi$  be a unary DFA operation. We say  $\Psi$  is *uniform* if for every pair of DFAs  $\mathcal{A} = (Q, \Sigma, T_{\mathcal{A}}, i, F)$  and  $\mathcal{B} = (Q, \Gamma, T_{\mathcal{B}}, i, F)$  with the same state configuration, the image DFAs  $\mathcal{A}\Psi = (Q'_{\mathcal{A}}, \Sigma, T'_{\mathcal{A}}, i'_{\mathcal{A}}, F'_{\mathcal{A}})$  and  $\mathcal{B}\Psi = (Q'_{\mathcal{B}}, \Gamma, T'_{\mathcal{B}}, i'_{\mathcal{B}}, F'_{\mathcal{B}})$  satisfy the following conditions:

1.  $(Q'_{\mathcal{A}}, i'_{\mathcal{A}}, F'_{\mathcal{A}}) = (Q'_{\mathcal{B}}, i'_{\mathcal{B}}, F'_{\mathcal{B}})$ .
2. Whenever  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$  for  $a \in \Sigma$  and  $b \in \Gamma$ , we have  $(T'_{\mathcal{A}})_a = (T'_{\mathcal{B}})_b$ .

We will say a unary regular operation  $\Phi$  is *uniform* if there exists a uniform unary DFA operation equivalent to  $\Phi$ .

We can interpret this definition intuitively as follows. The first condition says that the operation is uniform with respect to state configurations: if the operation is given two input DFAs with the same state configuration, it will produce two output DFAs with the same state configuration. The second condition says that the operation is uniform with respect to actions: if the operation is given two input DFAs with the same state configuration and a common action, then it will produce two output DFAs with a common action, and furthermore the same letters which induce the common action in the input DFAs will induce the common action in the output DFAs.

The definition of uniformity is heavily dependent on DFAs. Thus, it may come as a surprise that there is a simple and purely language-theoretic characterization of uniformity. A morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  is *1-uniform* if it maps letters to letters.

**Proposition 4.** *Let  $L$  and  $K$  be regular languages over  $\Sigma$  and  $\Gamma$  respectively. The following are equivalent:*

1. *The regular operation  $\Phi$  is uniform.*
2. *For all 1-uniform morphisms  $\varphi: \Sigma^* \rightarrow \Gamma^*$ , if  $L = K\varphi^{-1}$  then  $L\Phi = K\Phi\varphi^{-1}$ .*

*Proof.* (1)  $\implies$  (2): Since  $\Phi$  is uniform, there is a uniform DFA operation  $\Psi$  equivalent to  $\Phi$ . Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  such that  $L = K\varphi^{-1}$ . Let  $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, T_{\mathcal{A}}, i_{\mathcal{A}}, F_{\mathcal{A}})$  be a DFA for  $L$ , and let  $\mathcal{B} = (Q_{\mathcal{B}}, \Gamma, T_{\mathcal{B}}, i_{\mathcal{B}}, F_{\mathcal{B}})$  be a DFA for  $K$ . We write  $w_{\mathcal{A}}$  for  $(T_{\mathcal{A}})_w$ , and  $w_{\mathcal{B}}$  for  $(T_{\mathcal{B}})_w$ .

Note that we can choose our DFAs so that they have the same state configuration. This follows from the fact that  $L = K\varphi^{-1}$ , and thus we can take  $\mathcal{A} = \mathcal{B}\varphi^{-1}$  which has the same state configuration as  $\mathcal{B}$ . Henceforth write  $Q_{\mathcal{A}} = Q_{\mathcal{B}} = Q$ ,  $i_{\mathcal{A}} = i_{\mathcal{B}} = i$ , and  $F_{\mathcal{A}} = F_{\mathcal{B}} = F$ .

Let  $\mathcal{A}\Psi = \mathcal{A}' = (Q'_{\mathcal{A}}, \Sigma, T'_{\mathcal{A}}, i'_{\mathcal{A}}, F'_{\mathcal{A}})$  and let  $\mathcal{B}\Psi = \mathcal{B}' = (Q'_{\mathcal{B}}, \Gamma, T'_{\mathcal{B}}, i'_{\mathcal{B}}, F'_{\mathcal{B}})$ . Write  $w'_{\mathcal{A}}$  for  $(T'_{\mathcal{A}})_w$  and  $w'_{\mathcal{B}}$  for  $(T'_{\mathcal{B}})_w$ . The DFA  $\mathcal{A}'$  recognizes  $L\Phi$ , and the DFA  $\mathcal{B}'$  recognizes  $K\Phi$ . By the uniformity of  $\Psi$ , we can write  $Q'_{\mathcal{A}} = Q'_{\mathcal{B}} = Q'$ ,  $i'_{\mathcal{A}} = i'_{\mathcal{B}} = i'$ , and  $F'_{\mathcal{A}} = F'_{\mathcal{B}} = F'$ .

Now, we want to show that  $L\Phi = K\Phi\varphi^{-1}$ . Since  $\mathcal{A} = \mathcal{B}\varphi^{-1}$ , for all  $q \in Q$  and  $a \in \Sigma$  we have  $qa_{\mathcal{A}} = q(a\varphi)_{\mathcal{B}}$  by definition. Thus  $a_{\mathcal{A}}$  and  $(a\varphi)_{\mathcal{B}}$  are equal as transformations of  $Q$  for all  $a \in \Sigma$ . By the uniformity of  $\Psi$ ,  $a'_{\mathcal{A}}$  and  $(a\varphi)'_{\mathcal{B}}$  are equal as transformations of  $Q'$ . It follows that  $w'_{\mathcal{A}}$  and  $(w\varphi)'_{\mathcal{B}}$  are equal as transformations of  $Q'$  for all  $w \in \Sigma^*$ . Hence we have

$$w \in L\Phi \iff i'w'_{\mathcal{A}} \in F' \iff i'(w\varphi)'_{\mathcal{B}} \in F' \iff w\varphi \in K\Phi \iff w \in K\Phi\varphi^{-1}.$$

This proves that  $L\Phi = K\Phi\varphi^{-1}$ .

(2)  $\implies$  (1): We are given a regular operation  $\Phi$ . We want to produce a uniform DFA operation  $\Psi$  such that for all DFAs  $\mathcal{A}$ , we have  $\mathcal{L}(\mathcal{A}\Psi) = \mathcal{L}(\mathcal{A})\Phi$ .

Fix an  $n$ -state DFA  $\mathcal{A} = (Q, \Sigma, T, i, F)$  and let  $L$  be its language. We define  $\mathcal{A}\Psi$  as follows. By Proposition 2, we have  $L = \mathcal{T}_Q(i, F)\varphi_L^{-1}$ , where  $\varphi_L: \Sigma^* \rightarrow \mathcal{T}_Q^*$  is the standard transformation morphism of  $L$ . By assumption, we then have  $L\Phi = \mathcal{T}_Q(i, F)\Phi\varphi_L^{-1}$ . Let  $\mathcal{D}' = (Q', \mathcal{T}_Q, T', i', F')$  be a minimal DFA for  $\mathcal{T}_Q(i, F)\Phi$ , and set  $\mathcal{A}\Psi = \mathcal{D}'\varphi_L^{-1}$ .

It is clear that we have  $\mathcal{L}(\mathcal{A}\Psi) = \mathcal{T}_Q(i, F)\Phi\varphi_{\mathcal{L}(\mathcal{A})}^{-1} = \mathcal{L}(\mathcal{A})\Phi$  as required. To see that  $\Psi$  is uniform, fix DFAs  $\mathcal{A} = (Q, \Sigma, T_{\mathcal{A}}, i, F)$  and  $\mathcal{B} = (Q, \Gamma, T_{\mathcal{B}}, i, F)$ . We compute the images  $\mathcal{A}\Psi = \mathcal{A}' = (Q'_{\mathcal{A}}, \Sigma, T'_{\mathcal{A}}, i'_{\mathcal{A}}, F'_{\mathcal{A}})$  and  $\mathcal{B}\Psi = \mathcal{B}' = (Q'_{\mathcal{B}}, \Gamma, T'_{\mathcal{B}}, i'_{\mathcal{B}}, F'_{\mathcal{B}})$ . Now, let  $\mathcal{D}' = (Q', \mathcal{T}_Q, T'_{\mathcal{D}}, i', F')$  be the minimal DFA for  $\mathcal{T}_Q(i, F)\Phi$ . By definition, we have  $\mathcal{A}' = \mathcal{D}'\varphi_{\mathcal{L}(\mathcal{A})}^{-1}$  and  $\mathcal{B}' = \mathcal{D}'\varphi_{\mathcal{L}(\mathcal{B})}^{-1}$ . So  $\mathcal{A}'$  and  $\mathcal{B}'$  both have the same state configuration as  $\mathcal{D}'$ . It follows that  $(Q'_{\mathcal{A}}, i'_{\mathcal{A}}, F'_{\mathcal{A}}) = (Q'_{\mathcal{B}}, i'_{\mathcal{B}}, F'_{\mathcal{B}})$ , as required.

Next, fix  $a \in \Sigma$  and  $b \in \Gamma$  such that  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$ . We have  $q(T'_{\mathcal{A}})_a = q(T'_{\mathcal{D}})_{a\varphi_{\mathcal{L}(\mathcal{A})}}$  for all  $q$ . Also,  $q(T'_{\mathcal{B}})_b = q(T'_{\mathcal{D}})_{b\varphi_{\mathcal{L}(\mathcal{B})}}$  for all  $q$ . By the definition of the standard transformation morphism, we have  $a\varphi_{\mathcal{L}(\mathcal{A})} = b\varphi_{\mathcal{L}(\mathcal{B})}$ , since  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$ . It follows that  $(T'_{\mathcal{A}})_a = (T'_{\mathcal{B}})_b$ , as required. Thus  $\Psi$  is uniform.  $\square$

Now that we have established the definition of uniformity and the language-theoretic characterization for unary regular operations, we turn to operations of higher arity. Let  $\Psi$  be an  $m$ -ary DFA operation. We say  $\Psi$  is *uniform* if for every pair of  $m$ -tuples of DFAs  $(\mathcal{A}_1, \dots, \mathcal{A}_m)$  and  $(\mathcal{B}_1, \dots, \mathcal{B}_m)$ , where for each  $j$  with  $1 \leq j \leq m$ , the DFAs  $\mathcal{A}_j$  and  $\mathcal{B}_j$  have the same state configuration, the DFA  $\mathcal{A}_j$  has alphabet  $\Sigma$  and transition set  $T_{\mathcal{A}_j}$ , and the DFA  $\mathcal{B}_j$  has transition set  $T_{\mathcal{B}_j}$  and alphabet  $\Gamma$ ; the image DFAs  $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi$  and  $\mathcal{B} = (\mathcal{B}_1, \dots, \mathcal{B}_m)\Psi$  have transition sets  $T_{\mathcal{A}}$  and  $T_{\mathcal{B}}$  respectively, and the following conditions hold:

1. The image DFAs  $\mathcal{A}$  and  $\mathcal{B}$  have the same state configuration.
2. If there exist letters  $a \in \Sigma$  and  $b \in \Gamma$  such that  $(T_{\mathcal{A}_j})_a = (T_{\mathcal{B}_j})_b$  for each  $j$  with  $1 \leq j \leq m$ , then  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$ .

There is a corresponding language-theoretic characterization of the general definition of uniformity.

**Proposition 5.** *Let  $(L_1, \dots, L_m)$  and  $(K_1, \dots, K_m)$  be  $m$ -tuples of regular languages, where  $L_j$  is a language over  $\Sigma$  and  $K_j$  is a language over  $\Gamma$  for  $1 \leq j \leq m$ . The following are equivalent:*

1. *The  $m$ -ary regular operation  $\Phi$  is uniform.*

2. For all 1-uniform morphisms  $\varphi: \Sigma^* \rightarrow \Gamma^*$ , if  $L_j = K_j\varphi^{-1}$  for  $1 \leq j \leq m$ , then  $(L_1, \dots, L_m)\Phi = (K_1, \dots, K_m)\Phi\varphi^{-1}$ .

The proof is very similar to the proof of Proposition 4, except the general definition of uniformity is used and full transformation tuple languages are used instead of full transformation languages.

*Proof.* (1)  $\implies$  (2): Since  $\Phi$  is uniform, there is a uniform DFA operation  $\Psi$  equivalent to  $\Phi$ . Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  such that  $L_j = K_j\varphi^{-1}$  for  $1 \leq j \leq m$ . We want to show that  $(L_1, \dots, L_m)\Phi = (K_1, \dots, K_m)\Phi\varphi^{-1}$ .

Since  $L_j = K_j\varphi^{-1}$ , for each  $j$  we can find a DFA  $\mathcal{A}_j$  for  $L_j$  and a DFA  $\mathcal{B}_j$  for  $K_j$  such that  $\mathcal{A}_j = \mathcal{B}_j\varphi^{-1}$ . Each pair of DFAs  $\mathcal{A}_j$  and  $\mathcal{B}_j$  has a common state configuration  $(Q_j, i_j, F_j)$ . For  $1 \leq j \leq m$ , let  $\mathcal{A}_j = (Q_j, \Sigma, T_{\mathcal{A}_j}, i_j, F_j)$  be the DFA for  $L_j$  and let  $\mathcal{B}_j = (Q_j, \Gamma, T_{\mathcal{B}_j}, i_j, F_j)$  be the DFA for  $K_j$ . By the uniformity of  $\Psi$ , the image DFAs  $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi$  and  $\mathcal{B} = (\mathcal{B}_1, \dots, \mathcal{B}_m)\Psi$  have a common state configuration  $(Q, i, F)$ . Write  $\mathcal{A} = (Q, \Sigma, T_{\mathcal{A}}, i, F)$  and  $\mathcal{B} = (Q, \Sigma, T_{\mathcal{B}}, i, F)$ .

Since  $\mathcal{A}_j = \mathcal{B}_j\varphi^{-1}$  for  $1 \leq j \leq m$ , for all  $q \in Q_j$  and  $a \in \Sigma$ , we have  $q(T_{\mathcal{A}_j})_a = q(T_{\mathcal{B}_j})_{a\varphi}$  by definition. Thus  $(T_{\mathcal{A}_j})_a = (T_{\mathcal{B}_j})_{a\varphi}$  for all  $a \in \Sigma$  and all  $1 \leq j \leq m$ . By the uniformity of  $\Psi$ , it follows that  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_{a\varphi}$ . Hence  $(T_{\mathcal{A}})_w = (T_{\mathcal{B}})_{w\varphi}$  for all  $w \in \Sigma^*$ . Thus we have

$$\begin{aligned} w \in (L_1, \dots, L_m)\Phi &\iff i(T_{\mathcal{A}})_w \in F \iff i(T_{\mathcal{B}})_{w\varphi} \in F \\ &\iff w\varphi \in (K_1, \dots, K_m)\Phi \iff w \in (K_1, \dots, K_m)\Phi\varphi^{-1}. \end{aligned}$$

This proves that  $(L_1, \dots, L_m)\Phi = (K_1, \dots, K_m)\Phi\varphi^{-1}$ .

(2)  $\implies$  (1): We want to produce a uniform  $m$ -ary DFA operation  $\Psi$  such that for all tuples of DFAs  $(\mathcal{A}_1, \dots, \mathcal{A}_m)$  over a common alphabet, we have  $\mathcal{L}((\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi) = (\mathcal{L}(\mathcal{A}_1), \dots, \mathcal{L}(\mathcal{A}_m))\Phi$ .

Fix a tuple  $(\mathcal{A}_1, \dots, \mathcal{A}_m)$  of DFAs over  $\Sigma$ , where  $\mathcal{A}_j$  has state configuration  $(Q_j, i_j, F_j)$ , and let  $L_j = \mathcal{L}(\mathcal{A}_j)$  for  $1 \leq j \leq m$ . We define the image  $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi$  as follows. By Proposition 3 we have  $L_j = \mathbb{T}_j(i_j, F_j)\varphi_{(L_1, \dots, L_m)}^{-1}$ , where  $\varphi_{(L_1, \dots, L_m)}$  is the standard transformation tuple morphism of  $(L_1, \dots, L_m)$  with respect to  $(\mathcal{A}_1, \dots, \mathcal{A}_m)$ , and  $\mathbb{T} = \mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m}$ . Let  $\mathcal{D}$  be a minimal DFA for  $(\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi$ , and set  $(\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi = \mathcal{D}\varphi_{(L_1, \dots, L_m)}^{-1}$ .

We claim that  $\mathcal{L}((\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi) = (\mathcal{L}(\mathcal{A}_1), \dots, \mathcal{L}(\mathcal{A}_m))\Phi$ . Indeed, since  $L_j = \mathbb{T}_j(i_j, F_j)\varphi_{(L_1, \dots, L_m)}^{-1}$ , we have

$$(L_1, \dots, L_m)\Phi = (\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi\varphi_{(L_1, \dots, L_m)}^{-1}.$$

It follows that

$$\begin{aligned} \mathcal{L}((\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi) &= \mathcal{L}(\mathcal{D}\varphi_{(L_1, \dots, L_m)}^{-1}) = (\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi\varphi_{(L_1, \dots, L_m)}^{-1} \\ &= (L_1, \dots, L_m)\Phi = (\mathcal{L}(\mathcal{A}_1), \dots, \mathcal{L}(\mathcal{A}_m))\Phi, \end{aligned}$$

as required.

To see that  $\Psi$  is uniform, fix  $m$ -tuples of DFAs  $(\mathcal{A}_1, \dots, \mathcal{A}_m)$  and  $(\mathcal{B}_1, \dots, \mathcal{B}_m)$  such that for  $1 \leq j \leq m$ , the DFAs  $\mathcal{A}_j$  and  $\mathcal{B}_j$  have the same state configuration  $(Q_j, i_j, F_j)$ , the DFA  $\mathcal{A}_j$  has alphabet  $\Sigma$  and transition set  $T_{\mathcal{A}_j}$ , and the DFA  $\mathcal{B}_j$  has alphabet  $\Gamma$  and transition set  $T_{\mathcal{B}_j}$ . Write  $(\mathcal{A}_1, \dots, \mathcal{A}_m)\Psi = \mathcal{A} = (Q_{\mathcal{A}}, \Sigma, T_{\mathcal{A}}, i_{\mathcal{A}}, F_{\mathcal{A}})$  and  $(\mathcal{B}_1, \dots, \mathcal{B}_m)\Psi = \mathcal{B} = (Q_{\mathcal{B}}, \Gamma, T_{\mathcal{B}}, i_{\mathcal{B}}, F_{\mathcal{B}})$ . Let  $\mathcal{D}$  be a minimal DFA for  $(\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi$  used in the definition of  $\Psi$ . Then  $\mathcal{A}$  and  $\mathcal{B}$  are both inverse morphism DFAs constructed from  $\mathcal{D}$ , so they both have the same state configuration as  $\mathcal{D}$ . Write  $(Q, i, F)$  for this common state configuration.

It remains to show that whenever we have  $a \in \Sigma$  and  $b \in \Gamma$  such that  $(T_{\mathcal{A}_j})_a = (T_{\mathcal{B}_j})_b$  for  $1 \leq j \leq m$ , it follows that  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$ . Fix  $a \in \Sigma$  and  $b \in \Gamma$  with this property. Write  $\varphi_{\mathcal{A}}$  as shorthand for  $\varphi_{(\mathcal{L}(\mathcal{A}_1), \dots, \mathcal{L}(\mathcal{A}_m))}$ , and write  $\varphi_{\mathcal{B}}$  for  $\varphi_{(\mathcal{L}(\mathcal{B}_1), \dots, \mathcal{L}(\mathcal{B}_m))}$ . By definition, we have  $\mathcal{A} = \mathcal{D}\varphi_{\mathcal{A}}^{-1}$  and  $\mathcal{B} = \mathcal{D}\varphi_{\mathcal{B}}^{-1}$ . Let  $T_{\mathcal{D}}$  be the transition set of  $\mathcal{D}$ . Then for  $q \in Q$ , we have  $q(T_{\mathcal{A}})_a = q(T_{\mathcal{D}})_{a\varphi_{\mathcal{A}}}$  and  $q(T_{\mathcal{B}})_b = q(T_{\mathcal{D}})_{b\varphi_{\mathcal{B}}}$ . By the definition of the standard transformation tuple morphism, we have  $a\varphi_{\mathcal{A}} = ((T_{\mathcal{A}_1})_a, \dots, (T_{\mathcal{A}_m})_a)$  and  $b\varphi_{\mathcal{B}} = ((T_{\mathcal{B}_1})_b, \dots, (T_{\mathcal{B}_m})_b)$ . But we are assuming that  $(T_{\mathcal{A}_j})_a = (T_{\mathcal{B}_j})_b$  for  $1 \leq j \leq m$ , so in fact  $a\varphi_{\mathcal{A}} = b\varphi_{\mathcal{B}}$ . It then follows that  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$ , as required.

This proves that  $\Psi$  is uniform, and thus  $\Phi$  is uniform, since it is equivalent to a uniform DFA operation.  $\square$

## 5 The Main Theorem

The goal of this section is to prove that the OLPA approach works for all uniform operations. Thanks to Proposition 4 and its generalization in Proposition 5, this is not especially difficult.

First we consider unary operations. The following lemma formalizes a “weak” version of the OLPA approach for unary operations. The short proof contains all the essential ideas, but the expression it gives for the state complexity function is not practical, since it involves taking a maximum over all sets of size  $n$ . Obtaining a practical expression for the state complexity function is just a technical matter that we will deal with after this proof.

**Lemma 6.** *Let  $\Phi$  be a uniform unary regular operation. Let  $L$  be a regular language recognized by a DFA  $(Q, \Sigma, T, i, F)$ . Then  $\text{sc}(L\Phi) \leq \text{sc}(\mathcal{T}_Q(i, F)\Phi)$ . In particular, the state complexity of  $\Phi$  is given by the following function:*

$$n \mapsto \max\{\text{sc}(\mathcal{T}_Q(i, F)\Phi) : |Q| = n, i \in Q, F \subseteq Q\}.$$

*Proof.* Fix  $L$ , and recall that  $L = \mathcal{T}_Q(i, F)\varphi_L^{-1}$ , where  $\varphi_L$  is the standard transformation morphism of  $L$ . Since  $\Phi$  is uniform, we have  $L\Phi = \mathcal{T}_Q(i, F)\Phi\varphi_L^{-1}$  by Proposition 4. By Lemma 1, we have  $\text{sc}(L\Phi) \leq \text{sc}(\mathcal{T}_Q(i, F)\Phi)$  as required.  $\square$

Now, we show that to compute the state complexity function, it suffices to just consider the set  $\{1, \dots, n\}$  instead of all sets of size  $n$ . Furthermore, we may assume that  $i = 1$ , and that  $F$  is either  $\{1, \dots, k\}$  or  $\{n - k + 1, \dots, n\}$  for some  $k \leq n$ . Thus it suffices to just check  $2n$  OLPA witnesses. Write  $\mathcal{T}_n$  for  $\mathcal{T}_{\{1, \dots, n\}}$ , let  $F_{n,k,1} = \{1, \dots, k\}$ , and let  $F_{n,k,0} = \{n - k + 1, \dots, n\}$ .

**Theorem 7.** *Let  $\Phi$  be a uniform unary regular operation. Let  $L$  be a regular language recognized by a DFA  $(Q, \Sigma, T, i, F)$ . Then  $\text{sc}(L\Phi) \leq \text{sc}(\mathcal{T}_n(1, F_{n,k,j})\Phi)$ , where  $n = |Q|$ ,  $k = |F|$ , and  $j$  is defined to be 1 if  $i \in F$  and 0 if  $i \notin F$ . The state complexity of  $\Phi$  is the following function:*

$$n \mapsto \max\{\text{sc}(\mathcal{T}_n(1, F_{n,k,j})\Phi) : 0 \leq j \leq 1, 0 + j \leq k \leq n - 1 + j\}.$$

*Proof.* We know from Lemma 6 that  $\text{sc}(L\Phi) \leq \text{sc}(\mathcal{T}_Q(i, F)\Phi)$ . Let us prove that  $\text{sc}(\mathcal{T}_Q(i, F)\Phi) \leq \text{sc}(\mathcal{T}_n(1, F_{n,k,j})\Phi)$ , with  $n$ ,  $k$  and  $j$  defined as in the statement of the theorem.

By Lemma 1 and the uniformity of  $\Phi$ , it suffices to exhibit a morphism  $\varphi: \mathcal{T}_Q^* \rightarrow \mathcal{T}_n^*$  such that  $\mathcal{T}_Q(i, F) = \mathcal{T}_n(1, F_{n,k,j})\varphi^{-1}$ . To define  $\varphi$ , first we define a bijection  $\beta: Q \rightarrow \{1, \dots, n\}$ . We take  $\beta$  to be a bijection with the following properties:  $i\beta = 1$  and  $F\beta = F_{n,k,j}$ . The remaining elements of  $Q$  are mapped to the remaining elements of  $\{1, \dots, n\}$  arbitrarily. Note that this definition of  $\beta$  is only possible because of our choice of the parameter  $j$ . Indeed, we are mapping  $i\beta$  to 1, so if  $i \in F$  then we better have  $1 \in F_{n,k,j}$ ; but  $i \in F$  implies  $j = 1$  and thus  $F_{n,k,j} = \{1, \dots, k\}$ . On the other hand, if  $i \notin F$  then we better have  $1 \notin F_{n,k,j}$ , but in this case we have  $j = 0$  giving  $F_{n,k,j} = \{n - k + 1, \dots, n\}$ . Given this definition of  $\beta$ , for each  $t: Q \rightarrow Q$ , we define  $t\varphi$  be the transformation of  $\{1, \dots, n\}$  that sends  $m$  to  $m\beta^{-1}t\beta$ .

We show that  $w \in \mathcal{T}_Q(i, F)$  if and only if  $w \in \mathcal{T}_n(1, F_{n,k,j})\varphi^{-1}$ . Let  $w = t_1 \dots t_k$  for  $t_1, \dots, t_k \in \mathcal{T}_Q$ .

$$\begin{aligned} w \in \mathcal{T}_Q(i, F) &\iff iw \in F \iff 1\beta^{-1}w \in F \iff 1\beta^{-1}w\beta \in F\beta \\ &\iff 1\beta^{-1}w\beta \in F_{n,k,j} \iff 1\beta^{-1}t_1t_2 \dots t_k\beta \in F_{n,k,j} \\ &\iff 1\beta^{-1}t_1\beta\beta^{-1}t_2\beta \dots \beta^{-1}t_k\beta \in F_{n,k,j} \\ &\iff 1(t_1\varphi)(t_2\varphi) \dots (t_k\varphi) \in F_{n,k,j} \iff 1(w\varphi) \in F_{n,k,j} \\ &\iff w\varphi \in \mathcal{T}_n(1, F_{n,k,j}) \iff w \in \mathcal{T}_n(1, F_{n,k,j})\varphi^{-1}. \end{aligned}$$

Thus  $\mathcal{T}_Q(i, F) = \mathcal{T}_n(1, F_{n,k,j})\varphi^{-1}$ , as required. This completes the proof.  $\square$

We now consider uniform operations of arbitrary arity. The proof strategies in this case are much the same, except full transformation tuple languages are used as witnesses, rather than full transformation languages.

**Lemma 8.** *Let  $\Phi$  be a uniform  $m$ -ary regular operation. Let  $(L_1, \dots, L_m)$  be regular languages, where  $L_j$  is recognized by a DFA  $(Q_j, \Sigma, T_j, i_j, F_j)$ . Let  $\mathbb{T} = \mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m}$ . Then  $\text{sc}((L_1, \dots, L_m)\Phi) \leq \text{sc}((\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi)$ .*

*Proof.* Fix  $(L_1, \dots, L_m)$ , and recall from Proposition 3 that  $(L_1, \dots, L_m) = (\mathbb{T}_1(i_1, F_1)\varphi^{-1}, \dots, \mathbb{T}_m(i_m, F_m)\varphi^{-1})$ , where the morphism  $\varphi = \varphi_{(L_1, \dots, L_m)}$  is the standard transformation tuple morphism of  $(L_1, \dots, L_m)$ . Since  $\Phi$  is uniform, we have

$$(L_1, \dots, L_m)\Phi = (\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi\varphi^{-1},$$

by Proposition 5. Then by Lemma 1, we have

$$\text{sc}((L_1, \dots, L_m)\Phi) \leq \text{sc}((\mathbb{T}_1(i_1, F_1), \dots, \mathbb{T}_m(i_m, F_m))\Phi),$$

as required.  $\square$

As before, it suffices to only check a finite number of witnesses. Recall that we defined  $\mathcal{T}_n = \mathcal{T}_{\{1, \dots, n\}}$ , and for  $k \leq n$  we defined  $F_{n,k,1} = \{1, \dots, k\}$  and  $F_{n,k,0} = \{n - k + 1, \dots, n\}$ .

**Theorem 9 (The “Fundamental Theorem of the OLPA Approach”).**

Let  $\Phi$  be a uniform  $m$ -ary regular operation. Let  $(L_1, \dots, L_m)$  be regular languages, where  $L_j$  is recognized by a DFA  $(Q_j, \Sigma, T_j, i_j, F_j)$ . Let  $n_j = |Q_j|$  and let  $\mathbb{T} = \mathcal{T}_{n_1} \times \dots \mathcal{T}_{n_m}$ . Then we have

$$\text{sc}((L_1, \dots, L_m)\Phi) \leq \text{sc}((\mathbb{T}_1(1, F_{n_1,k_1,\ell_1}), \dots, \mathbb{T}_m(1, F_{n_m,k_m,\ell_m}))\Phi),$$

where  $k_j = |F_j|$ , and  $\ell_j$  is defined to be 1 if  $i_j \in F_j$  and 0 if  $i_j \notin F_j$ . The state complexity of  $\Phi$  is the function

$$(n_1, \dots, n_m) \mapsto \max \text{sc}((\mathbb{T}_1(1, F_{n_1,k_1,\ell_1}), \dots, \mathbb{T}_m(1, F_{n_m,k_m,\ell_m}))\Phi),$$

where the maximum is taken over all possible values in the following ranges:  $1 \leq j \leq m$ ,  $0 \leq \ell_j \leq 1$ , and  $0 + \ell_j \leq k_j \leq n_j - 1 + \ell_j$ .

To compute the worst-case state complexity of an  $m$ -ary operation for  $m$  input DFAs of sizes  $n_1$  through  $n_m$ , we use  $2(n_1 + \dots + n_m)$  different languages, each with an alphabet of size  $n_1^{n_1} \dots n_m^{n_m}$ . The number of input  $m$ -tuples that must be tested is  $2^m n_1 \dots n_m$ , since for the  $j$ -th component there are  $2n_j$  choices.

*Proof.* Define  $\mathbb{T}' = \mathcal{T}_{Q_1} \times \dots \times \mathcal{T}_{Q_m}$ . We know from Lemma 8 that

$$\text{sc}((L_1, \dots, L_m)\Phi) \leq \text{sc}((\mathbb{T}'_1(i_1, F_1), \dots, \mathbb{T}'_m(i_m, F_m))\Phi).$$

Let us prove the following:

$$\text{sc}((\mathbb{T}'_1(i_1, F_1), \dots, \mathbb{T}'_m(i_m, F_m))\Phi) \leq \text{sc}((\mathbb{T}_1(1, F_{n_1,k_1,\ell_1}), \dots, \mathbb{T}_m(1, F_{n_m,k_m,\ell_m}))\Phi),$$

where  $n_j, k_j, \ell_j$  for  $1 \leq j \leq m$  are defined as in the statement of the theorem.

By Lemma 1 and the uniformity of  $\Phi$ , it suffices to exhibit a morphism  $\varphi: (\mathbb{T}')^* \rightarrow \mathbb{T}^*$  such that  $\mathbb{T}'_j(i_j, F_j) = \mathbb{T}_j(1, F_{n_j,k_j,\ell_j})\varphi^{-1}$  for  $1 \leq j \leq m$ . To define  $\varphi$ , first we define bijections  $\beta_j: Q_j \rightarrow \{1, \dots, n_j\}$  for  $1 \leq j \leq m$ . As in the proof of Theorem 7, we take each  $\beta_j$  to be a bijection with the following properties:  $i_j\beta_j = 1$  and  $F_j\beta_j = F_{n_j,k_j,\ell_j}$ . The remaining elements of  $Q_j$  are mapped to the remaining elements of  $\{1, \dots, n_j\}$  arbitrarily. Then for each tuple  $(t_1, \dots, t_m) \in \mathbb{T}'$ , we define  $(t_1, \dots, t_m)\varphi$  to be the transformation tuple  $(\beta_1^{-1}t_1\beta_1, \dots, \beta_m^{-1}t_m\beta_m)$ , which lies in  $\mathbb{T}$ .

Now, for  $1 \leq j \leq m$ , we show that  $w \in \mathbb{T}'_j(i_j, F_j) \iff w\varphi \in \mathbb{T}_j(1, F_{n_j, k_j, \ell_j})$ . Let  $w = (t_{1,1}, \dots, t_{m,1}) \cdots (t_{1,k}, \dots, t_{m,k})$ , where each of these transformation tuples lies in  $\mathbb{T}'$ .

$$\begin{aligned} w \in \mathbb{T}'_j(i_j, F_j) &\iff i_j w \in F_j \iff i_j t_{j,1} t_{j,2} \cdots t_{j,k} \in F_j \\ &\iff 1\beta_j^{-1} t_{j,1} t_{j,2} \cdots t_{j,k} \beta_j \in F_{n_j, k_j, \ell_j} \\ &\iff 1\beta_j^{-1} t_{j,1} \beta_j \beta_j^{-1} t_{j,2} \beta_j \cdots \beta_j^{-1} t_{j,k} \beta_j \in F_{n_j, k_j, \ell_j} \\ &\iff 1w\varphi \in F_{n_j, k_j, \ell_j} \iff w\varphi \in \mathbb{T}_j(1, F_{n_j, k_j, \ell_j}). \end{aligned}$$

Thus  $\mathbb{T}'_j(i_j, F_j) = \mathbb{T}_j(1, F_{n_j, k_j, \ell_j})\varphi^{-1}$ , as required.  $\square$

## 6 Examples of Uniform and Non-Uniform Operations

In this section, we prove that a number of common operations (as well as a few more esoteric ones) are uniform, and that the class of uniform operations is closed under composition. We also give some examples of non-uniform operations.

### 6.1 Uniform Operations

First we consider a class of operations called *shuffles on trajectories* [11,17]. Operations in this class include shuffle, literal shuffle, balanced literal shuffle, insertion, balanced insertion, concatenation, and anti-concatenation [17, Remark 3.1]. We denote the shuffle of languages  $L$  and  $L'$  along the set of trajectories  $X \subseteq \{0,1\}^*$  by  $L \sqcup_X L'$ . The shuffle on trajectories  $L \sqcup_X L'$  is regular if and only if  $X$  is regular [17, Theorem 5.1]. For the definition of  $L \sqcup_X L'$ , see [17, Section 3]; for the following proof we only need to know the DFA construction.

**Proposition 10.** *For all regular languages  $X \subseteq \{0,1\}^*$ , the shuffle on trajectories operation  $(L, L') \mapsto L \sqcup_X L'$  is uniform.*

*Proof.* Following [17], we define a DFA operation  $\Psi$  equivalent to the shuffle on trajectories operation. Let  $\mathcal{D}_1 = (Q_1, \Sigma, T_1, i_1, F_1)$  and  $\mathcal{D}_2 = (Q_2, \Sigma, T_2, i_2, F_2)$  be arbitrary DFAs. Let  $\mathcal{D}_X = (Q_X, \{0,1\}, T_X, i_X, F_X)$  be a DFA for the set of trajectories  $X$ . We set  $(\mathcal{D}_1, \mathcal{D}_2)\Psi$  to be the determinization of the following FA  $\mathcal{D}$ . The FA  $\mathcal{D}$  has state set  $Q_1 \times Q_X \times Q_2$ , alphabet  $\Sigma$ , initial state  $(i_1, i_X, i_2)$ , final state set  $F_1 \times F_X \times F_2$ , and transition set  $T$  defined as follows: for each  $a \in \Sigma$ , we have

$$(q_1, q_X, q_2)T_a = \{(q_1(T_1)_a, q_X(T_X)_0, q_2), (q_1, q_X(T_X)_1, q_2(T_2)_a)\}.$$

It was proved in [17, Theorem 5.1] that  $\mathcal{L}((\mathcal{D}_1, \mathcal{D}_2)\Psi) = \mathcal{L}(\mathcal{D}_1) \sqcup_X \mathcal{L}(\mathcal{D}_2)$ .

Let  $(\mathcal{A}_1, \mathcal{A}_2)$  and  $(\mathcal{B}_1, \mathcal{B}_2)$  be pairs of DFAs such that for  $1 \leq j \leq 2$ , the DFAs  $\mathcal{A}_j$  and  $\mathcal{B}_j$  have the same state configuration  $(Q_j, i_j, F_j)$ , the DFA  $\mathcal{A}_j$  has alphabet  $\Sigma$  and transition set  $T_{\mathcal{A}_j}$ , and the DFA  $\mathcal{B}_j$  has alphabet  $\Gamma$  and transition set  $T_{\mathcal{B}_j}$ .



It is clear that the image DFAs  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)\Psi$  and  $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)\Psi$  will have the same state configuration. Additionally, by inspecting the definitions of the transition sets  $T_{\mathcal{A}}$  of  $\mathcal{A}$  and  $T_{\mathcal{B}}$  of  $\mathcal{B}$ , it is clear that if  $(T_{\mathcal{A}_j})_a = (T_{\mathcal{B}_j})_b$  for  $a \in \Sigma$ ,  $b \in \Gamma$  and  $1 \leq j \leq 2$ , then  $(T_{\mathcal{A}})_a = (T_{\mathcal{B}})_b$ . Indeed, let  $S \subseteq Q_1 \times Q_X \times Q_2$ . Then we have

$$\begin{aligned}
S(T_{\mathcal{A}})_a &= \bigcup_{(q_1, q_X, q_2) \in S} \{(q_1, q_X, q_2)\}(T_{\mathcal{A}})_a \\
&= \bigcup_{(q_1, q_X, q_2) \in S} \{(q_1(T_{\mathcal{A}_1})_a, q_X(T_X)_0, q_2), (q_1, q_X(T_X)_1, q_2(T_{\mathcal{A}_2})_a)\} \\
&= \bigcup_{(q_1, q_X, q_2) \in S} \{(q_1(T_{\mathcal{B}_1})_b, q_X(T_X)_0, q_2), (q_1, q_X(T_X)_1, q_2(T_{\mathcal{B}_2})_b)\} \\
&= \bigcup_{(q_1, q_X, q_2) \in S} \{(q_1, q_X, q_2)\}(T_{\mathcal{B}})_b = S(T_{\mathcal{B}})_b.
\end{aligned}$$

Thus  $\Psi$  is uniform, and it follows that the shuffle on trajectories operation is uniform.  $\square$

The above proof illustrates the fact that once one understands the definition of uniformity, it is often easy to determine whether an operation is uniform just by inspecting the DFA construction. There are no difficult ideas in this proof; it is just a statement of a DFA construction and a rudimentary calculation.

One can also use the language-theoretic characterization of uniformity to prove that operations are uniform. Typically, proofs using the language-theoretic characterization require somewhat more thought to write and read, but are shorter and have less of the “boilerplate” needed for DFA-based proofs. The rest of our uniformity proofs will use the language-theoretic characterization.

**Proposition 11.** *The reversal operation  $L \mapsto L^R$  is uniform.*

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L = K\varphi^{-1}$ . Since  $\varphi$  is 1-uniform, we have  $(w^R)\varphi = (w\varphi)^R$  for all  $w \in \Sigma^*$ . It follows that

$$\begin{aligned}
w \in L^R &\iff w^R \in L \iff w^R \in K\varphi^{-1} \iff (w^R)\varphi \in K \\
&\iff (w\varphi)^R \in K \iff w\varphi \in K^R \iff w \in K^R\varphi^{-1}.
\end{aligned}$$

Thus  $L^R = K^R\varphi^{-1}$ . Therefore, by Proposition 4, reversal is uniform.  $\square$

The *cyclic shift* operation [15] is defined by  $L^{\text{cyc}} = \{uv : vu \in L\}$ .

**Proposition 12.** *The cyclic shift operation  $L \mapsto L^{\text{cyc}}$  is uniform.*

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L = K\varphi^{-1}$ . We want to show that  $L^{\text{cyc}} = K^{\text{cyc}}\varphi^{-1}$ .

If  $w \in L^{\text{cyc}}$ , we can write  $w = uv$  for some  $u, v \in \Sigma^*$  such that  $vu \in L$ . Since  $L = K\varphi^{-1}$ , we have  $(vu)\varphi = (v\varphi)(u\varphi) \in K$ . Thus  $(u\varphi)(v\varphi) = w\varphi \in K^{\text{cyc}}$ , and it follows that  $L^{\text{cyc}} \subseteq K^{\text{cyc}}\varphi^{-1}$ .

If  $w \in K^{\text{cyc}}\varphi^{-1}$ , then  $w\varphi \in K^{\text{cyc}}$ . Thus we can write  $w\varphi = uv$  for some  $u, v \in \Sigma^*$  such that  $vu \in K$ . Since  $\varphi$  is 1-uniform,  $w$  has length  $|u| + |v|$ . Write  $w = xy$  where  $|x| = |u|$  and  $|y| = |v|$ . Then  $(yx)\varphi = (y\varphi)(x\varphi) = vu \in K$ . It follows that  $yx \in L$ , which implies  $xy = w \in L^{\text{cyc}}$ . Hence  $L^{\text{cyc}} = K^{\text{cyc}}\varphi^{-1}$ . By Proposition 4, cyclic shift is uniform.  $\square$

**Proposition 13.** *The star operation  $L \mapsto L^*$  is uniform.*

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L = K\varphi^{-1}$ . For a language  $M$ , let  $t(M)$  be the set of all finite-length tuples of elements of  $M$ , and let  $\psi_M: t(M) \rightarrow M^*$  be the map  $(w_1, \dots, w_n)\psi_M = w_1 \cdots w_n$  (the empty tuple is sent to  $\varepsilon$ ). We claim that  $w\psi_L^{-1} \neq \emptyset \iff (w\varphi)\psi_K^{-1} \neq \emptyset$ . Indeed, if  $(w_1, \dots, w_n) \in w\psi_L^{-1}$  then  $(w_1\varphi, \dots, w_n\varphi) \in (w\varphi)\psi_K^{-1}$ . Conversely, if we have  $(x_1, \dots, x_n) \in (w\varphi)\psi_K^{-1}$ , then  $w\varphi = x_1 \cdots x_n$ . Since  $\varphi$  is 1-uniform, we can write  $w = w_1 \cdots w_n$  with  $|w_j| = |x_j|$  and  $w_j\varphi = x_j$  for  $1 \leq j \leq n$ . Then since  $x_j = w_j\varphi \in K \implies w_j \in K\varphi^{-1} = L$ , we have  $(w_1, \dots, w_n) \in w\psi_L^{-1}$  as required. It follows that:

$$w \in L^* \iff w\psi_L^{-1} \neq \emptyset \iff (w\varphi)\psi_K^{-1} \neq \emptyset \iff w\varphi \in K^* \iff w \in K^*\varphi^{-1}.$$

Thus  $L^* = K^*\varphi^{-1}$ . Therefore, by Proposition 4, star is uniform.  $\square$

An  $m$ -ary *boolean function* is a function  $\beta: \{0, 1\}^m \rightarrow \{0, 1\}$ . Each  $m$ -ary boolean function defines a corresponding  $m$ -ary *boolean operation* on languages over  $\Sigma^*$ , as follows. For  $L \subseteq \Sigma^*$ , let  $\chi_L: \Sigma^* \rightarrow \{0, 1\}$  be the *characteristic function* of  $L$ : if  $w \in L$  then  $w\chi_L = 1$ , and if  $w \notin L$  then  $w\chi_L = 0$ . Then we define

$$(L_1, \dots, L_m)\beta = \{w \in \Sigma^* : (w\chi_{L_1}, \dots, w\chi_{L_m})\beta = 1\}.$$

Examples of commonly used boolean operations on languages include union and intersection ( $m$ -ary for  $m \geq 2$ ), difference and symmetric difference (binary), and complement (unary).

**Proposition 14.** *Boolean operations on languages are uniform.*

*Proof.* Let  $\beta$  be an  $m$ -ary boolean operation on languages. Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L_j = K_j\varphi^{-1}$  for  $1 \leq j \leq m$ . We have

$$\begin{aligned} w \in (L_1, \dots, L_m)\beta &\iff (w\chi_{L_1}, \dots, w\chi_{L_m})\beta = 1 \\ &\iff (w\chi_{K_1\varphi^{-1}}, \dots, w\chi_{K_m\varphi^{-1}})\beta = 1 \\ &\iff (w\varphi\chi_{K_1}, \dots, w\varphi\chi_{K_m})\beta = 1 \\ &\iff w\varphi \in (K_1, \dots, K_m)\beta \iff w \in (K_1, \dots, K_m)\beta\varphi^{-1}. \end{aligned}$$

Therefore, by Proposition 4,  $\beta$  is uniform.  $\square$

We have seen that binary concatenation is uniform, since concatenation belongs to the class of shuffles on trajectories. Next we give a direct proof that  $m$ -ary concatenation is uniform.

**Proposition 15.** *The  $m$ -ary concatenation operation  $(L_1, \dots, L_m) \mapsto L_1 \cdots L_m$  is uniform.*

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L_j = K_j \varphi^{-1}$  for  $1 \leq j \leq m$ . Define  $\psi_L: L_1 \times \cdots \times L_m \rightarrow L_1 \cdots L_m$  by  $(w_1, \dots, w_m) \psi_L = w_1 \cdots w_m$  and similarly define  $\psi_K: K_1 \times \cdots \times K_m \rightarrow K_1 \cdots K_m$ . Using similar arguments to the proof of Proposition 13, we can show that  $w \psi_L^{-1} \neq \emptyset \iff (w\varphi) \psi_K^{-1} \neq \emptyset$ . Thus:

$$w \in L_1 \cdots L_m \iff w \psi_L^{-1} \neq \emptyset \iff (w\varphi) \psi_K^{-1} \neq \emptyset \iff w\varphi \in K_1 \cdots K_m.$$

Therefore, by Proposition 4,  $m$ -ary concatenation is uniform.  $\square$

Next, we show that the class of uniform operations is closed under composition. It is easy to see that this holds for unary uniform operations: if  $\Phi$  and  $\Phi'$  are uniform and  $L = K\varphi^{-1}$  for a 1-uniform morphism  $\varphi$ , then  $L\Phi = K\Phi\varphi^{-1}$  and subsequently  $(L\Phi)\Phi' = (K\Phi)\Phi'\varphi^{-1}$ . The general case is not much harder; the only difficulty is in dealing with the notation.

**Proposition 16.** *Let  $\Phi$  be an  $m$ -ary uniform operation, and let  $\Phi_1, \dots, \Phi_m$  be uniform operations where  $\Phi_j$  has arity  $n_j$ . Set  $N_j = n_1 + \cdots + n_j$  for  $1 \leq j \leq m$ , and consider the operation of arity  $N_m$  that maps  $(L_1, \dots, L_{N_m})$  to*

$$((L_1, \dots, L_{N_1})\Phi_1, (L_{N_1+1}, \dots, L_{N_2})\Phi_2, \dots, (L_{N_{m-1}+1}, \dots, L_{N_m})\Phi_m)\Phi.$$

*This operation, which we denote by  $\Phi'$ , is uniform.*

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L_j = K_j \varphi^{-1}$  for  $1 \leq j \leq N_m$ . By Proposition 4, it suffices to show that  $(L_1, \dots, L_{N_m})\Phi' = (K_1, \dots, K_{N_m})\Phi' \varphi^{-1}$ . Let  $N_0 = 0$ ; then by the uniformity of  $\Phi_j$ , for  $1 \leq j \leq m$  we have

$$(L_{N_{j-1}+1}, \dots, L_{N_j})\Phi_j = (K_{N_{j-1}+1}, \dots, K_{N_j})\Phi_j \varphi^{-1}.$$

Set  $M_j = (L_{N_{j-1}+1}, \dots, L_{N_j})\Phi_j$  and  $M'_j = (K_{N_{j-1}+1}, \dots, K_{N_j})\Phi_j$ . Then  $M_j = M'_j \varphi^{-1}$  for  $1 \leq j \leq m$ . By the uniformity of  $\Phi$ , we have

$$(L_1, \dots, L_{N_m})\Phi' = (M_1, \dots, M_m)\Phi = (M'_1, \dots, M'_m)\Phi \varphi^{-1} = (K_1, \dots, K_{N_m})\Phi' \varphi^{-1},$$

as required.  $\square$

This shows that all “combined operations” formed by compositions of the uniform operations we have seen so far are also uniform.

The following “substitution lemma” can also be used to construct new uniform operations from known ones.

**Lemma 17.** *Let  $\Phi$  be a  $k$ -ary operation. Fix  $m \geq 1$  and  $i_1, \dots, i_k \in \{1, \dots, m\}$ . Then the operation  $\Phi'$  defined by  $(L_1, \dots, L_m) \mapsto (L_{i_1}, \dots, L_{i_k})\Phi$  is uniform.*

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L_j = K_j\varphi^{-1}$  for  $1 \leq j \leq m$ . Then by the uniformity of  $\Phi$ , we have

$$(L_1, \dots, L_m)\Phi' = (L_{i_1}, \dots, L_{i_k})\Phi = (K_{i_1}, \dots, K_{i_k})\Phi\varphi^{-1} = (K_1, \dots, K_m)\Phi'\varphi^{-1}.$$

Therefore, by Proposition 4, the operation  $\Phi'$  is uniform.  $\square$

As an example, we show that the power operation is uniform. Define  $L^0 = \{\varepsilon\}$  and for  $n \geq 1$ , set  $L^n = L^{n-1}L$ .

**Proposition 18.** *For  $n \geq 0$ , the power operation  $L \mapsto L^n$  is uniform.*

*Proof.* For  $n \geq 2$ , in Lemma 17, let  $\Phi$  be the  $n$ -ary concatenation operation, set  $m = 1$  and set  $i_1, \dots, i_n = 1$ . For  $n = 1$ , it is immediate that  $L \mapsto L$  is uniform. For  $n = 0$ , see Proposition 19 below.  $\square$

Another example is the anti-concatenation operation  $(L, L') \mapsto L'L$ . This belongs to the class of shuffles on trajectories, so we already know that it is uniform, but an alternate proof could be given using Lemma 17: let  $\Phi$  be binary concatenation, set  $m = 2$ , set  $i_1 = 2$  and set  $i_2 = 1$ .

Next we consider some operations which depend only on the alphabet of the input languages. These are not interesting from a state complexity perspective, but can be used to construct interesting combined operations.

**Proposition 19.** *Let  $S \subseteq \mathbb{N}$ . The operation  $(L_1, \dots, L_m) \mapsto \bigcup_{n \in S} \Sigma^n$ , where  $\Sigma$  is the common alphabet of the inputs, is uniform. In particular, the following operations are uniform for all arities  $m$ :*

1.  $(L_1, \dots, L_m) \mapsto \emptyset$ .
2.  $(L_1, \dots, L_m) \mapsto \{\varepsilon\}$ .
3.  $(L_1, \dots, L_m) \mapsto \Sigma^*$ .
4.  $(L_1, \dots, L_m) \mapsto \Sigma^+$ .

*Proof.* Fix a 1-uniform morphism  $\varphi: \Sigma^* \rightarrow \Gamma^*$  and suppose  $L_j = K_j\varphi^{-1}$  for  $1 \leq j \leq m$ . We claim that  $\Sigma^n = \Gamma^n\varphi^{-1}$  for all  $n \geq 0$ . Indeed, take a word  $w \in \Sigma^n$ ; then  $w\varphi$  is in  $\Gamma^n$  by 1-uniformity, and so  $w \in \Gamma^n\varphi^{-1}$ . Conversely, if  $w \in \Gamma^n\varphi^{-1} = \{x \in \Sigma^* : x\varphi \in \Gamma^n\}$  then certainly  $w \in \Sigma^n$ . It follows that:

$$\bigcup_{n \in S} \Sigma^n = \bigcup_{n \in S} \Gamma^n\varphi^{-1} = \left( \bigcup_{n \in S} \Gamma^n \right) \varphi^{-1}.$$

By Proposition 4, operations of this type are uniform.  $\square$

For a language  $L$  over  $\Sigma$ , the *right ideal* generated by  $L$  is  $\Sigma^*L$ , the *left ideal* generated by  $L$  is  $L\Sigma^*$ , the *two-sided ideal* generated by  $L$  is  $\Sigma^*L\Sigma^*$ , and the *all-sided ideal* generated by  $L$  is  $L\sqcup\Sigma^*$ , where  $\sqcup$  is (ordinary) shuffle. By combining our earlier results, we can show that the operations which map  $L$  to one of the ideals it generates are uniform. For example, let  $\Phi$  be ternary concatenation, let  $\Phi_1$  and  $\Phi_3$  be  $L \mapsto \Sigma^*$ , and let  $\Phi_2$  be  $L \mapsto L$ . Then the operation

$(L_1, L_2, L_3) \mapsto (L_1\Phi_1, L_2\Phi_2, L_3\Phi_3)\Phi$  is uniform by closure under composition. Then by substitution,  $L \mapsto (L\Phi_1, L\Phi_2, L\Phi_3)\Phi = \Sigma^*L\Sigma^*$  is uniform.

In summary, we have proved that the following operations are uniform: reversal, cyclic shift, star, power,  $m$ -ary concatenation,  $m$ -ary boolean operations (including union, intersection, difference, symmetric difference and complement), shuffles on trajectories (including shuffle, literal shuffle, balanced literal shuffle, insertion, balanced insertion, and anti-concatenation), and the “alphabet-dependent” operations of Proposition 19. We also proved that the class of uniform operations is closed under composition, meaning that *all combined operations* formed by composing the aforementioned operations are uniform, such as “star-complement-star” or “star of union”. Additionally, we proved a substitution lemma that gives another way to construct new uniform operations from old, such as the “ideal generated by” operations.

## 6.2 Non-Uniform Operations

First we remark that *constant operations*, which output a fixed language regardless of the input, are not in general uniform. One issue is that our theoretical framework assumes that all regular operations are *alphabet-preserving*, so we cannot even define true “constant operations” that take arbitrary regular languages as inputs; we must restrict the inputs to have the same alphabet as the constant output language. The more fundamental problem is that constant operations need not behave uniformly with respect to transformations. For example, let  $\Psi$  be a constant DFA operation, and suppose that in DFA  $\mathcal{A}$ , the letter  $a$  induces transformation  $t$ , and in DFA  $\mathcal{B}$ , the letter  $b$  also induces transformation  $t$ . If  $\Psi$  is uniform, then the transformation induced by  $a$  in  $\mathcal{A}\Psi$  will be the same as the transformation induced by  $b$  in  $\mathcal{B}\Psi$ . But the constant operation  $\Psi$  could produce a DFA in which  $a$  and  $b$  induce different transformations, violating uniformity. The only way to ensure uniformity is if  $\Psi$  produces a DFA in which every letter induces the same transformation; if we enforce this condition, we essentially obtain the alphabet-dependent operations of Proposition 19.

Our first example of an interesting non-uniform operation is the following:

$$\frac{1}{2}L = \{x \in \Sigma^* : xy \in L, |x| = |y|\}.$$

This “half” operation is an example of a *proportional removal*; the state complexity of proportional removals was studied by Domaratzki [7]. We could prove that this operation is not uniform directly from the definition, or using the language-theoretic characterization, but instead we will show something even stronger: the OLPA approach does not maximize the state complexity of this operation.

If the OLPA approach worked for this operation, then by Lemma 6, the state complexity of the operation would be maximized by a language of the form  $\frac{1}{2}T_Q(i, F)$  for some state configuration  $(Q, i, F)$ . However, it is not hard to see that if  $F$  is non-empty, then  $\frac{1}{2}T_Q(i, F)$  is either  $T_Q^*$  or  $T_Q^* \setminus \{\varepsilon\}$ , depending on whether  $i \in F$ . Indeed, let  $w$  be a non-empty word in  $T_Q^*$ . We have  $iw = q$

for some  $q \in Q$ . Let  $t$  be a transformation that sends  $q$  into  $F$ . Then  $w t \text{id}_Q^{|w|-1}$  maps  $i$  into  $F$ , and so this word is in the language  $T_Q(i, F)$ . But  $w$  is exactly half the length of this word, so  $w \in \frac{1}{2}T_Q(i, F)$ . This means that  $\text{sc}(\frac{1}{2}T_Q(i, F)) \leq 2$ ; but Domaratzki [7] shows that there are languages  $L$  of state complexity  $n$  such that  $\text{sc}(\frac{1}{2}L) = n$ . A similar argument shows that OLPA approach fails for many other proportional removal operations as well, although we have not tried to characterize the proportional removals for which the approach fails.

Next we consider *deletions along trajectories* [8,12], a class of operations which includes left quotient, right quotient, deletion, scattered deletion, bi-polar deletion, and  $k$ -deletion [8, p. 296]. We will show that the left quotient operation and the deletion operation are not uniform. We have not investigated uniformity for other deletions along trajectories.

The case of left quotient is interesting, because the OLPA approach actually works for this operation despite its non-uniformity. The left quotient of  $L$  by  $L'$  is  $L' \setminus L = \{x \in \Sigma^* : wx \in L \text{ for some } w \in L'\}$ . This operation satisfies a weak version of the language-theoretic characterization of uniformity:

*For all 1-uniform morphisms  $\varphi: \Sigma^* \rightarrow \Gamma^*$ , if  $L_j = K_j \varphi^{-1}$  and  $L_j \neq \emptyset$  for  $1 \leq j \leq 2$ , then  $(L_2 \setminus L_1) \Phi = (K_2 \setminus K_1) \Phi \varphi^{-1}$ .*

Because empty languages are excluded here, the OLPA approach would fail if maximizing the state complexity in certain cases required the use of empty languages. But this does not happen for left quotient.

To see that left quotient is not uniform, let  $\Sigma = \{a, b\}$  and define  $\varphi: \Sigma^* \rightarrow \Sigma^*$  by  $a\varphi = b\varphi = b$ . Then define  $K_1 = \{ab\}$ ,  $K_2 = \{a\}$ ,  $L_1 = K_1 \varphi^{-1}$ , and  $L_2 = K_2 \varphi^{-1}$ . If left quotient was uniform, we would have  $L_2 \setminus L_1 = (K_2 \setminus K_1) \varphi^{-1}$ . But  $L_1 = L_2 = \emptyset$ , and so  $L_2 \setminus L_1 = \emptyset$ . Meanwhile,  $(K_2 \setminus K_1) \varphi^{-1} = (\{a\} \setminus \{ab\}) \varphi^{-1} = \{b\} \varphi^{-1} = \{a, b\}$ .

The *deletion* of  $L'$  from  $L$  is  $L \rightsquigarrow L' = \{xz \in \Sigma^* : xyz \in L \text{ for some } y \in L'\}$ . We will show that the OLPA approach fails for this operation.

If the OLPA approach worked, the state complexity would be maximized by some pair of OLPA witnesses. Consider the language  $\mathbb{T}_1(i_1, F_1) \rightsquigarrow \mathbb{T}_2(i_2, F_2)$  where  $\mathbb{T} = \mathcal{T}_{Q_1} \times \mathcal{T}_{Q_2}$  for some finite sets  $Q_1$  and  $Q_2$ . We claim that  $\mathbb{T}_1(i_1, F_1) \rightsquigarrow \mathbb{T}_2(i_2, F_2) = \mathbb{T}^*$ , which has state complexity one.

Indeed, fix a word  $w \in \mathbb{T}^*$ . Write  $w = (t_{1,1}, t_{2,1}) \cdots (t_{1,k}, t_{2,k})$ . Let  $w_1 = t_{1,1} \cdots t_{1,k}$ , set  $q_1 = i_1 w_1$ , and choose a transformation  $t_1: Q_1 \rightarrow Q_1$  that sends  $q_1$  into  $F_1$ . Next, choose a transformation  $t_2: Q_2 \rightarrow Q_2$  that sends  $i_2$  into  $F_2$ . Then  $i_1 w_1 t_1 \in F_1$ , so  $w(t_1, t_2) \in \mathbb{T}_1(i_1, F_1)$ . However,  $i_2 t_2 \in F_2$ , so  $(t_1, t_2) \in \mathbb{T}_2(i_2, F_2)$ . It follows  $w \in \mathbb{T}_1(i_1, F_1) \rightsquigarrow \mathbb{T}_2(i_2, F_2)$  since it can be obtained by deleting a word in  $\mathbb{T}_2(i_2, F_2)$  from a word in  $\mathbb{T}_1(i_1, F_1)$ .

This shows that using OLPA witnesses for deletion only produces languages of state complexity one. However, Han, Ko and Salomaa [12] proved that if  $L$  has state complexity  $n$ , then  $n2^{n-1}$  is a tight upper bound on the state complexity of  $L \rightsquigarrow L'$ . Hence the state complexity of deletion is not maximized by the OLPA approach.

It is interesting that our main examples of operations for which the OLPA approach fails involve the idea of “deletion” in some sense.

## 7 Proofs using the OLPA Approach

In the introduction, we used the OLPA approach to give a simple proof of the worst-case state complexity of reversal. We give two additional examples of proofs using the OLPA approach in this section. First we consider the star operation.

**Proposition 20.** *Let  $L$  be a regular language recognized by  $\mathcal{D} = (Q, \Sigma, T, i, F)$ , where  $|Q| = n$  and  $|F| = k$ . Suppose  $1 \leq |F| \leq n - 1$ . If  $F = \{i\}$  then  $L = L^*$ , and otherwise we have the following tight upper bounds on  $\text{sc}(L^*)$ :*

$$\text{sc}(L^*) \leq \begin{cases} (2^{n-k} - 1) + (2^{n-1} - 2^{n-k-1}) + 1, & \text{if } i \notin F; \\ (2^{n-k} - 1) + (2^{n-1}), & \text{if } i \in F \text{ and } |F| \geq 2. \end{cases}$$

*Proof.* By Theorem 7, it suffices to just compute the state complexity of  $L^*$  for  $L \in \{\mathcal{T}_n(1, F_{n,k,j}) : 0 \leq j \leq 1, 0 + j \leq k \leq n - 1 + j\}$ . Given  $\mathcal{D}$ , an FA for  $L^*$  is  $\mathcal{A} = (Q \cup \{s\}, \Sigma, T', s, F \cup \{s\})$  where  $T' = T \cup \{(f, a, iT_a) : f \in F \cup \{s\}\}$ . It is easy to see that if  $F = \{i\}$ , then  $\mathcal{A}$  recognizes  $L$ . Henceforth assume  $F \neq \{i\}$ .

We show each non-empty set  $S \subseteq Q$  is reachable by induction on  $|S|$ . From  $\{s\}$  we reach  $\{q\}$  for each  $q \in Q$  by a transformation that sends  $i$  to  $q$ . Now suppose  $|S| \geq 2$  and smaller sets are reachable. Choose a set  $X$  of size  $|S| - 1$  which contains a final state but does not contain  $i$ ; this is possible since  $F \neq \{i\}$ . Fix  $q \in S$  and choose a transformation  $t$  that maps  $X$  onto  $S \setminus \{q\}$  and  $i$  to  $q$ ; then  $t$  sends  $X$  to  $S$ . Thus all non-empty subsets of  $Q$  are reachable.

We show that sets in the following collection are pairwise distinguishable:

$$\{S \subseteq Q : S \neq \emptyset \text{ and } S \cap F = \emptyset\} \cup \{S \subseteq Q : i \in S \text{ and } S \cap F \neq \emptyset\}.$$

If a non-empty set  $S$  is not in this collection, it contains a final state but does not contain  $i$ , and the set  $S$  is indistinguishable from  $S \cup \{i\}$ . To distinguish distinct sets  $S$  and  $X$  in this collection, choose an element  $q$  which appears (without loss of generality) in  $S$  but not in  $X$ , and apply a transformation  $t$  which maps  $q$  into  $F$  and  $Q \setminus \{q\}$  into  $Q \setminus F$ . Note also that if  $i \notin F$  then  $\{s\}$  is distinguishable from all states in this collection, but if  $i \in F$  then  $\{s\}$  and  $\{i\}$  are indistinguishable. Elementary counting arguments then yield the stated bounds.  $\square$

For our other example, we consider boolean operations, defined in Section 6 (see the discussion before Proposition 14). In this case the proof is complicated, but the result is very general, and we believe it would be considerably more difficult to prove without the OLPA approach or a similar construction.

It is a bit tricky to state a tight upper bound for the worst-case state complexity of an arbitrary  $m$ -ary boolean operation, because the operation's result might not depend on all of its arguments. For example, if the inputs to a binary boolean operation have state complexity  $n_1$  and  $n_2$  respectively, the worst-case state complexity can be 1,  $n_1$ ,  $n_2$ , or  $n_1 n_2$ , depending on which arguments (if any) are relevant to the result.

To state our upper bound, we introduce some notation. Given an  $m$ -ary boolean function  $\beta: \{0,1\}^m \rightarrow \{0,1\}$ , we define functions  $\beta_j: \mathbb{N} \rightarrow \mathbb{N}$  for  $1 \leq j \leq m$  as follows. If there exist two binary  $m$ -tuples  $(b_1, \dots, b_m)$  and  $(b'_1, \dots, b'_m)$  which differ only in the  $j$ -th bit (that is,  $b_j \neq b'_j$  and  $b_i = b'_i$  for all  $i \neq j$ ) such that  $(b_1, \dots, b_m)\beta \neq (b'_1, \dots, b'_m)\beta$ , then we define  $n\beta_j = n$  for all  $n \in \mathbb{N}$ . Otherwise, it must be the case that for all binary  $m$ -tuples, flipping the  $j$ -th bit does not change the result of  $\beta$ ; in this case we define  $n\beta_j = 1$  for all  $n \in \mathbb{N}$ . If  $\beta_j$  is the identity map, we say that  $\beta$  *depends on argument*  $j$ , and if  $\beta_j$  is the constant function sending everything to 1, we say that  $\beta$  *does not depend on argument*  $j$ .

**Proposition 21.** *Let  $\beta$  be an  $m$ -ary boolean operation. Let  $(L_1, \dots, L_m)$  be regular languages, where  $L_j$  is recognized by  $(Q_j, \Sigma, T_j, i_j, F_j)$  for  $1 \leq j \leq m$ . Set  $n_j = |Q_j|$ . Then  $\text{sc}((L_1, \dots, L_m)\beta) \leq (n_1\beta_1) \cdots (n_m\beta_m)$  and this bound is tight.*

*Proof.* Recall the usual direct product construction for boolean operations:  $\mathcal{D} = (Q, \Sigma, T, (i_1, \dots, i_m), F)$  where the state set is  $Q = Q_1 \times \dots \times Q_m$ , the transition set is  $T = \{((q_1, \dots, q_m), a, (q_1(T_1)_a, \dots, q_m(T_m)_a)) : (q_1, \dots, q_m) \in Q, a \in \Sigma\}$  and the final state set is  $F = \{(q_1, \dots, q_m) \in Q : (q_1\chi_{F_1}, \dots, q_m\chi_{F_m}) = 1\}$ . This construction gives an upper bound of  $n_1 \cdots n_m$ . To get a tighter bound, we must consider distinguishability. Consider the following set of states:

$$\{(q_1, \dots, q_m) \in Q : q_j = i_j \text{ whenever } \beta \text{ does not depend on argument } j\}.$$

There are precisely  $(n_1\beta_1) \cdots (n_m\beta_m)$  states in this set, and we claim that every state lying outside this set is indistinguishable from a state within the set. To see this, fix a state  $(q_1, \dots, q_m)$  which is not in the above set. Then there exists  $j$  such that  $q_j \neq i_j$  and  $\beta$  does not depend on argument  $j$ . We claim state  $(q_1, \dots, q_m)$  is indistinguishable from  $(q_1, \dots, q_{j-1}, i_j, q_{j+1}, \dots, q_m)$ . Indeed, if two states differ only in component  $j$ , and  $\beta$  does not depend on argument  $j$ , then either both states are final or both states are non-final. Also, starting from a pair of states which differ only in component  $j$ , we can only reach other pairs that differ only in component  $j$ . Thus there is no way to distinguish these states.

Now we show that the upper bound is tight. Our witnesses will be OLPA witnesses  $L_j = \mathbb{T}_j(1, F_j)$ , where  $\mathbb{T} = \mathcal{T}_{n_1} \times \dots \times \mathcal{T}_{n_m}$  and  $F_j = \{1\}$  for  $1 \leq j \leq m$  (it does not really matter what we choose for  $F_j$ , as long as for  $n_j \geq 2$  it is a proper non-empty subset of  $\{1, \dots, n_j\}$ ).

The initial state of the direct product DFA is  $(1, 1, \dots, 1)$ . For each state  $(q_1, \dots, q_m) \in Q$ , let  $t_j$  be a transformation that sends 1 to  $q_j$ . Then the letter  $(t_1, \dots, t_m) \in \mathbb{T}$  sends the initial state to  $(q_1, \dots, q_m)$ ; thus all states are reachable.

Next we show that all pairs of states in  $\{(q_1\beta_1, \dots, q_m\beta_m) : (q_1, \dots, q_m) \in Q\}$ , which has size  $(n_1\beta_1) \cdots (n_m\beta_m)$ , are distinguishable. Suppose we have two distinct states  $(q_1\beta_1, \dots, q_m\beta_m)$  and  $(q'_1\beta_1, \dots, q'_m\beta_m)$ . Since they are distinct, they must differ in some component  $j$ , and for this  $j$  we must have  $q_j\beta_j = q_j$  and  $q'_j\beta_j = q'_j$ . Hence there exist two binary  $m$ -tuples  $(b_1, \dots, b_m)$  and  $(b'_1, \dots, b'_m)$  which differ only in component  $j$  such that  $(b_1, \dots, b_m)\beta \neq (b'_1, \dots, b'_m)\beta$ . Assume without loss of generality that  $(b_1, \dots, b_m)\beta = 1$  and  $(b'_1, \dots, b'_m)\beta = 0$ .



Now, choose a tuple of transformations  $(t_1, \dots, t_m) \in \mathbb{T}$  as follows:

- Choose  $t_j$  so that  $q_j t_j \chi_{F_j} = b_j$  and  $q'_j t_j \chi_{F_j} = b'_j$ .
- For  $i \neq j$ , if  $\beta$  depends on argument  $i$ , choose  $t_i$  so that  $(q_i t_i) \chi_{F_i} = b_i = b'_i$ . If  $\beta$  does not depend on argument  $i$ , let  $t_i$  be the identity map.

Now, we claim that  $(q_1 \beta_1, \dots, q_m \beta_m)(t_1, \dots, t_m)$  is a final state. To determine whether the reached state  $(q_1 \beta_1 t_1, \dots, q_m \beta_m t_m)$  is final, we look at the binary  $m$ -tuple  $(q_1 \beta_1 t_1 \chi_{F_1}, \dots, q_m \beta_m t_m \chi_{F_m})$ . If  $\beta$  depends on argument  $i$  (including the case  $i = j$ ) then we have  $q_i \beta_i t_i \chi_{F_i} = b_i$ . If  $\beta$  does not depend on argument  $i$ , then  $q_i \beta_i = 1$ , transformation  $t_i$  is the identity map, and  $F_i = \{1\}$ , so we have  $q_i \beta_i t_i \chi_{F_i} = 1$ . Thus  $(q_1 \beta_1 t_1 \chi_{F_1}, \dots, q_m \beta_m t_m \chi_{F_m}) = (\widetilde{b}_1, \dots, \widetilde{b}_m)$ , where  $\widetilde{b}_i$  is  $b_i$  if  $\beta$  depends on argument  $i$ , and  $\widetilde{b}_i$  is 1 otherwise. Now, we know that if  $\beta$  does not depend on argument  $i$ , then flipping the  $i$ -th bit in a binary  $m$ -tuple will not change the result of applying  $\beta$  to that  $m$ -tuple. So by flipping bits if necessary, we see that

$$(\widetilde{b}_1, \dots, \widetilde{b}_m) \beta = (b_1, \dots, b_m) \beta = 1.$$

Hence  $(q_1 \beta_1, \dots, q_m \beta_m)(t_1, \dots, t_m)$  is a final state.

On the other hand, consider the state  $(q'_1 \beta_1, \dots, q'_m \beta_m)(t_1, \dots, t_m)$ . For this state we have  $(q'_1 \beta_1 t_1 \chi_{F_1}, \dots, q'_m \beta_m t_m \chi_{F_m}) = (\widetilde{b}'_1, \dots, \widetilde{b}'_m)$ , where  $\widetilde{b}'_i$  is  $b'_i$  if  $\beta$  depends on argument  $i$ , and  $\widetilde{b}'_i$  is 1 otherwise. Thus by the same bit-flipping argument, we have

$$(\widetilde{b}'_1, \dots, \widetilde{b}'_m) \beta = (b'_1, \dots, b'_m) \beta = 0.$$

Thus this state is not final, and we have distinguished the two states.  $\square$

## 8 Conclusions

The “one letter per action” (OLPA) approach gives an easy way to find witnesses that maximize the state complexity of many regular operations, at the expense of requiring large alphabets. We defined a class of “uniform” regular operations for which the OLPA approach provably works. This class contains many common operations and is also closed under composition. We hope this paper will spark interest in and further study of the OLPA approach.

We list a few open questions that we find interesting.

- To what extent does the OLPA approach work in *subclasses* of the regular languages? It seems it will work for some “nice” subclasses but not for others.
- Can the OLPA approach be generalized to other state complexity measures, like incomplete state complexity [16] or unrestricted state complexity [2]?
- Can we find a larger class than the class of uniform operations for which the OLPA approach provably works, without sacrificing the nice property of closure under composition?

- How do we maximize the state complexity of proportional removals like  $\frac{1}{2}L$ ? Domaratzki’s work [7] does not completely solve this problem. If we find a way to maximize their state complexity, can it be generalized to other operations for which the OLPA approach fails?

**Acknowledgements.** I thank Jason Bell, Janusz Brzozowski, and the referees of the DLT 2018 version of this paper for their careful proofreading and helpful comments. I thank Lukas Fleischer for pointing me to some important references I overlooked, which allowed me to give a much more complete history of the ideas presented in this paper.

## References

1. Birget, J.C.: Intersection and union of regular languages and state complexity. *Inform. Process. Lett.* 43(4), 185–190 (1992)
2. Brzozowski, J.: Unrestricted state complexity of binary operations on regular languages. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) DCFS 2016. LNCS, vol. 9777, pp. 60–72. Springer (2016)
3. Brzozowski, J.A.: In search of most complex regular languages. *Int. J. Found. Comput. Sc.* 24(06), 691–708 (2013)
4. Brzozowski, J.A., Jirásková, G., Liu, B., Rajasekaran, A., Szykuła, M.: On the state complexity of the shuffle of regular languages. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) DCFS 2016. pp. 73–86. Springer (2016)
5. Caron, P., Hamel-De le court, E., Luque, J.G., Patrou, B.: New tools for state complexity. CoRR abs/1807.00663 (2018), <https://arxiv.org/abs/1807.00663>
6. Cho, D.J., Han, Y.S., Ko, S.K., Salomaa, K.: State complexity of inversion operations. *Theoret. Comput. Sci.* 610, 2–12 (2016)
7. Domaratzki, M.: State complexity of proportional removals. *J. Autom. Lang. Comb.* 7(4), 455–468 (2002)
8. Domaratzki, M.: Deletion along trajectories. *Theoret. Comput. Sci.* 320(2), 293–313 (2004)
9. Domaratzki, M., Kisman, D., Shallit, J.: On the number of distinct languages accepted by finite automata with  $n$  states. *J. Autom. Lang. Comb.* 7(4), 469–486 (2002)
10. Domaratzki, M., Okhotin, A.: State complexity of power. *Theoret. Comput. Sci.* 410(24), 2377–2392 (2009)
11. Domaratzki, M., Salomaa, K.: State complexity of shuffle on trajectories. *J. Autom. Lang. Comb.* 9, 217–232 (2004)
12. Han, Y.S., Ko, S.K., Salomaa, K.: State complexity of deletion and bipolar deletion. *Acta Informatica* 53(1), 67–85 (2016)
13. Jirásková, G., Shallit, J.: The state complexity of star-complement-star. In: Yen, H.C., Ibarra, O.H. (eds.) DLT 2012. pp. 380–391. Springer (2012)
14. Jirásková, G., Šebej, J.: Reversal of binary regular languages. *Theoret. Comput. Sci.* 449, 85–92 (2012)
15. Jirásková, Galina, Okhotin, Alexander: State complexity of cyclic shift. *RAIRO-Theor. Inf. Appl.* 42(2), 335–360 (2008)
16. Maia, E., Moreira, N., Reis, R.: Incomplete operational transition complexity of regular languages. *Inform. and Comput.* 244, 1–22 (2015)

17. Mateescu, A., Rozenberg, G., Salomaa, A.: Shuffle on trajectories: Syntactic constraints. *Theoret. Comput. Sci.* 197(1), 1–56 (1998)
18. Ravikumar, B.: Some applications of a technique of sakoda and sipser. *SIGACT News* 21(4), 73–77 (1990)
19. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two way finite automata. In: *STOC 1978*. pp. 275–286. ACM (1978)
20. Salomaa, A., Wood, D., Yu, S.: On the state complexity of reversals of regular languages. *Theoret. Comput. Sci.* 320(2), 315–329 (2004)
21. Yan, Q.: Lower bounds for complementation of  $\omega$ -automata via the full automata technique. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) *ICALP 2006*. pp. 589–600. Springer (2006)
22. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. *Theor. Comput. Sci.* 125(2), 315–328 (1994)