State Complexity of Partial Word Finite Automata

Martin Kutrib(✉) and Matthias Wendlandt

Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
{kutrib,matthias.wendlandt}@informatik.uni-giessen.de

Abstract. Partial word finite automata are deterministic finite automata that may have state transitions on a special symbol $\diamond$ which represents an unknown symbol or a hole in the word. Together with a subset of the input alphabet that gives the symbols which may be substituted for the $\diamond$, a partial word finite automaton represents a regular language. However, this substitution implies a certain form of limited nondeterminism in the computations when the $\diamond$-transitions are replaced by ordinary transitions. In this paper we first reconsider the problem to prove the minimality of partial word finite automata and present a method to utilize minimal NFAs with certain properties for this purpose. Then we study the operational state complexity of partial word finite automata with respect to Boolean operations. It turns out that the upper and lower bounds for all these operations are exponential. Moreover, we establish a state complexity hierarchy on the number of productive $\diamond$-transitions that may appear in partial word finite automata. The levels of the hierarchy are separated by exponential state costs.

1 Introduction

Partial words are strings where certain positions are not specified. These positions are often called holes or don't cares and printed by a diamond symbol $\diamond$. Apart from theoretical reasons, the basic motivation for studying this mechanism comes from the study of biological operations in connection with DNA strands. In particular, DNA sequencing is a biological process to determine the base sequence of a given DNA strand. To this end, the two DNA strands are separated and cut into small pieces. Afterwards the small sequences are copied, multiplied, and then detected. Subsequently, the complete strand has to be derived out of the small pieces. This assembling can be done by aligning the fragments with the help of gaps (holes) which leads to the definition of partial words. The first time the idea of words with don't cares has been investigated goes back to [7], where they were considered in connection with string matching. The notation *partial word* has firstly been defined in [2].

Partial words were mainly investigated in connection with combinatorics on words. A survey can be found in [4]. An interesting motivation in theory for this model is that ordinary languages can be compressed by the usage of holes. Consider for example the language L over the ternary alphabet $\Sigma = \{a, b, c\}$,

Y.-S. Han and S.-K. Ko (Eds.): DCFS 2021, LNCS 13037, pp. 113–124, 2021.
https://doi.org/10.1007/978-3-030-93489-7_10

$L = \{aaa, aba, aca\}$. It can be compressed by using a hole into $L' = \{a \diamond a\}$. Simply by replacing the diamond by a, b, or c the original language L can be achieved.

In 2012, partial words were studied in connection with families of formal languages [6]. In particular, a regular language is represented by the image of a partial language under a substitution that only replaces the hole symbols. In connection with DFAs it turned out that the usage of holes can be somehow seen as a limited nondeterminism, since it allows to define DFAs with outgoing edges that are labeled with ordinary symbols and additionally with a diamond. If some of the ordinary symbols may be substituted for the hole symbol as well, the corresponding state allows a nondeterministic choice with respect to the target language. While in the original definition of partial words a hole represents a placeholder for all letters of the underlying alphabet, in investigations in connection with language families the substitution of the hole symbol can be an arbitrary subset of the alphabet (see for example [1,6,10]).

The applications of defining language families by partial words via partial word finite automata have also been investigated from a complexity point of view. Concerning the descriptional complexity, in [1] it has been shown that the state complexity for a DFA that simulates a partial word DFA is exponential in general. Moreover also the state complexity of the simulation of an NFA by a partial word DFA may become exponential. Concerning the computational complexity, different problems as, for example, minimization have been studied for partial word automata [5,10].

The main aim of this paper is to extend the investigations on the state complexity of partial word automata. In connection with lower bounds on the number of states necessary for an automaton to accept a given language, the problem arises to prove the minimality of a given automaton. In Sect. 3 we discuss this problem by referring to known results from the literature and provide methods to prove the minimality of partial word DFAs by utilizing minimal NFAs with certain properties. Section 4 considers the operational state complexity for Boolean operations. It turns out that upper and lower bounds are exponential. In the last section we consider the impact of the number of productive $\diamond$-transitions in a partial word finite automaton, where a transition is called productive, if it does not lead to the rejecting sink state. It comes out that even the reduction of one of these transitions may lead to an exponential state explosion, which leads to a state complexity hierarchy dependent on the number of $\diamond$-transitions.

2 Preliminaries

We denote the non-negative integers $\{0, 1, 2, \dots\}$ by $\mathbb{N}$. Let Σ^* denote the set of all words over the finite alphabet Σ. A subset $L \subseteq \Sigma^*$ is said to be a *formal language* over Σ. We write $\overline{L}$ for the *complement* of L with respect to Σ, that is for $\Sigma^* \setminus L$. The *empty word* is denoted by λ and the *reversal* of a word w by w^R. For the *length* of w we write $|w|$. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*.

Setting $\Sigma_\diamond = \Sigma \cup \{\diamond\}$, where $\diamond \notin \Sigma$ represents *undefined positions* or *holes*, a *partial word* over Σ is a sequence of symbols from $\Sigma_\diamond$. Denoting the set of all partial words over Σ by $\Sigma_\diamond^*$, a *partial language* over Σ is a subset of $\Sigma_\diamond^*$. Partial languages can be transformed to (ordinary) languages by using $\diamond$-substitutions over Σ. A $\diamond$-substitution $\sigma\colon \Sigma_\diamond^* \to 2^{\Sigma^*}$ satisfies $\sigma(a) = \{a\}$, for all $a \in \Sigma$, $\sigma(\diamond) \subseteq \Sigma$, and $\sigma(uv) = \sigma(u)\sigma(v)$, for $u, v \in \Sigma_\diamond^*$. As a result, σ is fully defined by $\sigma(\diamond)$, for example, if $\sigma(\diamond) = \{a, b\}$ and $L = \{\diamond b, \diamond c\}$ then $\sigma(L) = \{ab, bb, ac, bc\}$. So, applying σ to a partial language $L \subseteq \Sigma_\diamond^*$ results in a (ordinary) language $\sigma(L) \subseteq \Sigma^*$.

A *nondeterministic finite automaton* (NFA) is a system $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, where Q is the finite set of *internal states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\delta\colon Q \times \Sigma \to 2^Q$ is the *transition function*. In the forthcoming, we sometimes refer to δ as a subset of $Q \times \Sigma \times Q$. A finite automaton M is *deterministic* (DFA) if and only if $|\delta(q, a)| = 1$, for all $q \in Q$ and $a \in \Sigma$. In this case, we simply write $\delta(q, a) = q'$ for $\delta(q, a) = \{q'\}$ assuming that the transition function is a total mapping $\delta\colon Q \times \Sigma \to Q$. Note that here any DFA is complete, that is, the transition function is total, whereas it may be a partial function for NFAs in the sense that the transition function of nondeterministic machines may map to the empty set. A finite automaton is said to be *minimal* if there is no finite automaton of the same type with fewer states, accepting the same language. Note that a rejecting sink state is counted for DFAs, since they are always complete, whereas it is not counted for NFAs, since their transition function may map to the empty set.

Generally speaking, a language L can be represented by a partial language L' together with a $\diamond$-substitution σ such that $\sigma(L') = L$. In particular, for regular languages, from the descriptional complexity point of view it is an interesting question to what extent there are regular languages L' such that the minimal DFA accepting L' has less states than the minimal DFA accepting L? In order to distinguish between finite automata accepting (ordinary) languages from those accepting partial languages, we refer to the latter as *partial word deterministic finite automata* ($\diamond$-DFA). Thus, $\diamond$-DFAs treat the hole symbol $\diamond$ as an ordinary input letter.

The number of states of the (complete) minimal DFA accepting a regular language L is denoted by $\min_{DFA}(L)$. Similarly, $\min_{NFA}(L)$ denotes the minimal number of states necessary for some NFA to accept L. For partial languages, we write $\min_{\diamond\text{-}DFA}(L)$ to denote the minimal number of states of a $\diamond$-DFA accepting a language L' such that there exists a $\diamond$-substitution σ with $\sigma(L') = L$.

3 Basic Constructions

In connection with lower bounds on the number of states necessary for an automaton to accept a given language, the problem arises to prove the minimality of a given automaton. While a couple of techniques exist to prove the minimality of DFAs, only a few techniques exist for NFAs. The situation is much

worse for $\diamond$-DFAs. Clearly, a $\diamond$-DFA can be seen as a DFA over the alphabet $\Sigma_\diamond$. But, in general, the minimization of a $\diamond$-DFA M changes the language that it represents, that is, $\sigma(L(M))$. It has been shown in [10] that the problem to find a minimal $\diamond$-DFA M' (together with a $\diamond$-substitution) for a given regular language is PSPACE-complete. The problem has been studied in more detail in [5], where algorithms are given for the construction of minimal partial languages, associated with some $\diamond$-substitution, as well as approximation algorithms for the construction of minimal $\diamond$-DFAs. However, for particular languages that witness certain lower bounds, their minimality has to be proved almost from scratch. Here we continue with some observations that can nevertheless be applied in lower bound proofs.

First, we briefly recall the so-called (extended) *fooling set* technique (see, for example, [3,8,12]) that is widely used for proving lower bounds on the number of states necessary for an NFA to accept a given language.

Theorem 1. *Let $L \subseteq \Sigma^*$ be a regular language and suppose there exists a set of pairs $P = \{ (x_i, y_i) \mid 1 \leq i \leq n \}$ such that (1) $x_i y_i \in L$, for $1 \leq i \leq n$, and (2) $i \neq j$ implies $x_i y_j \notin L$ or $x_j y_i \notin L$, for $1 \leq i, j \leq n$. Then any nondeterministic finite automaton accepting L has at least n states. Here P is called an (extended)* fooling set *for L.*

Let M be a $\diamond$-DFA $= \langle Q, \Sigma_\diamond, \delta, q_0, F \rangle$ and σ be an associated $\diamond$-substitution. Then a minimal DFA M' that accepts the language $\sigma(L(M))$ can be constructed as follows. First, modify M to the NFA $\hat{M} = \langle Q, \Sigma, \hat{\delta}, q_0, F \rangle$ by replacing any transition $\delta(p, \diamond) = q$ by the transitions $\{ \hat{\delta}(p, a) = q \mid a \in \sigma(\diamond) \}$ and keeping all other transitions from δ. Then determinize $\hat{M}$ and minimize the outcome. We call M' constructed in this way the *canonical* DFA for M and σ. This construction is presented as Algorithm 1 in [1].

The intermediate NFA in the construction exhibits the limited nondeterminism provided by $\diamond$-DFAs. In fact, for each state of the NFA, there are at most two outgoing transitions for each input symbol. This is a valuable hint for the seek for suitable witness automata. For example, it is known that $2^n - 1$ is a tight bound on the number of states for a DFA that accepts the language $\sigma(L(M))$ of an n-state $\diamond$-DFA M with associated $\diamond$-substitution [1]. In order to find further witnesses for the lower bound it is sufficient to look for complete NFAs having (i) the required form of limited nondeterminism for just one input symbol and (ii) causing the maximal state blow-up of $2^n - 1$ when determinized. An example is depicted in Fig. 1.

In such an NFA M the nondeterminism can be removed by replacing one of the two nondeterministic outgoing transitions by a transition on $\diamond$, respectively, and setting $\sigma(\diamond) = \{x\}$, where x is the sole input symbol for which the nondeterminism occurs. Since M is complete, for all states of the resulting automaton on which a $\diamond$-transition is defined, transitions on all other input symbols are defined as well. So, in order to make the resulting automaton complete, it is sufficient to add a $\diamond$-transition to the states for which no $\diamond$-transition is defined so far. This can safely be done by copying the transition on x. The transition

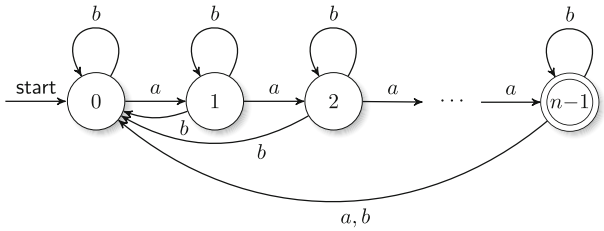

Fig. 1. A complete n-state NFA whose minimal equivalent DFA has $2^n - 1$ states.

on x must exist, since M is complete. Clearly, the resulting automaton M' is a $\diamond$-DFA with $\sigma(L(M')) = L(M)$ (see Fig. 2 for a possible $\diamond$-DFA obtained from the NFA of Fig. 1). Since $\min_{NFA}(L) \leq \min \diamond\text{-DFA}(L)$ [6] and M' has the same number of states as M has, the $\diamond$-DFA M' is minimal, that is, even for any other $\diamond$-substitution no smaller equivalent $\diamond$-DFA exists.

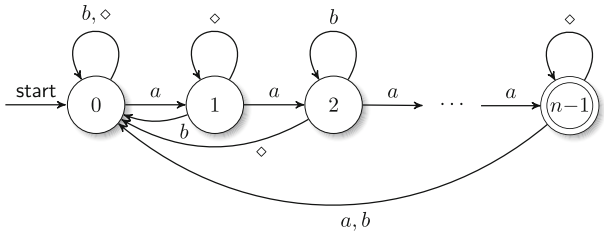

Fig. 2. A minimal $\diamond$-DFA obtained from the NFA depicted in Fig. 1, $\sigma(\diamond) = \{b\}$.

The example above dealt with the maximal state blow-up for "determinization". However, the method to prove the minimality of $\diamond$-DFAs by utilizing minimal NFAs with certain properties can be extended.

Lemma 1. *Let $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ be a possibly incomplete DFA, $S \subseteq \Sigma$ be a fixed subset of input symbols, $P \subseteq Q$, and $\alpha\colon Q \to Q$ be a total mapping. Moreover, let $\hat{M} = \langle Q, \Sigma, \hat{\delta}, q_0, F\rangle$ be an NFA obtained from M by adding the transitions $\{\hat{\delta}(p, a) = \alpha(p) \mid p \in P, a \in S\}$ to δ. Then $\min_{\diamond\text{-}DFA}(L(\hat{M})) \leq |Q|$ if M is complete and $P = Q$, and $\min_{\diamond\text{-}DFA}(L(\hat{M})) \leq |Q| + 1$ otherwise.*

Proof. We set $\sigma(\diamond) = S$ and construct a $\diamond$-DFA $M' = \langle Q, \Sigma_\diamond, \delta', q_0, F\rangle$ from the given NFA $\hat{M}$. To this end, for any state $p \in P$, a set of transitions $\{\hat{\delta}(p, a) = \alpha(p) \mid a \in S\}$ is replaced by the transition $\hat{\delta}(p, \diamond) = \alpha(p)$. By the construction of $\hat{M}$ from the DFA M it follows that M' is deterministic. If M is complete and $P = Q$, that is, for all states there is an outgoing $\diamond$-transition in M', then M' is complete. Otherwise, it is completed by adding missing transitions to a new rejecting sink state. This gives the transition function δ'. Since the input alphabet of M' is $\Sigma_\diamond$, it is a $\diamond$-DFA. Moreover, the canonical DFA

for M' and σ is equivalent to $\hat{M}$. Since apart from a possible new sink state the state set is the same for M' and $\hat{M}$, we conclude $\min_{\diamond\text{-}DFA}(L(\hat{M})) \leq |Q|$ if M is complete and $P = Q$. Otherwise, we have $\min_{\diamond\text{-}DFA}(L(\hat{M})) \leq |Q| + 1$. □

Let $L = \{aa, aaa, aaaa, aca, aaca, baca, baa, baaa\} \subset \{a, b, c\}^*$ be a finite language. It has been shown in [5] that any $\diamond$-DFA accepting L has at least seven states if $\sigma(\diamond) = \{a, b\}$, and at least eight states if $\sigma(\diamond) = \{a, c\}$. In order to show that any minimal NFA accepting L has five states, we apply Theorem 1 by providing the set $P = \{(\lambda, a^4), (b, a^3), (ba, a^2), (bac, a), (baca, \lambda)\}$ whose fooling set property for L is easily verified. A minimal NFA M accepting L is shown in Fig. 3.

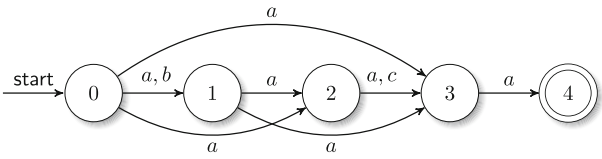

Fig. 3. A minimal NFA accepting a finite language.

This NFA M cannot serve as witness for the constructions from above because there are three transitions on input symbol a defined for state 0. Moreover, let α be the mapping of Lemma 1. Then, it can map state 0 to state 1 or to state 2 or to state 3 to remove the nondeterminism. However, in any case the nondeterminism is not removed entirely, when the transition $\delta(0, a) = \alpha(0)$ is deleted. It is not hard to show that *any* minimal NFA accepting L must have three a-transitions from the initial state. So, changing to a possibly different but equivalent minimal NFA does not help. Nevertheless, we can utilize M to show the minimality of a $\diamond$-DFA accepting the finite language L as follows.

By way of contradiction, assume that there is a 6-state $\diamond$-DFA M' and a $\diamond$-substitution σ such that $\sigma(L(M')) = L$. Since M' is complete and, for example, any input beginning with symbol c has to be rejected, M' has a rejecting sink state. We remove this sink state and all transitions to it, and obtain an equivalent incomplete 5-state $\diamond$-DFA M''. Next, we construct an NFA $\hat{M}$ from M'' as in the construction of the canonical DFA. That is, $\hat{M}$ is obtained from M'' by replacing any transition $\delta(p, \diamond) = q$ from M'' by the transitions $\{\, \hat{\delta}(p, a) = q \mid a \in \sigma(\diamond) \,\}$ and keeping all other transitions from δ. So, $\hat{M}$ has five states and is minimal. In particular, it has at most two transitions on input symbol a from the initial state. This is a contradiction, since any minimal NFA accepting L must have three a-transitions from the initial state. We conclude that, for *any* $\diamond$-substitution, a minimal $\diamond$-DFA accepting L has at least seven states.

In order to construct such a minimal $\diamond$-DFA with $\sigma(\diamond) = \{a\}$, we can resolve the nondeterminism by replacing two a-transitions from the initial state by a single a-transition to a new state $2,3$ which, in turn, has the outgoing transitions of the states 2 and 3. The newly introduced nondeterminism for this state can be removed by a $\diamond$-transition as depicted in Fig. 4.

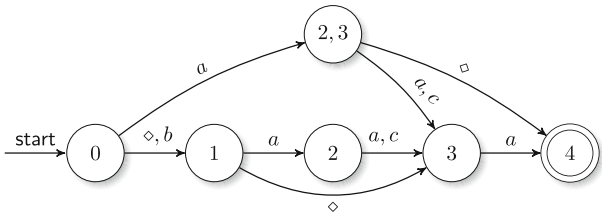

Fig. 4. A minimal $\diamond$-DFA accepting a finite language, where $\sigma(\diamond) = \{a\}$. The rejecting sink state is not depicted.

4 Operational State Complexity

Let $\circ$ be a fixed operation on languages that preserves regularity. Then the $\circ$-language operation problem for $\diamond$-DFAs is defined as follows:

- Given an n-state $\diamond$-DFA M_1 with $\diamond$-substitution σ_1 and an m-state $\diamond$-DFA M_2 with $\diamond$-substitution σ_2.
- How many states are sufficient and necessary in the worst case (in terms of n and m) for a $\diamond$-DFA M_3 with some $\diamond$-substitution σ_3 such that

$$\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \circ \sigma_2(L(M_2))?$$

Obviously, this problem generalizes to unary language operations like, for example, complementation or reversal.

We first consider the operation of complementation and show an upper bound and a lower bound that is tight up to a constant factor. The result reveals that complementation is an expensive operation from the state complexity point of view.

Proposition 1. *Let $n \geq 1$ be an integer and M_1 be an n-state $\diamond$-DFA with $\diamond$-substitution σ_1. Then $2^n - 1$ states are sufficient for a $\diamond$-DFA M_2 with some $\diamond$-substitution σ_2 such that $\sigma_2(L(M_2))$ is the complement of $\sigma_1(L(M_1))$. Therefore, we have $\min_{\diamond\text{-}DFA}(\overline{L}) \leq 2^{\min_{\diamond\text{-}DFA}(L)} - 1$, for all regular languages L.*

Theorem 2. *Let $n > 2$ be an integer. There exists a minimal n-state $\diamond$-DFA M_1 with $\diamond$-substitution σ_1 such that any $\diamond$-DFA M_2 with any $\diamond$-substitution σ_2, where $\sigma_2(L(M_2))$ is the complement of $\sigma_1(L(M_1))$, has at least 2^{n-3} states. Therefore, we have $\min_{\diamond\text{-}DFA}(\overline{L}) \leq 2^{\min_{\diamond\text{-}DFA}(L)-3}$, for infinitely many regular languages L.*

Proof. We are going to utilize Lemma 1 to construct minimal witness automata. To this end, consider the incomplete DFA M depicted in Fig. 5. We set $S = \{a\}$,

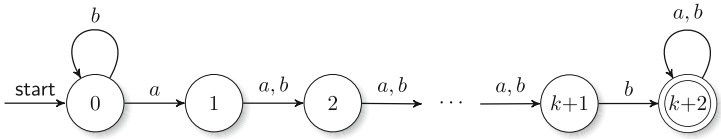

Fig. 5. An incomplete DFA.

$P = \{0\}$, and α to be the identity on the state set. After adding the required transitions to M, we obtain the NFA $\hat{M}$ depicted in Fig. 6.

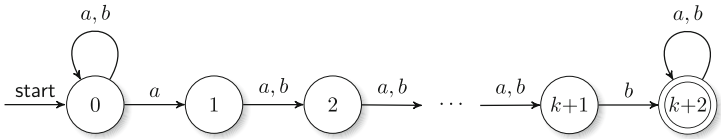

Fig. 6. The NFA obtained from the DFA of Fig. 5.

So, for $k \geq 0$, we consider the witness languages $L_k = \{a, b\}^* a \{a, b\}^k b \{a, b\}^*$. Now, by Lemma 1, we have $\min_{\diamond\text{-}DFA}(L_k) \leq k + 4$. On the other hand, it is not hard to see that the NFA of Fig. 6 is minimal. Therefore, by $\min_{NFA}(L_k) \leq \min_{\diamond\text{-}DFA}(L_k)$ we derive $k + 3 \leq \min_{\diamond\text{-}DFA}(L_k) \leq k + 4$. Since a minimal $\diamond$-DFA M' with $\diamond$-substitution σ' such that $\sigma'(L(M') = L_k$ is complete and, thus, has a rejecting sink state which the NFA of Fig. 6 does not have, we conclude $\min_{\diamond\text{-}DFA}(L_k) = k + 4$.

Essentially, in order to accept the complement of L_k an NFA has to verify that the input has no substring $a\{a, b\}^k b$. Therefore, after reading a symbol a the NFA must be able to remember the next k input symbols. Altogether this needs 2^{k+1} states. In fact, it has been shown in [11] that any NFA that accepts the complement of L_k needs at least 2^{k+1} states. Again, by $\min_{NFA}(\overline{L_k}) \leq \min_{\diamond\text{-}DFA}(\overline{L_k})$, we derive that any $\diamond$-DFA M_2 with any $\diamond$-substitution σ_2 where $\sigma_2(L(M_2)) = \overline{L_k}$ has at least 2^{k+1} states. Setting $n = k + 4$ shows the theorem. □

We continue with Boolean operations. In general, neither the union nor the intersection of partial languages gives a partial language whose substitution is the union or intersection of the substitutions of the given partial languages. So a simple cross-product construction does not help. The idea for the union is to take a $\diamond$-DFA for one of the given partial languages and the canonical DFA for the other one, and build their cross-product automaton to obtain a $\diamond$-DFA for the upper bound of the state costs. However, for the intersection, the idea does not apply. The reason is that a $\diamond$ in the input that can be substituted by at least two different symbols a_1 and a_2, must be treated by the canonical DFA as if the input were a_1 or a_2 and *both* symbols lead to accepting computations.

So, currently the best general upper bound for the intersection is the trivial one obtained by building the cross-product automaton of two canonical DFAs. In particular, let $m, n \geq 1$ be two integers, M_1 be an m-state $\diamond$-DFA with $\diamond$-substitution σ_1, and M_2 be an n-state $\diamond$-DFA with $\diamond$-substitution σ_2. Then $(2^m-1)\cdot(2^n-1)$ states are sufficient for a $\diamond$-DFA M_3 with some $\diamond$-substitution σ_3 such that $\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \cap \sigma_2(L(M_2))$. In fact, M_3 is a DFA.

For the special case that one of the two involved $\diamond$-substitutions is a singleton, a much better upper bound can be shown, which turns out to be tight in the order of magnitude.

Theorem 3. *Let $m, n \geq 1$ be two integers, M_1 be an m-state $\diamond$-DFA with $\diamond$-substitution σ_1, where $|\sigma_1(\diamond)| = 1$, and M_2 be an n-state $\diamond$-DFA with $\diamond$-substitution σ_2. Then $m \cdot (2^n - 1)$ states are sufficient for a $\diamond$-DFA M_3 with some $\diamond$-substitution σ_3 such that $\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \cap \sigma_2(L(M_2))$.*

Proof. To construct M_3, we take the $\diamond$-DFA M_1 with σ_1 as it is. Then we build the canonical DFA $M_2' = \langle Q, \Sigma, \delta, q_0, F\rangle$ for M_2 and σ_2. Let $\sigma_1(\diamond) = \{a\}$. We add a $\diamond$-transition to each state of M_2' by copying the a-transition. More precisely, for each state $q \in Q$, we define additionally $\delta(q, \diamond) = \delta(q, a)$. Finally, we construct the cross-product automaton from M_1 and M_2', call it M_3, and set $\sigma_3(\diamond) = \{a\}$.

Now let $w \in \sigma_1(L(M_1)) \cap \sigma_2(L(M_2))$. Then there is a word $w' \in \sigma_1^{-1}(w)$ accepted by M_1. Moreover, w is accepted by the canonical DFA for M_2 and σ_2. Since $\sigma_1(\diamond) = \{a\}$ and M_2' is this canonical DFA extended by a $\diamond$-transition in parallel to every a-transition, we derive that w' is accepted by M_2' as well. So, w' is accepted by M_3 and, thus, $w = \sigma_3(w') \in \sigma_3(L(M_3))$.

Conversely, let $w \in \sigma_3(L(M_3))$. Then there is a word $w' \in \sigma_3^{-1}(w)$ accepted by M_3. Therefore, w' is accepted by M_1 and by M_2'. Since $\sigma_1 = \sigma_3$, we have $w = \sigma_3(w') \in \sigma_1(L(M_1))$. Furthermore, by construction, $\sigma_2(L(M_2)) = \sigma_3(L(M_2'))$ and, hence, $w = \sigma_3(w') \in \sigma_2(L(M_2))$. We conclude $w \in \sigma_1(L(M_1)) \cap \sigma_2(L(M_2))$ and altogether have derived $\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \cap \sigma_2(L(M_2))$.

For the construction of M_3 as cross-product automaton of M_1 and M_2', a number of states that is the product of the number of states of M_1 and M_2', that is $m \cdot (2^n - 1)$, is sufficient. □

The proofs of the lower bounds are more involved, in a sense that the minimality of a $\diamond$-DFA accepting the intersection or union has to be shown.

Theorem 4. *Let $m \geq n \geq 1$ be two positive integers. There exist a $2m$-state $\diamond$-DFA M_1 with $\diamond$-substitution σ_1, and an n-state $\diamond$-DFA M_2 with $\diamond$-substitution σ_2, such that any $\diamond$-DFA M_3 with any $\diamond$-substitution σ_3 where $\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \cap \sigma_2(L(M_2))$ has at least $(m+1)\cdot(2^n-1)$ states. Therefore, we have*

$$\min_{\diamond\text{-}DFA}(L_1 \cap L_2) \geq (\min_{\diamond\text{-}DFA}(L_1)/2 + 1) \cdot 2^{\min_{\diamond\text{-}DFA}(L_2)} - 1,$$

for infinitely many regular languages L_1 and infinitely many regular languages L_2.

The last Boolean operation we are looking at is the union. As mentioned above, the idea for the upper bound is to take a $\diamond$-DFA for one of the given partial languages and the canonical DFA for the other one, and build their cross-product automaton.

Theorem 5. *Let $m \geq n \geq 1$ be two integers, M_1 be an m-state $\diamond$-DFA with $\diamond$-substitution σ_1, and M_2 be an n-state $\diamond$-DFA with $\diamond$-substitution σ_2. Then $m \cdot (2^n - 1)$ states are sufficient for a $\diamond$-DFA M_3 with some $\diamond$-substitution σ_3 such that $\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \cup \sigma_2(L(M_2))$.*

A lower bound for the union is shown in the next theorem. While it is exponential, it does not match the upper bound, since it consists of the sum of the number of states of the larger given automaton and two to the power of the number of states of the smaller given automaton. The upper bound was given by their product.

Theorem 6. *Let $m \geq n \geq 0$ be two positive integers. There exist a $(m+1)$-state $\diamond$-DFA M_1 with $\diamond$-substitution σ_1, and an $(n+1)$-state $\diamond$-DFA M_2 with $\diamond$-substitution σ_2, such that any $\diamond$-DFA M_3 with any $\diamond$-substitution σ_3 where $\sigma_3(L(M_3)) = \sigma_1(L(M_1)) \cup \sigma_2(L(M_2))$ has at least $m + 2^n$ states. Therefore, we have $\min_{\diamond\text{-}DFA}(L_1 \cup L_2) \geq (\min_{\diamond\text{-}DFA}(L_1) - 1) + 2^{\min_{\diamond\text{-}DFA}(L_2)-1}$, for infinitely many regular languages L_1 and infinitely many regular languages L_2.*

5 Hierarchy of $\diamond$-Transitions

Here we turn to considering the number of productive $\diamond$-transitions in a $\diamond$-DFA. Here a transition is called productive, if it does not lead to the rejecting sink state. By the tight bound of $2^n - 1$ states for the $\diamond$-DFA to DFA conversion, the state costs for removing all $\diamond$-transitions are already known. But this raises the question for the state costs when only some of the productive $\diamond$-transitions are removed. In other words, we consider the following (k_1, k_2)-$\diamond$-transition problem:

- Let $k_1 > k_2 \geq 0$ be two integers.
- Given an n-state $\diamond$-DFA M_1 with $\diamond$-substitution σ_1 having at most k_1 productive $\diamond$-transitions.
- How many states are sufficient and necessary in the worst case (in terms of n) for a $\diamond$-DFA M_2 with some $\diamond$-substitution σ_2 having at most k_2 productive $\diamond$-transitions such that $\sigma_2(L(M_2)) = \sigma_1(L(M_1))$?

Corollary 1. *For any $k_1 > 0$, the upper bound of the $(k_1, 0)$-$\diamond$-transition problem is $2^n - 1$.*

Next, we generalize the problem and derive exponential lower bounds. In particular, the lower bound for the $(k_1, k_1 - 1)$-$\diamond$-transition problem turns out to be exponential in the order of magnitude. Moreover, for every further productive $\diamond$-transition that is removed, an exponential number of states is additionally necessary in the worst case.

Theorem 7. *Let $k_1 > k_2 \geq 0$ be two constant integers. Then, for each $\ell \geq 2$, there exist a $(5k_1 + k_1\ell - 1)$-state $\diamond$-DFA M_1 with $\diamond$-substitution σ_1 having k_1 productive $\diamond$-transitions, such that any $\diamond$-DFA M_2 with any $\diamond$-substitution σ_2 having at most k_2 productive $\diamond$-transitions and $\sigma_2(L(M_2)) = \sigma_1(L(M_1))$ has at least $2k_1 + k_2(\ell + 1) + (k_1 - k_2)2^\ell - 1$ states.*

Proof. First, we construct a witness automaton. To this end, let $\ell \geq 2$ be an integer. We consider the $(\ell + 1)$-state $\diamond$-DFA $\hat{M}$ with $\sigma_1(\diamond) = \{a\}$ as depicted on the right-hand side of Fig. 7, where all transitions not depicted go into the rejecting sink-state that is not depicted as well. Clearly, $\hat{M}$ has exactly one productive $\diamond$-transition.

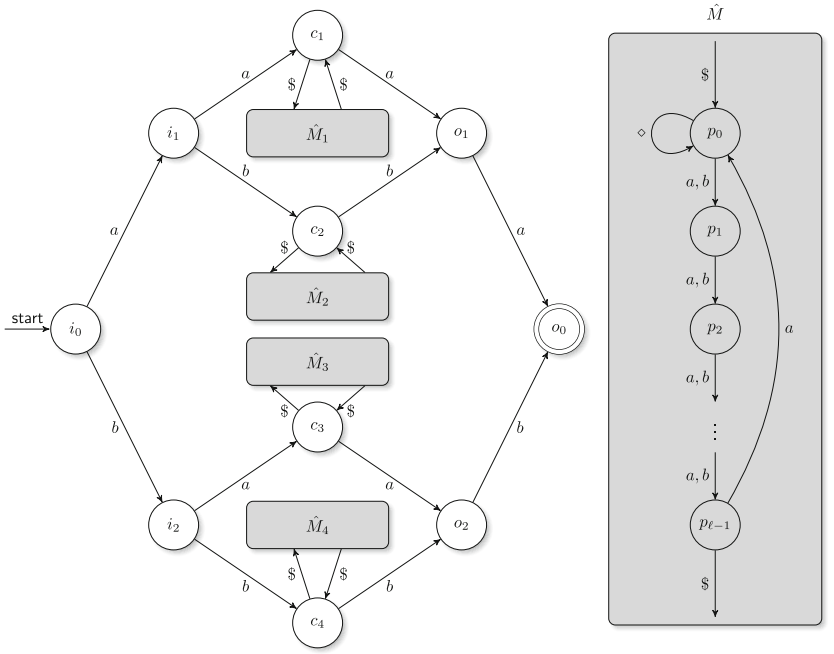

Fig. 7. A $\diamond$-DFA with $\sigma_1(\diamond) = \{a\}$ and 4 productive $\diamond$-transitions. Four copies of $\hat{M}$ are plugged in as $\hat{M}_i$, $1 \leq i \leq 4$. The common rejecting sink-state as well as the transitions to it are not depicted.

Next, we use k_1 copies $\hat{M}_i$, $1 \leq i \leq k_1$ of $\hat{M}$ that are distinct except for a common sink-state. Say the states are $p_{i,j}$, for $1 \leq i \leq k_1$ and $0 \leq j \leq \ell - 1$, and p_e, for the sink-state. Finally, these copies are assembled into one $\diamond$-DFA M_1 by selecting k_1 different words $z_1, z_2, \ldots, z_{k_1}$ of length $\lceil \log(k_1) \rceil$ from $\{a, b\}^*$. These words are processed from an initial state in a tree-like structure, where the initial state is the root and each of the k_1 leaves is connected to and from one copy by $\$$-transitions, where $\$$ is a new symbol (see the left-hand side of Fig. 7). In this way, each copy $\hat{M}_i$ is selected by an individual

prefix z_i. Again, all missing transitions are directed to the common sink-state p_e. Let $L(\hat{M})$ denote the language of words accepted by $\hat{M}$ with initial state p_0 and sole accepting state $p_{\ell-1}$. Then a word w is accepted by M_1 if and only if it has the form $z(\$L(\hat{M})\$)^* z^R$, where $z \in \{z_1, z_2, \ldots, z_{k_1}\}$ (see Fig. 7 for an example with $k_1 = 4$). In total, the $\diamond$-DFA M_1 has k_1 productive $\diamond$-transitions and at most $(2 \cdot (2^{\lceil \log(k_1) \rceil} - 1)) + k_1 + k_1\ell + 1 \leq 5k_1 + k_1\ell - 1$ states.

The rest of the proof is to show the claimed lower bound for the number of states necessary for any $\diamond$-DFA M_2 with any $\diamond$-substitution σ_2 having at most k_2 productive $\diamond$-transitions and $\sigma_2(L(M_2)) = \sigma_1(L(M_1))$. □

References

1. Balkanski, E., Blanchet-Sadri, F., Kilgore, M., Wyatt, B.J.: On the state complexity of partial word DFAs. Theor. Comput. Sci. **578**, 2–12 (2015)
2. Berstel, J., Boasson, L.: Partial words and a theorem of Fine and Wilf. Theor. Comput. Sci. **218**, 135–141 (1999)
3. Birget, J.C.: Intersection and union of regular languages and state complexity. Inform. Process. Lett. **43**, 185–190 (1992)
4. Blanchet-Sadri, F.: Algorithmic Combinatorics on Partial Words. CRC Press, Boca Raton (2008)
5. Blanchet-Sadri, F., Goldner, K., Shackleton, A.: Minimal partial languages and automata. RAIRO Inform. Théor. **51**, 99–119 (2017)
6. Dassow, J., Manea, F., Mercaş, R.: Regular languages of partial words. Inf. Sci. **268**, 290–304 (2014)
7. Fischer, M.J., Paterson, M.S.: String-matching and other products. In: Karp, R.M. (ed.) Complexity of Computation. SIAM-AMS Proceedings, vol. 7, pp. 113–125. AMS, New Jersey (1974)
8. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Process. Lett. **59**, 75–77 (1996)
9. Goldstine, J., Kintala, C.M.R., Wotschke, D.: On measuring nondeterminism in regular languages. Inform. Comput. **86**, 179–194 (1990)
10. Holzer, M., Jakobi, S., Wendlandt, M.: On the computational complexity of partial word automata problems. Fund. Inform. **148**, 267–289 (2016)
11. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. Int. J. Found. Comput. Sci. **14**, 1087–1102 (2003)
12. Holzer, M., Kutrib, M.: Nondeterministic finite automata - recent results on the descriptional and computational complexity. Int. J. Found. Comput. Sci. **20**, 563–580 (2009)