Optimal State Reductions of Automata with Partially Specified Behaviors☆

Nelma Moreira[a,1], Giovanni Pighizzini[b,2,*], Rogério Reis[a,1]

[a] *CMUP, Faculdade de Ciências da Universidade do Porto*
Rua do Campo Alegre, 4169-007 Porto, Portugal
[b] *Dipartimento di Informatica, Università degli Studi di Milano, Italy*

Abstract

Nondeterministic finite automata with *don't care* states, namely states which neither accept nor reject, are considered. A characterization of deterministic automata compatible with such a device is obtained. Furthermore, an optimal state bound for the smallest compatible deterministic automata is provided. It is proved that the problem of minimizing deterministic *don't care* automata is NP-complete and PSPACE-hard in the nondeterministic case. The restriction to the unary case is also considered.

Keywords: Finite automata, nondeterminism, *don't care* automata, descriptional complexity

1. Introduction

Finite state automata are well-known and widely investigated language acceptors. On each input string x, the behavior of a finite automaton is an answer

☆Preliminary version presented at *SOFSEM 2015: Theory and Practice of Computer Science – 41st International Conference on Current Trends in Theory and Practice of Computer Science*, Pec pod Sněžkou, Czech Republic, January 24–29, 2015 [*Lect. Notes Comput. Sci.*, 6224, pp. 339–351, Springer, 2015].

*Corresponding author
Email addresses: nam@dcc.fc.up.pt (Nelma Moreira), pighizzini@di.unimi.it (Giovanni Pighizzini), rvr@dcc.fc.up.pt (Rogério Reis)

[1]Authors partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under projects PEst-C/MAT/UI0144/2013.
[2]Author partially supported by MIUR under the project PRIN "Automi e Linguaggi Formali: Aspetti Matematici e Applicativi", code H41J12000190001.

yes/no to the question of the membership of x to the accepted language. In some situations, however, we could have some input sequences for which the answer of the automaton is not interesting, or even situations where the automaton does not need to consider all possible strings over the input alphabet. For example, an automaton could receive its input from another machine or program, which produces only sequences in a special form, thus excluding all the other sequences which are definable over the input alphabet. We give a couple of trivial but immediate examples over the alphabet $\{\texttt{-}, \texttt{0}, \texttt{1}, \ldots, \texttt{9}\}$. If the inputs of the automaton represent numbers in decimal notation produced by a (correct) program, the automaton cannot expect sequences starting by `0` (with the only exception of the sequence `0`) as `00123`, sequences starting by `-0`, and sequences containing the symbol `-` after the leftmost position, as `4-9-2014`. On the other hand, if the inputs represented calendar dates, the last string would be a valid input, while a string as `-1234` would be invalid (unless a strange and counterintuitive format is used).

In these cases, we do not need to define the behavior of the automaton, namely acceptance or rejection, on the strings which are not interesting or will never appear as input. This suggests us the idea of studying finite automata with three kinds of states: accepting states, rejecting states, and *don't care* states. We call these models automata with *don't care* states or, shortly, *don't care* automata. A quite natural problem we consider in the paper is the state reduction of these models. Of course, to perform this reduction, we can arbitrarily accept or reject strings on which the behavior of the automaton is not specified.

This idea is not completely new, if fact, in digital systems design, Moore automata (or equivalently Mealy automata) are used to specify several kinds of algorithms, protocols and processes which then are used in sequential circuits synthesis. Usually, the automata are incomplete (lacking either outputs or transitions for some inputs), and the elimination of redundant states reduces the size of the logic needed to be implemented, tested or verified. However, the stan-

dard algorithm for minimizing deterministic complete automata is not enough for incomplete ones. The first algorithm for the exact solution was described by Paull and Unger [15], and Pfleeger [17] proved that the minimization of incomplete deterministic Moore machines is a NP-complete problem. Since then many other exact and heuristic algorithms have been proposed, some considering that the initial machine is nondeterministic [18, 11, 16, 12, 3]. The standard Paull and Unger approach is based on the identification of sets of compatible states and the obtention of a minimal closed cover. The use of *don't care* states has been also considered for different purposes in the case of automata on infinite words [4].

In this paper, we mainly investigate *nondeterministic* automata with *don't care* states (dcNFA). Given such a device A, we are interested in finding a smallest deterministic finite automaton (DFA) B which is "compatible" with it, in the sense that all the strings accepted by A are also accepted by B and all the strings rejected by A are also rejected by B, while on the remaining strings B can have an arbitrary behavior. This problem can be reformulated as a *separation problem*: given two disjoint languages L_1 and L_2, find a language L with minimal state complexity that *separates* L_1 and L_2, i.e. such that $L_1 \subseteq L \subseteq L_2^c$ (where L^c is the complement of L). In the context of model checking, this version of the problem was considered by Chen et al. [1], but there the general Paull and Unger algorithm for deterministic automata was used.

Here we obtain a precise characterization of the DFAs which are compatible with a given dcNFA. This result is useful to obtain an upper bound for the number of states of the smallest compatible DFAs. We also show that this bound is tight. We also study computational complexity aspects. To this respect, we show that the problem of obtaining a smallest DFA compatible with a given dcDFA is NP-complete, and PSPACE-hard if the given *don't care* automaton is nondeterministic.

We conclude the paper by presenting some state complexity results concerning dcNFAs over a one-letter alphabet.

2. Automata with *don't care* States

Given an alphabet Σ, we consider the usual notions of deterministic finite automata (DFAs) and nondeterministic finite automata (NFAs) (with multiple initial states). Given an automaton A, we denote the language accepted by it as $\mathcal{L}(A)$. We also assume that the reader is familiar with the notion of *minimal* DFA. We now introduce the main notion we are interested in.

Definition 1. *A don't care* nondeterministic finite automaton *(dcNFA) A is a tuple $\langle Q, \Sigma, \delta, I, F^{\oplus}, F^{\ominus}\rangle$, where $A^{\oplus} = \langle Q, \Sigma, \delta, I, F^{\oplus}\rangle$ and $A^{\ominus} = \langle Q, \Sigma, \delta, I, F^{\ominus}\rangle$ are two NFAs such that $\mathcal{L}(A^{\oplus}) \cap \mathcal{L}(A^{\ominus}) = \emptyset$. A state $q \in Q$ is called an* accepting (rejecting) state *if $q \in F^{\oplus}$ ($q \in F^{\ominus}$, respectively). If $q \notin F^{\oplus} \cup F^{\ominus}$ then q is called a* don't care *state. There are two languages associated with A, the* accepted $\mathcal{L}^{\oplus}(A) = \mathcal{L}(A^{\oplus})$ *and the* rejected language $\mathcal{L}^{\ominus}(A) = \mathcal{L}(A^{\ominus})$.

The automaton A is a don't care deterministic finite automaton *(dcDFA) if the set I consists exactly of one element i and δ is a partial function from Q to Q, namely, for each $q \in Q$, $a \in \Sigma$, $\delta(q, a)$ contains at most one element.*

Two dcNFAs are equivalent if they have the same accepted language and the same rejected language. Notice that given a dcNFA A, its accepted (rejected) language consists of all words having a computation path from an initial state to an accepting (a rejecting, respectively) state. Hence, if all the states of A are reachable from the initial state, then the sets $F^{\oplus}$ and $F^{\ominus}$ must be disjoint.

A DFA can be immediately transformed into an equivalent dcDFA by choosing as sets of accepting and rejecting states, the set of final states and its complement, respectively. Hence, we can see dcDFAs as generalizations of DFAs. The situation is different for nondeterministic devices where, due to the different interpretation of the rejection, NFAs and dcNFAs are incomparable. Indeed, a string is rejected by a NFA when all computations on it do not lead to final states, while for the rejection by a dcNFA it is enough at least one computation leading to a rejecting state. For this purpose, the definition of dcNFA requires that the accepted and rejected languages are disjoint. This can restrict the non-

deterministic possibilities: fixed the set of states and the sets of accepting and rejecting states, not all the possible transition graphs define a dcNFA.

We also observe that unspecified transitions have different meanings for DFAs and for dcDFAs. In the first case, they lead to rejection. This can be explicitly represented, in a standard way, by inserting an extra state, called *trap* or *dead state*. All DFAs we consider in the paper have a total transition function. On the other hand, in the case of dcDFAs, unspecified transitions represent, according to Definition 1, *don't care* conditions. Since we do not need them to accept or to reject strings in the two languages associated to a dcDFA, we will never add any extra state to a dcDFA.

In this paper, given a *don't care* automaton A we are interested in finding automata that agree with A on its accepted and rejected languages. This leads to the following definition.

Definition 2. *Let A be a dcNFA. A language L is said to be* compatible *with A whenever $\mathcal{L}^{\oplus}(A) \subseteq L$ and $\mathcal{L}^{\ominus}(A) \subseteq L^c$. An NFA (or DFA) B is* compatible *with A when $\mathcal{L}(B)$ is compatible with A.*

Example 1. *Consider the dcDFA A represented in Figure 1. We label each*

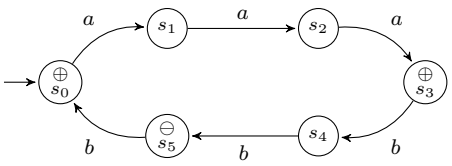

Figure 1: The dcDFA A of Example 1.

accepting state with a $\oplus$ and each rejecting state with a $\ominus$, leaving don't care *states unlabeled. Hence, $F^{\ominus} = \{s_5\}$ and $F^{\oplus} = \{s_0, s_3\}$. Trivially, $\mathcal{L}^{\oplus}(A) = (a^3b^3)^\star(\varepsilon + a^3)$ and $\mathcal{L}^{\ominus}(A) = (a^3b^3)^\star(a^3b^2)$. The language $L = (a^3b^3)^\star(\varepsilon + a + a^2 + a^3)$ is compatible with A. Notice that L is accepted by a DFA with the same transition graph of A (plus the trap state) and with set of final states $\{s_0, s_1, s_2, s_3\}$. In the next sections, we will present several smaller DFAs com-*

patible with A.

Let $G = (V, E)$ be an undirected graph. We recall that each complete subgraph of G is called a *clique*. We also say that a subset $\alpha \subseteq V$ forms a clique if the subgraph of G induced by α, namely the graph $(\alpha, E \cap (\alpha \times \alpha))$, is a clique. Furthermore, a clique $\alpha \subseteq V$ is maximal if any other subset of V which properly contains α does not form a clique. A *clique covering* of a graph G is a set of cliques such that every vertex of G belongs at least to one clique.

A *self-verifying* automaton (SVFA) A is a dcNFA where it is required that for each input string there exists at least one computation ending in an accepting or in a rejecting state, i.e., $\mathcal{L}^{\oplus}(A) = (\mathcal{L}^{\ominus}(A))^c$. This implies that the only language compatible with A is its accepted language $\mathcal{L}^{\oplus}(A)$. Hence, each SVFA can be transformed into an equivalent (and unique) minimal DFA. In [10] an optimal bound for the number of states of the minimal DFA equivalent to any given SVFA has been obtained. The authors associate with each n-state self-verifying automaton a graph with n vertices and prove that the state set of the minimum DFA equivalent to the given SVFA should be isomorphic to the set of the maximal cliques of such a graph. In the next sections, we use a similar approach to obtain minimal DFAs compatible with a given automaton with *don't care* states.

3. Conversion into Compatible Deterministic Automata

In this section we study how to convert any given dcNFA A into compatible DFAs. In particular we are interested in finding a minimal DFA compatible with A. As we will see, it is possible to have several minimal nonisomorphic smallest compatible DFAs.

Let us suppose that all the states of $A = \langle Q, \Sigma, \delta, I, F^{\oplus}, F^{\ominus} \rangle$ are reachable from the initial state. For each $q \in Q$, we denote by $L_q^{\oplus}$ and $L_q^{\ominus}$, respectively, the set of strings accepted and the set of strings rejected starting from q, that is, $L_q^{\oplus} = \{x \in \Sigma^* \mid \delta(q, x) \cap F^{\oplus} \neq \emptyset\}$ and $L_q^{\ominus} = \{x \in \Sigma^* \mid \delta(q, x) \cap F^{\ominus} \neq \emptyset\}$. Using the facts that q is reachable and that accepted and rejected languages by

A are disjoint (Definition 1), it can also be concluded that those two languages are disjoint.

For the same reason, applying the subset construction to A, it turns out that $L^{\oplus}_{\alpha} \cap L^{\ominus}_{\alpha} = \emptyset$ for each subset $\alpha \subseteq Q$ whose states are all reachable by a same string, where $L^{\times}_{\alpha} = \bigcup_{q \in \alpha} L^{\times}_{q}$, for $\times \in \{\oplus, \ominus\}$. So, by a suitable marking of accepting and rejecting states, from A we can get the *subset* dcDFA A_s (with only reachable states) with $L^{\times}(A_s) = L^{\times}(A)$, for $\times \in \{\oplus, \ominus\}$.

As in [10], to study the structure of DFAs which are compatible with A, we introduce a *compatibility relation* on the state set Q. Intuitively, two states p, q of A are compatible if and only if two computations starting from p and q cannot give contradictory answers on the same string. Formally:

Definition 3. *Two states p, q of A are* compatible *if and only if*

$$(L^{\oplus}_{p} \cup L^{\oplus}_{q}) \cap (L^{\ominus}_{p} \cup L^{\ominus}_{q}) = \emptyset.$$

The compatibility graph *of A is the undirected graph whose vertex set is Q, and which contains the edge $\{p, q\}$ if and only if states p and q are compatible.*

It follows from the above discussion that if α is a state of the automaton A_s, then all states p, q in the set α must be compatible. Hence, each reachable state of A_s is represented by a clique in the compatibility graph.

In the case of SVFAs, it was proved that if for two reachable subsets $\alpha, \beta \subseteq Q$ of the subset automaton the set $\alpha \cup \beta$ is a clique of the compatibility graph then α and β are equivalent [10]. In our case, since the automaton A_s deriving from the subset construction could contain *don't care* states, we cannot properly define a similar equivalence over A_s states. However, we can prove the following result which characterizes DFAs that are compatible with A in terms of functions that map states into cliques of the compatibility graph.

Theorem 4. *A DFA $A' = \langle Q', \Sigma, \delta', i', F' \rangle$ is compatible with a given dcNFA $A = \langle Q, \Sigma, \delta, I, F^{\oplus}, F^{\ominus} \rangle$ if and only if there is a function $\phi : Q' \to 2^{Q}$ such that:*

1. *$I \subseteq \phi(i')$,*

2. *for* $q \in Q'$, $a \in \Sigma$, $\delta(\phi(q), a) \subseteq \phi(\delta'(q, a))$,

3. *for* $q \in Q'$, $\phi(q) \cap F^{\oplus} \neq \emptyset$ *implies* $q \in F'$ *and* $\phi(q) \cap F^{\ominus} \neq \emptyset$ *implies* $q \notin F'$.

Furthermore, if A' *is compatible with* A *then:*

4. *for each* $x \in \Sigma^*$, $\delta(I, x) \subseteq \phi(\delta'(i', x))$,

5. *the set* $\phi(Q')$ *is a clique covering of the compatibility graph of* A.

Proof. First, let us suppose that A' is compatible with A. For each $q \in Q'$, we define

$$\phi(q) = \{p \in Q \mid \exists x \in \Sigma^* \text{ s.t. } q = \delta'(i', x) \text{ and } p \in \delta(I, x)\}.$$

By considering the empty string, we observe that $I \subseteq \phi(i')$, proving 1. Now, given $a \in \Sigma$, let $q' = \delta'(q, a)$. To prove 2 we show that $p' \in \delta(\phi(q), a)$ implies $p' \in \phi(q')$. To this aim, let us consider $p \in \phi(q)$ such that $p' \in \delta(p, a)$. By the definition of ϕ, there is a string $x \in \Sigma^*$ such that $q = \delta'(i', x)$ and $p \in \delta(I, x)$. Hence, $q' = \delta'(q, a) = \delta'(i', xa)$ and $p' \in \delta(p, a) \subseteq \delta(I, xa)$. According to the definition of ϕ this implies $p' \in \phi(q')$. Finally, the condition 3 follows from our definition of ϕ.

To prove the converse, first of all it is useful to derive 4 from 1 and 2. We use an induction on the length of the string x. The basis $x = \epsilon$ is trivial. Now, let us consider a nonempty string $x = ya$ with $y \in \Sigma^*$ and $a \in \Sigma$, and suppose condition 4 true for y. Given $p \in \delta(I, x)$, there is a state $p' \in \delta(I, y)$ such that $p \in \delta(p', a)$. Let $q' = \delta'(i', y)$. From the induction hypothesis we get that $p' \in \phi(\delta'(i', y)) = \phi(q')$ and, by condition 2, $\delta(\phi(q'), a) \subseteq \phi(\delta'(q', a))$ and, by putting all together, we complete the proof of 4:

$$p \in \delta(p', a) \subseteq \delta(\phi(q'), a) \subseteq \phi(\delta'(q', a)) = \phi(\delta'(i', x)).$$

Now, given $x \in \mathcal{L}^{\oplus}(A)$, let $p \in \delta(I, x) \cap F^{\oplus}$. Since $p \in \phi(\delta'(I, x))$, by condition 3 x should be accepted by A'. In a similar way, if $x \in \mathcal{L}^{\ominus}(A)$ then x should be rejected by A'. Hence we conclude that A' is compatible with A.

Concerning the second part of the theorem, we already proved 4. To prove 5, first we show that, for each $q \in Q'$, the set $\phi(q)$ is a clique of the compatibility graph of A, namely, each two states $p, r \in \phi(q)$ are compatible. Let $x, u \in \Sigma^*$ such that $q = \delta'(i', x) = \delta'(i', u)$, $p \in \delta(I, x)$, and $r \in \delta(I, u)$. By contradiction, suppose p and r not compatible. Then, there is $z \in \Sigma^*$ such that, without loss of generality, $\delta(p, z) \in F^{\oplus}$ and $\delta(r, z) \in F^{\ominus}$. It follows that z distinguishes strings x and u, which contradicts the fact that these strings lead to the same state in the automaton A'. This allows us to conclude that p, r must be compatible and, hence, $\phi(q)$ is a clique. Furthermore, since all the states of A are reachable, as a consequence of 4 for each $p \in Q$ there is a state $q \in Q'$ such that $p \in \phi(q)$. This completes the proof of 5. □

Using Theorem 4, we now derive a "pseudo-subset construction" which allows to find some DFAs compatible with A. We remind the reader that we suppose that all the states of A are reachable from the initial state. Then we define a DFA $A' = \langle Q', \Sigma, \delta', i', F' \rangle$ as follows:

- Q' is the set of *all maximal cliques* of the compatibility graph of A; in the following, given a maximal clique $\alpha \subseteq Q$, we use the same name α to denote the corresponding state in Q';
- i' is a clique that includes the set I of initial states of A;
- for $\alpha \in Q'$, $a \in \Sigma$, $\delta'(\alpha, a)$ is a state $\beta \in Q'$ such that $\delta(\alpha, a) \subseteq \beta$;
- the set F' of final states is a subset of Q' that contains all states α s.t. $\alpha \cap F^{\oplus} \neq \emptyset$ and does not contain any state α s.t. $\alpha \cap F^{\ominus} \neq \emptyset$, namely, each state of Q' that contains a state from $F^{\oplus}$ is marked as final, each state that contains a state from $F^{\ominus}$ is marked as nonfinal, while each one of the remaining states can be freely marked either as final or as nonfinal.

The above definition leaves some degrees of freedom, which allow to obtain different DFAs. For any possible choice, it can be immediately verified that the function $\phi : Q' \to 2^Q$ defined as $\phi(\alpha) = \alpha$ satisfies the conditions of Theorem 4. Hence, it turns out that each DFA A', defined as above, is compatible with A.

Example 2. *Let us consider the dcDFA A of Example 1 (Figure 1). Its compatibility graph is depicted in Figure 2 (left). Applying the above construction we obtain 4 different DFAs, which are summarized in the Figure 2 (right). We have two choices for the initial state and two choices for the transition from state $\{s_1, s_2, s_5\}$ on b. These choices are represented by dotted arrows.*

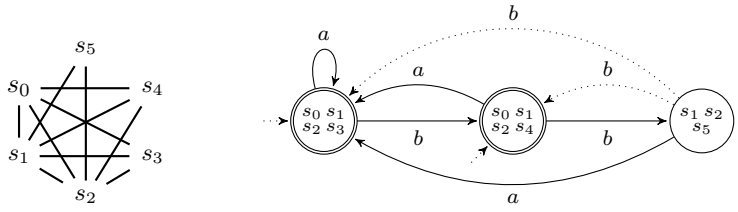

Figure 2: The compatibility graph of the dcDFA A in Fig. 1 and four compatible DFAs.

In the previous construction, we used the covering of the compatibility graph defined by maximal cliques. In general, we could also use a different covering, provided that the trivial function ϕ mapping each clique of the considered cover in itself satisfies the conditions 1, 2, and 3 of Theorem 4. For instance, further DFAs, compatible with the dcDFA A of Example 2, are depicted in Figure 3. We can observe that in this example the compatibility graph of A cannot be

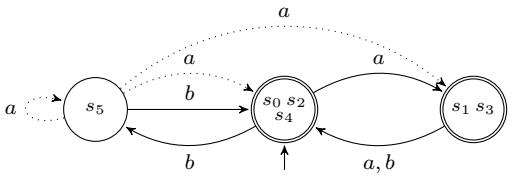

Figure 3: More DFAs compatible with the dcDFA A in Figure 2.

covered using less than 3 cliques. Hence, there are no DFAs compatible with A with less than 3 states, the number of maximal cliques in the compatibility graph. However, in general the situation can be different, as illustrated in the next example.

Example 3. *Let us consider the dcDFA A depicted in the upper part of Figure 4 with its compatibility graph, which contains 4 maximal cliques. This graph has*

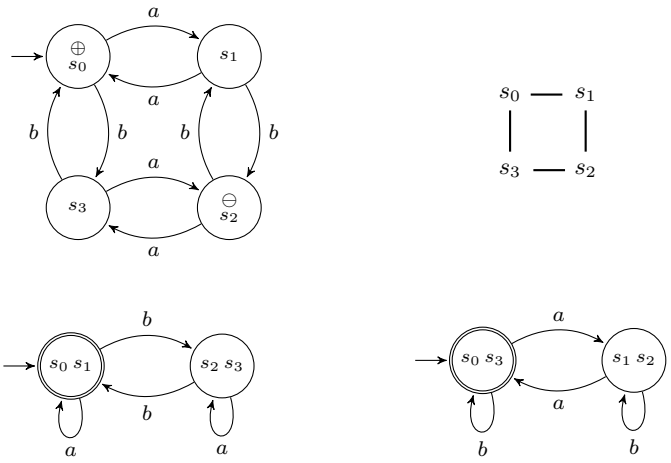

Figure 4: The dcDFA A of Example 3 with its compatibility graph, and two compatible DFAs.

the following two coverings consisting each one of two cliques: $\{\{s_0, s_1\}, \{s_2, s_3\}\}$ *and* $\{\{s_0, s_3\}, \{s_1, s_2\}\}$*. For these coverings we obtain two DFAs which are compatible with A (see also Figure 4). Since these DFAs have only two states and each DFA consisting only of one state cannot be compatible with A, it turns out that they are the smallest DFAs which are compatible with A. In Figure 5 it is depicted a dcNFA* $\widehat{A}$ *having the same compatibility graph as A, with two compatible DFAs whose states correspond to all maximal cliques of that graph. We observe that in the automaton* $\widehat{A}$ *the strings* a*,* b*,* bca*, and* bcb *lead to the set of states* $\{s_0, s_1\}$*,* $\{s_0, s_3\}$*,* $\{s_2, s_3\}$*, and* $\{s_1, s_2\}$*, respectively. Hence, observing the compatibility graph and using condition 4 of Theorem 4 we can conclude that in this example* all maximal cliques *of the compatibility graph are necessary. Hence, each DFA compatible with* $\widehat{A}$ *should have at least* 4 *states.*

Example 3 shows that we can have different dcNFAs A and $\widehat{A}$ with the same compatibility graph but with smallest compatible DFAs of different sizes. The following theorem summarizes the situation, providing bounds for such a size in

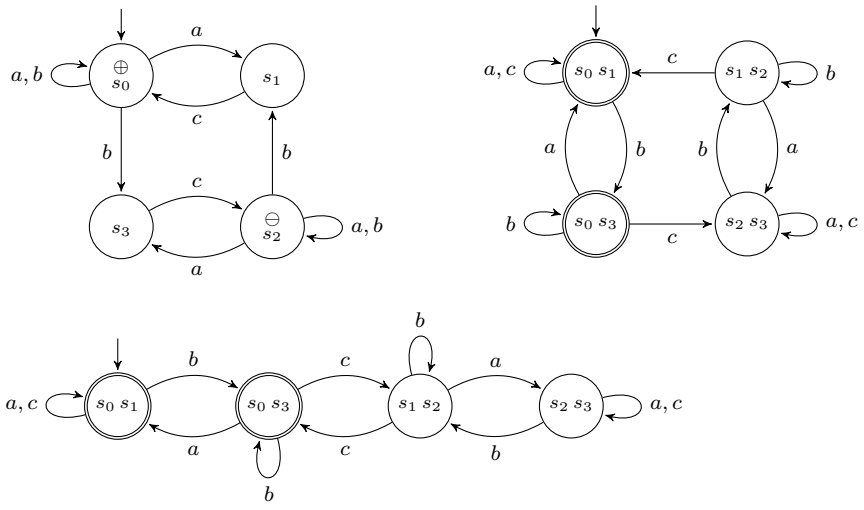

Figure 5: The dcNFA $\widehat{A}$ of Example 3 (top left), with two compatible DFAs (top right and bottom).

terms of cliques of the compatibility graph:

Theorem 5. *For each dcNFA A, there exists a compatible DFA whose number of states is bounded by the number of maximal cliques in the compatibility graph of A. Furthermore, each DFA compatible with A should have at least as many states as the smallest number of cliques that cover the compatibility graph of A.*

Proof. From the above pseudo-subset construction we know that there exists a DFA which is compatible with A and has a number of states equal to the number of maximal cliques of the compatibility graph. The lower bound follows from condition 5 of Theorem 4. □

4. State Complexity

In this section, we study descriptional complexity aspects. First we state an upper bound for the number of states of smallest DFAs compatible with a given dcNFA, showing that it can be effectively reached, i.e. it is tight. The arguments are adapted from those used for SVFAs [10].

Theorem 6. *For each integer $n \geq 2$ and each n-state dcNFA there exists a compatible DFA with at most $f(n)$ states, where*

$$f(n) = \begin{cases} 3^{\lfloor n/3 \rfloor}, & \text{if } n \equiv 0 \pmod 3, \\ 4 \cdot 3^{\lfloor n/3 \rfloor - 1}, & \text{if } n \equiv 1 \pmod 3, \\ 2 \cdot 3^{\lfloor n/3 \rfloor}, & \text{if } n \equiv 2 \pmod 3. \end{cases}$$

Furthermore this bound can be effectively reached.

Proof. The upper bound immediately derives from Theorem 5 and from a result by Moon and Moser [13] stating that the maximum number of maximal cliques in a graph with n vertices is given by the function $f(n)$. The lower bound is a consequence of Theorem 10 in [10], where for each integer n an n-state SVFA A_n with multiple initial states such that the smallest equivalent DFA requires $f(n)$ states was provided. Figure 6 illustrates the case of n multiple of 3. Notice that a maximal clique in the compatibility graph of A_n contains exactly one state from each column.

Since SVFAs with multiple initial states are a special case of dcNFAs, the claimed result follows. □

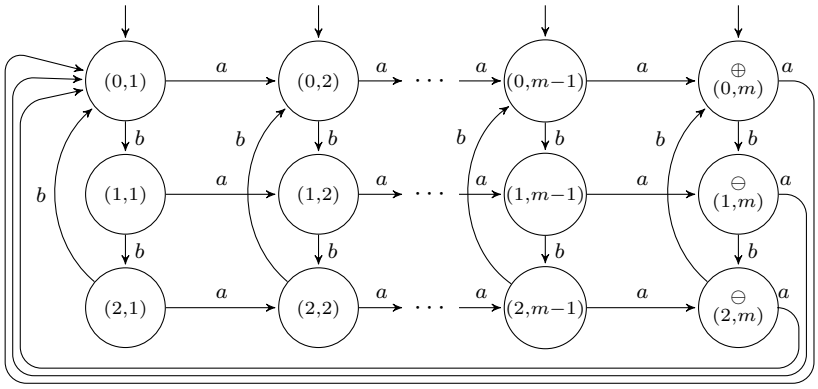

Figure 6: Automaton A_n of Theorem 6 in the case of $n = 3m$, $m \geq 1$.

The optimality proof in Theorem 6 is a consequence of the optimality of the same bound for SVFAs with multiple initial states. Since the optimal bound in

the case of SVFAs with a single initial state is slightly different ($1+f(n-1)$), one could ask what happens in the case of dcNFAs with a *single initial state*. We are going to prove that in this case the optimal bound remains that of Theorem 6.

To this aim, for each n we consider an automaton A'_n, obtained by modifying the automaton A_n used to give the optimality in Theorem 6, as follows. We start from the same set of states of A_n and from the same transition graph. One of the initial states of A_n is chosen as the initial state of A'_n. Furthermore, we add a transition on a new input symbol c from a selected state of A_n to all the states that in A_n are initial. (An example of such a modification is depicted in Figure 7.) In this way each time the automaton A'_n makes a transition on the letter c, it is able to simulate a computation of A_n on a factor $w \in \{a,b\}^*$. We

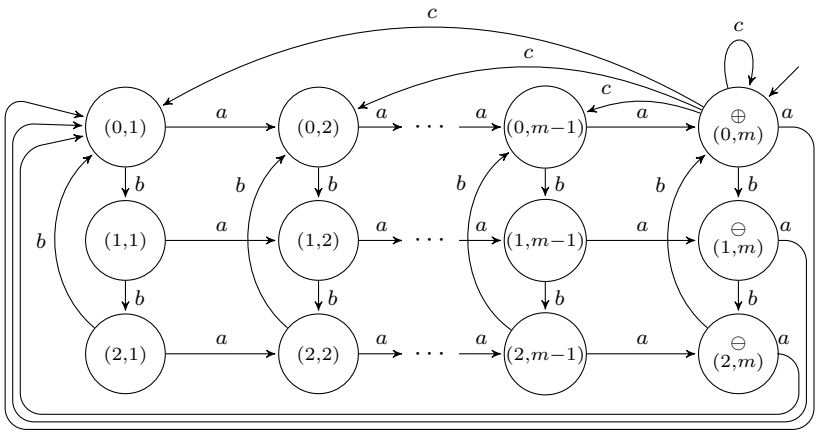

Figure 7: An example of automaton A'_n in the case of n multiple of 3. The state $(0,m)$ has been chosen as initial state. Furthermore, a transition on c from $(0,m)$ to each $(0,j)$, with $1 \leq j \leq m$ has been added.

will show that each DFA compatible with A'_n requires $f(n)$ states, where f is the function given in Theorem 6, by making use of the following general lemma:

Lemma 7. *Let Σ be an alphabet and $c \notin \Sigma$ be an extra symbol. Given a dcNFA A over Σ, a nonempty set $K \subseteq (\Sigma \cup \{c\})^*$, two languages $J', J'' \subseteq \Sigma^*$, consider the languages $L' = Kc\mathcal{L}^{\oplus}(A) \cup J'$ and $L'' = Kc\mathcal{L}^{\ominus}(A) \cup J''$. If $L' \cap L'' = \emptyset$ then each DFA accepting a language L, with $L' \subseteq L \subseteq L''^c$ should have at*

least as many states as a smallest DFA compatible with A.

Proof. Given a language $L \subseteq (\Sigma \cup \{c\})^*$, with $L' \subseteq L \subseteq L''^c$, let us fix a string $w \in K$ and consider the language $\widetilde{L} = \{x \in \Sigma^* \mid wcx \in L\}$. We observe that $x \in \mathcal{L}^{\oplus}(A)$ implies $wcx \in Kc\mathcal{L}^{\oplus}(A) \subseteq L' \subseteq L$ and so $x \in \widetilde{L}$. In a similar way, $x \in \mathcal{L}^{\ominus}(A)$ implies $wcx \in Kc\mathcal{L}^{\ominus}(A) \subseteq L'' \subseteq L^c$ and $x \in \widetilde{L}^c$. This means that each DFA accepting $\widetilde{L}$ should be compatible with the given dcNFA A.

Now we observe that if two strings $x, y \in \Sigma^*$ are distinguishable for the language $\widetilde{L}$ then the strings wcx and wcy are distinguishable for L. In fact, for $z \in \Sigma^*$, $xz \in \widetilde{L}$ and $yz \notin \widetilde{L}$ implies $wcxz \in L$ and $wcyz \notin L$. Hence, we can conclude that each DFA accepting L should have at least as many states as the smallest DFA accepting the language $\widetilde{L}$, which, as already noticed, is compatible with A. This completes the proof. □

Notice that in Lemma 7 we do not need that K, J', J'' are regular.

Theorem 8. *For each integer n there is an n-state dcNFA with a unique initial state such that the smallest compatible DFA requires $f(n)$ states, where $f(n)$ is the function given in Theorem 6.*

Proof. Given n, let us consider the automaton A_n used in Theorem 6 and a dcNFA A'_n defined as above explained. We denote by J' and J'' the languages accepted and rejected by computations of the original automaton A_n that start from the initial state of A'_n, and K the set of strings (over $\Sigma \cup \{c\}$) that in A'_n lead from the initial state to the initial state. Clearly, $J' \subseteq \mathcal{L}^{\oplus}(A_n)$ and $J'' \subseteq \mathcal{L}^{\ominus}(A_n)$. The reader can verify that $\mathcal{L}^{\oplus}(A'_n) = Kc\mathcal{L}^{\oplus}(A_n) \cup J'$ and $\mathcal{L}^{\ominus}(A'_n) = Kc\mathcal{L}^{\ominus}(A_n) \cup J''$, and that those two languages are disjoint. Hence, the claimed result follows from Lemma 7. □

After considering the restriction to dcNFAs having only one initial state, we further restrict to the case of deterministic transitions where, clearly, the bound of Theorem 6 can be reduced. In fact, given an n-state dcDFA A we can just arbitrarily mark each *don't care* state as accepting or rejecting in order to obtain a compatible DFA with the same number of states. Furthermore, if the set of

don't care states of A is empty and A is minimal then we clearly cannot obtain a smaller compatible DFA. Hence, in the deterministic case n is a tight bound.

We can also observe that if A contains a *don't care* trap state then a compatible DFA can be always obtained by moving each transition leading to the trap state to an arbitrarily chosen state and by arbitrarily choosing final states among the remaining *don't care* states. Hence, the resulting DFA contains $n-1$ states. For each n this bound cannot be further reduced. Consider in fact the n-state automaton consisting of a loop of $n-1$ states accepting the language $(ab^{n-2})^*$ and rejecting all the strings in $(ab^{n-2})^*ab^k$ with $0 \leq k < n-2$, plus a *don't care* trap state. Clearly, each two states on the loop are incompatible. Hence, they belong to different cliques of the compatibility graph. By Theorem 5, we conclude that each compatible DFA should have at least $n-1$ states.

5. Time Complexity

In this section we shortly study time complexity of the reductions of dcDFAs and dcNFAs to minimal compatible DFAs. In the first case we prove NP-completeness. Our starting point is the following problem, which has been proved to be NP-complete by Pfleeger [17]:

> Given an "incomplete" DFA A and $k > 0$, is there a way to assign a state to each unspecified transition so that the resulting complete automaton has a minimal equivalent DFA with at most k states?

A clarification is necessary to explain the meaning of "incomplete" in this context. As already mentioned in the Section 2, reaching an undefined transition in a DFA is conventionally interpreted as the definitive rejection of the input and, hence, undefined transitions can be made defined by introducing a trap state, which is not final and, so, rejecting. In the above mentioned problem, an undefined transition will never be reached (e.g., because some restrictions on the form of possible input words) and, hence, it represents a *don't care* condition. Hence, an "incomplete" DFA $A = \langle Q, \Sigma, \delta, i, F\rangle$ in the previous problem, can be transformed in a complete dcDFA by adding a trap state q_t which is the only

don't care state, and by choosing F as the set of accepting states and $Q - F$ as the set of rejecting states. We obtain the following result.

Theorem 9. *The problem of deciding if, given a dcDFA A and an integer $k > 0$ (in binary), there exists a compatible DFA with at most k states is NP-complete.*

Proof. The NP-hardness follows from the above discussion.[3] To show that the problem belongs to NP, we observe that in polynomial time it is possible to nondeterministically generate a DFA B with at most k states and verify if it is compatible with A. More into details, compatibility is verified by checking if $\mathcal{L}^{\oplus}(A) \subseteq \mathcal{L}(B)$ and $\mathcal{L}^{\ominus}(A) \subseteq (\mathcal{L}(B))^c$. To do that, from A and B we build a product automaton and verify that for each reachable state (p, q), when the component p is an accepting state of A then the component q is a final state of B and when the component p is a rejecting state of A then the component q is a nonfinal state of B. □

Notice that the compatible DFA B guessed in the procedure described in the proof of Theorem 9 has a description of size $O(k)$ which, since A is deterministic, is polynomial in the size of A. We are now wondering what happens in Theorem 9 when the automaton A is nondeterministic. The size of the "witness" compatible DFA B could be exponential in the size of A and in the length of k (written in binary), but polynomial in the *value* of k. Then we obtain the following result.

Theorem 10. *The problem of deciding if, given a dcNFA A and an integer $k > 0$ written in unary, there exists a compatible DFA with at most k states is NP-complete.*

Determinization of NFAs has been proved to be PSPACE-hard [9]. Hence, for k written in binay we obtain the following corollary.

[3] It could also be deduced using NP-completeness of the inference of a DFA from a finite set of words [8].

Corollary 11. *The problem of deciding if, given a dcNFA A and an integer $k > 0$ (in binary), there exists a compatible DFA with at most k states is PSPACE-hard.*

6. State Complexity in the Unary Case

In Theorem 6 we proved an upper bound $f(n)$ for the number of the states in the smallest DFA compatible with each given n-state dcNFA and we showed that it can be effectively reached by witness automata over a two letter alphabet. We now discuss the unary case, namely the case of languages and automata defined over a one letter alphabet, which in the following we assume to be $\Sigma = \{a\}$.

Since each unary n-state NFA can be simulated by a DFA with $e^{O(\sqrt{n \ln n})}$ many states [2], a number which is lower than $f(n)$, we can conclude that in the unary case the number of states which are necessary for a DFA compatible with a given dcNFA is lower than in the general case.

We further deepen the discussion concerning unary dcNFAs. First, we recall that a unary NFA is in *Chrobak normal form* if its transition graph consists of an initial deterministic path and $s \geq 0$ disjoint loops. The last state of the path is connected via s outgoing edges to each of the loops. This is the only place where a nondeterministic decision can be taken by the automaton. Chrobak proved that each unary n-state NFA can be turned into this normal form, obtaining an automaton having $O(n^2)$ states in the initial path and at most n states in the loops [2]. These upper bounds have been reduced respectively to $n^2 - n$ and $n - 1$ [5, 6] (except when the given NFA is the trivial loop of length n which, in the worst case, cannot be further reduced).

We now show that even unary dcNFAs can be converted in Chrobak normal form within the same state bounds. (A unary automaton with *don't care* states in Chrobak normal form and two compatible DFAs are represented in Figure 8.)

Theorem 12. *For each unary n-state dcNFA different from the trivial loop of length n there exists an equivalent dcNFA in Chrobak normal form with an initial*

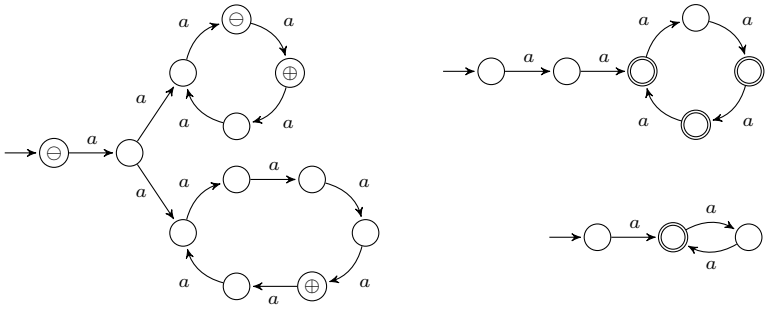

Figure 8: A unary dcDFA in Chrobak normal form and two compatible DFAs.

path of $\mu \leq n^2 - n$ *states and* $s \geq 0$ *disjoint loops of lengths* $\ell_1, \ell_2, \ldots, \ell_s > 0$ *with* $\ell_1 + \ell_2 + \cdots + \ell_s \leq n - 1$.

Proof. Given a unary n-state dcNFA A different from the trivial loop of length n, we can transform the unary NFAs $A^{\oplus}$ and $A^{\ominus}$ into two equivalent automata in Chrobak normal form $B_{\oplus}$ and $B_{\ominus}$, satisfying the required state bounds. We can also suppose that the initial paths of those two automata have the same length, otherwise we can modify the automaton with the shortest initial path by suitably "unrolling" its loops.

From $B_{\oplus}$ and $B_{\ominus}$, we could obtain a dcNFA equivalent to A (namely, with the same accepted and rejected languages) by merging the initial paths of $B_{\oplus}$ and $B_{\ominus}$ (namely, by identifying the corresponding states in the initial paths and marking any state as accepting or rejecting in the case it is marked as accepting or rejecting in one of the two automata), and by taking as set of loops the union of the set of loops of $B_{\oplus}$ and $B_{\ominus}$. Hence, the resulting automaton has at most $n^2 - n$ states in the initial path and at most $2n - 2$ in the loops.

However, by closely inspecting the conversion of unary NFAs in Chrobak normal form as presented in [6, Thm. 3.5], we can do better: Given a unary NFA M, some "cardinal loops" are selected from its transition graph. Their lengths will be the lengths $\ell_1, \ell_2, \ldots, \ell_s$ of the loops in the equivalent automaton in Chrobak normal form produced by the construction. Furthermore, the choice of the cardinal loops does not depend on the finality of the states in M. (The

interested reader can find all the details in [6, Def. 3.2, Th. 3.3].) Hence, since the two NFAs $A^{\oplus}$ and $A^{\ominus}$ derived from the given n-state dcNFA A have the same transition graph, we can conclude that the automata $B_{\oplus}$ and $B_{\ominus}$ in Chrobak normal form so obtained have the same set of loops and, thus, their transition graphs are isomorphic. The dcNFA B can be obtained just taking another copy of such transition graph and by marking any state q as accepting if q is accepting in $B_{\oplus}$ and by marking q as rejecting if it is accepting in $B_{\ominus}$. Thus, $B^{\oplus} = B_{\oplus}$ and $B^{\ominus} = B_{\ominus}$. This allows to conclude that $\mathcal{L}^{\oplus}(A) = \mathcal{L}^{\oplus}(B)$ and $\mathcal{L}^{\ominus}(A) = \mathcal{L}^{\ominus}(B)$. Hence, A and B are equivalent. □

The following lemmata will be used later to discuss the gap between unary dcNFAs and compatible DFAs:

Lemma 13. *A unary dcNFA (not necessarily in Chrobak normal form) cannot accept a string $a^{m'}$ along a path containing a state belonging to a loop of length ℓ' and reject another string $a^{m''}$ along a path containing a state belonging to a loop of length ℓ'', if ℓ' and ℓ'' are relatively prime.*

Proof. By contradiction, if the automaton accepts $a^{m'}$ and rejects $a^{m''}$ as in the statement, then by repeatedly traversing the loops under consideration, it also accepts strings $a^{m'+k\ell'}$ and rejects strings $a^{m''+h\ell''}$, for each $k, h \geq 0$. From results on diophantine equations it turns out that if $\gcd(\ell', \ell'') = 1$ then there are integers $\hat{h}, \hat{k} > 0$ such that $m' + \hat{k}\ell' = m'' + \hat{h}\ell''$. Hence the string $a^{m'+\hat{k}\ell'}$ should be accepted along a path and rejected along a different path, which is a contradiction. □

We observe that if a loop of a unary dcNFA in Chrobak normal form does not contain any accepting neither any rejecting state then it can be removed without affecting the accepted and the rejected languages.

Lemma 14. *Let A be a unary dcNFA in Chrobak normal form such that each loop contains at least one accepting or one rejecting state. If the lengths of all loops form a set of pairwise relatively prime numbers and A contains at least two loops, then there exists a compatible language which is either finite or co-finite.*

Proof. As a consequence of Lemma 13, if the length of the loops are pairwise relatively prime numbers then the automaton A cannot accept a string in a loop and reject another string in a different loop. This implies that the set Q_ℓ of all states of A which are in the loops cannot contain both accepting and rejecting states. If Q_ℓ does not contain any accepting state, then only a finite number of strings is accepted by A. Then, there exists a compatible language which is finite. In a similar way, if Q_ℓ does not contain any rejecting state, then we can conclude that there is a compatible language which is co-finite. □

We remind the reader that the transition graph of a unary DFA consists of a (possibly empty) initial path followed by one loop. When the accepted language is either finite or co-finite, a loop of one state is enough. Hence, it can be easily concluded that for the dcNFA A in Lemma 14 there exists a compatible DFA having the same initial path as A and a loop of length 1.

On the other hand, it is well-known that the maximum state gap between unary n-state NFAs and equivalent DFAs ($e^{\Theta(\sqrt{n \ln n})}$) is achieved only when the lengths of the loops of NFAs converted in Chrobak normal form are relatively prime numbers [2]. (The DFA is obtained by replacing the loops by one loop whose length is the product of the loop lengths of the given Chrobak normal form NFA. In the worst case, this loop cannot be reduced.) From Lemma 14 it turns out that such a gap cannot be achieved from unary dcNFAs to compatible DFAs. This is also known in the case of SVFAs, as observed in [10]. On the other hand, it has been shown that the state gap between unary n-state SVFAs and DFAs grows at least as $e^{\Omega(\sqrt[3]{n \ln^2 n})}$ [7]. Hence, this gives a lower bound for the gap from dcNFAs to DFAs:

Theorem 15. *For each sufficiently large integer n there is a dcNFA with at most n states such that each compatible DFA requires at least $e^{\Omega(\sqrt[3]{n \ln^2 n})}$.*

Let us now turn our attention to the deterministic case. As in the general case, we can easily construct examples of unary n-state dcDFAs having smallest compatible DFAs with n states. Concerning the separation problem, we are able to prove the following:

Theorem 16. *Let A' and A'' be two unary DFAs with initial paths of lengths μ', μ'', loops of lengths ℓ', ℓ'', and accepting languages L', L'', respectively, with $L' \subseteq L''$. Then there is a language L which is accepted by a DFA with an initial path of $\max(\mu', \mu'')$ states and a loop of $\gcd(\ell', \ell'')$ states such that $L' \subseteq L \subseteq L''$.*

Proof. Let us start by considering the case $\mu' = \mu'' = 0$. Given $\ell = \gcd(\ell', \ell'')$ we prove that the following language satisfies the claimed properties:

$$L = \{a^m \mid \exists a^k \in L' \text{ s.t. } m \equiv k \pmod{\ell}\}.$$

From the definition, it follows that $L' \subseteq L$ and L is accepted by a DFA consisting of a loop of ℓ states. Hence, it remains to prove that $L \subseteq L''$, namely $a^k \in L'$ implies $a^{k+\gamma\ell} \in L''$, for each $\gamma \geq 0$. Since the transition graphs of A' and A'' are loops of lengths ℓ' and ℓ'', respectively, and $L' \subseteq L''$, from $a^k \in L'$ we obtain that $a^{k+\alpha\ell'+\beta\ell''} \in L''$, for each $\alpha, \beta \geq 0$. By results on diophantine equations, it follows that $a^{k+\gamma\ell} \in L''$ for each $\gamma \geq \gamma_0$, for a suitable constant γ_0. We now consider $\gamma < \gamma_0$ and we choose an integer h such that $\gamma' = \gamma + h\frac{\ell''}{\ell} > \gamma_0$. Then $a^{k+\gamma'\ell} \in L''$. Furthermore, since $k + \gamma'\ell = k + \gamma\ell + h\ell''$ and A'' is a loop of ℓ'' states, we also get that $a^{k+\gamma\ell} \in L''$. As a consequence, we conclude that $L \subseteq L''$.

When $\mu' + \mu'' \neq 0$, we can modify one of the two automata by "unrolling" its loop in order to have both initial paths of the same length $\mu = \max(\mu', \mu'')$. We can now define an automaton A accepting a separating language which agrees with L' on strings of length less than μ, while it is defined by suitably shifting the previous construction for the remaining lengths, i.e.,

$$L = \{a^{m+\mu} \mid \exists a^{k+\mu} \in L' \text{ s.t. } m \equiv k \pmod{\ell}\} \cup \{a^m \mid m < \mu\}.$$

□

7. Final remarks

We studied automata with accepting states, rejecting states and states that neither accept nor reject. An obvious follow-up is the study of how standard

operations on finite automata can be extended. Some results on this subject have been recently presented in [14].

In the unary case the problems addressed in this paper are not fully solved and deserve further investigation. In particular it should be interesting to find a tight bound for the number of states of DFAs compatible with unary n-state dcNFAs.

Acknowledgments. The authors would like to thank the valuable remarks and corrections suggested by the anonymous referees.

References

[1] Chen, Y.-F., Farzan, A., Clarke, E. M., Tsay, Y.-K., Wang, B.-Y., 2009. Learning minimal separating DFA's for compositional verification. In: Kowalewski, S., Philippou, A. (Eds.), Proc. TACAS 2009. Vol. 5505 of Lecture Notes in Computer Science. Springer, pp. 31–45.

[2] Chrobak, M., 1986. Finite automata and unary languages. Theor. Comput. Sci. 47 (3), 149–158.

[3] Damiani, M., 1997. The state reduction of nondeterministic finite-state machines. IEEE Trans. CAD 16 (11), 1278–1291.

[4] Eisinger, J., Klaedtke, F., 2008. Don't care words with an application to the automata-based approach for real addition. Formal Methods in System Design 33 (1-3), 85–115.

[5] Gawrychowski, P., 2011. Chrobak normal form revisited, with applications. In: Bouchou-Markhoff, B., Caron, P., Champarnaud, J.-M., Maurel, D. (Eds.), Implementation and Application of Automata - 16th International Conference, CIAA 2011, Blois, France, July 13-16, 2011. Proceedings. Vol. 6807 of Lecture Notes in Computer Science. Springer, pp. 142–153.

[6] Geffert, V., 2007. Magic numbers in the state hierarchy of finite automata. Inf. Comput. 205 (11), 1652–1670.

[7] Geffert, V., Pighizzini, G., 2012. Pairs of complementary unary languages with "balanced" nondeterministic automata. Algorithmica 63 (3), 571–587.

[8] Gold, E. M., Dec. 1978. Complexity of automaton identification from given data. Inf. Contr. 37 (3), 302–320.

[9] Jiang, T., Ravikumar, B., 1993. Minimal NFA problems are hard. SIAM Journal on Computing, 1117–1141.

[10] Jirásková, G., Pighizzini, G., 2011. Optimal simulation of self-verifying automata by deterministic automata. Inf. Comput. 209 (3), 528–535.

[11] Kam, T., Villa, T., Brayton, R., Sangiovanni-Vincentelli, A., 1994. A fully implicit algorithm for exact state minimization. Proc. ACM/IEEE Design Automation Conf., 684–690.

[12] Kam, T., Villa, T., Brayton, R., Sangiovanni-Vincentelli, A., Nov. 1997. Theory and algorithms for state minimization of nondeterministic FSMs. IEEE Trans. CAD 16 (11), 1311–1322.

[13] Moon, J., Moser, L., 1965. On cliques in graphs. Israel J. Math 3, 23–28.

[14] Moreira, N., Pighizzini, G., Reis, R., 2015. Universal disjunctive concatenation and star. In: Shallit, J., Okhotin, A. (Eds.), Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings. Vol. 9118 of Lecture Notes in Computer Science. Springer, pp. 197–208.

[15] Paull, M. C., Unger, S. H., 1959. Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans. on Elect. Comput. 3, 356–367.

[16] Pena, J. M., Oliveira, A. L., 1999. A new algorithm for exact reduction of incompletely specified finite state machines. IEEE Trans. CAD 18 (11), 1619–1632.

[17] Pfleeger, C. P., Dec. 1973. State reduction in incompletely specified finite-state machines. IEEE Trans. Comput. 22 (C), 1099–1102.

[18] Rho, J.-K., Hachtel, G., Somenzi, F., Jacoby, R., 1994. Exact and heuristic algorithms for the minimization of incompletely specified state machines. IEEE Trans. CAD 13, 167–177.