Optimal State Reductions of Automata with Partially Specified Behaviors

Nelma Moreira[1], Giovanni Pighizzini[2], and Rogério Reis[1]

[1] Centro de Matemática e Faculdade de Ciências da Universidade do Porto, Portugal
{nam,rvr}@dcc.fc.up.pt*
[2] Dipartimento di Informatica, Università degli Studi di Milano, Italy
pighizzini@di.unimi.it**

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. Finally, it is proved that the problem of minimizing nondeterministic and deterministic *don't care* automata is NP-complete.

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 *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 would represent calendar dates, the last string will be a valid input, while a string as `-1234` will be invalid (unless a strange and counterintuitive format is used).

* 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 and FCOMP-01-0124-FEDER-020486.
** Author partially supported by MIUR under the project PRIN "Automi e Linguaggi Formali: Aspetti Matematici e Applicativi", code H41J12000190001.

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 the 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 (laking either outputs or transitions from some inputs), and the elimination of redundant states reduces the size of the logic needed to be implemented, tested or verified. However, the standard 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 [11], and Pfleeger [13] 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 [14, 8, 12, 9, 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 a 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 regular 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 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 dcNFA is NP-complete, and it remains NP-complete if the given *don't care* automaton is deterministic. The paper concludes with some considerations concerning dcNFAs over one-letter alphabets.

Due to the lack of space, some of the proofs are omitted from this version of the paper.

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. Associated to A there are the two languages $\mathcal{L}^{\oplus}(A) = \mathcal{L}(A^{\oplus})$ and $\mathcal{L}^{\ominus}(A) = \mathcal{L}(A^{\ominus})$ called the* accepted *and the* rejected language *by A, respectively.*

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.*

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, resp.) 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. As usual, in the deterministic case we will denote a dcNFA as $\langle Q, \Sigma, \delta, i, F^{\oplus}, F^{\ominus}\rangle$ and we will write $p = \delta(q, a)$ instead of $p \in \delta(q, a)$, when $\delta(q, a)$ is defined. The function δ can be made total, in a standard way, by inserting an extra state, called *trap* or *dead state*. However, while in DFAs this state is rejecting, according to our definition in the case of dcNFAs this state should be a *don't care* state. Hence, in a dcNFA with a partial transition function, a string $x \in \Sigma^*$ leading to an undefined transition is neither accepted nor rejected.

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.*

Since, as already observed, unspecified transitions have different meanings for DFAs (rejection) and for dcDFAs (*don't care* condition), while counting the number of the states in the case of DFAs we will add the trap state when the transition function is not total, while in the case of dcDFAs we will never add any extra state.

Example 3. Consider the dcDFA A represented in Figure 1. We label each 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, $\mathcal{L}^{\oplus}(A) = (a^3b^3)^{\star}(\varepsilon +$

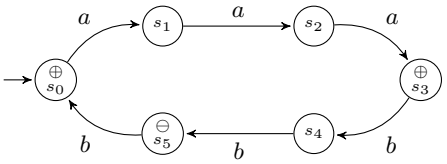

Fig. 1. The dcDFA A of Example 3.

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 an implicit 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 compatible 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 [7] 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 for obtaining minimal DFAs compatible with a given automata 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 fact that q is reachable, it can be immediately verified that those two languages are disjoint. For the same reason, applying the subset construction to A, it turns out that $L_\alpha^{\oplus} \cap L_\alpha^{\ominus} = \emptyset$ 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 suitable marking 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 [**?**], 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 4. *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 [7]. 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 will allow us to characterize DFAs that are compatible with A in terms of functions mapping states into cliques of the compatibility graph.

Theorem 5. *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 $q \in Q'$, $a \in \Sigma$, $\delta(\phi(q), a) \subseteq \phi\left(\delta'(q, a)\right)$,*
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 immediately from our choice 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 $p \in \delta(p', a)$. Furthermore, $q = \delta'(q', a)$ where $q' = \delta'(i', y)$. 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 5, 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 $\alpha \in Q'$, $\sigma \in \Sigma$, $\delta'(\alpha, \sigma)$ is a state $\beta \in Q'$ such that $\delta(\alpha, \sigma) \subseteq \beta$;
- the set F' of final states is a subset of Q' that contains those states α s.t. $\alpha \cap F^{\oplus} \neq \emptyset$ and does not contain those states 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 5. Hence, it turns out that each DFA A', defined as above, is compatible with A.

Example 6. Let us consider the dcDFA A of Example 3 (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.

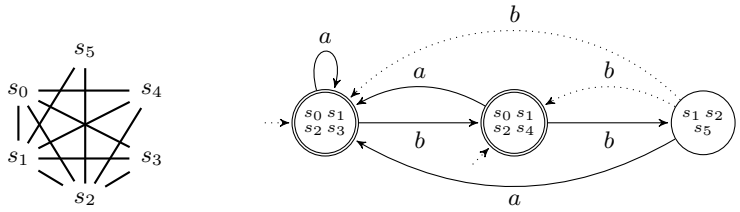

Fig. 2. The compatibility graph of the dcDFA A in Fig. 1 and four compatible DFAs.

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 5. For instance, further DFAs, compatible with the dcDFA A of Example 6, are depicted in Figure 3. We

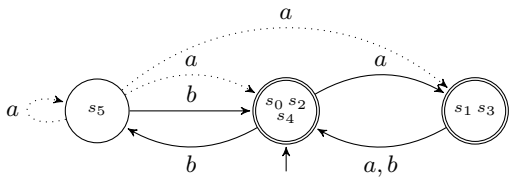

Fig. 3. More DFAs compatible with the dcDFA A in Figure 2.

can observe that in this example the compatibility graph of A cannot be 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 7. 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 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

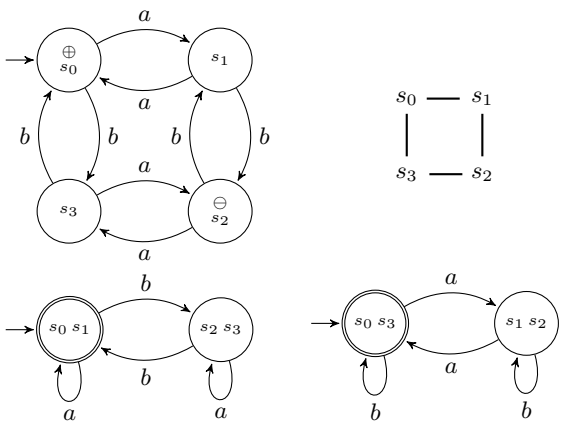

Fig. 4. The dcDFA A of Example 7 with its compatibility graph, and two compatible DFAs.

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 5 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 7 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 terms of cliques of the compatibility graph:

Theorem 8. *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 covering the compatibility graph of A.*

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 [7].

Theorem 9. *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.

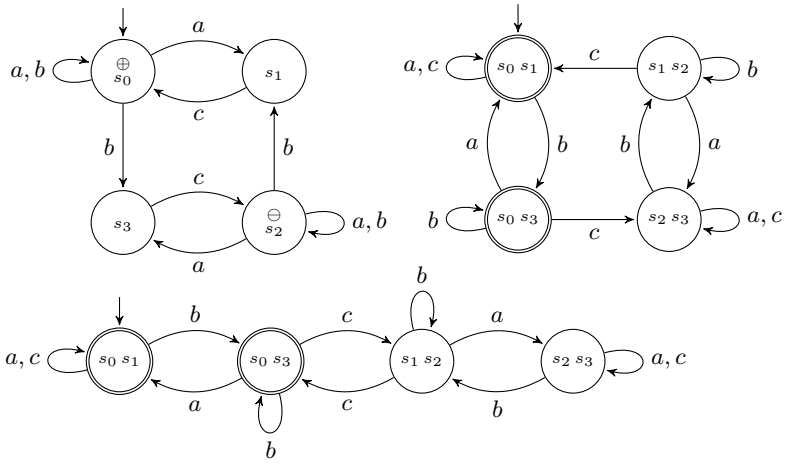

Fig. 5. The dcNFA $\widehat{A}$ of Example 7 (top left), with two compatible DFAs (top right and bottom).

Proof. The upper bound immediately derives from Theorem 8 and from a result by Moon and Moser [10] 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 [7], 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. (See Figure 6 for the case of n multiple of 3.) Since SVFAs with multiple initial states are a special case of dcNFAs, the claimed result follows. □

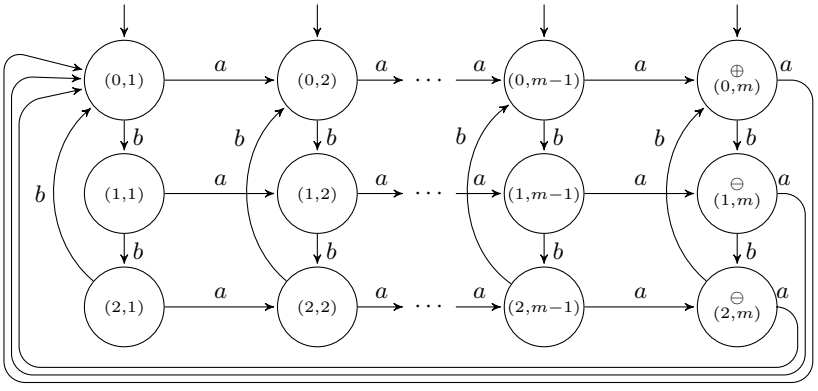

Fig. 6. Automaton A_n of Theorem 9 in the case of n multiple of 3.

The optimality proof in Theorem 9 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 9.

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 9, 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. 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 show that each DFA compatible with A'_n requires $f(n)$ states, where f is the function given in Theorem 9, by considering the following general lemma:

Lemma 10. *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.*

Theorem 11. *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 9.*

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 9 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 8, 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 both cases we prove NP-completeness. Our starting point is the following problem, which has been proved to be NP-complete by Pfleeger [13]:

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. From this discussion we immediately obtain the following result.

Theorem 12. *The problem of deciding if, given a dcDFA A and an integer $k > 0$, there exists a compatible DFA with at most k states is NP-hard.*

We note that the same result could also be deduced using NP-completeness of the inference of a DFA from a finite set of words [6]. We can also easily prove that the problem belongs to NP. However, we can do better, by proving that the problem is in NP even if A is nondeterministic. This allows us to obtain the main result of this section:

Theorem 13. *The problem of deciding if, given a dcNFA A and an integer $k > 0$, there exists a compatible DFA with at most k states is NP-complete.*

Proof. 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 (nondeterministic) 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.

NP-hardness follows from Theorem 12. □

6 The Unary Case

As well-known, in the unary case (namely the case of languages and automata defined over a one letter alphabet, which in the following we assume to be $\Sigma = \{a\}$) many state bounds are lower than in the general case. In this section we shortly present some considerations in this respect for dcNFAs. First of all, using standard results on diophantine equations, we can prove the following lemma:

Lemma 14. *A unary dcNFA 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.*

The maximal state gap between n-state unary NFAs and equivalent DFAs is $e^{\Theta(\sqrt{n \ln n})}$ [2]. Using Lemma 14, it is possible to show that the same gap cannot be reached starting from unary dcNFAs and compatible DFAs. This happens even in the case of SVFAs. However, it has been shown that the state gap between unary n-state SVFAs and DFAs grows at least as $e^{\Omega(\sqrt[3]{n \ln^2 n})}$ [5]. Hence, there is at least the same 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 $e^{\Omega(\sqrt[3]{n \ln^2 n})}$.*

References

1. Chen, Y.F., Farzan, A., Clarke, E.M., Tsay, Y.K., Wang, B.Y.: Learning minimal separating DFA's for compositional verification. In: Kowalewski, S., Philippou, A. (eds.) Proc. TACAS 2009. LNCS, vol. 5505, pp. 31–45. Springer (2009)
2. Chrobak, M.: Finite automata and unary languages. Theor. Comput. Sci. 47(3), 149–158 (1986)
3. Damiani, M.: The state reduction of nondeterministic finite-state machines. IEEE Trans. CAD 16(11), 1278–1291 (1997)
4. Eisinger, J., Klaedtke, F.: 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 (2008), http://dx.doi.org/10.1007/s10703-008-0057-6
5. Geffert, V., Pighizzini, G.: Pairs of complementary unary languages with "balanced" nondeterministic automata. Algorithmica 63(3), 571–587 (2012)
6. Gold, E.M.: Complexity of automaton identification from given data. Inf. Contr. 37(3), 302–320 (Dec 1978)
7. Jirásková, G., Pighizzini, G.: Optimal simulation of self-verifying automata by deterministic automata. Inf. Comput. 209(3), 528–535 (2011)
8. Kam, T., Villa, T., Brayton, R., Sangiovanni-Vincentelli, A.: A fully implicit algorithm for exact state minimization. Proc. ACM/IEEE Design Automation Conf. pp. 684–690 (1994)
9. Kam, T., Villa, T., Brayton, R., Sangiovanni-Vincentelli, A.: Theory and algorithms for state minimization of nondeterministic FSMs. IEEE Trans. CAD 16(11), 1311–1322 (Nov 1997)
10. Moon, J., Moser, L.: On cliques in graphs. Israel J. Math 3, 23—-28 (1965)
11. Paull, M.C., Unger, S.H.: Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans. on Elect. Comput. 3, 356–367 (1959)
12. Pena, J.M., Oliveira, A.L.: A new algorithm for exact reduction of incompletely specified finite state machines. IEEE Trans. CAD 18(11), 1619–1632 (1999)
13. Pfleeger, C.P.: State reduction in incompletely specified finite-state machines. IEEE Trans. Comput. 22(C), 1099–1102 (Dec 1973)
14. Rho, J.K., Hachtel, G., Somenzi, F., Jacoby, R.: Exact and heuristic algorithms for the minimization of incompletely specified state machines. IEEE Trans. CAD 13, 167–177 (1994)