IFIG Research Report

Institut für Informatik

Minimal and Hyper-Minimal Biautomata

Markus Holzer Sebastian Jakobi

IFIG Research Report 1401
March 2014

Institut für Informatik
JLU Gießen
Arndtstraße 2
35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

IFIG Research Report
IFIG Research Report 1401, March 2014

Minimal and Hyper-Minimal Biautomata

Markus Holzer[1] and Sebastian Jakobi[2]

Institut für Informatik, Universität Giessen
Arndtstraße 2, 35392 Giessen, Germany

Abstract. We compare deterministic finite automata (DFAs) and biautomata under the following two aspects: structural similarities between minimal and hyper-minimal automata, and computational complexity of the minimization and hyper-minimization problem. Concerning classical minimality, the known results such as isomorphism between minimal DFAs, and NL-completeness of the DFA minimization problem carry over to the biautomaton case. But surprisingly this is not the case for hyper-minimization: the similarity between almost-equivalent hyper-minimal biautomata is not as strong as it is between almost-equivalent hyper-minimal DFAs. Moreover, while hyper-minimization is NL-complete for DFAs, we prove that this problem turns out to be computationally intractable, i.e., NP-complete, for biautomata.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Reducibility and completeness*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*;

Additional Key Words and Phrases: biautomata, almost-equivalence, hyper-minimization, computational complexity

[1]E-mail: holzer@informatik.uni-giessen.de
[2]E-mail: sebastian.jakobi@informatik.uni-giessen.de

1 Introduction

The minimization problem for finite automata is well studied in the literature, see, e.g., [9] for a recent overview on some automata related problems. The problem asks for the smallest possible finite automaton that is equivalent to a given one. Because regular languages are used in many applications and one may like to represent the languages succinctly, this problem is also of practical relevance. It is well known that for a given n-state deterministic finite automaton (DFA) one can efficiently compute an equivalent minimal automaton in $O(n \log n)$ time [11]. More precisely, the DFA minimization problem is complete for NL, even for DFAs without inaccessible states [4]. On the other hand, minimization of nondeterministic finite automata (NFAs) is highly intractable, namely PSPACE-complete [14]. These results go along with the structural properties of minimal finite automata. While minimal DFAs are unique up to isomorphism, this is not the case for minimal nondeterministic state devices anymore [1]. In fact, the characterization of minimal DFAs is one of the basic building blocks for efficient DFA minimization algorithms.

When changing from minimization to hyper-minimization a quite similar picture as mentioned above emerges. Hyper-minimization asks for the smallest automaton that is equivalent to a given one up to a finite number of exceptions—this form of "equivalence" is referred to as almost-equivalence in the literature. Let us discuss the situation for hyper-minimal DFAs and NFAs in more detail. First, all of the above mentioned computational complexity results remain valid for hyper-minimization. Thus, computing a hyper-minimal DFA can be done in $O(n \log n)$ time [10] and the hyper-minimization problem is NL-complete [6]. In fact it is known that minimization for DFAs linearly reduces to hyper-minimization [10]. Moreover, the intractability result for NFAs remains, that is, hyper-minimization for NFAs is PSPACE-complete [6], just as it is for ordinary NFA minimization. What can be said about the structural properties of hyper-minimal finite state machines? Neither hyper-minimal DFAs nor hyper-minimal NFAs are unique up to isomorphism. Nevertheless, hyper-minimal DFAs obey a structural characterization as shown in [2]. Almost-equivalent hyper-minimal DFAs have isomorphic kernels and isomorphic preambles up to state acceptance. Here the kernel of an automaton consists of the states that are reachable from the start state by an *infinite* number of inputs; all other states belong to the preamble of the automaton.

Recently, an alternative automaton model to deterministic finite automata, the so called *biautomaton* (DBiA) [19] was introduced. Roughly speaking, a biautomaton consists of a *deterministic* finite control, a read-only input tape, and two reading heads, one reading the input from left to right (forward transitions), and the other head reading the input from the opposite direction, i.e., from right to left (backward transitions). An input word is accepted by a biautomaton, if there is an accepting computation starting the heads on the two ends of the word meeting somewhere in an accepting state. Although the choice of reading a symbol by either head is nondeterministic, a deterministic outcome of the computation of the biautomaton is enforced by two properties: (i) The heads read input symbols independently, i.e., if one head reads a symbol and

the other reads another, the resulting state does not depend on the order in which the heads read these single letters. (ii) If in a state of the finite control one head accepts a symbol, then this letter is accepted in this state by the other head as well. Later we call the former property the $\diamond$-property and the latter one the F-property. In [19] and a series of forthcoming papers [7, 8, 15, 18] it was shown that biautomata share a lot of properties with ordinary finite automata. For instance, as minimal DFAs, also minimal DBiAs are unique up to isomorphism [19]. Moreover, in [7] it was shown that classical DFA minimization algorithms can be adapted to biautomata as well. As a first result we show that biautomaton minimization is NL-complete as for ordinary DFAs.

Now the question arises, which of the structural similarities between almost-equivalent or hyper-minimal DFAs similarly hold for biautomata as well? Moreover, what can be said about the computational complexity of biautomaton hyper-minimization? We give answers to both questions in the forthcoming. Some of the structural similarities found for almost-equivalent and hyper-minimal DFAs carry over to the case of biautomata, but there are subtle differences. On the one hand we show that the kernel isomorphism for almost-equivalent DFAs carries over to almost-equivalent biautomata, but on the other hand, the isomorphism for the preamble for almost-equivalent hyper-minimal DFAs does not transfer to the biautomaton case. In fact, we present an example, of two almost-equivalent hyper-minimal biautomata the preambles of which are not isomorphic at all—observe, that the size of both preambles must be the same due to the hyper-minimality of the devices, and the kernel isomorphism. The observed phenomenon is related to the structure of the almost-equivalence classes of hyper-minimal biautomata. In contrast to hyper-minimal DFAs, where two different but almost-equivalent states can only appear in the kernel, the induced almost-equivalence classes in case of hyper-minimal biautomata may in addition also span between preamble and kernel states, or even between two preamble states. Later we use this fact in order to prove the main result of this paper, namely that hyper-minimizing biautomata is *not* as easy as for DFAs. More precisely, we show that hyper-minimization for biautomata is NP-complete. This is in sharp contrast to the case of hyper-minimal DFAs.

2 Preliminaries

A *deterministic finite automaton* (DFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of *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 $\delta\colon Q \times \Sigma \to Q$ is the *transition function*. The *language accepted* by the deterministic finite automaton A is $L(A) = \{\, w \in \Sigma^* \mid \delta(q_0, w) \in F \,\}$, where the transition function is recursively extended to $\delta\colon Q \times \Sigma^* \to Q$ as usual.

A *deterministic biautomaton* (DBiA) is a sixtuple $A = (Q, \Sigma, \cdot, \circ, q_0, F)$, where Q, Σ, q_0, and F are defined as for DFAs, and where $\cdot$ and $\circ$ are mappings from $Q \times \Sigma$ to Q, called the *forward* and *backward transition function*, respectively. It is common in the literature on biautomata to use an infix notation for these functions, i.e., writing $q \cdot a$ and $q \circ a$ instead of $\cdot(q, a)$ and $\circ(q, a)$. Similar as for the transition function of a DFA, the forward transition function $\cdot$

can be extended to $\cdot\colon Q \times \Sigma^* \to Q$ by $q \cdot \lambda = q$, and $q \cdot av = (q \cdot a) \cdot v$, for all states $q \in Q$, symbols $a \in \Sigma$, and words $v \in \Sigma^*$. The extension of the backward transition function $\circ$ to $\circ\colon Q \times \Sigma^* \to Q$ is defined as follows: $q \circ \lambda = q$ and $q \circ va = (q \circ a) \circ v$, for all states $q \in Q$, symbols $a \in \Sigma$, and words $v \in \Sigma^*$. Notice that $\circ$ consumes the input from right to left, hence the name backward transition function. The DBiA A *accepts* a word $w \in \Sigma^*$ if there are words $u_i, v_i \in \Sigma^*$, for $1 \leq i \leq k$, such that w can be written as $w = u_1 u_2 \dots u_k v_k \dots v_2 v_1$, and

$$((\dots((((q_0 \cdot u_1) \circ v_1) \cdot u_2) \circ v_2) \dots) \cdot u_k) \circ v_k \in F.$$

The language accepted by A is $L(A) = \{\, w \in \Sigma^* \mid A \text{ accepts } w \,\}$.

The DBiA A has the *$\diamond$-property*, if $(q \cdot a) \circ b = (q \circ b) \cdot a$, for all $q \in Q$ and $a, b \in \Sigma$, and it has the *F-property*, if we have $q \cdot a \in F$ if and only if $q \circ a \in F$, for all $q \in Q$ and $a \in \Sigma$. The biautomata as introduced in [19] always had to satisfy *both* these properties, while in [7, 8] also biautomata that lack one or both of these properties, as well as nondeterministic biautomata were studied. Throughout the current paper, when writing of biautomata, or DBiAs, we always mean deterministic biautomata that satisfy both the $\diamond$-property, and the F-property, i.e., the model as introduced in [19]. For such biautomata the following is known from the literature [7, 19]:

- $(q \cdot u) \circ v = (q \circ v) \cdot u$, for all states $q \in Q$ and words $u, v \in \Sigma^*$,
- $(q \cdot u) \circ vw \in F$ if and only if $(q \cdot uv) \circ w \in F$, for all states $q \in Q$ and words $u, v, w \in \Sigma^*$.

From this one can conclude that for all words $u_i, v_i \in \Sigma^*$, with $1 \leq i \leq k$, we have

$$((\dots((((q_0 \cdot u_1) \circ v_1) \cdot u_2) \circ v_2) \dots) \cdot u_k) \circ v_k \in F$$

if and only if

$$q_0 \cdot u_1 u_2 \dots u_k v_k \dots v_2 v_1 \in F.$$

Therefore, the language accepted by a biautomaton A can as well be defined as $L(A) = \{\, w \in \Sigma^* \mid q_0 \cdot w \in F \,\}$.

In the following we define the two DFAs contained in a DBiA, which accept the language, and the reversal of the language accepted by the biautomaton. Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ be a DBiA. We denote by Q_{fwd} (Q_{bwd}, respectively) the set of all states reachable from q_0 by only using forward (backward, respectively) transitions, i.e.,

$$Q_{\text{fwd}} = \{\, q \in Q \mid \exists u \in \Sigma^* : q_0 \cdot u = q \,\},$$

and

$$Q_{\text{bwd}} = \{\, q \in Q \mid \exists v \in \Sigma^* : q_0 \circ v = q \,\}.$$

Now we define the DFA $A_{\text{fwd}} = (Q_{\text{fwd}}, \Sigma, \delta_{\text{fwd}}, q_0, F_{\text{fwd}})$, with $F_{\text{fwd}} = Q_{\text{fwd}} \cap F$, and where $\delta_{\text{fwd}}(q, a) = q \cdot a$, for all states $q \in Q_{\text{fwd}}$ and symbols $a \in \Sigma$. Similarly, we define the DFA $A_{\text{bwd}} = (Q_{\text{bwd}}, \Sigma, \delta_{\text{bwd}}, q_0, F_{\text{bwd}})$, with $F_{\text{bwd}} = Q_{\text{bwd}} \cap F$, and $\delta_{\text{bwd}}(q, a) = q \circ a$, for all $q \in Q$ and $a \in \Sigma$. One readily sees that $L(A_{\text{fwd}}) = L(A)$. Moreover, since $q \circ uv = (q \circ v) \circ u$, one can also see $L(A_{\text{bwd}}) = L(A)^R$.

For a state q of an automaton A (DFA or DBiA), the *right language* of q is the language $L_A(q)$ accepted by the automaton that is obtained from A by making q its initial state. Notice that the right language of the initial state q_0 of A is $L_A(q_0) = L(A)$. We say that two automata A and A' are *equivalent*, denoted by $A \equiv A'$, if $L(A) = L(A')$. Similarly, if q is a state of A and q' a state of A', then q and q' are *equivalent*, for short $q \equiv q'$, if $L_A(q) = L_{A'}(q')$. An automaton A is *minimal* if there is no automaton B of the same type, that has fewer states than A and satisfies $A \equiv B$.

Let L be a language over Σ and let $u, v \in \Sigma^*$. The *left derivative* of L by u is the language $u^{-1}L = \{\, w \in \Sigma^* \mid uw \in L \,\}$, and the *right derivative* of L by v is $Lv^{-1} = \{\, w \in \Sigma^* \mid wv \in L \,\}$. Notice that $u^{-1}(Lv^{-1}) = (u^{-1}L)v^{-1}$, so we may denote *both-sided derivatives* by $u^{-1}Lv^{-1} = \{\, w \in \Sigma^* \mid uwv \in L \,\}$. Derivatives are used in [19] for the definition of the canonical biautomaton of a regular language, which is similar to the canonical DFA as described in [3]. The set of states of the canonical biautomaton for a regular language L consists of all derivatives of L—this is a finite set because L is regular—and the right language of a state $u^{-1}Lv^{-1}$ is the language $u^{-1}Lv^{-1}$. We often use regular expressions to describe languages—see, e.g., [12]. As usual we identify an expression with the language it describes, and by abuse of notation we also use regular expressions as names for states.

Recently, the notions of almost-equivalence and hyper-minimality were introduced [2]. Two languages L and L' are *almost-equivalent*, denoted by $L \sim L'$, if their symmetric difference $L \triangle L' := (L \setminus L') \cup (L' \setminus L)$ is finite. This notion naturally carries over to automata and states: two automata A and A' are *almost-equivalent*, for short $A \sim A'$, if $L(A) \sim L(A')$, and two states q and q' of A and, respectively, A' are *almost-equivalent*, for short $q \sim q'$, if $L_A(q) \sim L_{A'}(q')$. An automaton A is *hyper-minimal* if there is no automaton B of the same type, that has fewer states than A and satisfies $A \sim B$. A useful concept for the study of almost-equivalent automata is the partitioning of the state set into preamble and kernel states. A state q of an automaton A is a *kernel state* if it is reachable from the initial state of A by an infinite number of inputs, otherwise q is a *preamble state*. In case A is a biautomaton over alphabet Σ, this means that q is a kernel state if and only if there are infinitely many pairs of words $u, v \in \Sigma^*$ such that $(q_0 \cdot u) \circ v = q$—here q_0 is the initial state, and $\cdot$ and $\circ$ are the transition functions of A. The set of all preamble states of A is denoted by $\mathrm{Pre}(A)$, and the set of kernel states is $\mathrm{Ker}(A)$.

We assume familiarity with the basic concepts of complexity theory [12, 20] such as reductions, completeness, and the inclusion chain $\mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP}$. Here NL is the set of problems accepted by nondeterministic logarithmic space bounded Turing machines. Moreover, let P (NP, respectively) denote the set of problems accepted by deterministic (nondeterministic, respectively) polynomial time bounded Turing machines.

3 Structural Similarity Between Minimal Automata

The well-known fact that two equivalent minimal DFAs are isomorphic can be formulated as follows.

Theorem 1. *Let $A = (Q, \Sigma, \delta, q_0, F)$ and $A' = (Q', \Sigma, \delta', q_0', F')$ be two minimal deterministic finite automata with $A \equiv A'$. Then there exists a mapping $h\colon Q \to Q'$ that is bijective, and that satisfies the following conditions:*

1. *$q \equiv h(q)$, for all $q \in Q$ (in particular $q \in F$ if and only if $h(q) \in F'$).*
2. *$h(q_0) = q_0'$.*
3. *$h(\delta(q, a)) = \delta'(h(q), a)$, for all $q \in Q$ and $a \in \Sigma$.*

Further, the following characterization of minimal DFAs is well known.

Theorem 2. *A deterministic finite automaton is minimal if and only if all its states are reachable, and there is no pair of distinct, but equivalent states.*

An isomorphism as in Theorem 1 can also be found between equivalent minimal biautomata, which follows from results from [19].

Theorem 3. *Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ and $A' = (Q', \Sigma, \cdot', \circ', q_0', F')$ be two minimal biautomata[1] with $A \sim A'$. Then there exists a mapping $h\colon Q \to Q'$ that is bijective, and that satisfies the following conditions:*

1. *$q \equiv h(q)$, for all $q \in Q$ (in particular $q \in F$ if and only if $h(q) \in F'$).*
2. *$h(q_0) = q_0'$.*
3. *$h(q \cdot a) = h(q) \cdot' a$, and $h(q \circ a) = h(q) \circ' a$, for all $q \in Q$ and $a \in \Sigma$.*

Also the following characterization of minimal DBiAs, which is similar to Theorem 2 for DFAs, was shown in [7].

Theorem 4. *A biautomaton is minimal if and only if all its states are reachable, and there is no pair of distinct, but equivalent states.*

We can draw another connection between biautomata and finite automata. Recall that any DBiA A contains the two DFAs A_{fwd} and A_{bwd}, accepting the languages $L(A_{\text{fwd}}) = L(A)$ and $L(A_{\text{bwd}}) = L(A)^R$. In fact, if A is a minimal biautomaton, then the two contained DFAs are minimal, too, as the following result shows.

Lemma 5. *Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ be a minimal biautomaton. Then A_{fwd} is a minimal deterministic finite automaton for $L(A)$ and A_{bwd} is a minimal deterministic finite automaton for $L(A)^R$.*

Proof. Notice that if q is a state of A_{fwd}, then $L_{A_{\text{fwd}}}(q) = L_A(q)$, and if q is a state of A_{bwd}, then $L_{A_{\text{bwd}}}(q) = L_A(q)^R$. Therefore, if A_{fwd} or A_{bwd} contains a pair of equivalent states, then these states are also equivalent in the biautomaton A. Now if A is a minimal biautomaton, then Theorem 4 implies that it does not contain a pair of distinct, but equivalent states. Therefore also the DFAs A_{fwd}, and A_{bwd} contain no such pair. Since by definition all states of A_{fwd} and A_{bwd} are reachable, both DFAs must be minimal, due to Theorem 2. □

[1] Remember that throughout this paper a biautomaton is always a deterministic biautomaton which satisfies both the $\diamond$-property and the F-property.

The following example shows that the converse of Lemma 5 is not true, which means that a DBiA A where both DFAs A_{fwd} and A_{bwd} are minimal needs not to be minimal itself.

Example 6. Consider the biautomaton $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ with the state set $Q = \{q_0, q_1, \ldots, q_6\}$, initial state q_0, final states $F = \{q_2, q_4, q_5\}$ and the transition functions $\cdot$ and $\circ$ of which can be read from Figure 1—solid arrows denote forward transitions by $\cdot$, and dashed arrows denote backward transitions by $\circ$. Obviously the three accepting states q_2, q_4, and q_5 are equivalent, so we

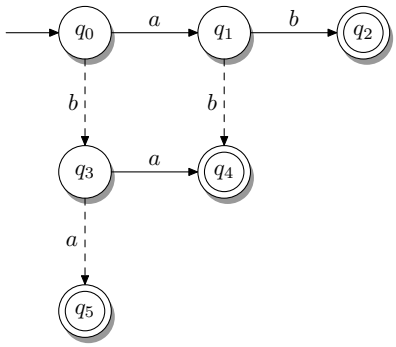

Fig. 1. A non-minimal biautomaton A where both contained DFAs A_{fwd} and A_{bwd} are minimal. The sink state q_6 and all transitions to it are not shown.

know by Theorem 4 that this is not a minimal biautomaton. However, one can easily see that both contained DFAs A_{fwd} and A_{bwd} are minimal. This shows that the converse of Lemma 5 does not hold.

4 Structural Similarity Between Hyper-Minimal Automata

The notions of almost-equivalence and hyper-minimality were introduced in [2]. There it was shown that two almost-equivalent hyper-minimal DFAs are isomorphic in their kernels, and isomorphic in their preambles (up to acceptance values of preamble states). The following theorem, which summarizes results from [2], should be compared to the corresponding Theorem 1 for equivalent minimal DFAs.

Theorem 7. *Let $A = (Q, \Sigma, \delta, q_0, F)$ and $A' = (Q', \Sigma, \delta', q_0', F')$ be two minimal deterministic finite automata with $A \equiv A'$. Then there exists a mapping $h\colon Q \to Q'$ satisfying the following conditions.*

1. *If $q \in \mathrm{Pre}(A)$ then $q \sim h(q)$, and if $q \in \mathrm{Ker}(A)$ then $q \equiv h(q)$.*
2. *If $q_0 \in \mathrm{Pre}(A)$ then $h(q_0) = q_0'$, and if $q_0 \in \mathrm{Ker}(A)$ then $h(q_0) \sim q_0'$.*
3. *The restriction of h to $\mathrm{Ker}(A)$ is a bijection between the kernels of A and A', that is compatible with taking transitions:*
   - 3.a *We have $h(\mathrm{Ker}(A)) = \mathrm{Ker}(A')$, and if $q_1, q_2 \in \mathrm{Ker}(A)$ with $h(q_1) = h(q_2)$ then $q_1 = q_2$.*
   - 3.b *We have $h(\delta(q, a)) = \delta'(h(q), a)$, for all $q \in \mathrm{Ker}(A)$ and all $a \in \Sigma$.*

Further, if A and A' are hyper-minimal then also the following condition holds.

4. *The restriction of h to $\mathrm{Pre}(A)$ is a bijection between the preambles of A and A', that is compatible with taking transitions, except for transitions from preamble to kernel:*
   - 4.a *We have $h(\mathrm{Pre}(A)) = \mathrm{Pre}(A')$, and if $q_1, q_2 \in \mathrm{Pre}(A)$ with $h(q_1) = h(q_2)$ then $q_1 = q_2$.*
   - 4.b *We have $h(\delta(q,a)) = \delta'(h(q),a)$, for all $q \in \mathrm{Pre}(A)$ and all $a \in \Sigma$, that satisfy $\delta(q,a) \in \mathrm{Pre}(A)$.*

Notice that the bijection between the preamble states does not preserve finality of states. Further, the mapping h does not necessarily respect the transitions from preamble states to kernel states—see Condition 4.b of Theorem 7. Thus, two almost-equivalent hyper-minimal DFAs can differ in the following:

- acceptance values of preamble states,
- transitions leading from preamble to kernel states,
- the initial state, if the preamble is empty.

However, the transitions connecting preamble and kernel of almost-equivalent DFAs cannot differ arbitrarily. Assume that we have a state $q \in \mathrm{Pre}(Q)$, and some symbol $a \in \Sigma$, such that $\delta(q,a) \in \mathrm{Ker}(Q)$. Then it could be that the two states $h(\delta(q,a))$ and $\delta'(h(q),a)$ are different, but they must at least be almost-equivalent. This follows from the following result from [2].

Lemma 8. *Let $A = (Q, \Sigma, \delta, q_0, F)$ and $A' = (Q', \Sigma, \delta', q_0', F')$ be two (not necessarily distinct) deterministic finite automata, with $q \in Q$ and $q' \in Q'$. Then $q \sim q'$ if and only if $\delta(q,w) \sim \delta'(q',w)$, for all $w \in \Sigma^*$. Moreover, $q \sim q'$ implies $\delta(q,w) \equiv \delta'(q',w)$, for all words $w \in \Sigma^*$ with $|w| \geq k = |Q \times Q'|$.*

Also a characterization of hyper-minimal DFAs, which is similar to Theorem 2, was shown in [2]:

Theorem 9. *A deterministic finite automaton is hyper-minimal if and only if it is minimal, and there is no pair of distinct but almost-equivalent states such that one of them is in the preamble.*

Now let us investigate, which of the structural similarity results for almost-equivalent hyper-minimal DFAs carry over to biautomata. We first show that a result similar to Lemma 8 also holds for biautomata.

Lemma 10. *Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ and $A' = (Q', \Sigma, \cdot', \circ', q_0', F')$ be two biautomata. Let $q \in Q$ and $q' \in Q'$, then $q \sim q'$ if and only if $(q \cdot u) \circ v \sim (q' \cdot' u) \circ' v$, for all words $u, v \in \Sigma^*$. Moreover, $q \sim q'$ implies $(q \cdot u) \circ v \equiv (q' \cdot' u) \circ' v$, for all words $u, v \in \Sigma^*$ with $|uv| \geq k = |Q \times Q'|$.*

Proof. Assume there are words $u, v \in \Sigma^*$ such that the states $p = (q \cdot u) \circ v$ and $p' = (q' \cdot' u) \circ' v$ are not almost-equivalent. This means that there are infinitely many words $w \in L_A(p) \triangle L_{A'}(p')$, which implies that there are infinitely many words $uwv \in L_A(q) \triangle L_{A'}(q')$. Thus states q and q' are *not*

almost-equivalent. Therefore, if $(q \cdot u) \circ v \sim (q' \cdot' u) \circ' v$, for all $u, v \in \Sigma^*$, then states q and q' are almost-equivalent. The reverse implication is trivial: if $(q \cdot u) \circ v \sim (q' \cdot' u) \circ' v$, for every $u, v \in \Sigma^*$, then we obtain $q \sim q'$ by choosing $u = v = \lambda$. This proves the first part of the Lemma.

For the second part assume $q \sim q'$, and consider two words $u = a_1 a_2 \dots a_\ell$ and $v = a_m \dots a_{\ell+2} a_{\ell+1}$, with $a_1, a_2, \dots, a_m \in \Sigma$ such that $|uv| = m \geq k$. Consider the sequence of state pairs (q_i, q'_i), for $0 \leq i \leq m$, that the automata pass through in their computations $(q \cdot u) \circ v$ and $(q' \cdot' u) \circ' v$:

$$(q_i, q'_i) = \begin{cases} (q, q'), & \text{for } i = 0, \\ (q_{i-1} \cdot a_i,\ q'_{i-1} \cdot' a_i) & \text{for } 1 \leq i \leq \ell, \\ (q_{i-1} \circ a_i,\ q'_{i-1} \circ' a_i) & \text{for } \ell + 1 \leq i \leq m. \end{cases}$$

Because $m \geq |Q \times Q'|$, there must be integers i, j with $0 \leq i < j \leq m$, for which we have $(q_i, q'_i) = (q_j, q'_j)$.

If $j \leq \ell$ then the word u can be written as $u = u_1 u_2 u_3$ such that

$$\begin{array}{lll} q \cdot u_1 = q_i, & q_i \cdot u_2 = q_i, & (q_i \cdot u_3) \circ v = p, \\ q' \cdot' u_1 = q'_i, & q'_i \cdot' u_2 = q'_i, & (q'_i \cdot' u_3) \circ' v = p'. \end{array}$$

If the states $p = (q \cdot u) \circ v$ and $p' = (q' \cdot' u) \circ' v$ are not equivalent, then there is a word $w \in L_A(p) \triangle L_{A'}(p')$, and it follows that $u_1 u_2^n u_3 w v \in L_A(q) \triangle L_{A'}(q')$, for all $n \geq 0$. This is a contradiction to $q \sim q'$.

If $\ell + 1 \leq i$ then we can find a similar partition of the word $v = v_3 v_2 v_1$, such that a word w in the symmetric difference of the two states $p = (q \cdot u) \circ v$ and $p' = (q' \cdot' u) \circ' v$ induces infinitely many words $u w v_3 v_2^n v_1 \in L_A(q) \triangle L_{A'}(q')$, for all $n \geq 0$.

It remains to discuss the case $i \leq \ell < j$. Now the words u and v can be written as $u = u_1 u_2$ and $v = v_2 v_1$ such that

$$\begin{array}{lll} q \cdot u_1 = q_i, & (q_i \cdot u_2) \circ v_1 = q_i, & q_i \circ v_2 = p, \\ q' \cdot' u_1 = q'_i, & (q'_i \cdot' u_2) \circ' v_1 = q'_i, & q'_i \circ' v_2 = p'. \end{array}$$

Now, if the states $p = (q \cdot u) \circ v$ and $p' = (q' \cdot' u) \circ' v$ are not equivalent, then there is a word $w \in L_A(p) \triangle L_{A'}(p')$, and it follows that $u_1 u_2^n w v_2 v_1^n \in L_A(q) \triangle L_{A'}(q')$, for all $n \geq 0$. This is again a contradiction to $q \sim q'$, hence the two states $(q \cdot u) \circ v$ and $(q' \cdot' u) \circ' v$ must be equivalent. □

Now we come to a mapping between the states of two almost-equivalent biautomata. As in the case of finite automata, we can find an isomorphism between the kernels of the two automata. However, we cannot find a similar isomorphism between their preambles. Of course, two almost-equivalent hyper-minimal biautomata must have the same number of states, and if their kernels are isomorphic, then also their preambles must be of same size. But still we cannot always find a bijective mapping that preserves almost-equivalence, as in the case of finite automata. We will later see an example for this phenomenon, but first we present our result on the structural similarity between almost-equivalent minimal biautomata.

Theorem 11. *Let $A = (Q, \Sigma, \cdot, \circ, q_0, F)$ and $A' = (Q', \Sigma, \cdot', \circ', q_0', F')$ be two minimal biautomata with $A \sim A'$. There exists a mapping $h\colon Q \to Q'$ that satisfies the following conditions.*

1. *If $q \in \mathrm{Pre}(A)$ then $q \sim h(q)$, and if $q \in \mathrm{Ker}(A)$ then $q \equiv h(q)$.*
2. *If $q_0 \in \mathrm{Pre}(A)$ then $h(q_0) = q_0'$, and if $q_0 \in \mathrm{Ker}(A)$ then $h(q_0) \sim q_0'$.*
3. *The restriction of h to $\mathrm{Ker}(A)$ is a bijection between the kernels of A and A', that is compatible with taking transitions:*
   - 3.a *We have $h(\mathrm{Ker}(A)) = \mathrm{Ker}(A')$, and if $q_1, q_2 \in \mathrm{Ker}(A)$ with $h(q_1) = h(q_2)$ then $q_1 = q_2$.*
   - 3.b *We have $h(q \cdot a) = h(q) \cdot' a$ and $h(q \circ a) = h(q) \circ' a$, for all $q \in \mathrm{Ker}(A)$ and all $a \in \Sigma$.*

Proof. In order to define the mapping h we choose for every state $q \in Q$ tow words u_q and v_q as follows: If q is a kernel state of A then there are infinitely many pairs $u, v \in \Sigma^*$ such that $(q_0 \cdot u) \circ v = q$. Hence we can choose for every kernel state $q \in \mathrm{Ker}(A)$ two words u_q and v_q with $|u_q v_q| \geq k = |Q \times Q'|$ such that $q = (q_0 \cdot u_q) \circ v_q$. Moreover, for every preamble state $q \in \mathrm{Pre}(A)$, we fix some shortest words u_q and v_q, i.e., where $|u_q v_q|$ is shortest possible, such that also $(q_0 \cdot u_q) \circ v_q = q$. The mapping $h\colon Q \to Q'$ is then defined by $h(q) = (q_0' \cdot' u_q) \circ' v_q$. In the following we show that this mapping satisfies the statements of the theorem. Since $A \sim A'$ it must be $q_0 \sim q_0'$. If $q_0 \in \mathrm{Pre}(Q)$, then we have $u_{q_0} = v_{q_0} = \lambda$, and so $h(q_0) = q_0'$, which proves one part of Statement 2. From Lemma 10 we obtain $(q_0 \cdot u_q) \circ v_q \sim (q_0' \cdot' u_q) \circ' v_q$, for all preamble states $q \in \mathrm{Pre}(A)$, and further even $(q_0 \cdot u_q) \circ v_q \equiv (q_0' \cdot' u_q) \circ' v_q$, for all kernel states $q \in \mathrm{Ker}(A)$. This proves Statement 1. Now, if $q_0 \in \mathrm{Ker}(A)$, then $q_0' \sim q_0 \equiv h(q_0)$, which proves the other part of Statement 2. Moreover, for all kernel states $q \in \mathrm{Ker}(A)$ we have $q \equiv h(q)$, which implies

$$h(q \cdot a) \equiv q \cdot a \equiv h(q) \cdot' a \qquad \text{and} \qquad h(q \circ a) \equiv q \circ a \equiv h(q) \circ' a,$$

for all symbols $a \in \Sigma$. Since A' is a minimal biautomaton, it does not contain a pair of different, but equivalent states. Therefore it must be $h(q \cdot a) = h(q) \cdot' a$ and $h(q \circ a) = h(q) \circ' a$, for all $a \in \Sigma$, which proves Statement 3.b.

It remains to prove Statement 3.a, namely that the mapping is bijective between the kernels of the two DBiAs. If $q \in \mathrm{Ker}(A)$ then $|u_q v_q| \geq |Q'|$, so the computation path in A' that leads from q_0' to $h(q)$ must contain a cycle. Hence state $h(q)$ must be a kernel state of A', and we get $h(\mathrm{Ker}(A)) \subseteq \mathrm{Ker}(A')$. Moreover, the mapping h is injective, which can be seen as follows: if $h(p) = h(q)$, then we know $p \equiv h(p) = h(q) \equiv q$, but since A is a minimal biautomaton, this implies $p = q$. Therefore it is $|\mathrm{Ker}(A)| \leq |\mathrm{Ker}(A')|$. By exchanging the roles of the automata A and A' we can also find an injective mapping $h'\colon Q' \to Q$ that satisfies $h'(\mathrm{Ker}(A')) \subseteq \mathrm{Ker}(A)$, which in turn shows that $|\mathrm{Ker}(A')| \leq |\mathrm{Ker}(A)|$. Altogether we obtain $|\mathrm{Ker}(A)| = |\mathrm{Ker}(A')|$, so the mapping h must also be surjective on $\mathrm{Ker}(A')$. This concludes our proof. □

Notice that Theorem 11 requires the almost-equivalent DBiAs A and A' to be minimal, but not necessarily hyper-minimal. Of course, the theorem also

holds for hyper-minimal automata, since these are always minimal. However, the question is whether we can find more structural similarities—like Statement 4.b from Theorem 7 on DFAs—if both DBiAs are hyper-minimal. Unfortunately the answer is no, as the following example demonstrates.

Example 12. Consider the biautomaton A which is depicted on top in Figure 2—as usual, transitions which are not shown lead to a non-accepting sink state, which is also not shown. The state labels of the eight states in the lower two rows of the automaton denote the right languages of the respective states. The kernel of A consists of those states, and the sink state. The right languages of the states q_0, q_1, and q_2, which constitute the preamble of A, are as follows: $L_A(q_0) = L(A) = (a+b)ba^*b + c^*a$, $L_A(q_1) = ba^*b + \lambda$, and $L_A(q_2) = ba^*b$. One can verify that A satisfies the $\diamond$- and the F-property. Let us first show that A is hyper-minimal.

Claim. The biautomaton A depicted on top in Figure 2 is hyper-minimal.

Proof. Assume B is a minimal biautomaton that is almost-equivalent to A. We have to show that B has at least as many states as A. We know from Theorem 11, that the kernels of A and B are isomorphic, hence it suffices to show that B has at least three states in its preamble. Let us denote the initial state of B by q_0^B, and its forward and backward transition functions by $\cdot_B$ and $\circ_B$, respectively. We have $q_0^B \sim q_0$ because $B \sim A$, and by Lemma 10 follows $(q_0^B \cdot_B u) \circ_B v \sim (q_0 \cdot u) \circ v$, for all $u, v \in \Sigma^*$. Since state q_0 is not almost-equivalent to any kernel state of A, also state q_0^B is not a kernel state of B either—remember that the kernels of A and B are isomorphic. Further, also the state $q_1 = q_0 \cdot a$ is not almost-equivalent to any kernel state, and not almost-equivalent to q_0, therefore the state $q_0^B \cdot_B a$ must be a another preamble state in B, too. Let us denote this state by q_1^B.

If we can show that B has another preamble state, then we know that A is hyper-minimal. Therefore assume for the sake of contradiction, that q_0^B and q_1^B are the only preamble states of B. Because no kernel state of A is almost-equivalent to q_1, and due to the isomorphism between the kernels of A and B, state q_1^B is the only state of B that is almost-equivalent to state q_1 of automaton A. Because $q_1 \sim q_2$, state q_1^B is also the only state of B almost-equivalent to q_2. By Lemma 10, the state $q_2 = q_0 \cdot b$ of A must be almost-equivalent to state $q_0^B \cdot_B b$ of B, so we conclude $q_0^B \cdot_B b = q_1^B = q_0^B \cdot_B a$. Now we consider two cases, namely whether q_1^B is an accepting state, or not.

If $q_1^B = q_0^B \cdot_B a$ is not accepting, then also the state $q_0^B \circ_B a$ must not be accepting due to the F-property of B. However, state $q_0^B \circ_B a$ must be almost-equivalent to state $q_0 \circ a$, which is the *accepting* kernel state c^* in A. By Theorem 11, also the corresponding kernel state of B must be accepting, therefore this cannot be the target of the transition $q_0^B \circ_B a$. Since there is no other kernel state which is almost-equivalent to c^*, we conclude that the automaton B must have yet another preamble state $q_0^B \circ_B a$, different from q_0^B and q_1^B—a contradiction.

The other case is similar: if $q_1^B = q_0^B \cdot_B b$ is an accepting state, then also state $q_0^B \circ_B b$ must be accepting. Moreover, this state must be almost-equivalent

to the kernel state $q_0 \circ b$, i.e., the state $(a+b)ba^*$ of A. The corresponding kernel state of B is also non-accepting, so it cannot be the target of the transition $q_0^B \circ_B b$. Again, there is no other kernel state in B that is almost-equivalent to the state $(a+b)ba^*$, so B must possess another preamble state, different from q_0^B and q_1^B. This concludes our proof. □

Now consider the DBiA A', depicted on the bottom of Figure 2. This biautomaton accepts the language $L(A') = (a+b)ba^*b + cc^*a$, so it is almost-equivalent to A. Since A and A' have the same number of states and A is hyper-minimal, the automaton A' is hyper-minimal, too. Consider a mapping h from the states of A to the states of A', that satisfies the conditions of Theorem 11. Between the kernels of the automata, the mapping is clear. Moreover, since q_0 and q_0' are preamble states, it must be $h(q_0) = q_0'$. This can even be concluded if q_0 and q_0' were not the initial states, because state q_0' of A' is the only state that is equivalent to q_0, and h must satisfy $q \sim h(q)$ for all states q. With the same argumentation we obtain $h(q_1) = h(q_2) = q_2'$. The mapping h is now fully defined, so in this example, there is no other possible mapping from the states of A to the states of A' that preserves almost-equivalence.

Notice that mapping h is not a bijection between the preambles: because we have $h(q_1) = h(q_2) = q_2'$, it is not injective, and it neither is surjective, since no state of A is mapped to state q_1' of B. This shows that the bijection Condition 4.a of Theorem 7 for preambles of deterministic finite automata does not hold for biautomata.

Similarly, also Condition 4.b of Theorem 7 cannot be satisfied here, which is witnessed by the following. We have $h(q_0 \circ a) = h(c^*) = c^*$—here c^* in $h(c^*)$ denotes the kernel state of A, and c^* after the equation symbol denotes the kernel state of A'—but it is $h(q_0) \circ_B a = q_0' \circ_B a = q_1'$, so $h(q_0 \circ a) \neq h(q_0) \circ_B a$.

Of course there exist bijective mappings between the state sets of the two automata A and A', but none of these can preserve almost-equivalence because the corresponding almost-equivalence classes in the state sets are not always of same size. For example, there are two states in A that are almost-equivalent to q_1, namely q_1 itself and q_2, but in A' there is only state q_2' in its equivalence class.

In the previous example we have seen two hyper-minimal biautomata, where one biautomaton (the lower automaton from Figure 2) has an almost-equivalence class of states that is cut by the preamble-kernel border: the preamble state q_1' is almost-equivalent to the kernel state c^*. In the other biautomaton (the upper automaton from Figure 2) all almost-equivalence classes lie entirely in either the preamble or the kernel. Now one may ask, whether for a given biautomaton one can always find an almost-equivalent hyper-minimal biautomaton where no almost-equivalence class contains both a kernel and a preamble state. But even this is not possible: in the proof of the forthcoming Theorem 14 we will see a biautomaton where *every* almost-equivalent biautomaton must contain a preamble state that is almost-equivalent to some kernel state—cf. Figure 4.

Another question is whether two almost-equivalent states, from which one is a preamble state, can only differ in acceptance, i.e., whether their symmetric

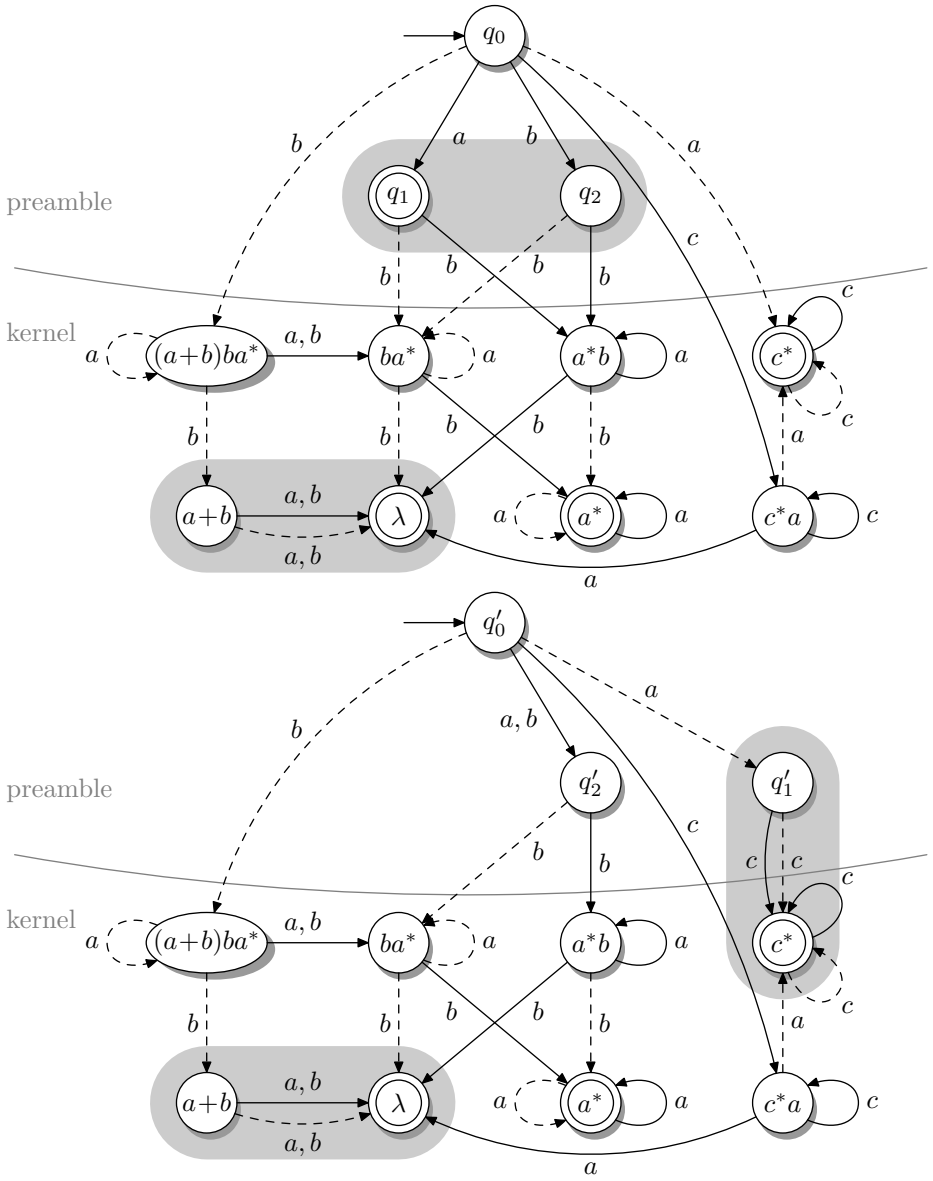

Fig. 2. Two hyper-minimal biautomata that are almost-equivalent. Biautomaton A (top) accepts the language $L(A) = (a + b)ba^*b + c^*a$, and A' (bottom) accepts the language $L(A') = (a+b)ba^*b+cc^*a$. The preambles are $\mathrm{Pre}(A) = \{q_0, q_1, q_2\}$ and $\mathrm{Pre}(A') = \{q'_0, q'_1, q'_2\}$. The states q_1, q_2, q'_1, and q'_2 have the following right languages: $L_A(q_1) = ba^*b + \lambda$, $L_A(q_2) = L_{A'}(q'_2) = ba^*b$, and $L_{A'}(q'_1) = cc^*$. The gray shading of state pairs denotes almost-equivalence, i.e., we have $q_1 \sim q_2$, $a + b \sim \lambda$, and $q'_1 \sim c^*$.

difference is at most $\{\lambda\}$. Also this is not the case, which can also be seen in the automaton from Figure 4, namely for the states q_1 and q_1'.

We have seen in Lemma 5 that the DFAs A_{fwd}, and A_{bwd} contained in a minimal DBiA A are minimal DFAs for the language $L(A)$, and $L(A)^R$, respectively. Notice that this relation does not hold if we consider hyper-minimal automata: the biautomaton A from Example 12 is hyper-minimal. But the contained DFA A_{fwd} is not hyper-minimal because the two preamble states q_1 and q_2 almost-equivalent, which contradicts the characterization of hyper-minimal DFAs by Theorem 9. In fact, one can check that for every hyper-minimal biautomaton B that is almost-equivalent to A, either B_{fwd} or B_{bwd} is not hyper-minimal.

Due to the lack of structural similarity in the preambles of almost-equivalent hyper-minimal biautomata, we do not hope for a nice characterization of hyper-minimal biautomaton, as we have seen in Theorems 2, 4, and 9. Another effect related to these unsatisfying structural properties of hyper-minimal biautomata will show up in the following section, where we show that hyper-minimizing biautomata is computationally hard.

5 Computational Complexity of (Hyper)-Minimization

Given a deterministic finite automaton, it is an easy task to construct an equivalent minimal automaton. A lot of minimization algorithms for DFAs are known, the most effective of them being Hopcroft's algorithm [11] with a running time of $O(n \log n)$, where n is the number of states of the input DFA. In fact, the decision version of the DFA minimization problem—given a DFA A and an integer n, decide whether there exists an n-state DFA B with $A \equiv B$—is NL-complete [4].

Concerning minimization of biautomata, it was discussed in [7] how classical DFA minimization techniques can also be applied to DBiAs. In the following we investigate the computational complexity of the minimization problem for biautomata, and show that it is NL-complete, too. For proving NL-hardness we give a reduction from the following variant of the graph reachability problem [16, 17] which is NL-complete, too.

> **Reachability:** given a directed graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$, where every vertex has at most two successors, and at most two predecessors, decide whether v_n is reachable from v_1.[2]

The next theorem reads as follows:

Theorem 13 (DBiA Minimization Problem). *The problem of deciding for a given biautomaton A, and an integer n, whether there exists an n-state biautomaton B with $A \equiv B$, is* NL*-complete.*

[2] The general graph reachability problem can be reduced to the case where every vertex has at most two successors by appending after each vertex that has more than two successors a small tree-like subgraph to simulate the multiple outgoing edges. A similar construction can be used to reduce the number of predecessors, to obtain a graph where also the number of predecessors of a vertex is at most two.

Proof. For the NL upper bound we use the following algorithm for computing the number k of equivalence classes of the state set of A. Let $q_1, q_2, \ldots q_m$ be some fixed order of the states of A, and initially set $k = 0$. For all states q_i (in ascending order) do the following: if $q_i \not\equiv q_j$, for all $j < i$, then increment k. Finally, if $k \leq n$ then the answer is yes, otherwise it is no. Because A has the $\diamond$-property, and the F-property, it suffices to consider only forward transitions to decide whether $q_i \not\equiv q_j$. Therefore, in order to check whether $q_i \not\equiv q_j$ holds, we can check equivalence of the two DFAs obtained from A by making q_i, and q_j, respectively, the initial state, and considering only forward transitions. Since (in-)equivalence of DFAs can be decide in NL, the whole algorithm can be implemented on a non-deterministic logarithmic space-bounded Turing machine.

For NL-hardness we give a reduction from the above described variant of the graph reachability problem. The idea is to transform the graph into a DFA A_1 that accepts the empty language if the instance of the graph reachability problem is a "no" instance, and it accepts a non-empty, and non-universal language otherwise. Then an equivalent DBiA A is built by a cross-product construction of A_1 and the reverse of A_1. The biautomaton A is equivalent to a single-state biautomaton if and only if the graph reachability problem is a "no" instance. The only problem in this approach is to make sure that the construction of the reverse of A_1 can be done by a logarithmic space-bounded Turing machine, because in general this construction induces an exponential blow-up in the number of states of the finite automaton. Therefore we construct the DFA A_1 such that its reversal is also a deterministic automaton.

Let $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$ be a directed graph where every vertex has at most two successors, and at most two predecessors. We construct the partial DFA $A_1 = (Q, \Sigma, \delta, q_0, F)$—here partial means that some transitions of A_1 may be undefined—over the alphabet $\Sigma = \{a, b\}$ as follows: The states set consists of the vertices and edges of G, i.e., $Q = V \cup E$, the initial state is $q_0 = v_1$, and set of final states is $F = \{v_n\}$. The transitions in states $v_i \in V$ are defined as follows:

- if v_i has two successors v_{j_1} and v_{j_2}, i.e., if $(v_i, v_{j_1}), (v_i, v_{j_2}) \in E$, and $j_1 < j_2$, then $\delta(v_i, a) = (v_i, v_{j_1})$ and $\delta(v_i, b) = (v_i, v_{j_2})$,
- if v_i has one successors v_j, i.e., if (v_i, v_j) is the only edge in E with v_i as on the left-hand side, then $\delta(v_i, a) = (v_i, v_j)$,

Finally, the transitions in states $(v_i, v_j) \in E$ are defined as follows:

- if (v_i, v_j) is the only edge in E with vertex v_j on the right-hand side, then $\delta((v_i, v_j), a) = v_j$,
- if there are two edges (v_{i_1}, v_j) and (v_{i_2}, v_j) in E with vertex v_j on the right-hand side, then $\delta((v_{i_1}, v_j), a) = v_j$ and $\delta((v_{i_2}, v_j), b) = v_j$.

All other transitions are undefined. Note that every state $(v_i, v_j) \in E$ has exactly one outgoing and one ingoing transition, and every state $v_i \in V$ has at most one outgoing and at most one ingoing transition for each alphabet symbol. Therefore, the reverse automaton $A_1^R = (Q, \Sigma, \delta^R, v_n, \{v_1\})$, where all transitions are reversed, and the initial and (single) final state are interchanged is also a partial

DFA. The biautomaton A can now be constructed by a cross-product construction, simulating A_1 in the first component using its forward transitions, and simulating A_1^R in the second component using backward transitions—see [19] for details of this construction. Clearly this construction can be realized by a logarithmic space-bounded deterministic Turing machine.

It remains to prove the correctness of the reduction. First assume that in the graph G the vertex v_n is *not* reachable from v_1. Then clearly the language $L(A_1) = L(A)$ is empty, so there exists a single-state biautomaton B that is equivalent to A. Next assume v_n is reachable from v_1 in G. Then clearly the language $L(A) = L(A_1)$ is not empty, and because $v_1 \neq v_n$ it is also not Σ^*. Therefore there exists *no* single-state biautomaton B that is equivalent to A. Since $\mathsf{NL} = \mathsf{coNL}$ [13, 21], the theorem is proven. □

Now we turn to hyper-minimization. For deterministic finite automata the situation is similar as in the case of classical minimization: efficient hyper-minimization algorithms with running time $O(n \log n)$ are known [5, 10], and it was shown in [6] that the hyper-minimization problem for DFAs is NL-complete. On the one hand, since classical DFA minimization methods also work well for DBiAs, one could expect that hyper-minimization of DBiAs is as easy as for ordinary DFAs. On the other hand, the problems related to the structure of hyper-minimal biautomata, which we discussed in Section 4, already give hints that hyper-minimization of DBiAs may not be so easy. In fact, we show in the following that the hyper-minimization problem for biautomata is NP-complete. To prove NP-hardness we give a reduction from the NP-complete MAX-2-SAT problem [20] which is defined as follows.

> MAX-2-SAT: given a Boolean formula φ in conjunctive normal form, where each clause has exactly two literals, and an integer k, decide whether there exists an assignment that satisfies at least k clauses of φ.

Before we give a detailed proof of NP-hardness, we want to describe the key idea of the reduction. Given as instance of MAX-2-SAT a formula φ and number k, we construct a DBiA A_φ such that for every clause that can be satisfied in φ, we can save one state of A_φ, obtaining an almost-equivalent DBiA. Every clause of φ will be translated to a part of the biautomaton using a separate alphabet, so that the clause gadgets in A_φ are mostly independent from each other. Assume that $\varphi_i = (\ell_{i_1} \vee \ell_{i_2})$ is a clause of φ, and the first literal is $\ell_{i_1} = x_u$, and the second is $\ell_{i_2} = \overline{x_v}$, for some variables x_u and x_v. Then the DBiA A_φ contains the structure which is depicted in Figure 3.

The states q_1 and q_1' correspond to literal ℓ_{i_1}, and states q_2 and q_2' to the literal ℓ_{i_2}. States p_u and p_v correspond to the variables x_u and x_v, respectively, and are shared by all clause gadgets related to these variables. Now assume that there is a truth assignment ξ to the variables that satisfies clause φ_i, say by $\xi(x_u) = 1$. Then we make the preamble state p_u accepting, and merge the preamble state q_1' to the almost-equivalent state q_1. To preserve the $\diamond$-property of the automaton, we further re-route the backward c transition of the initial state, making state s_1 the target of the transition. The case where the clause φ_i is satisfied by the second literal corresponds to the similar situation,

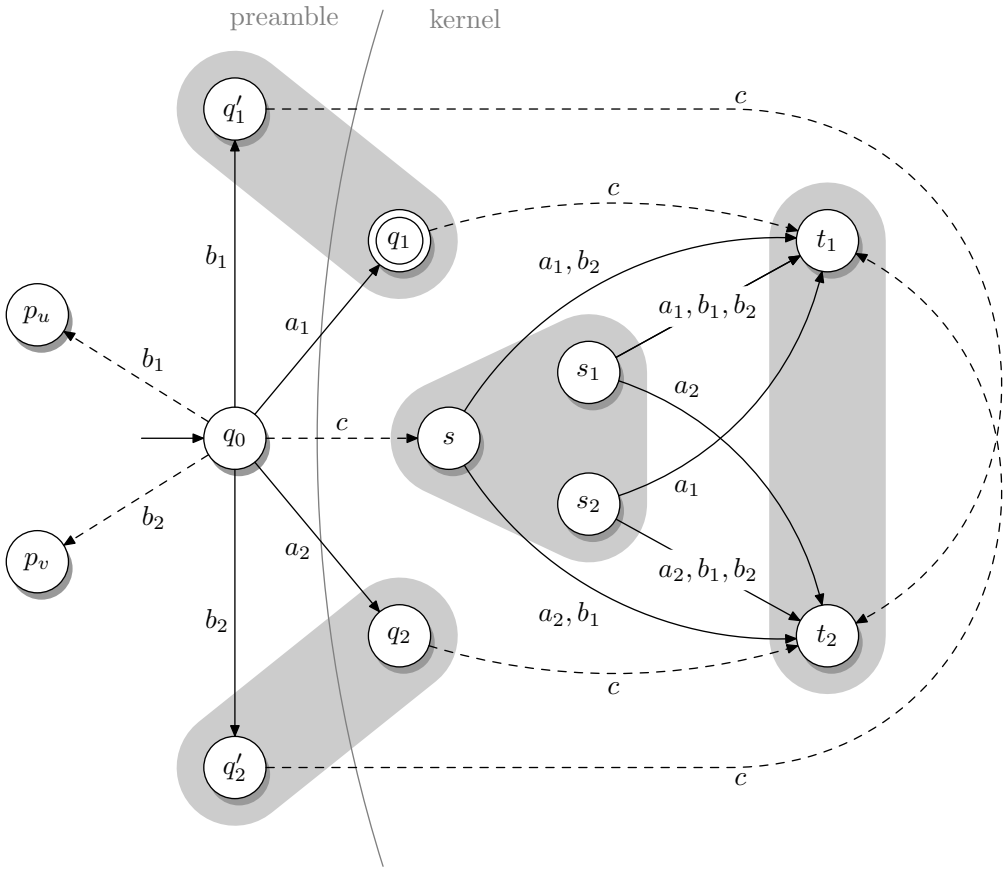

Fig. 3. Simplified structure of A_φ corresponding to the clause $\varphi_i = (x_u \vee \overline{x_v})$. The gray shading denotes almost-equivalence of states.

where state p_v stays non-accepting, state q_2' is merged to q_2, and the target of the backward c transition from q_0 is state s_2. The changing of acceptance of preamble states only introduces a finite number of errors. Further, the merging of preamble states to almost-equivalent kernel states also yields an almost-equivalent automaton.[3] Therefore, if k clauses of φ can be satisfied, then k states of A_φ can be saved.

The other direction, i.e., the deduction of a truth assignment ξ from a given automaton B that is almost-equivalent to A_φ, is similar: let $\xi(x_u) = 1$ if and only if state p_u of automaton B is accepting. Now assume that state B has k states less than A_φ. The reduction will make sure that only the states q_1' and q_2' can be saved. If for example state q_1' is not present in B, then the initial state of B must enter state q_1 on reading symbol b_1 with a forward transition. Due to the F-property of B, the state p_u reached from the initial state by taking a

[3] In general, this needs some more argumentation. Here the described changes in the automaton preserved the $\diamond$-property, and the F-property. Therefore the languages accepted by the original and the modified biautomata are the same as the languages accepted by the contained DFAs (using only forward transitions). Now almost-equivalence of these DFAs, and thus, of the biautomata, follows from the fact that merging preamble states to almost-equivalent kernel states in a DFA preserves almost-equivalence.

backward b_1 transition must be accepting. Since the variable states are shared by all clause gadgets, the information that p_u is accepting—i.e., that variable x_u should be assigned truth value 1—is transported to all other clause gadgets that use variable x_u. Therefore, no state corresponding to the negative literal $\overline{x_u}$ can be saved, i.e., no clause can be satisfied by a literal $\overline{x_u}$.[4] It may be the case that both states q_1' and q_2' are merged to their almost-equivalent kernel states q_1 and q_2, respectively. But then, due to the $\diamond$-property, the initial state must go to some state s' on reading a c symbol with a backward transition, and this state s' must go to state t_1 on a forward b_1 transition, and it must go to state t_2 on a forward b_2 transition. Such a state is not present in the automaton A_φ, so this state s' is an additional state in the preamble of B. Hence, even if both states q_1' and q_2' are merged into the kernel, the clause gadget in B cannot save more than one state compared to the clause gadget in A_φ. Altogether, for every state that B has less than A, there is a clause of φ that is satisfied by ξ.

We now present our result on the NP-hardness of the hyper-minimization problem with a detailed proof.

Theorem 14. *The problem of deciding for a given biautomaton A, and an integer n, whether there exists an n-state biautomaton B, with $A \sim B$, is* NP*-hard.*

Proof. Let φ and k form an instance of the MAX-2-SAT problem where k is an integer, and $\varphi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_m$ is a Boolean formula in conjunctive normal-form over the set of variables $X = \{x_1, x_2, \ldots, x_n\}$, and where each clause φ_i contains exactly two literals ℓ_{i_1} and ℓ_{i_2}. Instead of directly constructing the DBiA A_φ for the instance of the hyper-minimization problem, we describe the language L_φ accepted by this biautomaton. The integer k' for the instance of the hyper-minimization problem will be the number of states of A_φ minus k, i.e., for each clause that has to be satisfied in φ a state of A_φ is to be saved. From the definition of L_φ one can see that A_φ can indeed be constructed from φ in polynomial time. After the definition of L_φ we will analyze the structure of A_φ and prove the correctness of the described reduction.

Let us define the language L_φ. The alphabet Σ over which L_φ is defined is

$$\Sigma = \{\$\} \cup \bigcup_{i=1}^{m} \Sigma^{(i)} \cup \bigcup_{j=1}^{n} \bigcup_{h=1}^{m+1} \{\#_{j,h}\},$$

where for $1 \leq i \leq m$ the alphabet $\Sigma^{(i)}$ is

$$\Sigma^{(i)} = \{a_1^{(i)}, a_2^{(i)}\} \cup B_1^{(i)} \cup B_2^{(i)} \cup \{c^{(i)}, d^{(i)}, e_1^{(i)}, e_2^{(i)}, f_1^{(i)}, f_2^{(i)}, f_3^{(i)}, f_4^{(i)}, f_5^{(i)}\},$$

with

$$B_1^{(i)} = \bigcup_{h=1}^{m+1} \{b_{1,h}^{(i)}\} \qquad \text{and} \qquad B_2^{(i)} = \bigcup_{h=1}^{m+1} \{b_{2,h}^{(i)}\}.$$

[4] The reader may have noticed that there is still a possibility to "cheat:" one could use accepting and non-accepting copies of variable states in the preamble in order to satisfy a lot more clauses than possible. We take care of this problem in the detailed proof. (The problem can be solved by using many copies $b_{1,j}$ and $b_{2,j}$ of the b_1 and b_2 symbols, each connected to a different copy $p_{u,j}$ of variable states. If the number of these copies is larger than the number of clauses, then the "cheat" turns out to be a bad trade-off.)

The language L_φ consists of *clause languages* L_{φ_i} and *variable languages* L_{x_j}:

$$L_\varphi = \bigcup_{i=1}^{m} L_{\varphi_i} \cup \bigcup_{j=1}^{n} L_{x_j}.$$

The variable languages L_{x_j}, for $1 \leq j \leq n$, are defined as follows:

$$L_{x_j} = \bigcup_{h=1}^{m+1} \left(\{\#_{j,h}\} \cdot \{\$\}^* \cdot \{\#_{j,h}\} \cdot B_{x_j,h} \right),$$

where the set $B_{x_j,h}$ contains the symbol $b_{1,h}^{(i)}$ ($b_{2,h}^{(i)}$, respectively) if and only if the first (second, respectively) literal in clause φ_i corresponds to the variable x_j. More formally,

$$\begin{aligned} B_{x_j,h} = &\{\, b_{1,h}^{(i)} \mid \varphi_i = (\ell_{i_1} \vee \ell_{i_2}) \text{ and } \ell_{i_1} \in \{x_j, \overline{x_j}\} \,\} \\ &\cup \{\, b_{2,h}^{(i)} \mid \varphi_i = (\ell_{i_1} \vee \ell_{i_2}) \text{ and } \ell_{i_2} \in \{x_j, \overline{x_j}\} \,\}. \end{aligned}$$

For $1 \leq i \leq m$, the clause language L_{φ_i} is defined over the alphabet $(\{\$\} \cup \Sigma^{(i)})^*$ as follows—for a better readability we omit the upper index (i) for symbols from $\Sigma^{(i)}$:

– If $\varphi_i = (x_{i_1} \vee x_{i_2})$ then

$$\begin{aligned} L_{\varphi_i} = {} & a_1 \cdot (e_1^* + d^+c) + B_1 \cdot (e_1^+ + d^*c) + a_2 \cdot (e_2^* + d^*c) + B_2 \cdot (e_2^+ + d^+c) \\ & + \$^+ \cdot \Big(f_1 \cdot (e_1^* + d^+c) \cdot f_1 + f_2 \cdot (e_2^* + d^*c) \cdot f_2 \\ & \quad + f_3 \cdot ((a_1 + B_2) \cdot d^+ + (a_2 + B_1) \cdot d^*) \cdot f_3 \\ & \quad + f_4 \cdot ((a_1 + B_1 + B_2) \cdot d^+ + a_2 \cdot d^*) \cdot f_4 \\ & \quad + f_5 \cdot (a_1 \cdot d^+ + (a_2 + B_1 + B_2) \cdot d^*) \cdot f_5 \Big) \cdot \$^+. \end{aligned}$$

– If $\varphi_i = (x_{i_1} \vee \overline{x_{i_2}})$ then L_{φ_i} is defined similar as in the first case, but we use $(e_2^+ + d^*c)$ instead of $(e_2^* + d^*c)$: (parts which are the same as in the first case are shown grayed out)

$$\begin{aligned} L_{\varphi_i} = {} & a_1 \cdot (e_1^* + d^+c) + B_1 \cdot (e_1^+ + d^*c) + a_2 \cdot (e_2^+ + d^*c) + B_2 \cdot (e_2^+ + d^+c) \\ & + \$^+ \cdot \Big(f_1 \cdot (e_1^* + d^+c) \cdot f_1 + f_2 \cdot (e_2^+ + d^*c) \cdot f_2 \\ & \quad + f_3 \cdot ((a_1 + B_2) \cdot d^+ + (a_2 + B_1) \cdot d^*) \cdot f_3 \\ & \quad + f_4 \cdot ((a_1 + B_1 + B_2) \cdot d^+ + a_2 \cdot d^*) \cdot f_4 \\ & \quad + f_5 \cdot (a_1 \cdot d^+ + (a_2 + B_1 + B_2) \cdot d^*) \cdot f_5 \Big) \cdot \$^+. \end{aligned}$$

– If $\varphi_i = (\overline{x_{i_1}} \vee x_{i_2})$ then L_{φ_i} is defined as in the first case, but we use $(e_1^+ + d^+c)$ instead of $(e_1^* + d^+c)$.
– If $\varphi_i = (\overline{x_{i_1}} \vee \overline{x_{i_2}})$ then L_{φ_i} is defined as in the first case, but we use $(e_1^+ + d^+c)$ instead of $(e_1^* + d^+c)$, and $(e_2^+ + d^*c)$ instead of $(e_2^* + d^*c)$.

This concludes the definition of the language L_φ.

Let $A_\varphi = (Q, \Sigma, \cdot, \circ, q_0, F)$ be the canonical biautomaton [19] for L_φ, with state set $Q = \{\, u^{-1}L_\varphi v^{-1} \mid u, v \in \Sigma^* \,\}$, initial state $q_0 = L_\varphi$, set of final states $F = \{\, q \in Q \mid \lambda \in q \,\}$, and where $q \cdot a = a^{-1}q$ and $q \circ a = qa^{-1}$ for all $q \in Q$ and $a \in \Sigma$. Further let $k' = |Q| - k$. That the instance (A_φ, k') can be constructed in polynomial time from the given instance (φ, k) of the MAX-2-SAT problem can be seen as follows. Every clause language induces a fixed number of states in A_φ, and except for the \$ symbol the clause languages are defined over disjoint alphabets. Hence the number of states corresponding to clause languages is linear in the number of clauses. Further, the number of states induced by the variable languages is $O(mn)$, so the automaton A_φ can be constructed in polynomial time. Before we show the correctness of the reduction, let us analyze the structure of A_φ. The preamble of A_φ consists of the states

$$\mathrm{Pre}(A_\varphi) = \{q_0\} \cup \{\, q_0 \cdot b^{(i)}_{x,h}, q_0 \circ b^{(i)}_{x,h} \mid 1 \le i \le m, 1 \le h \le m+1, x \in \{1,2\} \,\},$$

which can be seen as follows. Let $1 \le i \le m$, $1 \le h \le m+1$, and assume $\varphi_i = (\ell_{i_1} \vee \ell_{i_2})$ with $\ell_{i_1} \in \{x_u, \overline{x_u}\}$ and $\ell_{i_1} \in \{x_v, \overline{x_v}\}$, then

$$\begin{aligned} q_0 \cdot b^{(i)}_{1,h} &= {e^{(i)}_1}^+ + {d^{(i)}}^* c^{(i)}, & q_0 \circ b^{(i)}_{1,h} &= \#_{u,h}\$^*\#_{u,h}, \\ q_0 \cdot b^{(i)}_{2,h} &= {e^{(i)}_2}^+ + {d^{(i)}}^+ c^{(i)}, & q_0 \circ b^{(i)}_{2,h} &= \#_{v,h}\$^*\#_{v,h}. \end{aligned}$$

One can see from the descriptions of the languages L_{φ_i} and L_{x_j}, that these states are only reachable by reading a single symbol from $\bigcup_{i=1}^m (B^{(i)}_1 \cup B^{(i)}_2)$, so they are preamble states. By examining all other states that can be reached from the initial state by reading a symbol which is not in $\bigcup_{i=1}^m (B^{(i)}_1 \cup B^{(i)}_2)$, one can see that there are no further preamble states. For example consider the state $q_0 \cdot a^{(i)}_1$. This state can also be reached from q_0 by reading words from $\$^+ f_1$ with forward transitions and words from $f_1\$^+$ with backward transitions. Therefore, state $q_0 \cdot a$, and all states reachable from it are kernel states. The reader is invited to verify that all other states of A_φ are kernel states, too.

Figure 4, which is similar to Figure 3 shown before the proof, exemplarily shows a part of the structure of A_φ that corresponds to a clause language L_{φ_i}, for $\varphi_i = (x_u \vee \overline{x_v})$. The states are renamed as follows:

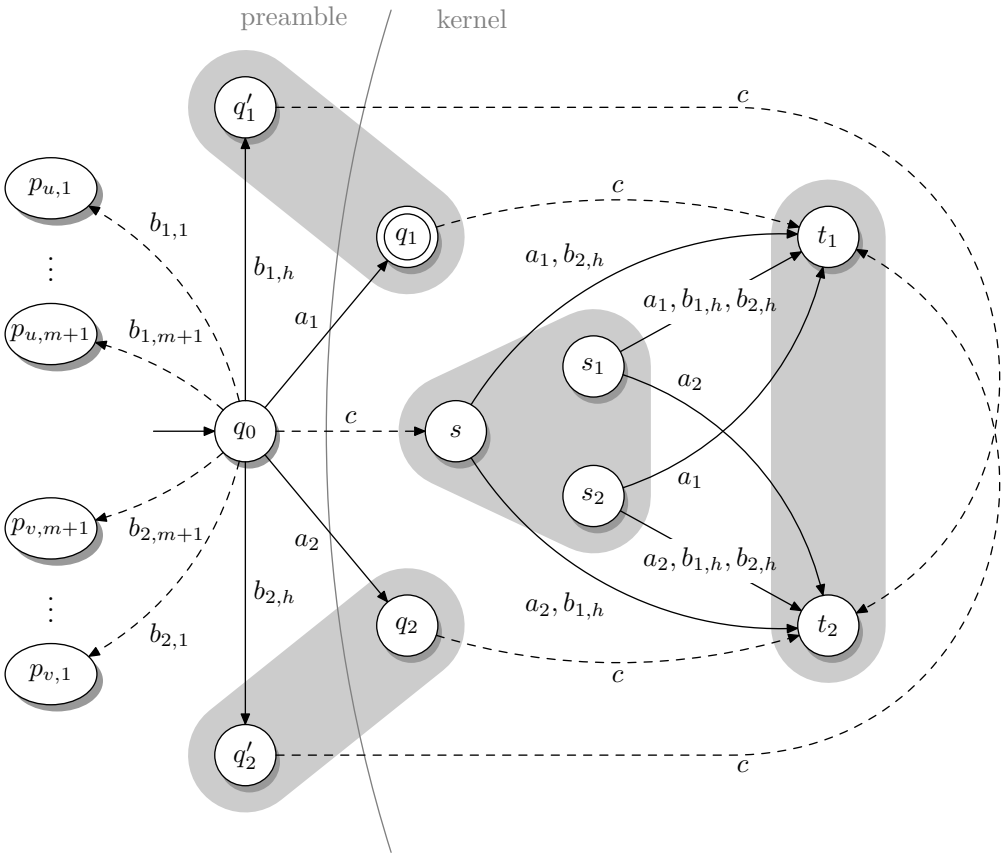

Fig. 4. Structure of A_φ corresponding to clause $\varphi_i = (x_u \vee \overline{x_v})$. The gray shading denotes almost-equivalence of states.

$$
\begin{aligned}
q_1^{(i)} &:= q_0 \cdot a_1^{(i)} = {e_1^{(i)}}^* + {d^{(i)}}^+ c^{(i)},\\
{q_1'}^{(i)} &:= q_0 \cdot b_{1,h}^{(i)} = {e_1^{(i)}}^+ + {d^{(i)}}^* c^{(i)}, \text{ for } 1 \le h \le m+1,\\
q_2^{(i)} &:= q_0 \cdot a_2^{(i)} = {e_2^{(i)}}^+ + {d^{(i)}}^* c^{(i)},\\
{q_2'}^{(i)} &:= q_0 \cdot b_{2,h}^{(i)} = {e_2^{(i)}}^+ + {d^{(i)}}^+ c^{(i)}, \text{ for } 1 \le h \le m+1,\\
s^{(i)} &:= q_0 \circ c^{(i)} = (a_1^{(i)} + B_2^{(i)}) \cdot {d^{(i)}}^+ + (a_2^{(i)} + B_1^{(i)}) \cdot {d^{(i)}}^*,\\
s_1^{(i)} &:= (q_0 \cdot \$f_4^{(i)}) \circ f_4^{(i)}\$ = (a_1^{(i)} + B_1^{(i)} + B_2^{(i)}) \cdot {d^{(i)}}^+ + a_2^{(i)} \cdot {d^{(i)}}^*,\\
s_2^{(i)} &:= (q_0 \cdot \$f_5^{(i)}) \circ f_5^{(i)}\$ = a_1^{(i)} \cdot {d^{(i)}}^+ + (a_2^{(i)} + B_1^{(i)} + B_2^{(i)}) \cdot {d^{(i)}}^*,\\
t_1^{(i)} &:= q_1^{(i)} \circ c^{(i)} = {d^{(i)}}^+,\\
t_2^{(i)} &:= q_2^{(i)} \circ c^{(i)} = {d^{(i)}}^*.
\end{aligned}
$$

Note that the following almost-equivalences hold between those states:

$$q_1^{(i)} \sim {q_1'}^{(i)}, \quad q_2^{(i)} \sim {q_2'}^{(i)}, \quad s^{(i)} \sim s_1^{(i)} \sim s_2^{(i)}, \quad t_1^{(i)} \sim t_2^{(i)},$$

In fact, the mentioned states are not almost-equivalent to any other states, which can be seen as follows. The right languages of above described states for

the clause φ_i contain infinitely many words with $d^{(i)}$ symbols. Therefore, these states cannot be almost-equivalent to states which correspond to derivatives of some clause language L_{φ_j}, with $j \neq i$, nor to states corresponding to derivatives of variable languages. Thus, the above states could only be almost-equivalent to states that correspond to some derivative of the clause language L_{φ_i}. But this is also not the case, which can be verified by inspection of the regular expression for L_{φ_i}. Finally, notice that also the states

$$p_{u,h} := q_0 \circ b_{1,h}^{(i)} = \#_{u,h}\$^*\#_{u,h}, \qquad \text{and} \qquad p_{v,h} := q_0 \circ b_{2,h}^{(i)} = \#_{v,h}\$^*\#_{v,h},$$

for $1 \leq h \leq m+1$, which correspond to the variables x_u and x_v, are not almost-equivalent to any other state.

Now we are ready to show the correctness of the reduction. To do this, we prove the following two claims:

Claim (1). If there exists a biautomaton that is almost-equivalent to A_φ and that has $k' = |Q| - k$ states, where Q is the state set of A_φ, then there exists a truth assignment that satisfies at least k clauses of φ.

Claim (2). If there is a truth assignment satisfying k clauses of φ, then there exists a biautomaton that is almost-equivalent to A_φ and that has $k' = |Q| - k$ states, where Q is the state set of A_φ.

The first claim is proven as follows:

Proof (of Claim (1)). Let $B = (Q', \Sigma, \cdot', \circ', q_0', F')$ be a minimal biautomaton with at most k' states, satisfying $B \sim A_\varphi$. Then we know from Theorem 11 that the kernels of the two automata are isomorphic. So in the following we may identify the kernel states of B with the derivatives of L_φ that constitute the kernel states of A_φ. Since $|Q'| \leq |Q| - k$, and the kernels of both automata are isomorphic, the preamble of B contains at most $|\mathrm{Pre}(A_\varphi)| - k$ states. The only states in $\mathrm{Pre}(A_\varphi)$ that can be saved are $q_1'^{(i)} = q_0 \cdot b_{1,h}^{(i)}$ and $q_2'^{(i)} = q_0 \cdot b_{2,h}^{(i)}$, with $1 \leq i \leq m$ and $1 \leq h \leq m+1$, because the other preamble states q_0 and $q_0 \circ b_{x,h}^{(i)}$, for $x \in \{1,2\}$ and $1 \leq i \leq m$, are not almost-equivalent to any other states. A state $q_1'^{(i)}$ (or $q_2'^{(i)}$, respectively) of A_φ can be saved if and only if in B we have $q_0' \cdot' b_{1,h}^{(i)} = q_0' \cdot' a_1^{(i)}$ (or $q_0' \cdot' b_{2,h}^{(i)} = q_0' \cdot' a_2^{(i)}$, respectively), for all integers h, with $1 \leq h \leq m+1$, i.e., if state $q_1'^{(i)}$ (or $q_2'^{(i)}$, respectively) is merged to the almost-equivalent kernel state $q_1^{(i)}$ (or $q_2^{(i)}$, respectively).

We first show that at most one state can be saved in each clause gadget. Consider the case where for some clause $\varphi_i = (\ell_{i_1} \vee \ell_{i_2})$, with $1 \leq i \leq m$, both $q_1'^{(i)}$ and $q_2'^{(i)}$ are merged to the kernel states $q_1^{(i)}$ and $q_2^{(i)}$, respectively. This means that in B we have for all h, with $1 \leq h \leq m+1$:

$$q_0' \cdot' b_{1,h}^{(i)} = \begin{cases} {e_1^{(i)}}^* + {d^{(i)}}^+ c^{(i)} & \text{if } \ell_{i_1} \text{ is positive,} \\ {e_1^{(i)}}^+ + {d^{(i)}}^+ c^{(i)} & \text{if } \ell_{i_1} \text{ is negative,} \end{cases}$$

$$q_0' \cdot' b_{2,h}^{(i)} = \begin{cases} {e_2^{(i)}}^* + {d^{(i)}}^* c^{(i)} & \text{if } \ell_{i_2} \text{ is positive,} \\ {e_2^{(i)}}^+ + {d^{(i)}}^* c^{(i)} & \text{if } \ell_{i_2} \text{ is negative.} \end{cases}$$

It follows $(q'_0 \cdot' b^{(i)}_{1,h}) \circ' c^{(i)} = d^{(i)^+}$ and $(q'_0 \cdot' b^{(i)}_{2,h}) \circ' c^{(i)} = d^{(i)^*}$, for $1 \leq h \leq m+1$. Since B has the $\diamond$-property, the state $s'^{(i)} := q'_0 \circ c^{(i)}$ must lead to state $d^{(i)^+}$ on symbols $b^{(i)}_{1,h}$ and $a^{(i)}_1$, and it must lead to state $d^{(i)^*}$ on symbols $b^{(i)}_{2,h}$ and $a^{(i)}_2$. Such a state $s'^{(i)}$ is not present in automaton A_φ (though it is almost equivalent to the states $s^{(i)}$, $s^{(i)}_1$, and $s^{(i)}_2$), so it must be an additional preamble state of B.

Thus, for each clause φ_i, the number of states in $\mathrm{Pre}(B)$, that can be reached by reading a symbol from $\Sigma^{(i)}$ is equal to, or by one smaller than the number of states in $\mathrm{Pre}(A_\varphi)$ reachable by such symbols. Since $|\mathrm{Pre}(B)| \leq |\mathrm{Pre}(A_\varphi)| - k$, there must be k clauses $\varphi_{\alpha_1}, \varphi_{\alpha_2}, \ldots \varphi_{\alpha_k}$, such that for each $\alpha \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ we have that $q'^{(\alpha)}_1$ is merged to $q^{(\alpha)}_1$, or $q'^{(\alpha)}_2$ is merged to $q^{(\alpha)}_2$. We use this to construct the truth assignment $\xi\colon X \to \{0,1\}$ for φ as follows. For each index $\alpha \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ consider the clause $\varphi_\alpha = (\ell_{\alpha_1} \vee \ell_{\alpha_2})$.

- If $q'^{(\alpha)}_1$ is merged to $q^{(\alpha)}_1$, and $\ell_{\alpha_1} = x_u$, for some $x_u \in X$, then set $\xi(x_u) = 1$.
- If $q'^{(\alpha)}_1$ is merged to $q^{(\alpha)}_1$, and $\ell_{\alpha_1} = \overline{x_u}$, for some $x_u \in X$, then set $\xi(x_u) = 0$.
- If $q'^{(\alpha)}_2$ is merged to $q^{(\alpha)}_2$, and $\ell_{\alpha_2} = x_v$, for some $x_v \in X$, then set $\xi(x_v) = 1$.
- If $q'^{(\alpha)}_2$ is merged to $q^{(\alpha)}_2$, and $\ell_{\alpha_2} = \overline{x_v}$, for some $x_v \in X$, then set $\xi(x_v) = 0$.

For all remaining variables $x_j \in X$ that have not yet been assignment a truth value, set $\xi(x_j) = 0$.

Let us first see, that there is no variable that gets assigned both values 0 and 1. Assume that there is such a variable x_j. Then there must be clauses φ_α and $\varphi_{\alpha'}$ with $\alpha, \alpha' \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, such that the variable x_j appears positive in φ_α and negative in $\varphi_{\alpha'}$. Assume that $x_j = \ell_{\alpha_1}$ is the first literal of φ_α, and $\overline{x_j} = \ell_{\alpha'_2}$ is the second literal of α'—the other cases can be handled similarly. Since x_j gets assigned truth value 1, the state $q'^{(\alpha)}_1$ must have been merged to $q^{(\alpha)}_1$, which means that for all h, with $1 \leq h \leq m+1$ we have

$$q'_0 \cdot' b^{(\alpha)}_{1,h} = e^{(\alpha)^*}_1 + d^{(\alpha)^+} c^{(\alpha)} \in F'.$$

Since B has the F-property, also the preamble states $q'_0 \circ' b^{(\alpha)}_{1,h}$, for $1 \leq h \leq m+1$, must be in F'. Recall that the state $(q'_0 \circ' b^{(\alpha)}_{1,h})$ of automaton B is almost-equivalent to state $q_0 \circ b^{(\alpha)}_{1,h} = p_{j,h}$ of A_φ, for $1 \leq h \leq m+1$. Since x_j also gets assigned truth value 0, the state $q'^{(\alpha')}_2$ must have been merged to $q^{(\alpha')}_2$, which means that for all h, with $1 \leq h \leq m+1$, we have

$$q'_0 \cdot' b^{(\alpha')}_{2,h} = e^{(\alpha')^+}_2 + d^{(\alpha')^*} c^{(\alpha')} \notin F'.$$

Since B has the F-property, also the preamble states $q'_0 \circ' b^{(\alpha')}_{2,h}$, for $1 \leq h \leq m+1$ are not in F'. Notice that the state $q'_0 \circ' b^{(\alpha')}_{2,h}$ is almost-equivalent to the state $(q_0 \circ b^{(\alpha')}_{2,h}) = p_{j,h}$ of A, for $1 \leq h \leq m+1$. It follows that for each state $p_{j,h}$ from $\mathrm{Pre}(A_\varphi)$ with $1 \leq h \leq m+1$, there are two different states $q'_0 \circ b^{(\alpha)}_{1,h}$ and $q'_0 \circ b^{(\alpha')}_{2,h}$ in $\mathrm{Pre}(B)$. But with these $m+1$ additional states, the preamble

of B is definitely larger than the preamble of A_φ, since for each of the m clauses at most one state can be saved. This is a contradiction, so every variable gets assigned exactly one truth value.

From the definition of ξ it clearly follows that ξ satisfies each clause φ_α, for $\alpha \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. We have shown that if there is a k'-state biautomaton B with $A_\varphi \sim B$, then there is a truth assignment ξ that satisfies k clauses of φ. □

It remains to prove the second claim.

Proof (of Claim (2)). Let $\xi\colon X \to \{0,1\}$ be a truth assignment that satisfies the clauses $\varphi_{\alpha_1}, \varphi_{\alpha_2}, \ldots, \varphi_{\alpha_k}$. We construct the biautomaton B from automaton A_φ as follows. For all variables $x_j \in X$ with $\xi(x_j) = 1$, we make the states $p_{j,h}$ accepting, for $1 \leq h \leq m+1$. Note that $p_{j,h}$ is only reachable from q_0 by reading a symbol from $B_{x_j,h}$. In order to keep the F-property, we also make the states $q_0 \cdot b$ accepting, for all $b \in B_{x_j,h}$. Since all these states are in the preamble, we still have an almost-equivalent biautomaton.

Now let $i \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ and consider the clause $\varphi_i = (\ell_{i_1} \vee \ell_{i_2})$. If φ_i is satisfied by the first literal ℓ_{i_1}, then we can merge state $q_1'^{(i)}$ to state $q_1^{(i)}$ by re-routing all forward transitions from q_0 on symbols $b_{1,h}^{(i)}$ to state $q_1^{(i)}$, and re-routing the backward transition from q_0 on symbol $c^{(i)}$ to state s_1. Now the state $q_1'^{(i)}$ is not reachable anymore and can be removed. If the clause φ_i is not satisfied by literal ℓ_{i_1}, then it must be satisfied by the second literal ℓ_{i_2}. In this case, we can re-route all forward transitions from q_0 on symbols $b_{2,h}^{(i)}$ to state $q_2^{(i)}$, and re-route the backward transition from q_0 on symbol $c^{(i)}$ to state s_2, so that state $q_2'^{(i)}$ can be removed. Note that the resulting biautomaton still has the $\diamond$-property and the F-property. Further, it is almost-equivalent to A_φ, since we only re-routed transitions starting from a preamble state, and the new target states for these transitions are almost-equivalent to the original target states. In this way we can remove k states from A_φ, and obtain an almost-equivalent biautomaton A' that has $k' = |Q| - k$ states. We have shown that if k clauses of φ can be satisfied, then k states of A_φ can be saved. This concludes the proof of Claim (2). □

The theorem now follows from Claims (1) and (2). □

Containment of the hyper-minimization problem in NP can be seen by an easy guess-and-check-algorithm, therefore, we obtain the following theorem.

Theorem 15 (DBiA Hyper-Minimization Problem). *The problem of deciding for a given biautomaton A, and an integer n, whether there exists an n-state biautomaton B, with $A \sim B$, is* NP*-complete.*

Proof. We know from Theorem 14 that the problem is NP-hard, hence it remains to prove containment in NP. Given as an instance of the problem a DBiA A and an integer n, a non-deterministic Turing machine first guesses an n-state DBiA B and writes it on its working tape. Checking whether A and B are almost-equivalent can be done by testing whether the contained DFAs A_{fwd}

and B_{fwd} are almost-equivalent. It is shown in [6] that this question for DFAs can be decided in NL. Therefore, the hyper-minimization problem for biautomata can be decided by a non-deterministic Turing machine in polynomial time. □

In [6] another form of equivalence of languages and automata was considered, namely E-equivalence. Given an *error language* $E \subseteq \Sigma^*$, two languages L and L' over Σ are *E-equivalent*, for short $L \sim_E L'$, if $L \triangle L' \subseteq E$, and two automata A and A' are E-equivalent, if $L(A) \sim_E L(A')$. It is shown in [6] that the following E-minimization problem for DFAs is already NP-complete: given two DFAs A and A_E, and an integer n, decide whether there exists an n-state DFA B, with $A \sim_E B$, for $E = L(A_E)$. That result cannot directly be used to obtain a similar result for biautomata, but one can show that the E-minimization problem for biautomata is NP-complete, too. Concerning the definition of the E-minimization problem for biautomata, one could consider the following two possibilities for specifying the error set E: either by a deterministic finite automaton, or by a biautomaton. Both variants turn out to be NP-complete, as the following result shows.

Theorem 16 (DBiA E-Minimization Problem). *The problem of deciding for a given biautomaton A, a deterministic finite automaton A_E, and an integer n, whether there exists an n-state biautomaton B, such that $A \sim_E B$, for $E = L(A_E)$, is* NP*-complete. This also holds, if A_E is a biautomaton instead of a finite automaton.*

Proof (Sketch). We can use the fact that almost-equivalence is a special case of E-equivalence: if two languages L and L' are accepted by DFAs having n and n' states, respectively, and if $L \sim L'$, then $L \sim_E L'$, for $E = \Sigma^{\leq \max(n,n')}$. With this, NP-hardness of the E-minimization problem for biautomata follows from Theorem 14. The NP upper bound can be seen by an easy guess-and-check algorithm. □

6 Conclusions

We compared structural and computational complexity results for deterministic finite automata to similar results for biautomata. These two different automaton models behave very similar in many aspects, but we found a notable difference in the complexity of the hyper-minimization problem for these machines: for DFAs this problem is known to be NL-complete [6], while we proved in this paper, that it is NP-complete for biautomata.

References

1. Arnold, A., Dicky, A., Nivat, M.: A note about minimal non-deterministic automata. Bull. EATCS **47** (1992) 166–169
2. Badr, A., Geffert, V., Shipman, I.: Hyper-minimizing minimized deterministic finite state automata. RAIRO–Informatique théorique et Applications / Theoretical Informatics and Applications **43**(1) (2009) 69–94
3. Brzozowski, J.A.: Derivatives of regular expressions. J. ACM **11** (1964) 481–494

4. Cho, S., Huynh, D.T.: The parallel complexity of finite-state automata problems. Inform. Comput. **97** (1992) 1–22
5. Gawrychowski, P., Jeż, A.: Hyper-minimization made efficient. In Královic, R., Niwinski, D., eds.: Proceedings of the 34th Conference on Mathematical Foundations of Computer Science. Number 5734 in LNCS, Novy Smokovec, High Tatras, Slovakia, Springer (2011) 356–368
6. Holzer, M., Jakobi, S.: From equivalence to almost-equivalence, and beyond: Minimizing automata with errors. Internat. J. Found. Comput. Sci. **24**(7) (2013) 1083–1134
7. Holzer, M., Jakobi, S.: Minimization and characterizations for biautomata. In Bensch, S., Drewes, F., Freund, R., Otto, F., eds.: Proceedings of the 5th International Workshop on Non-Classical Models of Automata and Applications. Number 294 in books@ocg.at, Umeå, Sweden, Österreichische Computer Gesellschaft (2013) 179–193
8. Holzer, M., Jakobi, S.: Nondeterministic biautomata and their descriptional complexity. In Jürgensen, H., Reis, R., eds.: Proceedings of the 15th International Workshop on Descriptional Complexity of Formal Systems. Number 8031 in LNCS, London, Ontario, Canada, Springer (2013) 112–123
9. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata—a survey. Inform. Comput. **209**(3) (2011) 456–470
10. Holzer, M., Maletti, A.: An $n \log n$ algorithm for hyper-minimizing a (minimized) deterministic automaton. Theoret. Comput. Sci. **411**(38–39) (2010) 3404–3413
11. Hopcroft, J.: An $n \log n$ algorithm for minimizing the state in a finite automaton. In Kohavi, Z., ed.: The Theory of Machines and Computations. Academic Press, New York (1971) 189–196
12. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
13. Immerman, N.: Nondeterministic space is closed under complementation. SIAM J. Comput. **17**(5) (1988) 935–938
14. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. In: Proceedings of the International Colloquim on Automata, Languages, and Programming. Number 510 in LNCS, Springer (1991) 629–640
15. Jirásková, G., Klíma, O.: Descriptional complexity of biautomata. In Kutrib, M., Moreira, N., Reis, R., eds.: Proceedings of the 14th International Workshop Descriptional Complexity of Formal Systems. Number 7386 in LNCS, Braga, Portugal, Springer (2012) 196–208
16. Jones, N.: Space-bounded reducibility among combinatorial problems. J. Comput. System Sci. **11** (1975) 68–85
17. Jones, N.D., Lien, Y.E., Laaser, W.T.: New problems complete for nondeterministic log space. Math. Systems Theory **10** (1976) 1–17
18. Klíma, O., Polák, L.: Biautomata for k-piecewise testable languages. In Yen, H.C., Ibarra, O.H., eds.: Proceedings of the 16th International Conference Developments in Language Theory. Number 7410 in LNCS, Taipei, Taiwan, Springer (2012) 344–355
19. Klíma, O., Polák, L.: On biautomata. RAIRO–Informatique théorique et Applications / Theoretical Informatics and Applications **46**(4) (2012) 573–592
20. Papadimitriou, C.H.: On the complexity of the parity argument and other ineffienct proofs of existence. J. Comput. System Sci. **48**(3) (1994) 498–532
21. Szelepcsényi, R.: The method of forced enumeration for nondeterministic automata. Acta Inform. **26**(3) (1988) 279–284

Institut für Informatik

Justus-Liebig-Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany

Recent Reports

(Further reports are available at `www.informatik.uni-giessen.de`.)

J. Kari, M. Kutrib, A. Malcher (Eds.), *19th International Workshop on Cellular Automata and Discrete Complex Systems AUTOMATA 2013 Exploratory Papers*, Report 1302, September 2013.

M. Holzer, S. Jakobi, *Minimization, Characterizations, and Nondeterminism for Biautomata*, Report 1301, April 2013.

A. Malcher, K. Meckel, C. Mereghetti, B. Palano, *Descriptional Complexity of Pushdown Store Languages*, Report 1203, May 2012.

M. Holzer, S. Jakobi, *On the Complexity of Rolling Block and Alice Mazes*, Report 1202, March 2012.

M. Holzer, S. Jakobi, *Grid Graphs with Diagonal Edges and the Complexity of Xmas Mazes*, Report 1201, January 2012.

H. Gruber, S. Gulan, *Simplifying Regular Expressions: A Quantitative Perspective*, Report 0904, August 2009.

M. Kutrib, A. Malcher, *Cellular Automata with Sparse Communication*, Report 0903, May 2009.

M. Holzer, A. Maletti, *An $n \log n$ Algorithm for Hyper-Minimizing States in a (Minimized) Deterministic Automaton*, Report 0902, April 2009.

H. Gruber, M. Holzer, *Tight Bounds on the Descriptional Complexity of Regular Expressions*, Report 0901, February 2009.

M. Holzer, M. Kutrib, and A. Malcher (Eds.), *18. Theorietag Automaten und Formale Sprachen*, Report 0801, September 2008.

M. Holzer, M. Kutrib, *Flip-Pushdown Automata: Nondeterminism is Better than Determinism*, Report 0301, February 2003

M. Holzer, M. Kutrib, *Flip-Pushdown Automata: $k+1$ Pushdown Reversals are Better Than k*, Report 0206, November 2002

M. Holzer, M. Kutrib, *Nondeterministic Descriptional Complexity of Regular Languages*, Report 0205, September 2002

H. Bordihn, M. Holzer, M. Kutrib, *Economy of Description for Basic Constructions on Rational Transductions*, Report 0204, July 2002

M. Kutrib, J.-T. Löwe, *String Transformation for n-dimensional Image Compression*, Report 0203, May 2002

A. Klein, M. Kutrib, *Grammars with Scattered Nonterminals*, Report 0202, February 2002

A. Klein, M. Kutrib, *Self-Assembling Finite Automata*, Report 0201, January 2002

M. Holzer, M. Kutrib, *Unary Language Operations and its Nondeterministic State Complexity*, Report 0107, November 2001

A. Klein, M. Kutrib, *Fast One-Way Cellular Automata*, Report 0106, September 2001

M. Holzer, M. Kutrib, *Improving Raster Image Run-Length Encoding Using Data Order*, Report 0105, July 2001

M. Kutrib, *Refining Nondeterminism Below Linear-Time*, Report 0104, June 2001

M. Holzer, M. Kutrib, *State Complexity of Basic Operations on Nondeterministic Finite Automata*, Report 0103, April 2001

M. Kutrib, J.-T. Löwe, *Massively Parallel Fault Tolerant Computations on Syntactical Patterns*, Report 0102, March 2001

A. Klein, M. Kutrib, *A Time Hierarchy for Bounded One-Way Cellular Automata*, Report 0101, January 2001

M. Kutrib, *Below Linear-Time: Dimensions versus Time*, Report 0005, November 2000