From Equivalence to Almost-Equivalence, and Beyond—Minimizing Automata with Errors
(Extended Abstract)

Markus Holzer and Sebastian Jakobi

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{holzer,jakobi}@informatik.uni-giessen.de

Abstract. We introduce E-equivalence, which is a straightforward generalization of almost-equivalence. While almost-equivalence asks for ordinary equivalence up to a finite number of exceptions, in E-equivalence these exceptions or *errors* must belong to a (regular) set E. The computational complexity of minimization problems and their variants w.r.t. almost- and E-equivalence are studied. Roughly speaking, whenever nondeterministic finite automata (NFAs) are involved, most minimization problems, and their equivalence problems they are based on, become PSPACE-complete, while for deterministic finite automata (DFAs) the situation is more subtle. For instance, hyper-minimizing DFAs is NL-complete, but E-minimizing DFAs is NP-complete, even for finite E. The obtained results nicely fit to the known ones on ordinary minimization for finite automata. Moreover, since hyper-minimal and E-minimal automata are not necessarily unique (up to isomorphism as for minimal DFAs), we consider the problem of counting the number of these minimal automata. It turns out that counting hyper-minimal DFAs can be done in FP, while counting E-minimal DFAs is #P-hard, and belongs to the counting class $\# \cdot \mathsf{coNP}$.

1 Introduction

The study of the minimization problem for finite automata dates back to the early beginnings of automata theory. This problem is also of practical relevance, because regular languages are used in many applications, and one may like to represent the languages succinctly. 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 [14]. More precisely, the DFA-to-DFA minimization problem is complete for NL, even for DFAs without inaccessible states [5]. This is contrary to the nondeterministic case since the nondeterministic finite automaton (NFA) minimization problem is known to be computationally hard [17]. Minimization remains intractable even if either the input or the output automaton is deterministic [17,21].

Recently another form of minimization for DFAs, namely hyper-minimization, was considered in the literature [2,3,7,13]. While minimization aims to find an

H.-C. Yen and O.H. Ibarra (Eds.): DLT 2012, LNCS 7410, pp. 190–201, 2012.

equivalent automaton that is as small as possible, hyper-minimization intends to find an almost-equivalent automaton that is as small as possible. Here two languages are considered to be *almost-equivalent*, if they are equivalent up to a finite number of exceptions. Thus, an automaton is hyper-minimal if every other automaton with fewer states disagrees on acceptance for an *infinite* number of inputs. Hence, equivalence or almost-equivalence can be interpreted as an "*error profile*:" minimization becomes exact compression and hyper-minimization is a sort of lossy compression. Minimal and hyper-minimal automata, share a lot of similar traits, e.g., minimal and hyper-minimal DFAs have a nice structural description [3,15] and computing a minimal representation from a given n-state DFA can be done efficiently in $O(n \log n)$ time [7,13,14]. Nevertheless, there are subtle differences. The most important one is that ordinary minimal DFAs are unique up to isomorphism, but this property doesn't hold anymore for hyper-minimal DFAs [7]. Novel investigations on hyper-minimization performed in [8] and [20] show that hyper-minimization that returns a DFA that commits the smallest number of errors can be efficiently computed, while simultaneously bounding the size and the errors of the output DFA results in an NP-complete decision problem. This is the starting point of our investigation.

We provide a general framework for error profiles of automata. To this end we introduce the concept of E-equivalence. Two languages L_1 and L_2 are *E-equivalent*[1] if their symmetric difference lies in E, i.e., $L_1 \triangle L_2 \subseteq E$. Here E is called the *error* language. Although E-equivalence ($\sim_E$) is a generalization of equivalence ($\equiv$) and almost-equivalence ($\sim$), the problems to decide whether two languages given by finite automata are equivalent, almost-equivalent, or E-equivalent, respectively, are all of same complexity. To be more precise, whenever NFAs are involved in the language specification the decision problem is PSPACE-complete, while for DFAs it is NL-complete. When turning to minimization w.r.t. the above mentioned relations $\sim$ and $\sim_E$, the results mirror those for ordinary DFA and NFA minimization, with some notable exceptions. For instance, hyper-minimizing deterministic machines, that is the DFA-to-DFA minimization problem w.r.t. almost-equivalence, is shown to be NL-complete while E-minimization of DFAs in general turns out to be NP-complete, even for some finite E. Note, that the finiteness of E does not contradict the NL-completeness of hyper-minimizing DFAs. We also study some problems related to minimization such as canonicity, minimality, and variants thereof; a precise definition of these problems is given in the sections to come. For all these problems we obtain precise complexity bounds depending on whether NFAs or DFAs are given as inputs—see, e.g., Table 2. Moreover, since hyper-minimal and E-minimal automata are not necessarily unique (up to isomorphism as for minimal DFAs),

[1] A close inspection shows that E-equivalence allows us to cover a lot of prominent "equivalence" concepts from the literature such as, e.g., (i) equivalence—$E = \emptyset$, (ii) almost-equivalence and k-equivalence—E is finite and $E = \Sigma^{\leq k}$, respectively, (iii) equivalence modulo the empty word—$E = \{\lambda\}$, (iv) closeness—E is a sparse set, and (v) cover automata—$E = \Sigma^{\geq k}$. A detailed discussion on this subject is given in the full version of the paper.

we consider the problem of counting the number of these minimal automata. It turns out that counting hyper-minimal DFAs can be done in FP, while counting E-minimal DFAs is #P-hard, and belongs to the counting class $\# \cdot \mathsf{coNP}$. The upper bound for counting minimal NFAs is #PSPACE. Due to space constraints almost all proofs are omitted.

2 Preliminaries

We assume familiarity with the basic concepts of complexity theory [22] such as the inclusion chain $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$. Here L (NL, respectively) is the set of problems accepted by deterministic (nondeterministic, respectively) 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 and let PSPACE be the set of problems accepted by deterministic or nondeterministic polynomial space bounded Turing machines. We are also interested in counting the number of solutions to particular problems. Let FP be the class of polynomial time computable functions. Higher counting complexity classes are introduced *via* a predicate based approach—see, e.g., [11]. If C is a complexity class of decision problems, let $\# \cdot \mathsf{C}$ be the class of all functions f such that $f(x) = |\{\, y \mid R(x,y) \text{ and } |y| = p(|x|) \,\}|$, for some C-computable two-argument predicate R and some polynomial p. Observe, that $\# \cdot \mathsf{P}$ coincides with Valiant's counting class #P, i.e., $\#\mathsf{P} = \# \cdot \mathsf{P}$, introduced in his seminal paper on computing the permanent of a matrix [26]. Moreover, in particular we have the inclusion chain $\#\mathsf{P} = \# \cdot \mathsf{P} \subseteq \# \cdot \mathsf{NP} \subseteq \# \cdot \mathsf{P}^{\mathsf{NP}} = \# \cdot \mathsf{coNP}$, by Toda's result [25].

Next we need some notations on finite automata as contained in [15]. A *nondeterministic finite automaton* (NFA) 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 $\delta : Q \times \Sigma \to 2^Q$ is the *transition function*. The *language accepted* by the finite automaton A is defined as $L(A) = \{\, w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset \,\}$, where the transition function is recursively extended to $\delta : Q \times \Sigma^* \to 2^Q$. A finite automaton is *deterministic* (DFA) if and only if $|\delta(q,a)| = 1$, for all states $q \in Q$ and letters $a \in \Sigma$. In this case we simply write $\delta(q,a) = p$ for $\delta(q,a) = \{p\}$, assuming that the transition function is a mapping $\delta : Q \times \Sigma \to Q$. So any DFA is complete, i.e., the transition function is total, whereas for NFAs it is possible that δ maps to the empty set.

Two finite automata A and B are *equivalent*, $A \equiv B$, if and only if they accept the same language, i.e., $L(A) = L(B)$. Recently, hyper-minimal automata were studied in the literature [2,3]. Two finite automata A and B are *almost-equivalent*, $A \sim B$, if and only if the symmetric difference $L(A) \triangle L(B)$ is finite, i.e., $|L(A) \triangle L(B)| < \infty$. A finite automaton is *minimal* (*hyper-minimal*, respectively) if it admits no smaller equivalent (almost-equivalent, respectively) automaton. While minimal DFAs are unique up to isomorphism, this is not necessarily true for hyper-minimal DFAs anymore—see Figure 1. Nevertheless, hyper-minimal DFAs obey a nice structural characterization [3]. In this paper we

Fig. 1. Two hyper-minimal DFAs for the language $(a^* + bb^*a)^*$ which are not isomorphic to each other; the symmetric difference of the languages accepted by these two DFAs is equal to the finite set $\{\lambda\}$

consider another form of equivalence, by explicitly parameterizing the difference that is allowed between related languages. Let E be any subset of Σ^*, called the *error language*. Then we say that two languages L_1 and L_2 over the alphabet Σ are E*-equivalent*, $L_1 \sim_E L_2$, if and only if their symmetric difference lies in E, i.e., if $L_1 \triangle L_2 \subseteq E$—alternatively, the equivalent condition $L_1 \cup E = L_2 \cup E$ can be used. Moreover, this naturally carries over to finite automata; namely two finite automata A and B are E*-equivalent*, $A \sim_E B$, for some error language E, if and only if $L(A) \sim_E L(B)$. A finite automaton is E*-minimal* if it admits no smaller E-equivalent automaton. It is easy to see that that $\sim_E$ is an equivalence relation. These equivalence relations defined on languages or automata naturally carry over to relations on states. For instance, let $A = (Q, \Sigma, \delta, q_0, F)$. Then $p \sim_E q$, for $p, q \in Q$, if and only if $A_p \sim_E A_q$. Here A_p (A_q, respectively) is the automaton A, where the initial state is p (q, respectively) instead of q_0.

3 Finite Automata Equivalence, Minimization, and Related Problems

This section is four folded. First we consider the problem of testing equivalence of automata w.r.t almost- and E-equivalence. Then we consider the canonicity and the E-canonicity problem. Finally, in the last two subsections, we deal with the hyper-minimization and E-minimization problem. The precise definitions of these problems will be given in the appropriate subsections.

3.1 Equivalence Problems

The easiest problem for automata is that of ordinary equivalence. This is the problem of deciding for two given automata A and B, whether $A \equiv B$ holds. The complexity of this classical problem is well known. For DFAs the problem is NL-complete [5], and it is PSPACE-complete for NFAs [21]. This situation is resembled for the almost-equivalence and the E-equivalence problem, where for the latter problem, besides the automata A and B, a DFA A_E specifying the error language E is given as input. One can show that when the error language E is given by an NFA instead of a DFA, the E-equivalence problem instantly becomes PSPACE-complete, which is why we only consider DFAs for the description of the error language.

Theorem 1 (Almost- and E-Equivalence). *The problem of deciding for two given finite automata A and B, whether $A \sim B$, is* NL*-complete for DFAs and* PSPACE*-complete for NFAs. The statement also holds for the relation $\sim_E$ instead of $\sim$, where for $\sim_E$, a third input DFA A_E is given, that specifies the error language $E = L(A_E)$.* □

Hence, for all three error profiles the complexity of the equivalence problem is the same. For the next problems to come, in particular for the minimization problems, this will be not the case anymore. In most cases there will be a significant difference in complexity between problems on DFAs based on almost-equivalence and E-equivalence.

3.2 Canonical Languages

In general a hyper-minimal or E-minimal DFA for a language L can be smaller than the minimal DFA that accepts L. But this is not always the case, which leads to the notion of a language L being *canonical*, which means that the minimal DFA accepting L is also hyper-minimal [3]. When using E-minimality instead of hyper-minimality we speak of an *E-canonical* language. Recently canonical languages were studied in [24] from a descriptional complexity perspective. We start our investigations on the canonicity problem.

Theorem 2 (Canonicity). *The problem of deciding for a given finite automaton A, whether the language $L(A)$ is canonical, is* NL*-complete for DFAs, and* PSPACE*-complete for NFAs.*

Proof (Sketch). We only discuss the statement for DFAs, where we consider the complement of our problem. Then the result follows by the complementation closure of NL [16,23]. For NL-hardness we reduce the directed graph reachability problem 2GAP for acyclic graphs [18] to the non-canonicity problem. Given an acyclic graph $G = (V, E)$ and two vertices s and t, we construct in a natural way a DFA A with initial state s and final state t, whose transitions correspond to the edges of the graph. Undefined transitions lead to a sink state. If there is no path from s to t in G, then $L(A) = \emptyset$ is canonical, and otherwise $L(A)$ is finite (since G is acyclic) but not empty, and thus, not canonical.

The containment within NL boils down to the following property: a minimal DFA A is *not* hyper-minimal [3] if and only if the automaton contains a preamble state[2] that is almost-equivalent to some other state, by the structural characterization of hyper-minimal DFAs [3]. We can decide this property on a not necessarily minimal input DFA A in NL by checking whether there exists a pair of states p and r in A, satisfying the following properties: (i) p is a preamble state, (ii) $p \not\equiv r$, (iii) $p \sim r$, and (iv) $p \not\equiv q$, for all kernel states q. This proves the NL upper bound, and shows the stated NL-completeness of the canonicity problem for DFAs. □

[2] A state p in the finite automaton A is a *preamble state* if it is reachable from the start state of A by a *finite* number of inputs, only; otherwise the state is called a *kernel state*.

Table 1. Results on the computational complexity of deciding canonicity and E-canonicity of regular languages

Canonicity problem	Finite automata	
	DFA	NFA
$\sim$	NL	PSPACE
$\sim_E$, for DFA A_E with $E = L(A_E)$	coNP	PSPACE $\leq \cdot$ $\cdot \in$ coNEXP

Now we turn to the problem of deciding, whether the language accepted by some given finite automaton is E-canonical. For NFAs, this problem is PSPACE-hard, and contained in coNEXP. Here we could not conclude a PSPACE upper bound from an NL upper bound for the DFA problem as before, because the E-canonicity problem for DFAs is significantly harder than the canonicity problem. In contrast to the NL-completeness result for canonical languages, it turns out that the problem of deciding E-canonicity for a language $L(A)$ for some given DFA A and a given error language E is coNP-complete.

Theorem 3 (E-Canonicity). *The problem of deciding for two given DFAs A and A_E, whether the language $L(A)$ is E-canonical, for $E = L(A_E)$, is* coNP*-complete, even if E is finite. If the automaton A is an NFA, then the problem becomes* PSPACE*-hard, and is contained in* coNEXP*.* □

We summarize our results on canonical and E-canonical languages in Table 1. Note that ordinary equivalence is not included, since the corresponding decision problem—is the minimal DFA for the given language minimal?—is trivial.

3.3 Minimization Problems

Mostly, the decision version of the minimization problem is studied. For instance the DFA-to-DFA problem is defined as follows: given a DFA A and an integer[3] n, does there exist an *equivalent* n-state DFA B? This notation naturally generalizes to other types of finite automata. The DFA-to-DFA minimization problem is complete for NL, even for DFAs without inaccessible states [5]. This is contrary to the nondeterministic case since the NFA minimization problem is known to be PSPACE-complete [17], even if the input is given as a DFA.

Now the question arises, whether the complexity of the minimization problem changes, when equivalence is replaced by almost- or E-equivalence, respectively. Note, that by the results on equivalence, almost-equivalence, and E-equivalence in Subsection 3.1, one deduces upper bounds on the minimization since the problem description gives rise to simple guess-and-check algorithms. For instance, the DFA-to-DFA E-minimization belongs to NP, because for a DFA A one can guess

[3] When considering NFA-to-DFA minimization problems, we assume n to be given in unary notation. In all other cases, n may as well be given in binary notation.

an n-state DFA B and verify whether $A \sim_E B$ on a nondeterministic polynomial time bounded Turing machine by Theorem 1. In fact, this problem will be classified to be NP-complete.

Let us turn our attention to hyper-minimization. For the DFA-to-NFA hyper-minimization result, we use nearly the same automaton as constructed in [17] for the classical DFA-to-NFA minimization problem, together with an extended fooling set [4] for this automaton, which was presented in [10]. Then the following result on the descriptional complexity of hyper-minimal NFAs, which is interesting on its own, leads to a classification of the hyper-minimization problem.

Lemma 4. *Let $L \subseteq \Sigma^*$ be a regular language, and let F be an* extended fooling set *for L, i.e., $F = \{ (x_i, y_i) \mid 1 \leq i \leq n \}$, such that for $1 \leq i \leq n$ it is $x_i y_i \in L$, and for $1 \leq i, j \leq n$ with $i \neq j$, it is $x_i y_j \notin L$ or $x_j y_i \notin L$. Further let $L_0 \subseteq \Sigma^*$ be an infinite language satisfying $vw \in L \iff w \in L$ for every $v \in L_0$ and $w \in \Sigma^*$. Then any NFA A with $L(A) \sim L$ needs at least $|F|$ states.* □

Then the result on the hyper-minimization problem reads as follows.

Theorem 5 (Hyper-Minimization). *The problem of deciding for a given DFA A and an integer n, whether there exists a DFA B with n states, such that $A \sim B$, is* NL*-complete. The problem becomes* PSPACE*-complete for NFAs, even if the input is given as a DFA.* □

For the E-minimization problems, the situation is a bit different, since the DFA-to-DFA E-minimization is NP-complete—even if E is finite. To prove NP-hardness, it is tempting to use the NP-complete problem MINIMUM INFERRED FINITE STATE AUTOMATON which is defined in [6], and where [9] is given as reference. Unfortunately, in [9] this problem is defined for Mealey machines instead of DFAs as studied here. This makes a direct application complicated due to subtle differences between these machines—a detailed discussion on this subject is given in the full version of this paper. Nevertheless, we are able to succeed proving the following complexity result on E-minimization.

Theorem 6 (E-Minimization). *The problem of deciding for two given DFAs A and A_E, and an integer n, whether there exists a DFA B with n states, such that $A \sim_E B$, for $E = L(A_E)$, is* NP*-complete. This even holds, if the language E is finite. The problem becomes* PSPACE*-complete for NFAs, even if the input is given as a DFA.*

Proof (Sketch). We only sketch the proof for NP-completeness of the DFA-to-DFA E-minimization. Since $A \sim_E B$ can be verified for DFAs in deterministic polynomial time by Theorem 1, the problem description gives rise to a straightforward guess-and-check algorithm on a nondeterministic polynomial time bounded Turing machine. Hence the problem belongs to NP.

For NP-hardness we use a reduction from MONOTONE 3SAT [6]. Given a Boolean formula $\varphi = c_0 \wedge c_1 \wedge \cdots \wedge c_{k-1}$ with variables $X = \{x_0, x_1, \ldots, x_{n-1}\}$, where each c_i is either a positive clause of the form $c_i = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$ or

a negative clause of the form $c_i = (\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3})$, we construct a DFA $A = (Q \cup P \cup \{r, f, s\}, \{a, b, c\}, \delta, q_0, \{f\})$, where $Q = \{q_0, q_1, \ldots, q_{k-1}\}$, and $P = \{p_0, p_1, \ldots, p_{n-1}\}$. Its transition function δ is depicted in Figure 2. The integer for the E-minimization instance is set to $n + k + 2$, which is exactly one less than the number of states in A. Finally, the finite error language is

$$E = \{\, a^i b a^{n-j} \mid 0 \le i \le k-1,\ c_i \text{ contains } x_j \text{ or } \neg x_j \,\} \cup \{\, a^i b a^j b, a^i b a^j c \mid 0 \le i \le k-1,\ 1 \le j \le n-1 \,\} \cup \{\, a^{n+j} b \mid 0 \le j \le n-1 \,\}.$$

A DFA A_E accepting this language can easily be constructed in polynomial time.

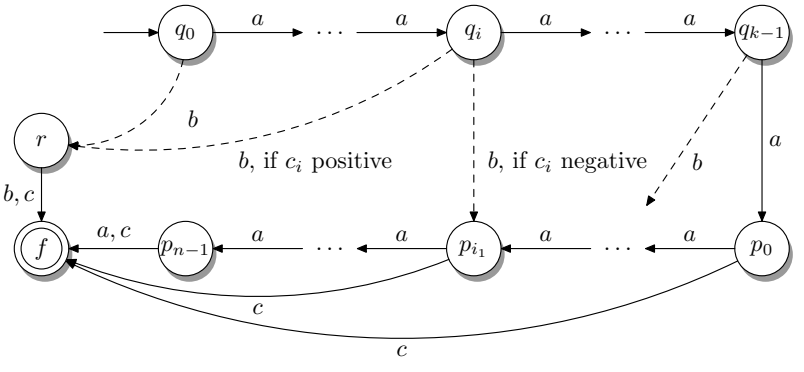

Fig. 2. The DFA A constructed from the Boolean formula φ. The b-transitions from states $q_0, q_1, \ldots, q_{k-1}$ are only sketched—it is $\delta(q_i, b) = p_{i_1}$, if $c_i = (\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3})$, and $\delta(q_i, b) = r$ otherwise. All undefined transitions go to the sink state s, which is not shown.

One can then show that φ is satisfiable if and only if there exists a DFA B, with $A \sim_E B$, that has $n + k + 2$ states—in this case, only state r is missing. The overall idea is the following. Since every word in E contains at least one b symbol, the error set does not allow E-equivalent automata to differ on inputs a or c. Further, since words $a^i bb$ and $a^i bc$ with $0 \le i \le k-1$ do not belong to E, the b-transitions from states q_i must end in states p_j, and the b-transitions from p_j must end in state f or the sink state s. The connection to φ is the following: a state p_i, corresponding to variable x_i, goes to state f on input b if and only if the variable x_i should be assigned the Boolean value 1. And a state q_i, corresponding to a clause c_i, goes to state p_j on input b if and only if the clause c_i gets satisfied by the variable x_j. In this way, any E-minimal DFA B, with $A \sim_E B$, corresponds to a satisfying truth assignment for φ, and *vice versa*. □

A slight variant of these minimization problems is the following problem, where also the number of errors is taken into account:

Table 2. Results on the computational complexity of minimizing finite automata with respect to different equivalence relations. The input to all problems is a finite automaton A and an integer n, and the question is, whether there exists an n-state finite automaton B, that is in the corresponding relation to A. For the problems on E-minimization, a DFA A_E specifying the error language E is given as additional input.

Equivalence relation	Minimization problem			
	DFA-to-...		NFA-to-...	
	DFA	NFA	DFA	NFA
$\equiv$	NL	PSPACE	PSPACE	
$\sim$				
$\sim_E$, for DFA A_E with $E = L(A_E)$	NP			

INSTANCE: An NFA A and integers e and n.
QUESTION: Is there an NFA B with n states, such that $|L(A) \triangle L(B)| \leq e$?

By adapting the proof on the PSPACE-hardness of the E-canonicity problem for NFAs, we can show that this problem is PSPACE-hard, and belongs to NEXP, even if the automaton B in the problem description is restricted to be deterministic. This is a generalization of a recently obtained result [8] on simultaneously restricting the size of the automata *and* the number of errors in DFA-to-DFA minimization w.r.t almost-equivalence. This problem was classified to be NP-complete. Thus, for the simultaneously restricted minimization problems, only DFA-to-NFA minimization for almost equivalence lacks a characterization. Since ordinary DFA-to-NFA minimization is PSPACE-complete, the lower bound transfers to the simultaneously restricted problem instance by setting $e = 0$, and the upper bound follows from the more general NFA-to-NFA result from above. In conclusion we obtain:

Corollary 7. *The problem of deciding for a given NFA A and given integers e and n, whether there is an NFA B with n states, such that $|L(A) \triangle L(B)| \leq e$, is* PSPACE*-hard and belongs to* NEXP*. This statement also holds if at most one of the automata A and B is restricted to be deterministic, and the problem becomes* NP*-complete, if both automata A and B are restricted to be deterministic.* □

We summarize our results on the decision versions of minimization problems in Table 2.

3.4 Deciding Minimality

Here we consider the computational complexity of minimality problems. The minimality problem is to decide for a given (deterministic or nondeterministic) finite automaton, whether it is minimal w.r.t. some error profile. For ordinary equivalence, deciding minimality is NL-complete for DFAs [5] and PSPACE-complete for NFAs [17]. For deciding hyper-minimality, we obtain a similar result.

Theorem 8 (Deciding Hyper-Minimality). *The problem of deciding for a given finite automaton A, whether A is hyper-minimal, is* NL*-complete for DFAs, and* PSPACE*-complete for NFAs.* □

When deciding E-minimality, the complexity changes dramatically for DFAs, but remains the same for NFAs.

Theorem 9 (Deciding E-Minimality). *The problem of deciding for two given DFAs A and A_E, whether A is E-minimal, for $E = L(A_E)$, is* coNP*-complete. The problem is* PSPACE*-complete, if A is given as an NFA.* □

4 Counting Minimal Automata

It is well known that for each regular language there is a unique minimal DFA (up to isomorphism) accepting this language. Since hyper-minimal and E-minimal DFAs are not necessarily unique anymore, we are led with the following counting problem: given a DFA A, what is the number of hyper-minimal DFAs B, with $A \sim B$? Naturally, this generalizes to determine the number of E-minimal DFAs, and further to NFAs as input. Counting problems for finite automata were previously investigated in, e.g., [1,12,19]. We show that there is again a significant difference between the computational complexities of questions concerning almost- and E-equivalence. Our first goal is to prove that the counting problem for hyper-minimal DFAs lies in FP. For this we first derive the following lemma.

Lemma 10. *Let A be a hyper-minimal DFA with p states in its preamble, let $K_1, K_2, \ldots, K_m$ be the almost-equivalence classes in the kernel, and let p_i, for $1 \leq i \leq m$, be the number of transitions that lead from preamble states to some state in K_i. Then the number of hyper-minimal DFAs that are almost-equivalent to A is $2^p \cdot \prod_{i=1}^{m} |K_i|^{p_i}$, if $p > 0$, and $|K_s|$, if $p = 0$ and the initial state lies in K_s.* □

Applying this lemma to one of the DFAs in Figure 1 gives $p = 0$, and $|K_s| = 2$, with $s = m = 1$, which means that besides the two depicted automata, there are no other hyper-minimal DFAs, that are almost-equivalent to the depicted ones. Since the values p, $|K_i|$, and p_i, for $1 \leq i \leq m$, from Lemma 10 can be derived from a given DFA in polynomial time, we obtain the following result.

Theorem 11 (Counting Hyper-Minimal DFAs). *Given a DFA A, then the number of hyper-minimal DFAs B satisfying $A \sim B$ can be computed in polynomial time, i.e., it belongs to* FP*.* □

Our next goal is to show that the counting problem for E-minimal DFAs is at least #P-hard, where we use a result from [27], that allows us to compute the coefficients σ_i of a formal power series $\sum \sigma_i x^i$ under certain conditions.

Theorem 12 (Counting E-Minimal DFAs). *Given two DFAs A and A_E, the problem of computing the number of E-minimal DFAs B, with $A \sim_E B$ and $E = L(A_E)$, is* #P*-hard and can be computed in $\# \cdot$* coNP*.*

Proof (Sketch). We only sketch the proof of #P-hardness by a reduction from the MONOTONE 2SAT counting problem, which is shown to be #P-complete in [27], and which is defined as follows. Given a Boolean formula $\varphi = c_0 \wedge c_1 \wedge \cdots \wedge c_{k-1}$ in conjunctive normal form over a set of variables $X = \{x_0, x_1, \ldots, x_{n-1}\}$, where each clause c_i contains exactly two positive literals, i.e., $c_i = (x_{i_1} \vee x_{i_2})$, with $x_{i_1}, x_{i_2} \in X$, for all $i \in \{0, 1, \ldots, k-1\}$, compute the number of satisfying truth assignments. Given an instance φ of this problem, we use the same technique as in the NP-completeness proof of Theorem 6, to construct the DFAs A and A_E. Due to the special structure of these automata, whenever for some truth assignment α exactly t clauses of φ are satisfied by both of their literals, and all other clauses only by one literal, then exactly 2^t different E-minimal DFAs B with $A \sim_E B$ can be derived from α, and any E-minimal DFA that is E-equivalent to A corresponds to a specific assignment. Then the number of E-minimal DFAs B with $A \sim_E B$ is $\sum_{t=0}^{k} \sigma_t 2^t$, where σ_t is the number of assignments, that satisfy exactly t clauses of φ twice, and the others once. Using a technique from [27], we can compute the number of satisfying truth assignments for φ from this value in polynomial time. □

Instead of counting the number of hyper-minimal or E-minimal DFAs, one could also count minimal NFAs. It is easy to see, using a similar strategy as in the proof of Theorem 12, that these three NFA counting problems belong to #PSPACE, which is equal to FPSPACE, the class of functions computable in polynomial space [19]. What can be said about the lower bound on these NFA counting problems? We have to leave open this question for counting minimal and hyper-minimal NFAs. For counting the number of E-minimal NFAs, we can prove #P-hardness with nearly the same proof as for Theorem 12. We summarize our result as follows:

Theorem 13 (Counting E-Minimal NFAs). *Given an NFA A and an additional DFA A_E, the problem of computing the number of E-minimal NFAs B, with $A \sim_E B$ and $E = L(A_E)$, is #P-hard and contained in #PSPACE.* □

References

1. Àlvarez, C., Jenner, B.: A very hard log-space counting class. Theoret. Comput. Sci. 107(1), 3–30 (1993)
2. Badr, A.: Hyper-minimization in $O(n^2)$. Internat. J. Found. Comput. Sci. 20(4), 735–746 (2009)
3. Badr, A., Geffert, V., Shipman, I.: Hyper-minimizing minimized deterministic finite state automata. RAIRO–Inform. Théori. Appl./Theoret. Inform. Appl. 43(1), 69–94 (2009)
4. Birget, J.C.: Intersection and union of regular languages and state complexity. Inform. Process. Lett. 43, 185–190 (1992)
5. Cho, S., Huynh, D.T.: The parallel complexity of finite-state automata problems. Inform. Comput. 97, 1–22 (1992)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. Freeman (1979)

7. Gawrychowski, P., Jeż, A.: Hyper-minimisation Made Efficient. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 356–368. Springer, Heidelberg (2009)
8. Gawrychowski, P., Jeż, A., Maletti, A.: On Minimising Automata with Errors. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 327–338. Springer, Heidelberg (2011)
9. Gold, E.M.: Complexity of automaton identification from given data. Inform. Control 37(3), 302–320 (1978)
10. Gruber, H., Holzer, M.: Finding Lower Bounds for Nondeterministic State Complexity Is Hard (Extended Abstract). In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 363–374. Springer, Heidelberg (2006)
11. Hemaspaandra, L.A., Vollmer, H.: The satanic notations: Counting classes beyond #P and other definitional adventures. SIGACT News 26(1), 2–13 (1995)
12. Holzer, M.: On emptiness and counting for alternating finite automata. In: Developments in Language Theory II (DLT); at the Crossroads of Mathematics, Computer Science and Biology, pp. 88–97. World Scientific (1996)
13. Holzer, M., Maletti, A.: An $n \log n$ algorithm for hyper-minimizing a (minimized) deterministic automaton. Theoret. Comput. Sci. 411(38-39), 3404–3413 (2010)
14. Hopcroft, J.: An $n \log n$ algorithm for minimizing the state in a finite automaton. In: The Theory of Machines and Computations, pp. 189–196. Academic Press (1971)
15. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
16. Immerman, N.: Nondeterministic space is closed under complementation. SIAM J. Comput. 17(5), 935–938 (1988)
17. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. SIAM J. Comput. 22(6), 1117–1141 (1993)
18. Jones, N.D., Lien, Y.E., Laaser, W.T.: New problems complete for nondeterministic log space. Math. Systems Theory 10, 1–17 (1976)
19. Ladner, R.E.: Polynomial space counting problems. SIAM J. Comput. 18(6), 1087–1097 (1989)
20. Maletti, A., Quernheim, D.: Optimal hyper-minimization. Internat. J. Found. Comput. Sci. 22(8), 1877–1891 (2011)
21. Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential time. In: Switching and Automata Theory (SWAT), pp. 125–129. IEEE Society Press (1972)
22. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
23. Szelepcsényi, R.: The method of forced enumeration for nondeterministic automata. Acta Inform. 26(3), 279–284 (1988)
24. Szepietowski, A.: Closure properties of hyper-minimized automata. RAIRO–Inform. Théori. Appl./Theoret. Inform. Appl. 45(4), 459–466 (2011)
25. Toda, S.: Computational Complexity of Counting Complexity Classes. PhD thesis, Tokyo Institute of Technology, Department of Computer Science, Tokyo, Japan (1991)
26. Valiant, L.G.: The complexity of computing the permanent. Theoret. Comput. Sci. 8(2), 189–201 (1979)
27. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)