Antichains: A New Algorithm for Checking Universality of Finite Automata*

M. De Wulf[1], L. Doyen[1**], T. A. Henzinger[2,3], and J.-F. Raskin[1]

[1] CS, Université Libre de Bruxelles, Belgium
[2] I&C, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[3] EECS, University of California at Berkeley, U.S.A.

Abstract. We propose and evaluate a new algorithm for checking the universality of nondeterministic finite automata. In contrast to the standard algorithm, which uses the subset construction to explicitly determinize the automaton, we keep the determinization step implicit. Our algorithm computes the least fixed point of a monotone function on the lattice of antichains of state sets. We evaluate the performance of our algorithm experimentally using the random automaton model recently proposed by Tabakov and Vardi. We show that on the difficult instances of this probabilistic model, the antichain algorithm outperforms the standard one by several orders of magnitude. We also show how variations of the antichain method can be used for solving the language-inclusion problem for nondeterministic finite automata, and the emptiness problem for alternating finite automata.

1 Introduction

The *universality problem* asks, given a nondeterministic finite automaton A over the alphabet Σ, if the language of A contains all finite words over Σ, that is, if $\mathsf{Lang}(A) = \Sigma^*$. This problem is fundamental in automata theory, and several important problems in verification reduce polynomially to this problem. The standard algorithm for universality is to first determinize the automaton using the *subset construction*, and then check for the reachability of a set containing only nonaccepting states. The subset construction may construct a deterministic automaton that is exponentially larger than the original automaton. This explosion is in some sense unavoidable, as the universality problem is known to be PSpace-complete [MS72]. Explicit determinization via the subset construction is also useful to solve a wide range of other problems, such as checking the emptiness of alternating finite automata [CKS81,KV01], checking language inclusion and language equivalence for two nondeterministic finite automata [HMU01], and solving two-player safety games of incomplete information [Rei84].

* This research was supported in part by the NSF grants CCR-0234690 and CCR-0225610, and the Belgian FNRS grant 2.4530.02 of the FRFC project "Centre Fédéré en Vérification."

** Research fellow supported by the Belgian National Science Foundation (FNRS).

Recently, we showed that explicit determinization via the subset construction can be avoided when solving two-player safety games of incomplete information. To avoid the subset construction, we proposed in [DDR06] a *lattice-theoretic solution* that comes in the form of a monotone function on the lattice of antichains of state sets (an *antichain* is a set of $\subseteq$-incomparable sets). The greatest fixed point of this monotone function contains the solution to the strategy synthesis problem. The three main advantages of the antichain method over the subset construction are as follows. First, the new algorithm keeps determinization implicit. Second, the antichain algorithm takes into account the safety objective of the game and computes only what is necessary to establish the existence of a winning strategy for that particular objective. Third, antichains of state sets allow us to store only maximal subsets of states for which a winning strategy exists. This is because if Player I has a strategy to keep the game in safe states starting from a set s of states, then she also has such a strategy for all starting sets $s' \subseteq s$. We show in this paper that the idea of keeping determinization implicit using antichains can also be applied to important problems of automata theory, such as universality and language inclusion for nondeterministic automata, and emptiness for alternating automata.

First, we show that the universality problem for nondeterministic finite automata can be solved on the lattice of antichains of state sets using a variation of the monotone function proposed in our previous work. We reduce the universality problem to a two-player reachability game of incomplete information, which can be solved by computing the least fixed point of this monotone function. We implemented this solution using NuSMV [CCGR99] and the CUDD library [Som98]. To compare the performance of the antichain algorithm to the performance of various implementations of subset-construction based algorithms, we used a large set of examples generated in the probabilistic framework by Tabakov and Vardi [TV05]. This framework was proposed with the express purpose of comparing the performances of algorithms on finite automata. In their experiments, the authors conclude that explicit determinization as implemented in [Mø04] outperforms the algorithm of Brzozowski [BL80] as well as newer implementations, which use symbolic methods for the subset construction. Our experimental results show that our implementation of the antichain algorithm is considerably faster, on the entire parameter space of the probabilistic framework, than the most efficient implementation of the standard algorithm. In particular, on the most difficult instances of the probabilistic framework, the antichain algorithm outperforms [Mø04] by two orders of magnitude. For this comparison, we are limited to automata with approximately 175 states, which is the limit that the explicit-determinization approach can handle on the most expensive instances of the probabilistic framework. On these difficult instances, the antichain approach scales much better: we are able to successfully check universality for automata with several thousands of states in less than 10 seconds.

Second, to show the generality of the antichain approach, we also give new algorithmic solutions to the language-inclusion problem for nondeterministic automata, and to the emptiness problem for alternating automata. Again, no ex-

plicit determinization is performed. To solve the emptiness problem for alternating automata, we use the same lattice as for universality and only change the monotone function that operates on the lattice. To solve the language-inclusion problem for nondeterministic automata, we need a slightly richer lattice.

Structure of the paper In Section 2, we review some basic notions about finite automata. In Section 3, we introduce the lattice of antichains of state sets, and we present the antichain algorithm for the universality problem for nondeterministic automata. In Section 4, we report on two different symbolic implementations of the antichain algorithm, and we compare their performances with the classical algorithm that uses explicit determinization. In Section 5, we give antichain-based solutions for nondeterministic language inclusion and alternating emptiness.

2 Finite Automata

Definitions A (*nondeterministic*) *finite automaton*, NFA for short, is a tuple $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$, where Loc is a finite set of states (or locations), $\mathsf{Init} \subseteq \mathsf{Loc}$ is the set of initial states, $\mathsf{Fin} \subseteq \mathsf{Loc}$ is the set of accepting (or final) states, Σ is a finite alphabet, and $\delta \subseteq \mathsf{Loc} \times \Sigma \times \mathsf{Loc}$ is a (nondeterministic) transition relation. A *deterministic* finite automaton, DFA for short, is an NFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$ such that for all states $\ell \in \mathsf{Loc}$ and all letters $\sigma \in \Sigma$, there exists a unique state $\ell' \in \mathsf{Loc}$ such that $\delta(\ell, \sigma, \ell')$. A *run* of the NFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$ over a finite word $w = \sigma_1 \dots \sigma_n$ is a sequence $r = \ell_0 \ell_1 \dots \ell_n$ of states such that (1) $\ell_0 \in \mathsf{Init}$ and (2) $\delta(\ell_i, \sigma_{i+1}, \ell_{i+1})$ for all $0 \leq i < n$. The run r is *accepting* iff $\ell_n \in \mathsf{Fin}$. The *language* $\mathsf{Lang}(A)$ accepted by A is the set of words $w \in \Sigma^*$ such that A has an accepting run over w.

Notations Given a finite word $w = \sigma_1 \dots \sigma_n$ of size $|w| = n$, we write $w(i) = \sigma_i$ for the i-th letter of w, and $w(0) = \varepsilon$ for the empty word. Given an NFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$, a state set $s \subseteq \mathsf{Loc}$, and a letter $\sigma \in \Sigma$, we define $\mathsf{post}^A_\sigma(s) = \{\ell' \in \mathsf{Loc} \mid \exists \ell \in s : \delta(\ell, \sigma, \ell')\}$, $\mathsf{pre}^A_\sigma(s) = \{\ell \in \mathsf{Loc} \mid \exists \ell' \in s : \delta(\ell, \sigma, \ell')\}$, and $\mathsf{cpre}^A_\sigma(s) = \{\ell \in \mathsf{Loc} \mid \forall \ell' \in \mathsf{Loc} : \delta(\ell, \sigma, \ell') \rightarrow \ell' \in s\}$. Note that $\mathsf{Loc} \setminus \mathsf{cpre}^A_\sigma(s) = \mathsf{pre}^A_\sigma(\mathsf{Loc} \setminus s)$.

Operations Given two NFAs A and B, we denote by $A \otimes B$ the synchronous product of the two automata, and by $A \oplus B$ the sum of the automata. The language accepted by the product is $\mathsf{Lang}(A \otimes B) = \mathsf{Lang}(A) \cap \mathsf{Lang}(B)$ and the language accepted by the sum is $\mathsf{Lang}(A \oplus B) = \mathsf{Lang}(A) \cup \mathsf{Lang}(B)$. Given a DFA A, we denote by $\overline{A}$ the complement of A, which accepts the language $\mathsf{Lang}(\overline{A}) = \Sigma^* \setminus \mathsf{Lang}(A)$.

Problems The *emptiness problem* for NFAs is to decide, given an NFA A, if $\mathsf{Lang}(A) = \emptyset$. This problem is solvable in time linear in the size of A. The *universality problem* for NFAs is to decide, given an NFA A, if $\mathsf{Lang}(A) = \Sigma^*$. This problem is much harder than emptiness: it is complete for PSPACE [MS72].

The classical algorithm for deciding universality first determinizes A, and then checks emptiness of the complement. The difficult step is the determinization, as it may cause an exponential blow-up in the number of states of the automaton. The *language-inclusion problem* for NFAs is to decide, given two NFAs A and B, if $\mathsf{Lang}(A) \subseteq \mathsf{Lang}(B)$. This problem is also complete for PSPACE. The classical algorithm for deciding language inclusion checks emptiness of the product of A with the complement of B. In the next section, we propose a new approach to solve the universality problem, which does not involve explicit determinization, and later we extend the approach to solve also language inclusion.

3 A Fixed Point to Solve Universality

Two lattices of antichains Let Loc be a set (in our case, a set of states of some automaton). An *antichain* over Loc is a set $q \subseteq 2^{\mathsf{Loc}}$ such that $\forall s, s' \in q : s \not\subset s'$. Thus q is a set of pairwise incomparable subsets of Loc (with regard to set inclusion). We denote by L the set of antichains over Loc. We define the following partial orders: for two antichains $q, q' \in L$, let $q \sqsubseteq q'$ if $\forall s \in q \cdot \exists s' \in q' : s \subseteq s'$, and let $q \mathrel{\widetilde{\sqsubseteq}} q'$ if $\forall s \in q \cdot \exists s' \in q' : s' \subseteq s$. The two partial orders $\sqsubseteq$ and $\widetilde{\sqsubseteq}$ yield complete lattices on the set L of antichains. This can be seen as follows. Given a set $q \subseteq 2^{\mathsf{Loc}}$ (not necessarily an antichain), a set $s \in q$ is *maximal* in q iff $\forall s' \in q : s \not\subset s'$. Similarly, $s \in q$ is *minimal* in q iff $\forall s' \in q : s' \not\subset s$. We write $\lceil q \rceil$ (resp. $\lfloor q \rfloor$) for the set of maximal (resp. minimal) elements of q. Given two antichains $q, q' \in L$, the $\sqsubseteq$-lub (least upper bound) of q and q' is the antichain $q \sqcup q' = \lceil \{s \mid s \in q \vee s \in q'\} \rceil$; the $\sqsubseteq$-glb (greatest lower bound) is the antichain $q \sqcap q' = \lceil \{s \cap s' \mid s \in q \wedge s' \in q'\} \rceil$. Similarly, the $\widetilde{\sqsubseteq}$-lub is $q \mathrel{\widetilde{\sqcup}} q' = \lfloor \{s \mid s \in q \vee s \in q'\} \rfloor$, and the $\widetilde{\sqsubseteq}$-glb is $q \mathrel{\widetilde{\sqcap}} q' = \lfloor \{s \cup s' \mid s \in q \wedge s' \in q'\} \rfloor$.

These definitions can be extended to lub's and glb's of arbitrary sets in the obvious way, yielding the operators $\bigsqcup$, $\bigsqcap$, $\widetilde{\bigsqcup}$, and $\widetilde{\bigsqcap}$. Adding suitable bottom and top elements, we obtain the following lemma.

Lemma 1 $\langle L, \sqsubseteq, \bigsqcup, \bigsqcap, \emptyset, \{\mathsf{Loc}\}\rangle$ *and* $\langle L, \widetilde{\sqsubseteq}, \widetilde{\bigsqcup}, \widetilde{\bigsqcap}, \emptyset, \{\emptyset\}\rangle$ *are complete lattices.*

We call these two lattices the *lattice of antichains* and the *dual lattice of antichains*, respectively. We show how to solve the universality problem for nondeterministic finite automata using either lattice.

Game interpretation of universality Consider the following game played by a protagonist and an antagonist. The protagonist wants to establish that a given NFA A does not accept the language Σ^*. The protagonist has to provide a finite word w such that, no matter which run of A over w the antagonist chooses, the run does not end in an accepting state. This game is a one-shot game. However, to obtain a fixed point solution to the universality problem, we can consider a multi-round game interpretation of this problem: in each round of the game, the protagonist provides a single letter σ, and the antagonist decides how to update the state of A on input σ according to the nondeterministic transition

relation. To be equivalent to the one-shot game, the protagonist must not be able to observe the state of the automaton, which is chosen by the antagonist. So, we have to consider a game where the protagonist cannot distinguish between states of the automaton: this is a *game of imperfect information*. We can solve the universality problem by looking for the existence of winning strategies in such games. In a recent paper, we showed that safety games of imperfect information can be solved by computing the greatest fixed point of a monotone function on the lattice of antichains [DDR06]. We show here that reachability games of imperfect information can be solved by computing a least fixed point on this lattice. This gives a new algorithm for checking universality.

Using the lattice of antichains to solve universality Given an NFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$, we define the following monotone function on the lattice L of antichains over Loc. For an antichain $q \in L$, let

$$\mathsf{CPre}^A(q) = \lceil \{ s \mid \exists s' \in q \cdot \exists \sigma \in \Sigma : s = \mathsf{cpre}^A_\sigma(s') \} \rceil.$$

So, a set s of states belongs to the antichain $\mathsf{CPre}^A(q)$ iff it is maximal and there exist a state set $s' \in q$ and a letter $\sigma \in \Sigma$ such that for all states $\ell \in s$, the set of states ℓ' with $\delta(\ell, \sigma, \ell')$ is in s'. This monotone function can be used to solve the universality problem for NFAs. This is formalized in the next theorem.

Theorem 2 *Let* $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$ *be an NFA, and let* $\mathcal{F} = \bigsqcap \big\{ q \mid q = \mathsf{CPre}^A(q) \sqcup \{\overline{\mathsf{Fin}}\} \big\}$. *Then* $\mathsf{Lang}(A) \neq \Sigma^*$ *iff* $\{\mathsf{Init}\} \sqsubseteq \mathcal{F}$.

Proof. First, assume that $\mathsf{Lang}(A)$ is not universal. Let $w \in \Sigma^* \setminus \mathsf{Lang}(A)$ be a word of size $|w| = n$. Consider the sequence $s_0, s_1, \ldots, s_n$ of state sets such that (1) $s_0 = \mathsf{Init}$, (2) $s_i = \mathsf{post}^A_{w(i)}(s_{i-1})$ for all $1 \leq i \leq n$, and (3) $s_n \subseteq \overline{\mathsf{Fin}}$ (recall that A has no accepting run over w). We prove by induction on k that $\{s_{n-k}\} \sqsubseteq \mathcal{F}$. For $k = 0$, since $s_n \subseteq \overline{\mathsf{Fin}}$, we obtain immediately $\{s_n\} \sqsubseteq \mathcal{F}$. For the inductive case, assume that $\{s_{n-k}\} \sqsubseteq \mathcal{F}$ for all $0 \leq k < i$, and let us show that $\{s_{n-i}\} \sqsubseteq \mathcal{F}$. Observe that by definition, for $\sigma = w(n-i+1)$ we have $\mathsf{post}^A_\sigma(s_{n-i}) = s_{n-i+1}$. Therefore $\{s_{n-i}\} \sqsubseteq \mathsf{CPre}^A(\{s_{n+1-i}\})$, and by the monotonicity of CPre^A and the induction hypothesis, we get $\{s_{n-i}\} \sqsubseteq \mathsf{CPre}^A(\mathcal{F})$ and $\{s_{n-i}\} \sqsubseteq \mathsf{CPre}^A(\mathcal{F}) \sqcup \{\overline{\mathsf{Fin}}\}$, which is equivalent to $\{s_{n-i}\} \sqsubseteq \mathcal{F}$, as $\mathcal{F}$ is a fixed point. In particular, we have $\{s_0\} \sqsubseteq \mathcal{F}$, that is, $\{\mathsf{Init}\} \sqsubseteq \mathcal{F}$.

Second, assume that $\{\mathsf{Init}\} \sqsubseteq \mathcal{F}$. We construct a word $w \notin \mathsf{Lang}(A)$. Consider the infinite sequence $q_0, q_1, q_2, \ldots$ of antichains defined by (1) $q_0 = \emptyset$ and (2) $q_i = \mathsf{CPre}^A(q_{i-1}) \sqcup \{\overline{\mathsf{Fin}}\}$ for all $i \geq 1$. By Tarski's fixed point theorem, we know that $\mathcal{F} = q_n$ for some $n \in \mathbb{N}$. We construct an integer $k < n$, a sequence $s_0, s_1, \ldots, s_k$ of $k+1$ state sets, and a word w of size k such that $\{s_i\} \sqsubseteq \mathsf{CPre}^A(q_{n-i-1})$ and $\mathsf{post}^A_{w(i+1)}(s_i) \subseteq s_{i+1}$ for all $0 \leq i < k$. We start with $s_0 = \mathsf{Init}$ so that $\{s_0\} \sqsubseteq q_n$. Then, we have either $\{s_0\} \sqsubseteq \{\overline{\mathsf{Fin}}\}$ or $\{s_0\} \sqsubseteq \mathsf{CPre}^A(q_{n-1})$ (because $\{s_0\}$ is a singleton). In the first case, we stop the construction with $k = 0$ and $w = \varepsilon$. In the second case, we continue the construction inductively. Assume that we have constructed $\{s_{i-1}\} \sqsubseteq \mathsf{CPre}^A(q_{n-i})$ for some $i \geq 1$. By the definition of CPre^A, we

know that there are $\sigma_i \in \Sigma$ and $s_i \in q_{n-i}$ such that $\mathsf{post}^A_{\sigma_i}(s_{i-1}) \subseteq s_i$. We choose $w(i) = \sigma_i$. Then $\{s_i\} \sqsubseteq q_{n-i}$, and thus either $\{s_i\} \sqsubseteq \{\overline{\mathsf{Fin}}\}$ and we stop with $k = i$ and $w = \sigma_1 \ldots \sigma_i$, or $\{s_i\} \sqsubseteq \mathsf{CPre}^A(q_{n-i-1})$. This construction stops for some $k < n$, as $q_1 = \{\overline{\mathsf{Fin}}\}$ and $\{s_k\} \sqsubseteq \{\overline{\mathsf{Fin}}\}$. The sequence $s_0, s_1, \ldots, s_k$ shows that A has no accepting run over w, because (1) $s_0 = \mathsf{Init}$, (2) $\mathsf{post}^A_{w(i)}(s_{i-1}) \subseteq s_i$ for all $1 \leq i \leq k$, and (3) $s_k \subseteq \overline{\mathsf{Fin}}$. Hence $w \notin \mathsf{Lang}(A)$. ■

The algorithm that consists in computing the least fixed point $\mathcal{F}$ from Theorem 2 through the successive approximation sequence $q_0 \sqsubseteq q_1 \sqsubseteq q_2 \sqsubseteq \cdots$ (as defined in the proof) is called the *backward antichain algorithm*. The computation is similar to the subset construction used in the backward determinization of A, with the essential difference that it maintains only sets of states that are *maximal* in the subset-inclusion order.

Using the dual lattice of antichains to solve universality In the previous algorithm, the automaton is traversed backward starting from the set of nonaccepting states. Using the dual lattice of antichains, we can formulate a solution that traverses the automaton forward starting from the set of initial states. Given an NFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$ and an antichain $q \in L$, let

$$\mathsf{Post}^A(q) = \lfloor \{s \mid \exists s' \in q \cdot \exists \sigma \in \Sigma : s = \mathsf{post}^A_\sigma(s')\} \rfloor.$$

This function is monotone on the dual lattice of antichains. We can solve the universality problem for NFAs by iterating Post as follows, defining a *forward antichain algorithm*.

Theorem 3 *Let $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$ be an NFA, and let $\tilde{\mathcal{F}} = \widetilde{\bigsqcap} \{q \mid q = \mathsf{Post}^A(q) \mathbin{\tilde{\sqcup}} \{\mathsf{Init}\}\}$. Then $\mathsf{Lang}(A) \neq \Sigma^*$ iff $\{\overline{\mathsf{Fin}}\} \mathrel{\tilde{\sqsubseteq}} \tilde{\mathcal{F}}$.*

The computation of the least fixed point $\tilde{\mathcal{F}}$ is similar to the standard, forward subset construction used in the determinization of A, with the essential difference that it maintains only *minimal* sets of states.

Relationship between forward and backward algorithms Given an NFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$, the *reverse* of A is the NFA $B = \langle \mathsf{Loc}, \overline{\mathsf{Fin}}, \overline{\mathsf{Init}}, \Sigma, \delta' \rangle$, where for all states $\ell, \ell' \in \mathsf{Loc}$ and all letters $\sigma \in \Sigma$, we have $\delta'(\ell, \sigma, \ell')$ iff $\delta(\ell', \sigma, \ell)$. Note that for all $\sigma \in \Sigma$ and all $s \subseteq \mathsf{Loc}$, we have $\mathsf{pre}^A_\sigma(s) = \mathsf{post}^B_\sigma(s)$. For a set $s \subseteq \mathsf{Loc}$, let $\overline{s}$ be the complement of s relative to Loc, that is, $\overline{s} = \mathsf{Loc} \setminus s$. For a set $q \subseteq 2^{\mathsf{Loc}}$, let $\tilde{q} = \{\overline{s} \mid s \in q\}$. Note that $\tilde{q}$ is an antichain iff q is an antichain, and $\widetilde{\lfloor q \rfloor} = \lceil \tilde{q} \rceil$.

Lemma 4 *Let $A = (\mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta)$ be an NFA, let B be its reverse, and let q be an antichain over Loc. Then $q' = \mathsf{CPre}^A(q)$ iff $\tilde{q}' = \mathsf{Post}^B(\tilde{q})$.*

From this lemma, it follows that the forward and backward approaches are equivalent in the following sense: for every instance A of the universality problem that

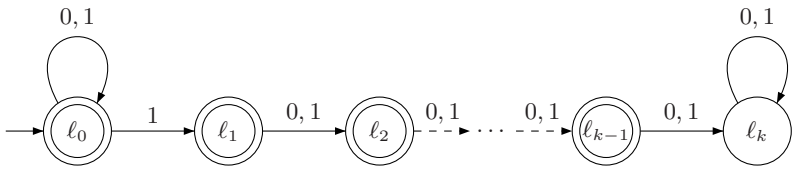

Fig. 1. A family of NFAs A_k, $k \geq 2$, for Theorem 5.

is difficult for the forward antichain algorithm, there is an equally difficult instance (namely, the reverse of A) for the backward antichain algorithm, and vice versa. Indeed, let $q_0 \sqsubseteq q_1 \sqsubseteq q_2 \sqsubseteq \cdots$ be the sequence of antichains that are constructed when computing the least fixed point $\mathcal{F}$ from Theorem 2 (as defined in the proof of the theorem); and let $q'_0 \mathrel{\widetilde{\sqsubseteq}} q'_1 \mathrel{\widetilde{\sqsubseteq}} q'_2 \mathrel{\widetilde{\sqsubseteq}} \cdots$ be the sequence of antichains that are constructed when computing the least fixed point $\tilde{\mathcal{F}}$ from Theorem 3, defined as follows: (1) $q'_0 = \emptyset$ and (2) $q'_i = \mathsf{Post}^B(q'_{i-1}) \mathbin{\widetilde{\sqcup}} \{\overline{\mathsf{Fin}}\}$ for all $i \geq 1$ (where B is the reverse of A[1]). Using Lemma 4 and induction, we can prove that $q_i = \tilde{q}'_i$ for all $i \geq 0$.

Comparison with explicit determinization We call the classical algorithm for solving the universality problem for NFAs the *subset algorithm*: it first determinizes the NFA using a subset construction, and then checks if every reachable state in the resulting DFA is accepting. The determinization is stopped whenever a rejecting state is encountered. Usually, the DFA is constructed in a breadth-first forward search, but it can also be done in a backward fashion.

Theorem 5 *For checking universality, there exists an infinite family of NFAs A_k, with $k \geq 2$ states, for which the forward subset algorithm is exponential, and the (forward and backward) antichain algorithms are polynomial. There also exists an infinite family of NFAs B_k for which the backward subset algorithm is exponential, and the antichain algorithms are polynomial.*

Proof. Consider the family of NFAs A_k, $k \geq 2$, over the alphabet $\Sigma = \{0, 1\}$ shown in Fig. 1. The automaton A_k has $k + 1$ states, $\ell_0, \ldots, \ell_k$, all accepting except ℓ_k. There is only one initial state: $\mathsf{Init} = \{\ell_0\}$. Every A_k is universal, as the initial state has a self-loop labeled with Σ. The forward determinization of A_k has 2^k states. Hence the forward subset algorithm is exponential on the family A_k, $k \geq 2$. However, the backward antichain algorithm terminates in polynomial time, as the sequence $q_0 = \{\{\ell_k\}\}$, and $q_{i+1} = \mathsf{CPre}^{A_k}(q_i) \sqcup \{\{\ell_k\}\}$ for $i \geq 0$, stabilizes after k iterations with $q_i = \{\{\ell_{k-i}, \ldots, \ell_k\}\}$ for $i < k$, and $q_k = q_{k-1}$. The test $\{\mathsf{Init}\} \sqsubseteq q_i$ requires linear time. The forward antichain algorithm terminates after a single iteration with $\tilde{\mathcal{F}} = \{\mathsf{Init}\}$, and the test $\{\{\ell_k\}\} \mathrel{\widetilde{\sqsubseteq}} \tilde{\mathcal{F}}$ is done in constant time.

[1] Remember that $\overline{\mathsf{Fin}}$ is the set of initial states in B.

Algorithm 1: Backward antichain algorithm for testing universality.

Data : a nondeterministic finite automaton $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$.

begin

1 $\mathsf{Start} \leftarrow \{\mathsf{Init}\}$;

2 $F \leftarrow \{\overline{\mathsf{Fin}}\}$;

3 $\mathsf{Frontier} \leftarrow F$;

4 **while** $(\mathsf{Frontier} \neq \emptyset) \wedge (\mathsf{Start} \not\sqsubseteq \mathsf{Frontier})$ **do**

5 $\quad \mathsf{Frontier} \leftarrow \{q \in \mathsf{CPre}^A(\mathsf{Frontier}) \mid q \not\sqsubseteq F\}$;

6 $\quad F \leftarrow F \sqcup \mathsf{Frontier}$;

7 **return** $(\mathsf{Start} \not\sqsubseteq \mathsf{Frontier})$;

end

A similar proof holds for the second part of the theorem: for the family B_k, $k \geq 2$, choose each B_k to be the reverse of A_k. ■

4 Implementation and Practical Evaluation

Two symbolic implementations of antichains We implemented our new algorithm for testing universality on top of NuSMV [CCGR99] and the BDD library CUDD [Som98]. We considered two encodings of NFAs in NuSMV, and correspondingly, two encodings of antichains of state sets using BDDs.

Fully symbolic encoding In the first encoding, we associate a boolean variable with each state of an NFA. A valuation of the variables corresponds to a state set, and a BDD represents a set of state sets. Two valuations v_1 and v_2 for a set X of variables are incomparable iff there exist $x, y \in X$ such that $v_1(x) > v_2(x)$ and $v_1(y) < v_2(y)$. If the BDD contains only valuations that are incomparable, then it symbolically represents an antichain of state sets. We call this encoding *fully symbolic*.

Semi-symbolic encoding In the second encoding, we associate an integer with each state of the automaton. Then a single integer counter is used to encode the current state. A BDD represents a set of integer values and so a set of states. An antichain of state sets is represented by a set of BDDs that are incomparable for valuation inclusion. We call this encoding *semi-symbolic*.

Algorithm For both encodings, we use the backward Algorithm 1 to check universality. To avoid computing CPre twice for the same set, the algorithm computes iteratively CPre only on the frontier sets, which are the sets that were added to the approximation F of the least fixed point $\mathcal{F}$ in the previous iteration. When the automaton is not universal, then $\mathcal{F}$ is not fully computed, because we stop the computation as soon as one of the sets in F contains all initial states.

The randomized model To evaluate the antichain algorithm and compare with the subset algorithm, we use a random model to generate NFAs. This model was recently proposed by Tabakov and Vardi to compare the efficiency of some algorithms for automata [TV05]. In the model, the input alphabet is fixed to $\Sigma = \{0, 1\}$, and for each letter $\sigma \in \Sigma$, a number k_σ of different state pairs $(\ell, \ell') \in \mathsf{Loc} \times \mathsf{Loc}$ are chosen uniformly at random before the corresponding transitions (ℓ, σ, ℓ') are added to the automaton. The ratio $r_\sigma = \frac{k_\sigma}{|\mathsf{Loc}|}$ is called the *transition density* for σ. This ratio represents the average outdegree of each state for σ. In all experiments, we choose $r_0 = r_1$, and denote the transition density by r. The model contains a second parameter: the *density f of accepting states*. There is only one initial state, and the number m of accepting states is linear in the total number of states, as determined by $f = \frac{m}{|\mathsf{Loc}|}$. The accepting states themselves are chosen uniformly at random. Observe that since the transition relation is not always total, automata with $f = 1$ are not necessarily universal.

Tabakov and Vardi have studied the space of parameter values for this model and argue that "interesting" automata are generated by the model as the two parameters r and f vary. They have run large tests to evaluate the probability for an automaton to be universal as a function of the parameters. We reproduced those experiments for a greater space of parameter values and obtained a similar distribution (Fig. 2). To generate each sample point, we checked the universality of 200 random automata with 30 states.

Performance comparison We compare the performance of the backward antichain algorithm with the tool `dk.brics.automaton` developed by Møller [Mø04], which implements the forward subset algorithm and stops determinization whenever a rejecting state is encountered. According to the experiments of Tabakov and Vardi, this tool, which uses explicit state representation, is the most efficient one for checking universality [TV05]. For the comparison, we use the semi-symbolic encoding of antichains, as that turns out to be much more efficient than the fully symbolic encoding. The comparison is carried out on the whole parameter space of the randomized model. All experiments are conducted on a biprocessor Linux station (two 3.06Ghz Intel Xeons with 4GB of RAM). We only measure the execution times for the universality test in both approaches, not the time for parsing the input files and constructing the initial data structures.

In Fig. 3, Fig. 4, and Fig. 5, we present the execution times for checking universality by the explicit subset algorithm and the semi-symbolic antichain algorithm. To generate each sample point, we check the universality of 100 random automata with $|\mathsf{Loc}| = 175$ (this is roughly the largest size that the subset algorithm is able to handle on the entire parameter space with the available memory). In Fig. 3, we present the median execution times for testing universality by the subset approach as a function of r (transition density) and f (density of accepting states). The figure shows that the universality test is most difficult when $r = 2$ and $f = 1$. For the same instances, the median execution time of our algorithm is always less than the time unit of the system clock (1ms).

In Fig. 4 and Fig. 5, we present the average execution times for testing universality by the subset approach and the semi-symbolic antichain approach, re-

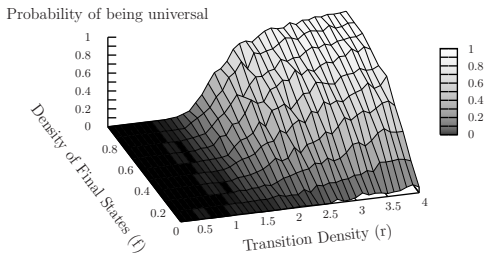

Fig. 2. Probability of universal automata ($|\mathsf{Loc}| = 30$).

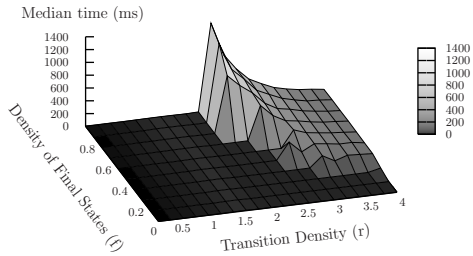

Fig. 3. Median execution time for the subset algorithm ($|\mathsf{Loc}| = 175$).

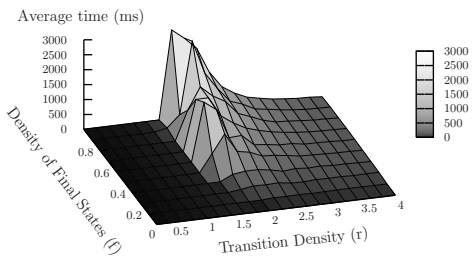

Fig. 4. Average execution time for the subset algorithm ($|\mathsf{Loc}| = 175$).

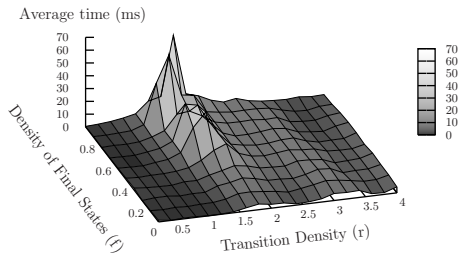

Fig. 5. Average execution time for the semi-symbolic antichain algorithm ($|\mathsf{Loc}| = 175$).

spectively. Both figures exhibit similar peaks, showing that the difficult instances are roughly the same for both approaches. However, the antichain algorithm is much faster. For the most difficult parameter values ($r = 2$ and $f = 1$), the antichain algorithm is 165 times faster than the subset algorithm. Intuitively, these instances are difficult for both algorithms for the following two reasons. First, the probability to be universal for these parameter values is around 50 percent, and we believe that most of these instances are neither trivially universal nor trivially nonuniversal. Second, when an automaton is universal, the subset method has to build the entire deterministic automaton, and the antichain method has to complete the computation of the least fixed point.

In Fig. 6 we present the ratio of the average time for the subset approach and the average time for the antichain approach as a function of the densities. The comparison for $r \leq 1.4$ and $f \leq 0.2$ is not very significant, because the execution times are very close to the precision of the system clock (1ms). For the rest of the parameter space, the antichain algorithm performs always better (up to 200 times better). Finally, in Fig. 7, we show that the semi-symbolic antichain approach scales well when the size of the automaton increases, in contrast to

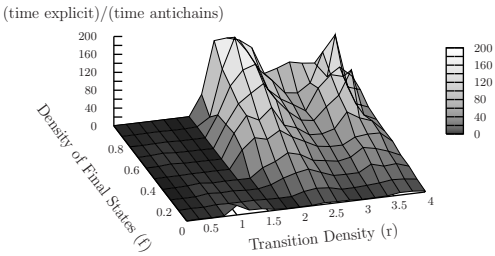

Fig. 6. Average execution time ratio ($|\mathsf{Loc}| = 175$).

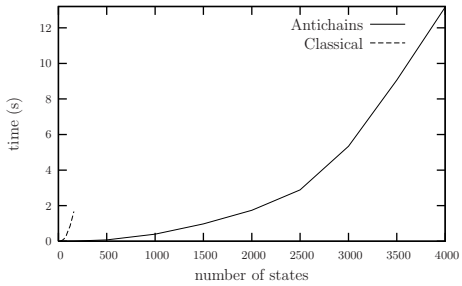

Fig. 7. Average execution times for the subset and semi-symbolic antichain algorithms (transition density 2; accepting-states density 1).

the subset approach. For the experiments we generated randomly 100 automata per sample point for automaton sizes under 200 states, and 30 automata per sample point for sizes over 200 states. The densities are again $r = 2$ and $f = 1$. The antichain algorithm is able to handle random automata with 4000 states in the average time of 12s. The average size of the final antichain (for universal automata) is 217 state sets for automata with 4000 states. We did not pursue experiments with larger automata, because we would have had to modify the automaton generator, as it is not designed for such large automaton sizes. The subset algorithm quickly exceeds the memory limit when the number of states nears 200, so the curve is quite short in the left corner of Fig. 7.

As mentioned above, the semi-symbolic antichain encoding gives far better performances on the random model than the fully symbolic encoding, as shown in Table 1 for the difficult instances ($r = 2$ and $f = 1$). It also turns out that the fully symbolic encoding does not scale well when the size of the automaton increases. Each sample point is computed on a set of 50 random automata with less than 100 states. For 175 states, the sample size is 100, and for more states, the sample size is 30. The number of boolean variables of the BDDs that encode antichains seems to be the reason for the difference in performances: the number of boolean variables grows linearly with the number of states in the fully symbolic encoding, but logarithmically in the semi-symbolic encoding. We have also implemented the forward antichain algorithm with the semi-symbolic encoding. On the random model, this approach is roughly twice as slow as the backward antichain algorithm, which is still better by several orders of magnitude than the subset algorithm. See Fig. 8 for the experimental results.

5 Beyond Universality

Language inclusion We show that language inclusion can be checked using an antichain algorithm based on a slightly richer lattice. Consider two NFAs

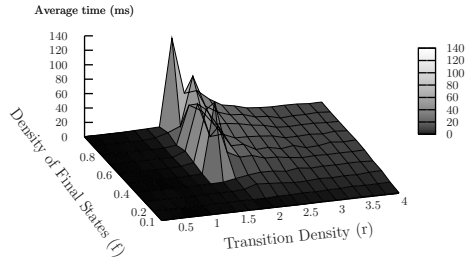

Fig. 8. Average execution time for the forward semi-symbolic antichain algorithm ($|\mathsf{Loc}|$=175).

Table 1. Average execution times (ms) for checking universality with $r = 2$ and $f = 1$.

number of states	20	40	60	80	100	175	500	1000	1500	2000	2500	3000	3500	4000
subset algorithm	23	50	141	309	583	2257	–	–	–	–	–	–	–	–
fully symb. antich.	3	14	70	175	421	6400	–	–	–	–	–	–	–	–
semi-symb. antich.	1	2	2	3	5	14	76	400	973	1741	2886	5341	9063	13160

$A = \langle \mathsf{Loc}_A, \mathsf{Init}_A, \mathsf{Fin}_A, \Sigma, \delta_A \rangle$ and $B = \langle \mathsf{Loc}_B, \mathsf{Init}_B, \mathsf{Fin}_B, \Sigma, \delta_B \rangle$ over the same alphabet. We wish to check whether $\mathsf{Lang}(A) \subseteq \mathsf{Lang}(B)$. An *antichain* over $\mathsf{Loc}_A \times 2^{\mathsf{Loc}_B}$ is a set $q \in 2^{\mathsf{Loc}_A \times 2^{\mathsf{Loc}_B}}$ such that for all $(\ell_1, s_1), (\ell_2, s_2) \in q$ with $\ell_1 = \ell_2$ and $s_1 \neq s_2$, we have neither $s_1 \subseteq s_2$ nor $s_2 \subseteq s_1$. Given a set $q \in 2^{\mathsf{Loc}_A \times 2^{\mathsf{Loc}_B}}$, an element $(\ell, s) \in q$ is *maximal* iff for every s' with $s' \supset s$, we have $(\ell, s') \notin q$. We denote by $\lceil q \rceil$ the set of maximal elements of q. Given two antichains q and q', we define

$$
\begin{aligned}
&q \sqsubseteq_l q' \text{ iff } \forall (\ell, s) \in q \cdot \exists (\ell, s') \in q' : s \subseteq s';\\
&q \sqcup_l q' = \lceil \{(\ell, s) \mid (\ell, s) \in q \lor (\ell, s) \in q'\} \rceil;\\
&q \sqcap_l q' = \lceil \{(\ell, s \cap s') \mid (\ell, s) \in q \land (\ell, s') \in q'\} \rceil.
\end{aligned}
$$

Let $\mathsf{CPre}_l(q) = \lceil \{(\ell, s) \mid \exists \sigma \in \Sigma \cdot \exists (\ell', s') \in q : \ell' \in \delta_A(\ell, \sigma) \land \mathsf{post}^B_\sigma(s) \subseteq s'\} \rceil$.

Theorem 6 *Let A and B be two finite automata, and let $\mathcal{F}_l = \bigsqcap_l \{q \mid q = \mathsf{CPre}_l(q) \sqcup_l (\mathsf{Fin}_A \times \{\overline{\mathsf{Fin}}_B\})\}$. Then $\mathsf{Lang}(A) \not\subseteq \mathsf{Lang}(B)$ iff there exists a state $\ell \in \mathsf{Init}_A$ such that $\{(\ell, \mathsf{Init}_B)\} \sqsubseteq_l \mathcal{F}_l$.*

Typically, A is an "implementation" automaton, and B a "specification" automaton. Often A is given as a synchronous product of automata, that is, $A = A_1 \otimes \cdots \otimes A_n$. Then we can apply our method with antichains over $\mathsf{Loc}_{A_1} \times \cdots \times \mathsf{Loc}_{A_n} \times 2^{\mathsf{Loc}_B}$. However, in the common case where the implementation components A_i are deterministic (but the specification B is nondeterministic), an alternative approach is possible, and likely more efficient. The

following lemma shows that in this case, the language-inclusion problem can be reduced in polynomial time to the universality problem. This reduction has the advantage of avoiding the construction of the product of the implementation components.

Lemma 7 *For a set $A_1, \ldots, A_n$ of DFAs and an NFA B, we define the sum $C = \overline{A_1} \oplus \cdots \oplus \overline{A_n} \oplus B$. Then $\mathsf{Lang}(A_1) \cap \ldots \cap \mathsf{Lang}(A_n) \subseteq \mathsf{Lang}(B)$ iff $\mathsf{Lang}(C) = \Sigma^*$.*

Emptiness of alternating automata The antichain algorithm for checking the universality of NFAs can be generalized to checking the emptiness of alternating automata, using the same lattice with a slight modification of the function CPre. In alternating automata, the transitions are given by boolean formulas. For example, $\rho(\ell, \sigma) = \ell_1 \vee (\ell_2 \wedge \ell_3)$ means that in state ℓ, a word of the form $\sigma \cdot w$ is accepted if either w is accepted in ℓ_1, or w is accepted in both ℓ_2 and ℓ_3. Our formal definitions follow [KV01]. Let $\mathcal{B}^+(\mathsf{Loc})$ be the set of monotone boolean formulas over Loc, defined by the grammar $\varphi ::= \mathsf{true} \mid \ell \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$, where $\ell \in \mathsf{Loc}$. A set $s \subseteq \mathsf{Loc}$ of states *satisfies* a formula $\varphi \in \mathcal{B}^+(\mathsf{Loc})$ (denoted $s \models \varphi$) iff φ is equivalent to true when the states in s are replaced by true, and the states in $\mathsf{Loc} \setminus s$ by false.

An *alternating finite automaton*, or AFA, is a tuple $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \rho \rangle$, where Loc, Init, Fin, and Σ are as for NFAs, and ρ: $\mathsf{Loc} \times \Sigma \to \mathcal{B}^+(\mathsf{Loc})$ is a transition function. The NFAs can be seen as a subclass of the AFAs: the transition relation δ of an NFA can be translated into the transition function ρ of AFA such that $\rho(\ell, \sigma) = \ell_1 \vee \ldots \vee \ell_n$ for $\{\ell_1, \ldots, \ell_n\} = \{\ell' \in \mathsf{Loc} \mid (\ell, \sigma, \ell') \in \delta\}$. A *run* of the AFA A over a finite word w is a tree $T = (N, \Rightarrow)$, whose nodes are a prefix-closed set $N \subseteq \mathsf{Loc}^+$ of nonempty sequences of states. The level of a node $x = \ell_1 \ldots \ell_n$ in N is its size $|x| = n$, and the last element of x is $\mathsf{last}(x) = \ell_n$. The set N contains a single node at level 1, the root, which is a state in Init. We require that for all $x \in N$, we have $|x| \leq |w| + 1$. The child relation $\Rightarrow \subseteq N \times N$ satisfies the following condition: for all nodes $x \in N$, we have (1) if $x \Rightarrow x'$, then $x' = x \cdot \ell$ for some $\ell \in \mathsf{Loc}$, and (2) if $|x| \leq |w|$, then the set $s = \{\mathsf{last}(x') \mid x \Rightarrow x'\}$ is such that $s \models \rho(\mathsf{last}(x), w(|x|))$. A leaf of T is a node x of level $|x| = |w| + 1$. A run T is *accepting* iff $\mathsf{last}(x) \in \mathsf{Fin}$ for all leaves x of T. The *language* $\mathsf{Lang}(A)$ accepted by A is the set of words $w \in \Sigma^*$ such that A has an accepting run over w.

The *emptiness problem* for AFAs is to decide, given an AFA A, whether $\mathsf{Lang}(A) = \emptyset$. Since complementation of AFAs is easy (by dualizing the transition function and complementing the set of accepting states), the universality problem for AFAs (to decide, given an AFA A, if $\mathsf{Lang}(A) = \Sigma^*$) is polynomially equivalent to emptiness. Given an AFA $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \rho \rangle$, consider the following monotone function on the lattice L of antichains over Loc: for an antichain $q \in L$, let

$$\mathsf{CPre}_a(q) = \lceil \{s \mid \exists s' \in q \cdot \exists \sigma \in \Sigma \cdot \forall \ell \in s : s' \models \rho(\ell, \sigma)\} \rceil.$$

This monotone function on L can be used to decide the emptiness problem for AFAs, as shown in the following theorem.

Theorem 8 *Let* $A = \langle \mathsf{Loc}, \mathsf{Init}, \mathsf{Fin}, \Sigma, \delta \rangle$ *be an AFA, and let* $\mathcal{F}_a = \bigsqcap \{ q \mid q = \mathsf{CPre}_a(q) \sqcup \{\mathsf{Fin}\} \}$. *Then* $\mathsf{Lang}(A) \neq \emptyset$ *iff* $\{\mathsf{Init}\} \sqsubseteq \mathcal{F}_a$.

6 Conclusions

We showed that explicit determinization can be avoided when solving several problems related to NFAs on finite words. Our new solutions to the universality and language-inclusion problems for NFAs, and to the emptiness problem for AFAs, evaluate the least fixed point of simple monotone functions on lattices of antichains. They are goal-directed and leave determinization implicit. We implemented the new algorithm for the universality problem and compared its performance to that of the classical algorithm (which uses explicit determinization). Our method outperforms the classical one dramatically on the entire parameter space of a randomized model. On the difficult instances of the randomized model, our algorithm is several orders of magnitude faster than the classical one.

We plan to pursue several future directions. First, as the performance of the new algorithm on the randomized model is very encouraging, we want to apply antichain algorithms to practical problems. Second, the antichain method does not extend trivially to automata over infinite words. We need further research to see if our results can be extended to such cases.

Acknowledgements We thank Deian Tabakov for his code and helpful answers about the randomized model.

References

[BL80] J.A. Brzozowski and E.L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[CCGR99] A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model verifier. In *Computer Aided Verification*, LNCS 1633, pages 495–499. Springer, 1999.

[CKS81] A.K. Chandra, D. Kozen, and L.J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[DDR06] M. De Wulf, L. Doyen, and J.-F. Raskin. A lattice theory for solving games of imperfect information. In *Hybrid Systems—Computation and Control*, LNCS 3927, pages 153–168. Springer, 2006.

[HMU01] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[KV01] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Computational Logic*, 2:408–429, 2001.

[Mø04] A. Møller. dk.brics.automaton. http://www.brics.dk/automaton, 2004.

[MS72] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Symp. Foundations of Computer Science*, pages 125–129. IEEE Computer Society, 1972.

[Rei84] J.H. Reif. The complexity of two-player games of incomplete information. *J. Computer and System Sciences*, 29:274–301, 1984.

[Som98] F. Somenzi. CUDD: CU Decision Diagram Package Release 2.3.0. University of Colorado at Boulder, 1998.

[TV05] D. Tabakov and M.Y. Vardi. Experimental evaluation of classical automata constructions. In *Logic for Programming, Artificial Intelligence, and Reasoning*, LNCS 3835, pages 396–411. Springer, 2005.