

Image-Binary Automata

Stefan Kiefer and Cas Widdershoven

Department of Computer Science, University of Oxford

Abstract. We introduce a certain restriction of weighted automata over the rationals, called *image-binary automata*. We show that such automata accept the regular languages, can be exponentially more succinct than corresponding NFAs, and allow for polynomial complementation, union, and intersection. This compares favourably with unambiguous automata whose complementation requires a superpolynomial state blowup. We also study an infinite-word version, *image-binary Büchi automata*. We show that such automata are amenable to probabilistic model checking, similarly to unambiguous Büchi automata. We provide algorithms to translate k -ambiguous Büchi automata to image-binary Büchi automata, leading to model-checking algorithms with optimal computational complexity.

1 Introduction

A weighted automaton assigns weights to words; i.e., it defines mappings of the form $f : \Sigma^* \rightarrow D$, where D is some domain of weights. Weighted automata are well-studied. Many variations have been discussed, such as max-plus automata [8] and probabilistic automata [25,24], both over finite words and over infinite words, in the latter case often combined with ω -valuation monoids [9]. However, it has been shown that many natural questions are undecidable for many kinds of weighted automata [10,1], including inclusion and equivalence. These problems become decidable for finitely ambiguous weighted automata [11].

In this paper we consider only numerical weights, where D is a subfield of the reals. A language $L \subseteq \Sigma^*$ can be identified with its characteristic function $\chi_L : \Sigma^* \rightarrow \{0,1\}$. We explore weighted automata that encode characteristic functions of languages, i.e., weighted automata that map each word either to 0 or to 1. We call such automata *image-binary finite automata (IFAs)* and view them as acceptors of languages $L \subseteq \Sigma^*$. We do not require, however, that individual transitions have weight 0 or 1. This makes IFAs a “semantic” class: it may not be obvious from the transition weights whether a given weighted automaton over, say, the rationals is image-binary. However, we will see that it can be checked efficiently whether a given \mathbb{Q} -weighted automaton is an IFA (Theorem 12).

An immediate question is on the expressive power of IFAs. Deterministic finite automata (DFAs) can be viewed as IFAs. On the other hand, in Section 2.2 we show that all languages accepted by IFAs are regular. It follows that IFAs accept exactly the regular languages. Moreover, IFAs are efficiently closed under

Boolean operations; i.e., given two IFAs that accept L_1, L_2 , respectively, one can compute in polynomial time IFAs accepting $L_1 \cup L_2$, $L_1 \cap L_2$, and $\Sigma^* \setminus L_1$.

The latter feature, efficient closure under complement, might be viewed as a key advantage of IFAs over unambiguous finite automata (UFAs). UFAs are nondeterministic finite automata (NFAs) such that every word has either zero or one accepting runs. UFAs can be viewed as a special case of IFAs. Whereas we show that IFAs can be complemented in polynomial time, UFAs are known to be not polynomially closed under complement [26].

The next question is then on the succinctness of IFAs, and on the complexity of converting other types of finite automata to IFAs and vice versa. We study such questions in Section 2.4. In Section 2.5, we also study the relationship of IFAs to mod-2 multiplicity automata, which are weighted automata over $GF(2)$, the field $\{0, 1\}$ where $1 + 1 = 0$. Such automata [2] share various features with IFAs, in particular efficient closure under complement.

In the second part of the paper we put IFAs “to work”. Specifically, we consider an infinite-word version, which we call *image-binary Büchi automata (IBAs)*. Following the theme that image-binary automata naturally generalise and relax unambiguous automata, we show that IBAs can be used for model checking Markov chains in essentially the same way as *unambiguous Büchi automata (UBAs)* [3]. Specifically, we show in Section 4 that given an IBA and a Markov chain, one can compute in NC (hence in polynomial time) the probability that a random word produced by the Markov chain is accepted by the IBA.

It was shown in [22] that a nondeterministic Büchi automaton (NBA) with n states can be converted to an NBA with at most 3^n states whose ambiguity is bounded by n . Known conversions from NBAs to UBAs have a state blowup of roughly n^n , see, e.g., [17]. We show in Section 3.2 that NBAs with logarithmic ambiguity (as produced by the construction from [22]) can be converted to IBAs in polylogarithmic space. This suggests that in order to translate NBAs into an automaton model suitable for probabilistic model checking (such as IBAs), it is reasonable to first employ the partial disambiguation procedure from [22] (which does most of the work). More specifically, by combining the partial disambiguation procedure from [22] with our translation to an IBA, we obtain a PSPACE transducer (i.e., a Turing machine whose work tape is polynomially bounded) that translates an NBA into an IBA. For example, combining that with the mentioned probabilistic model checking procedure for IBAs we obtain an (optimal) PSPACE procedure for model checking Markov chains against NBA specifications.

2 Image-Binary Finite Automata

2.1 Definitions

Let \mathbb{F} be one of the fields \mathbb{Q} or \mathbb{R} (with ordinary addition and multiplication). An \mathbb{F} -weighted automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, \eta)$ consists of a set of states Q , a

finite alphabet Σ , a map $M : \Sigma \rightarrow \mathbb{F}^{Q \times Q}$, an initial (row) vector $\alpha \in \mathbb{F}^Q$, and a final (column) vector $\eta \in \mathbb{F}^Q$. Extend M to Σ^* by setting $M(a_1 \cdots a_k) \stackrel{\text{def}}{=} M(a_1) \cdots M(a_k)$. The *language* $L_{\mathcal{A}}$ of an automaton \mathcal{A} is the map $L_{\mathcal{A}} : \Sigma^* \rightarrow \mathbb{F}$ with $L_{\mathcal{A}}(w) = \alpha M(w) \eta$. Automata \mathcal{A}, \mathcal{B} over the same alphabet Σ are said to be *equivalent* if $L_{\mathcal{A}} = L_{\mathcal{B}}$.

Let $\mathcal{A} = (Q, \Sigma, M, \alpha, \eta)$ be a \mathbb{Q} -weighted automaton. We call \mathcal{A} an *image-binary (weighted) finite automaton (IFA)* if $L_{\mathcal{A}}(\Sigma^*) \subseteq \{0, 1\}$, i.e., $L_{\mathcal{A}}(w) \in \{0, 1\}$ holds for all $w \in \Sigma^*$. An \mathbb{R} -IFA is defined like an IFA, but with \mathbb{Q} replaced by \mathbb{R} . An (\mathbb{R}) -IFA \mathcal{A} defines a language $L(\mathcal{A}) := \{w \in \Sigma^* \mid L_{\mathcal{A}}(w) = 1\}$. Note that we call both $L_{\mathcal{A}}$ and $L(\mathcal{A})$ the language of \mathcal{A} ; strictly speaking, the former is the characteristic function of the latter.

If an IFA $\mathcal{A} = (Q, \Sigma, M, \alpha, \eta)$ is such that $\alpha \in \{0, 1\}^Q$ and $\eta \in \{0, 1\}^Q$ and $M(a) \in \{0, 1\}^{Q \times Q}$ for all $a \in \Sigma$, then \mathcal{A} is called an *unambiguous finite automaton (UFA)*. Note that this definition of a UFA is essentially equivalent to the classical one, which says that a UFA is an NFA (nondeterministic finite automaton) where each word has at most 1 accepting run. Similarly, a *deterministic finite automaton (DFA)* is essentially a special case of a UFA, and hence of an IFA.

Example 1. Figure 1 shows an IFA and a UFA in a graphical notation. Formally, the IFA on the left is $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, M_{\mathcal{A}}, \alpha_{\mathcal{A}}, \eta_{\mathcal{A}})$ with $Q = \{1, 2, 3\}$ and $\Sigma = \{a, b\}$ and

$$M_{\mathcal{A}}(a) = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad M_{\mathcal{A}}(b) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and $\alpha_{\mathcal{A}} = (1 \ 0 \ 0)$ and $\eta_{\mathcal{A}} = (0 \ 0 \ 1)^T$. Both automata recognise the language of words that start in an even (positive) number of as .

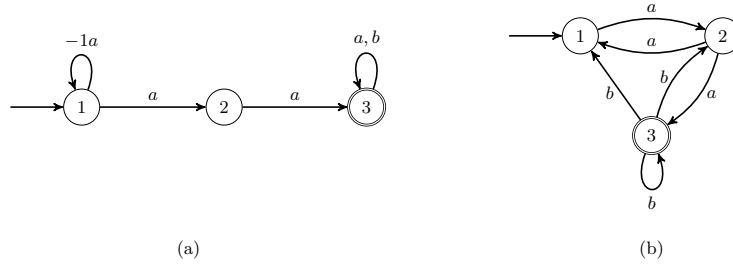


Fig. 1. The IFA in (a) is a forward conjugate of, and hence equivalent to, the UFA in (b). Unless indicated otherwise, edges in (a) have weight 1.

Let $\mathcal{A} = (Q, \Sigma, M, \alpha, \eta)$ be an \mathbb{R} -weighted automaton. We call $\vec{\mathcal{A}} := (\vec{Q}, \Sigma, \vec{M}, \vec{\alpha}, F\eta)$ a *forward conjugate* of \mathcal{A} with base F if $F \in \mathbb{R}^{\vec{Q} \times Q}$ and

$FM(a) = \vec{M}(a)F$ for all $a \in \Sigma$ and $\alpha = \vec{\alpha}F$. Such \mathcal{A} and $\vec{\mathcal{A}}$ are equivalent: indeed, let $w \in \Sigma^*$; by induction we have $FM(w) = \vec{M}(w)F$ and hence

$$L_{\mathcal{A}}(w) = \alpha M(w)\eta = \vec{\alpha} FM(w)\eta = \vec{\alpha} \vec{M}(w)F\eta = L_{\vec{\mathcal{A}}}(w).$$

A backward conjugate can be defined analogously.

Example 2. The IFA \mathcal{A} on the left of Figure 1 is a forward conjugate of the UFA on the right with base

$$F = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Indeed, we have $(1 \ 0 \ 0)F = (1 \ 0 \ 0)$ and $(0 \ 0 \ 1)^T = F(0 \ 0 \ 1)^T$, where \vec{v}^T denotes the transpose of a vector \vec{v} , and

$$F \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} F \quad \text{and} \quad F \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} F.$$

For some proofs we need the following definition. Let $L : \Sigma^* \rightarrow \mathbb{F}$, where \mathbb{F} is any field. Then the *Hankel matrix* of L is the infinite matrix $H_L \in \mathbb{F}^{\Sigma^* \times \Sigma^*}$ with $H_L[x, y] = L(xy)$. It was shown by Carlyle and Paz [6] and Fliess [12] that the rank of H_L is equal to the number of states of the minimal (in number of states) \mathbb{F} -weighted automaton \mathcal{A} with $L_{\mathcal{A}} = L$.

Proposition 3 ([6,12]). *Let L be an \mathbb{F} -weighted regular language, i.e. a function $L : \Sigma^* \rightarrow \mathbb{F}$ that can be represented by an \mathbb{F} -weighted automaton. Let $\mathcal{A} = (Q, \Sigma, M, \alpha, \eta)$ be a minimal \mathbb{F} -weighted automaton such that $L_{\mathcal{A}} = L$. Then $\text{rank } H_L = |Q|$.*

2.2 Regularity

Since a DFA is an IFA, for each regular language there is an IFA that defines it. Conversely, we show that the language of an IFA is regular:

Theorem 4. *Let $\mathcal{A} = (Q, \Sigma, M, \alpha, \eta)$ be an \mathbb{R} -IFA. Then $L(\mathcal{A})$ is regular, and there is a DFA \mathcal{B} with at most $2^{|Q|}$ states and $L(\mathcal{A}) = L(\mathcal{B})$.*

In order to do so, we will need some auxiliary lemmas. View $\mathbb{Z}_2 = \{0, 1\}$ as the field with two elements. In the proof of Theorem 4 we consider vector spaces over \mathbb{Z}_2 , i.e., where the scalars are from \mathbb{Z}_2 . In particular, we will argue with the vector space $\mathbb{Z}_2^{\mathbb{N}} \cong \mathbb{Z}_2^{\Sigma^*}$ over \mathbb{Z}_2 . We first show:

Lemma 5. *Let V be a set of n vectors. Consider the vector space $\langle V \rangle$ spanned by V over \mathbb{Z}_2 . Then $|\langle V \rangle| \leq 2^n$.*

Proof. Let $V = \{v_1, \dots, v_n\}$. Then $\langle V \rangle = \{\sum_{i=1}^n \lambda_i v_i \mid \lambda_i \in \mathbb{Z}_2\}$. □

Corollary 6. *Let V be a vector space over \mathbb{Z}_2 . For any $n \in \mathbb{N}$, if $\dim V \leq n$ then $|V| \leq 2^n$.*

The following two lemmas show that if an \mathbb{R} -weighted automaton is image-binary, then the rank over \mathbb{R} of its Hankel matrix H is at least the rank of H over \mathbb{Z}_2 .

Lemma 7. *Let $V \subseteq \{0, 1\}^{\mathbb{N}}$ be a set of vectors. If V is linearly dependent over \mathbb{R} then V is linearly dependent over \mathbb{Q} . Hence $\dim \langle V \rangle_{\mathbb{Q}} \leq \dim \langle V \rangle_{\mathbb{R}}$.*

Proof. Let V be linearly dependent (over \mathbb{R}), and let $n = \dim \langle V \rangle$. Then there are $v_0, v_1, \dots, v_n \in V$ such that $V' := \{v_1, \dots, v_n\}$ is linearly independent and $v_0 \notin V'$ but $v_0 \in \langle V' \rangle$. So there are unique $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ with $v_0 = \sum_{i=1}^n \lambda_i v_i$. It suffices to show that $\lambda_1, \dots, \lambda_n \in \mathbb{Q}$. Since V' is linearly independent, there exists $J \subseteq \mathbb{N}$ with $|J| = n$ such that $\{v_1[J], \dots, v_n[J]\}$ is linearly independent, where $v_i[J] \in \{0, 1\}^n$ is the restriction of v_i to the entries indexed by J . Hence the λ_i are the unique solution of a linear system of equations

$$v_0[j] = \sum_{i=1}^n \lambda_i v_i[j] \quad \text{for all } j \in J.$$

Linear systems of equations with rational coefficients and constants have rational solutions. Hence $\lambda_1, \dots, \lambda_n \in \mathbb{Q}$. \square

Lemma 8. *Let $V \subseteq \{0, 1\}^{\mathbb{N}}$ be a set of vectors. If V is linearly dependent over \mathbb{Q} then V is linearly dependent over \mathbb{Z}_2 . Hence $\dim \langle V \rangle_{\mathbb{Z}_2} \leq \dim \langle V \rangle_{\mathbb{Q}}$.*

Proof. Let $V = \{v_1, \dots, v_n\}$ and $\sum_{i=1}^n \lambda_i v_i = \vec{0}$, where $\lambda_i \in \mathbb{Q}$ are not all zero. By multiplying all λ_i with a common denominator, we can assume without loss of generality that $\lambda_i \in \mathbb{Z}$ where λ_i are not all zero. By dividing all λ_i with the largest power of 2 that divides all λ_i , we can assume without loss of generality that the λ_i are not all even. In the equation $\sum_{i=1}^n \lambda_i v_i = \vec{0}$, by regarding every λ_i and every entry of every v_i modulo 2, we have $\sum_{i=1}^n \lambda_i v_i = \vec{0}$ over \mathbb{Z}_2 , i.e., V is linearly dependent over \mathbb{Z}_2 . \square

Hence we can prove Theorem 4:

Proof (of Theorem 4). Write $n = |Q|$. Let H be the Hankel matrix of $L_{\mathcal{A}}$. We have $H \in \{0, 1\}^{\Sigma^* \times \Sigma^*}$. By proposition 3 we have $\text{rank } H \leq n$, where the rank is over \mathbb{R} . By lemmas 7 and 8 it follows that $\text{rank } H \leq n$, where the rank is over \mathbb{Z}_2 . By corollary 6 it follows that H has at most 2^n different rows, say $H[w_1, \cdot], \dots, H[w_\ell, \cdot]$ with $\ell \leq 2^n$. So every word is Myhill-Nerode equivalent to a word in $\{w_1, \dots, w_\ell\}$. Thus, there is an equivalent DFA with ℓ states.

Explicitly, the following DFA $\mathcal{B} = (Q', \Sigma, \delta, q_0, F)$ is equivalent to \mathcal{A} :

$$\begin{aligned} Q' &= \{H[w_1, \cdot], \dots, H[w_\ell, \cdot]\} \\ \delta(H[w_i, \cdot], a) &= H[w_i a, \cdot] \\ q_0 &= H[\varepsilon, \cdot] \\ F &= \{H[w_i, \cdot] \mid 1 \leq i \leq \ell, H[w_i, \varepsilon] = 1\}. \end{aligned} \quad \square$$

2.3 Boolean Operations and Checking Image-Binariness

IFAs are *polynomially closed* under all boolean operations, by which we mean:

Theorem 9. *Let $\mathcal{A}_1, \mathcal{A}_2$ be IFAs over Σ . One can compute in polynomial time IFAs $\mathcal{B}_-, \mathcal{B}_\cap, \mathcal{B}_\cup$ with $L(\mathcal{B}_-) = \Sigma^* \setminus L(\mathcal{A}_1)$ and $L(\mathcal{B}_\cap) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ and $L(\mathcal{B}_\cup) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.*

By De Morgan's laws it suffices to construct \mathcal{B}_- and \mathcal{B}_\cap . Since \mathcal{B}_- and \mathcal{B}_\cap need to satisfy only $L_{\mathcal{B}_-} = 1 + (-L_{\mathcal{A}_1})$ (where $1 : \Sigma^* \rightarrow \{1\}$ denotes the constant 1 function) and $L_{\mathcal{B}_\cap} = L_{\mathcal{A}_1} \cdot L_{\mathcal{A}_2}$, it suffices to know that \mathbb{Q} -weighted automata are polynomially closed under negation and pointwise addition and multiplication:

Hadamard product

Proposition 10 (see, e.g., [5, Chapter 1]). *Let $\mathcal{A}_1, \mathcal{A}_2$ be \mathbb{Q} -weighted automata. One can compute in polynomial time \mathbb{Q} -weighted automata $\mathcal{B}_-, \mathcal{B}_+, \mathcal{B}_\times$ with $L_{\mathcal{B}_-} = -L_{\mathcal{A}_1}$ and $L_{\mathcal{B}_+} = L_{\mathcal{A}_1} + L_{\mathcal{A}_2}$ and $L_{\mathcal{B}_\times} = L_{\mathcal{A}_1} \cdot L_{\mathcal{A}_2}$.*

Proof. For $i \in \{1, 2\}$ let $\mathcal{A}_i = (Q_i, \Sigma, M_i, \alpha_i, \eta_i)$. For \mathcal{B}_- , replace α_1 by $-\alpha_1$.

For \mathcal{B}_+ , assume $Q_1 \cap Q_2 = \emptyset$, take $Q = Q_1 \cup Q_2$, and $M(a) = \begin{pmatrix} M_1(a) & 0 \\ 0 & M_2(a) \end{pmatrix}$

for all $a \in \Sigma$, and $\alpha = (\alpha_1 \ \alpha_2)$, and $\eta = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}$. For \mathcal{B}_\times , take $Q = Q_1 \times Q_2$, and

$M(a) = M_1(a) \otimes M_2(a)$ for all $a \in \Sigma$, and $\alpha = \alpha_1 \otimes \alpha_2$, and $\eta = \eta_1 \otimes \eta_2$, where \otimes stands for the Kronecker product. It follows from the mixed-product property of \otimes (i.e., $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$) that indeed $L_{\mathcal{B}_\times} = L_{\mathcal{A}_1} \cdot L_{\mathcal{A}_2}$. \square

While DFAs are also polynomially closed under complement (switch accepting and non-accepting states), NFAs and UFAs are not. For NFAs, it was shown in [14] that the (worst-case) blowup in the number n of states is $\Theta(2^n)$. For UFAs, it was shown recently:

Proposition 11 ([26]). *For any $n \in \mathbb{N}$ there exists a unary (i.e., on an alphabet Σ with $|\Sigma| = 1$) UFA \mathcal{A}_n with n states such that any NFA for the complement language has at least $n^{(\log \log \log n)^{\Theta(1)}}$ states.*

This super-polynomial blowup (even for unary alphabet and even if the output automaton is allowed to be ambiguous) refuted a conjecture that it may be possible to complement UFAs with a polynomial blowup [7]. An upper bound (for general alphabets and requiring the output to be a UFA) of $O(2^{0.79n})$ was shown in [16]; see also [15] for an (unpublished) improvement.

The authors believe Proposition 11 shows the strength of Theorem 9: while UFAs cannot be complemented efficiently, the more general IFAs are polynomially closed under all boolean operations.

Proposition 10 can be used to show:

Theorem 12. *Given a \mathbb{Q} -weighted automaton, one can check in polynomial time if it is an IFA.*

Proof. Let \mathcal{A} be a \mathbb{Q} -weighted automaton. By Proposition 10 one can compute in polynomial time a \mathbb{Q} -weighted automaton \mathcal{B} with $L_{\mathcal{B}} = L_{\mathcal{A}} \cdot L_{\mathcal{A}}$ (pointwise multiplication). Then \mathcal{A} is an IFA if and only if \mathcal{A} and \mathcal{B} are equivalent. Equivalence of \mathbb{Q} -weighted automata can be checked in polynomial time, see [29,30]. \square

2.4 Succinctness

It is known that UFAs can be exponentially more succinct than DFAs: for each $n \in \mathbb{N}$ with $n \geq 3$ there is a UFA with n states such that the smallest equivalent DFA has 2^n states, see [21, Theorem 1]. Since UFAs are IFAs, Theorem 4 is optimal:

Corollary 13. *For converting IFAs to DFAs, a state blowup of 2^n is sufficient and necessary.*

It is also known from [21] that converting NFAs to UFAs can require $2^n - 1$ states. The argument carries over to IFAs:

Proposition 14. *For converting NFAs to IFAs, a state blowup of $\Theta(2^n)$ is sufficient and necessary.*

Proof. Sufficiency is clear via the subset construction. For necessity, Leung [21, Theorem 3] considers for each $n \geq 3$ an NFA (even an MDFA, which is a DFA with multiple initial states) such that its Hankel matrix has rank $2^n - 1$, which he proved in [20]. He then invokes an analogue of Proposition 3, due to Schmidt [28], to show that any equivalent UFA needs at least $2^n - 1$ states. But by Proposition 3 this holds also for IFAs. \square

It follows from Theorem 9 and Proposition 11 that IFAs cannot be converted to NFAs in polynomial time:

Proposition 15. *Converting IFAs to NFAs requires a super-polynomial state blowup.*

Proof. Let $n \in \mathbb{N}$, and let \mathcal{A}_n be the UFA from proposition 11. By theorem 9 there is a polynomial-size IFA, say \mathcal{B}_n , for the complement of $L(\mathcal{A}_n)$. If converting IFAs to NFAs required only a polynomial state blowup, there would exist an NFA, say \mathcal{B}'_n , with $L(\mathcal{B}'_n) = L(\mathcal{B}_n) = \Sigma^* \setminus L(\mathcal{A}_n)$, of size polynomial in \mathcal{A}_n , contradicting proposition 11. \square

2.5 Mod-2-Multiplicity Automata

We compare IFAs with the mod-2-multiplicity automata (mod-2-MAs) as introduced in [2], which are weighted automata over the field \mathbb{Z}_2 . Given a mod-2-MA \mathcal{A} and a word w , w is accepted iff $\mathcal{A}(w) = 1$. Like with IFAs, mod-2-MAs are exponentially more succinct than DFAs [2, Lemma 6]. Converting NFAs to mod-2-MAs requires a super-polynomial state blowup [2, Lemma 10] while (under the assumption that there are infinitely many Mersenne primes) converting mod-2-MAs to NFAs requires an exponential blowup [2, Lemma 11].

We can convert IFAs to mod-2-MAs without incurring a blowup:

Proposition 16. *For any IFA \mathcal{A} with n states there exists a mod-2-MA \mathcal{A}' of at most n states with $L_{\mathcal{A}} = L_{\mathcal{A}'}$.*

Proof. Let H be the Hankel matrix of $L_{\mathcal{A}}$. By Proposition 3, $\text{rank } H \leq n$, where the rank is taken over \mathbb{R} . Invoking Lemma 7 and Lemma 8 then shows that $\text{rank } H \leq n$ also when the rank is taken over \mathbb{Z}_2 . Then by Proposition 3 there exists a mod-2-MA with $\text{rank } H \leq n$ (over \mathbb{Z}_2) states that accepts the same language as \mathcal{A} . \square

However, the converse requires an exponential blowup. Inspired by Angluin et al.'s [2] proof that mod-2 automata can be exponentially more succinct than NFAs, this proof makes use of shift register sequences. However, note that this proof does not require the assumption that there are infinitely many Mersenne primes. For further information on shift register sequences, see [13]. A *shift register sequence* of dimension d is an infinite periodic sequence $\{a_n\}$ of bits defined by initial conditions $a_i = b_i$ for $i = 0, \dots, d-1$ and $b_i \in \{0, 1\}$, and a linear recurrence

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_d a_{n-d},$$

for all $n \geq d$, where each $c_i \in \{0, 1\}$ and addition is done modulo 2. The *minimum period* of a periodic sequence $\{a_n\}$ is the lowest $p \in \mathbb{N}$ such that $a_n = a_{n \bmod p}$ for every n . The maximum possible minimum period of a shift register sequence is $2^d - 1$, and it is known that for each positive integer d there are shift register sequences of maximum period. These are known as *maximal length* or *pseudo-noise* sequences [13].

Given $d > 0$, let $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_d a_{n-d}$ define a maximum period shift register sequence. Let L_d be the language over a unary alphabet $\{\#\}$ defined by $\#^n \in L_d$ if and only if $a_n = 1$. We have the following:

Proposition 17. *The language L_d is accepted by a mod-2-MA with d states, but not by any IFA with fewer than $2^d - 1$ states.*

Proof. The existence of a mod-2-MA with d states accepting L_d is shown in [2, Lemma 11].

Consider the Hankel matrix over \mathbb{R} of L_d . Since $\{a_n\}$ is $2^d - 1$ periodic, we have that the i 'th row of the Hankel matrix is equal to the $i + 2^d - 1$ 'st row for any i , and similar for the columns. Hence, the rank (over \mathbb{R}) of the Hankel matrix is equal to the rank of the submatrix of size $(2^d - 1) \times (2^d - 1)$ in the top left corner. We will call this matrix H . Notice that $H_{i,j} = a_{i+j}$. The *auto-correlation* of $\{a_n\}$ is defined as

$$C(\tau) = \sum_{k=1}^{2^d-1} a_k a_{k+\tau},$$

for $\tau \geq 0$. By Equation 10 on page 82 in [13], we have that $C(\tau) = 2^{d-1}$ if $\tau = 0$ and $C(\tau) = 2^{d-2}$ otherwise. Consider H^2 . If H^2 has full rank, then so does H .

We have that

$$\begin{aligned}
(H^2)_{i,j} &= \sum_{k=1}^{2^d-1} H_{i,k} H_{k,j} \\
&= \sum_{k=1}^{2^d-1} a_{i+k} a_{k+j} \\
&= \sum_{k=1}^{2^d-1} a_k a_{k+|j-i|} = C(|j-i|),
\end{aligned}$$

where the indices are taken modulo $2^d - 1$. Hence, H^2 is the matrix with 2^{d-1} on the diagonal and 2^{d-2} elsewhere. We show that the matrix with $2^{-d+2} - 2^{-2d+2}$ on the diagonal and -2^{-2d+2} elsewhere is an inverse of H^2 . Let $H' \in \mathbb{R}^{(2^d-1) \times (2^d-1)}$ as follows:

$$H'_{i,j} = \begin{cases} 2^{-d+2} - 2^{-2d+2} & \text{if } i = j \\ 2^{-2d+2} & \text{otherwise} \end{cases}$$

Now we have that $(H^2 H')_{i,j} = 1$ if $i = j$:

$$\begin{aligned}
(H^2 H')_{i,i} &= \sum_{k=1}^{2^d-1} H_{i,k}^2 H'_{k,i} \\
&= 2^{d-1}(2^{-d+2} - 2^{-2d+2}) + (2^d - 2)(2^{d-2}(-2^{-2d+2})) \\
&= 2^1 - 2^{-d+1} - 2^0 + 2^{-d+1} \\
&= 1
\end{aligned}$$

We also have $(H^2 H')_{i,j} = 0$ if $i \neq j$:

$$\begin{aligned}
(H^2 H')_{i,j} &= \sum_{k=1}^{2^d-1} H_{i,k}^2 H'_{k,j} \\
&= 2^{d-1}(-2^{-2d+2}) + 2^{d-2}(2^{-d+2} - 2^{-2d+2}) + (2^d - 3)(2^{d-2}(-2^{-2d+2})) \\
&= -2^{-d+1} + 2^0 - 2^{-d} - 2^0 + 3 * 2^{-d} \\
&= 0
\end{aligned}$$

Thus, H' is an inverse of H^2 . Hence, H^2 and H have full rank and thus the Hankel matrix of L_d has rank $2^d - 1$. This means that the smallest IFA accepting L_d has $2^d - 1$ states. \square

3 Image-binary Büchi Automata

3.1 Definitions

Let $\mathcal{A} = (Q, \Sigma, M, \alpha)$ be like in a weighted automaton over a field \mathbb{F} and let F be a set of final states. We call \mathcal{A} *ultimately stable* if for any $q, q' \in Q$ and $a \in \Sigma$

such that there exists a word w with $M(w)_{q',q} \neq 0$ (i.e., there is a path from q' to q over some word w), $M(a)_{q,q'} = 0$ or $M(a)_{q,q'} = 1$, meaning that any edges in a loop have weight 1. For any infinite word $w = w_0w_1 \dots$ we call $q_0q_1 \dots \in Q^\omega$ a *path* over w if $\alpha(q_0) \neq 0$ and for all i , $M(w_i)_{q_i,q_{i+1}} \neq 0$. We call $q_0q_1 \dots$ a *final path* if $\text{infinite}(q_0q_1 \dots) \cap F \neq \emptyset$, where $\text{infinite}(q_0q_1 \dots)$ denotes the set of states in Q that occur infinitely often in the path. We will write $\text{FinalPaths}_{\mathcal{A}}(w)$ to denote the set of final paths of an automaton \mathcal{A} over a word w .

It is clear that for any path $q_0q_1 \dots$ over a word $w = w_0w_1 \dots$ there exists an i such that for any $j \geq i$, q_j lies on a loop, and therefore we can define the weight of the path $q_0q_1 \dots$ over w to be $\lim_{i \rightarrow \infty} \prod_{n \leq i} M(w_n)_{q_n,q_{n+1}}$, denoted by $\text{weight}(q_0q_1 \dots, w)$. If w is clear from context, we may simply write $\text{weight}(q_0q_1 \dots)$. For any word w with finitely many final paths, we define the weight of w to be the sum of the weights of the final paths over w , denoted by $L_{\mathcal{A}}(w)$. We call $\mathcal{A} = (Q, \Sigma, M, \alpha, F)$ an *image-binary (weighted) Büchi automaton (IBA)* if it is ultimately stable, there exists a bound $N \in \mathbb{N}$ such that $|\text{FinalPaths}_{\mathcal{A}}(w)| \leq N$ for any word w , and $L_{\mathcal{A}}(\Sigma^\omega) \subseteq \{0, 1\}$, i.e. for all $w \in \Sigma^\omega$, $L_{\mathcal{A}}(w) \in \{0, 1\}$.

If an IBA \mathcal{A} is such that $\alpha \in \{0, 1\}^Q$ and $M(a) \in \{0, 1\}^{Q \times Q}$ for all $a \in \Sigma$, then \mathcal{A} is called an *unambiguous Büchi automaton (UBA)*. Similarly to the finite word case, we note that this definition of a UBA is essentially equivalent to the classical one, which says that a UBA is an NBA (nondeterministic Büchi automaton) where each word has at most 1 final path. We also see that a *deterministic Büchi automaton (DBA)* essentially is a special case of a UBA, and hence of an IBA.

We will use the following notation: given a finite sequence $a = a_1a_2 \dots a_n$, we will write $\text{last}(a)$ to denote a_n . Given a (possibly finite) sequence $a = a_1a_2 \dots$ and a character a_0 we will write $a_0 \cdot a$ to denote the concatenation $a_0a_1a_2 \dots$. We will write \mathbb{B} to denote the two element set $\{\perp, \top\}$ and for any set S we will write $\mathcal{P}^{\leq k}(S)$ to denote the set $\{S' \subseteq S \mid |S'| \leq k\}$. For ease of notation we will treat \mathbb{B} as the set of true (\top) and false (\perp) predicates, on which the standard Boolean operations apply.

3.2 IBAs and k -Ambiguous NBAs

In this section we introduce k -ambiguous NBAs (k -ABAs) and show that they can be exponentially more concise than IBAs. We give a procedure to translate a k -ABA into an equivalent IBA using a PSPACE transducer.

A non-deterministic Büchi automaton (NBA) is a tuple $(Q, \Sigma, \delta, Q_0, F)$ where Q is a state set, Σ is an alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is a transition relation, $Q_0 \subseteq Q$ is a set of initial states, and F is a set of final states. For any infinite word $w = w_0w_1 \dots$ we call $q_0q_1 \dots \in Q^\omega$ a *path* over w if $q_0 \in Q_0$ and for all i , $q_{i+1} \in \delta(q_i, w_i)$. We call $q_0q_1 \dots$ a *final path* if $\text{infinite}(q_0q_1 \dots) \cap F \neq \emptyset$, where $\text{infinite}(q_0q_1 \dots)$ denotes the set of states in Q that occur infinitely often in the path. We will write $\text{FinalPaths}_{\mathcal{A}}(w)$ to denote the set of final paths of an automaton \mathcal{A} over a word w . The *language* of an NBA is the set of those words w such that $\text{FinalPaths}_{\mathcal{A}}(w) \neq \emptyset$. A k -ABA is an NBA such that for every word

w , $|FinalPaths_{\mathcal{A}}(w)| \leq k$. For the rest of the section, fix a k -ambiguous NBA $\mathcal{A}_k = (Q, \Sigma, \delta, Q_0, F)$.

We have that k -ABAs can be exponentially more succinct than equivalent IBAs:

Lemma 18. *Let \mathcal{A} be a k -ABA with n states. The minimal IBA accepting the same language as \mathcal{A} may require at least 2^n states, even if $k = n$.*

Proof. By Proposition 14, there exists a family of n -ambiguous NFAs with exponentially fewer states than the minimal IFAs accepting the same languages. Let \mathcal{A} be such an NFA on n states, and let $L \subseteq \Sigma^*$ be its language. Consider the language $(L)_{\$}$ given by $w \in (L)_{\$}$ if and only if w can be decomposed as $u\$v$, where $\$$ does not occur in Σ and $u \in L$. We can create an n -ABA that accepts this language with $n + 1$ states by adding a $\$$ -labeled arrow from the final states of \mathcal{A} to an accepting sink state. However, let \mathcal{A}' be any IBA accepting $(L)_{\$}$. Let $\$w$ be any infinite word starting in $\$$, and let $\eta_q = L_{\mathcal{A}'[q]}(\$w)$. Let \mathcal{A}'' be IFA defined as follows: the state set of \mathcal{A}'' is equal to the state set of \mathcal{A}' , the alphabet is the alphabet of \mathcal{A} (i.e. the alphabet of \mathcal{A}' without $\$$), there exists an a -edge between two states q, q' if and only if there exists an a -edge between q and q' in \mathcal{A}' , and the final vector is given by η . We claim that \mathcal{A}'' accepts L . Indeed, we have $L_{\mathcal{A}''}(u) = L_{\mathcal{A}'}(u\$w)$. Hence, \mathcal{A}' has at least 2^n states, because otherwise \mathcal{A}'' would be an IFA with fewer than 2^n states accepting L . \square

The rest of this section will be dedicated to converting k -ABAs to equivalent IBAs, resulting in an IBA with at most a singly exponential state set size blowup.

By the binomial theorem, $(1+x)^n = \sum_{i=0}^n \binom{n}{i} x^i$, and hence, setting $x = -1$, $1 = 1 - \sum_{i=0}^n (-1)^i \binom{n}{i} = 1 - (-1)^0 \binom{n}{0} - \sum_{i=1}^n (-1)^i \binom{n}{i} = \sum_{i=1}^n (-1)^{i-1} \binom{n}{i}$. Hence, for any set S , $\sum_{S' \in \mathcal{P}(S) \setminus \{\emptyset\}} (-1)^{|S'| - 1} = \sum_{i=1}^{|S|} (-1)^{i-1} \binom{|S|}{i} = 1$. Let R be the set of final paths of \mathcal{A}_k over a word w . We can design an (infinite state) IBA $\mathcal{A}'_k = (Q', \Sigma, \Delta', \alpha, F')$ where final paths correspond to subsets of R , and where for each $R' \subseteq R$, the final path corresponding to R' has weight $(-1)^{|R'| - 1}$. This IBA is given as follows:

- $Q' = \mathcal{P}^{\leq k}(Q^* \times \mathbb{B}) \setminus \{\emptyset\}$,
- $\alpha_P = (-1)^{|P| - 1}$ for each $P \in \mathcal{P}(Q_0 \times \{\perp\}) \setminus \{\emptyset\}$ and $\alpha_P = 0$ otherwise,
- $F' = \mathcal{P}(Q^* \times \{\top\})$, and
- for any $P \in Q'$, let $b'' = \perp$ if for all $(r, b) \in P$, $b = \top$, and let $b'' = \top$ otherwise. Let P' be such that:
 - For any $(r, b) \in P$, there exists $(r', b') \in P'$ and $q \in \delta(last(r), a)$ such that $r' = r \cdot q$ and $b' = ((last(r) \in F) \vee b) \wedge b''$, and
 - For any $(r', b') \in P'$, there exists $(r, b) \in P$ and $q \in \delta(last(r), a)$ such that $r' = r \cdot q$ and $b' = ((last(r) \in F) \vee b) \wedge b''$.

Then $\Delta(a)_{P, P'} = (-1)^{|P'| - |P|}$. For any other P' , $\Delta(a)_{P, P'} = 0$.

Intuitively, the bit b in a (r, b) -tuple flips to true every time r reaches a final state, and back to false every time all the prefixes have reached a final state.

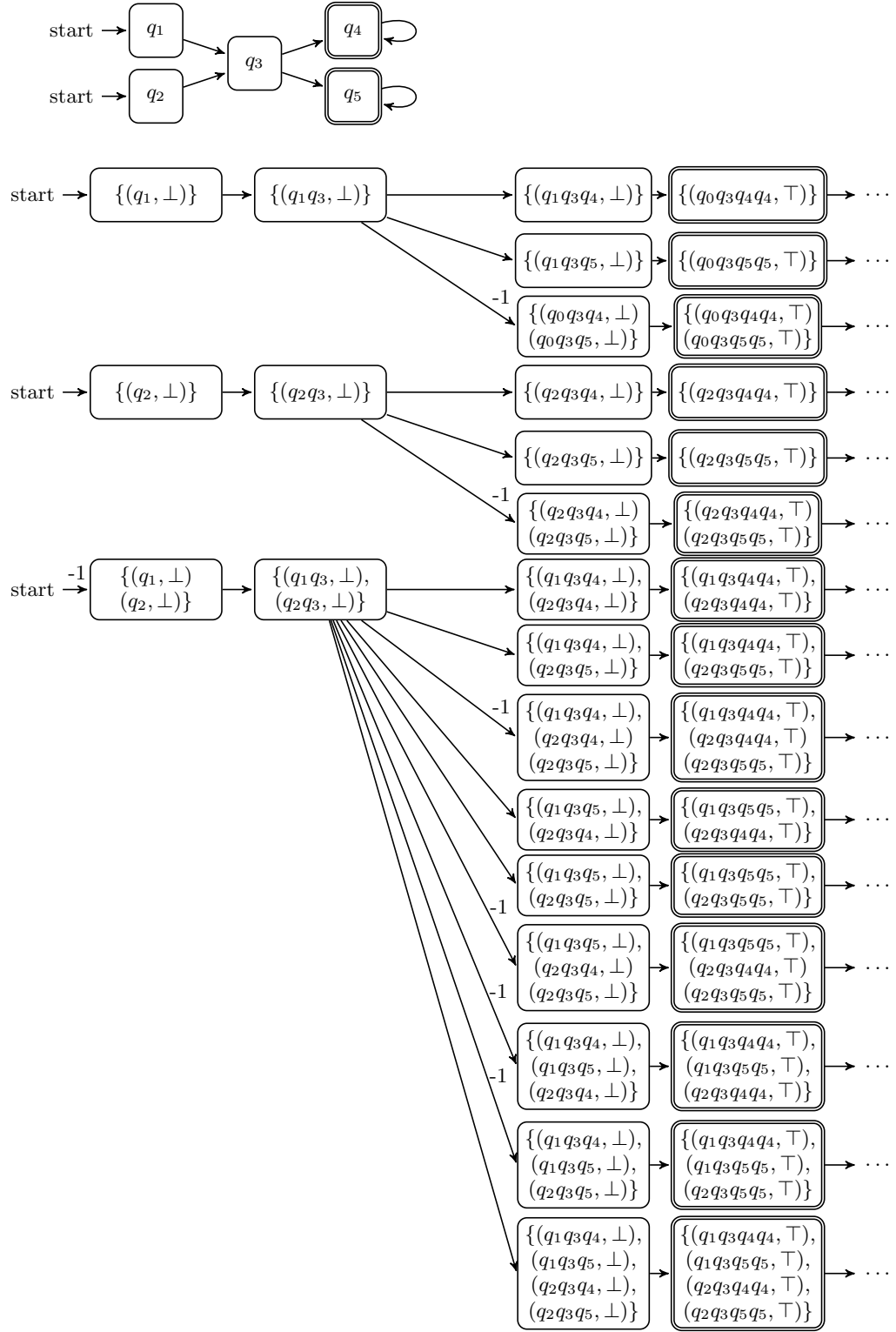


Fig. 2. An example 4-ambiguous automaton (above) and its infinite IBA counterpart (below). In the IBA, the weights of the unlabeled edges are 1.

This ensures that every sequence of prefixes in a final path of \mathcal{A}'_k visits final states infinitely often. This technique mirrors for instance Safra's construction [27].

In Figure 2 we give an example of a 4-ambiguous automaton over a unary alphabet, together with the corresponding infinite \mathcal{A}'_k .

Lemma 19. \mathcal{A}'_k is an infinite-state IBA equivalent to \mathcal{A}_k .

Proof. For \mathcal{A}'_k to be an IBA equivalent to \mathcal{A}_k , it needs to satisfy three properties - the edges of \mathcal{A}'_k over loops have weight 1, any word w has at most N final paths for some global bound N , and the sum of the weights of final paths over w is 1 if w is accepted by \mathcal{A}_k and 0 otherwise. Since states in a path of \mathcal{A}'_k consist of sets of prefixes of increasing length, we see that \mathcal{A}'_k does not have any loops and hence the first property is trivially true. For the second and third properties, let w be any fixed word.

Suppose w is not accepted by \mathcal{A}_k , then by König's lemma there exists a bound m such that any path of \mathcal{A}_k over w has at most m occurrences of final states. Let $\rho_1\rho_2\dots\rho_i$ be any prefix of a path of \mathcal{A}'_k over w . Since for any $(r, b) \in \rho_i$, r has at most m occurrences of final states, and all the bits are flipped to false any time every prefix in $\rho_1\rho_2\dots\rho_i$ has visited a final state, we see that $\rho_1\rho_2\dots\rho_i$ also visits final states at most m times. Therefore, \mathcal{A}'_k has no final paths (and hence the sum of the weights is 0) and the second and third properties are true.

Suppose, then, that w is accepted by \mathcal{A}_k . Let $\rho_1\rho_2\dots$ be a path of \mathcal{A}'_k over w . Suppose that for some i there exists $(r, b) \in \rho_i$ such that r is not a prefix of a final path of \mathcal{A}_k over w . Let r be the shortest such prefix. Then for large enough m and any $m' > m$ there exists $(r', \perp) \in \rho_{m'}$ where r is a prefix of r' . Hence, $\rho_{m'} \notin F'$ and $\rho_1\rho_2\dots$ is not a final path.

Finally, let $\rho_1\rho_2\dots$ be a path of \mathcal{A}'_k over w such that for every i and every $(r, b) \in \rho_i$, r is a prefix of a final path of \mathcal{A}_k over w . Let R be the set of those paths ρ' of \mathcal{A}_k over w such that for all i there exists a tuple $(r, b) \in \rho_i$ where r is a prefix of ρ' . We have that for any i and any $(r, b) \in \rho_i$ there exists $(r', b') \in \rho_{i+1}$ such that $r' = r \cdot q$ with $q \in \delta(\text{last}(r), w_i)$ and for any $(r', b') \in \rho_{i+1}$ there exists $(r, b) \in \rho_i$ such that $r' = r \cdot q$ with $q \in \delta(\text{last}(r), w_i)$. Therefore R is nonempty and ρ_i is precisely the set of prefixes of length i of paths in R (paired with some boolean b). Hence, paths $\rho_1\rho_2\dots$ in \mathcal{A}'_k over w correspond uniquely to sets of paths of \mathcal{A}_k over w .

We claim that R is a set of final paths, and that the weight of $\rho_1\rho_2\dots$ is $(-1)^{|R|-1}$. Indeed, suppose $\rho'_1\rho'_2\dots \in R$ is not a final path. Since for any i , $\rho'_1\rho'_2\dots\rho'_i$ prefixes a final path, there exists $j > i$ such that $\rho'_1\rho'_2\dots\rho'_i$ is a prefix of a final path $\rho''_1\rho''_2\dots$, and $\rho'_j \neq \rho''_j$. Hence \mathcal{A}_k has infinitely many final paths over w , which contradicts its k -ambiguity. Moreover, the weight of any prefix of length i of $\rho_1\rho_2\dots$ is $(-1)^{|\rho_1|-1} \prod_j (-1)^{|\rho_{j+1}|-|\rho_j|} = (-1)^{|\rho_i|-1}$. Since the size of ρ_i is non-decreasing and bounded from above by k , and for any i , ρ_i contains prefixes of length i of paths in R , for large enough i we have that $|\rho_i| = |R|$, and hence the weight of $\rho_1\rho_2\dots$ is $(-1)^{|R|-1}$.

Thus, final paths in \mathcal{A}'_k over w correspond uniquely to sets of final paths in \mathcal{A}_k over w , and hence there are at most 2^k final paths in \mathcal{A}'_k over w . Moreover,

since by the binomial theorem we have $\sum_{i=1}^{|S|} (-1)^{i-1} \binom{|S|}{i} = 1$ for any set S , and any path in \mathcal{A}'_k corresponding to a set R of final paths in \mathcal{A}_k has weight $(-1)^{|R|-1}$, we see that the sum of the weights of final paths of \mathcal{A}'_k over w is precisely equal to 1 if and only if \mathcal{A}_k has an accepting path over w . \square

We will construct a finite IBA called the k -disambiguation of \mathcal{A}_k based on \mathcal{A}'_k that accepts the same language as \mathcal{A}_k .

Let $\pi_{last} : Q' \rightarrow [k]^{Q \times \mathbb{B}}$ be defined as $\pi_{last}(P)_{q,b} = |\{(r,b) \in P \mid last(r) = q\}|$. We extend π_{last} over finite and infinite sequences of elements of Q' in the natural way: $\pi_{last}(P_1 P_2 \dots) = \pi_{last}(P_1) \cdot \pi_{last}(P_2 \dots)$.

Lemma 20. *Let $\rho = \rho_1 \rho_2 \dots \in (Q')^\omega$ be a path of \mathcal{A}'_k over a word w and let $\rho'_1 \rho'_2 \dots = \pi_{last}(\rho)$. Then $\rho_1 \rho_2 \dots$ is final if and only if for infinitely i , $(\rho'_i)_{q,\perp} = 0$ for all $q \in Q$.*

Proof. Trivial. \square

Paths in our k -disambiguation will be sequences in $([k]^{Q \times \mathbb{B}})^\omega$ such that there exist paths in \mathcal{A}'_k that map to that sequence. However, there is not a one-to-one correspondence between sequences over $[k]^{Q \times \mathbb{B}}$ and paths in \mathcal{A}'_k : in Figure 2, for instance, both $\{(q_1, \perp), (q_2, \perp)\}$ $\{(q_1 q_3, \perp), (q_2 q_3, \perp)\}$ $\{(q_1 q_3 q_4, \perp), (q_2 q_3 q_5, \perp)\}$ $\{(q_1 q_3 q_4 q_4, \top), (q_2 q_3 q_5 q_5, \top)\} \dots$ and $\{(q_1, \perp), (q_2, \perp)\}$ $\{(q_1 q_3, \perp), (q_2 q_3, \perp)\}$ $\{(q_1 q_3 q_5, \perp), (q_2 q_3 q_4, \perp)\}$ $\{(q_1 q_3 q_5 q_5, \top), (q_2 q_3 q_4 q_4, \top)\} \dots$ map to the sequence $(1, 1, 0, 0, 0, 0, 0, 0, 0, 0)^\top$ $(0, 0, 2, 0, 0, 0, 0, 0, 0, 0)^\top$ $((0, 0, 0, 1, 1, 0, 0, 0, 0, 0)^\top$ $(0, 0, 0, 0, 0, 0, 0, 0, 1, 1)^\top)^\omega$, where the order of the vector components is given by $((q_1, \perp), (q_2, \perp), (q_3, \perp), (q_4, \perp), (q_5, \perp), (q_1, \top), (q_2, \top), (q_3, \top), (q_4, \top), (q_5, \top))$.

Fix any $\vec{r}, \vec{r}' \in [k]^{Q \times \mathbb{B}}$ and $a \in \Sigma$. Let P be any state in Q' such that $\pi_{last}(P) = \vec{r}$. As it turns out, the number of states P' with $\pi_{last}(P') = \vec{r}'$ where P' is an a -successor of P only depends on \vec{r} , a , and \vec{r}' . We call this number $w(\vec{r}, a, \vec{r}')$.

Lemma 21. *The number $w(\vec{r}, a, \vec{r}')$ is unique and at most exponential in k .*

Proof. Given $\vec{r}, \vec{r}' \in [k]^{Q \times \mathbb{B}}$ and $a \in \Sigma$, let

$$C = \{f : Q \times \mathcal{B} \times [k] \rightarrow \mathcal{P}(Q) \mid \begin{aligned} &\forall (q, b, i) \in Q \times \mathcal{B} \times [k] : \\ &((i > \vec{r}_{q,b}) \rightarrow f(q, b, i) = \emptyset) \wedge \\ &((i \leq \vec{r}_{q,b}) \rightarrow f(q, b, i) \in \mathcal{P}(\delta(q, a)) \setminus \{\emptyset\}) \}. \end{aligned}$$

Intuitively, functions in C map the final states of prefixes in a state P with $\pi_{last}(P) = \vec{r}$ to nonempty sets of successor states. Let $b'' = \perp$ if $b = \top$ for all $(r, b) \in P$, and let $b'' = \top$ otherwise. Let C' be the set of those $f \in C$ such that for any $(q', b') \in Q \times \mathcal{B}$, $|\{(q, b, i) \in Q \times \mathcal{B} \times [k] \mid (b' = (q \in F \vee b) \wedge b'') \wedge q' \in f(q, b, i)\}| = \vec{r}'_{q', b'}$. We claim that $w(\vec{r}, a, \vec{r}') = |C'|$.

Fix an ordering on Q^* . Since $\pi_{last}(P) = \vec{r}$, we see that for any $q \in Q, b \in \mathcal{B}$ there are $\vec{r}_{q,b}$ tuples (s, b) in P where $last(s) = q$. Let $s_{q,b,i}$ denote the i 'th (under the ordering on Q^*) path in P that is paired with bit b and ends in state q . Then

for any $f \in C$, $f(q, b, i)$ is a nonempty set of a -successors of q . Let $b'' = \perp$ if $b' = \top$ for every $(s', b') \in P$ and let $b'' = \perp$ otherwise. If $q \in F$, let $b' = \top \wedge b''$, and otherwise let $b' = b \wedge b''$. Hence, for any $f \in C$ the following is an a -successor of P in \mathcal{A}'_k :

$$P' = \{(s_{q,b,i} \cdot q', b') \mid (s_{q,b,i}, b) \in P \wedge q' \in f(q, b, i)\}.$$

If $f \in C'$, then by definition of C' , for any q', b' , $\pi_{last}(P')_{q', b'} = \vec{r}_{q', b'}$, and hence $\pi_{last}(P') = \vec{r}'$. Since each $f \in C$ gives rise to a unique successor, this means P has $|C'|$ a -successors P' with $\pi_{last}(P') = \vec{r}'$. Moreover, $|C'| \leq |C| \leq 2^{2k|Q|^2}$. \square

For a state $\vec{r} \in Q''$ we will write $size(\vec{r}) = \sum_{(q,b)} \vec{r}_{q,b}$ to denote the size of \vec{r} . We define the IBA $k\text{-dis}'(\mathcal{A}_k) = (Q'', \Sigma, \Delta'', \alpha', F'')$ where:

- $Q'' = ([k]^{Q \times \mathbb{B}}) \setminus \{\vec{0}\}$,
- $\Delta''(a)_{\vec{r}, \vec{r}'} = (-1)^{size(\vec{r}') - size(\vec{r})} w(\vec{r}, a, \vec{r}')$
- $\alpha'_{\vec{r}} = (-1)^{size(\vec{r})-1}$ for every \vec{r} with:
 - $\vec{r}_{q,b} = 0$ if $q \notin Q_0$ or $b = \top$, and
 - $\vec{r}_{q,b} \leq 1$ otherwise.
- $\alpha'_{\vec{r}} = 0$ otherwise.
- $F'' = \{\vec{r} \in Q'' \mid \forall (q, \perp) \in Q'' : \vec{r}_{q,\perp} = 0\}$.

The IBA $k\text{-dis}(\mathcal{A}_k)$ (the k -disambiguation of \mathcal{A}_k) is then defined as $k\text{-dis}'(\mathcal{A}_k)$ restricted to those reachable states in Q'' that can reach a loop over a final state. This trimness condition helps the proofs later on, but could be omitted. In Figure 3 we see the k -disambiguation of the automaton in Figure 2.

The weights in $k\text{-dis}(\mathcal{A}_k)$ count the number of equivalent paths in \mathcal{A}'_k , where two paths $\rho, \rho' \in (Q')^\omega$ are equivalent if $\pi_{last}(\rho) = \pi_{last}(\rho')$:

Lemma 22. *Let $\rho \in (Q'')^\omega$ be a path in $k\text{-dis}(\mathcal{A}_k)$ over a word w . Let R be the set of those paths ρ' in \mathcal{A}'_k over w such that $\pi_{last}(\rho') = \rho$, and let $n = \max_i size(\rho_i)$. Then $weight(\rho) = (-1)^{n-1} |R|$.*

Proof. Let R_i be the set of prefixes of length i of paths in R . Let ρ'_i be any path in R_i . By Lemma 21, ρ'_i has $w(\rho_i, w_i, \rho_{i+1})$ w_i -successors that map to ρ_{i+1} under π_{last} . Hence, $|R_{i+1}| = w(\rho_i, w_i, \rho_{i+1}) |R_i|$ and in particular, $|R| = \lim_{n \rightarrow \infty} \prod_{i=1}^n w(\rho_i, w_i, \rho_{i+1})$. Moreover, $\lim_{n \rightarrow \infty} (-1)^{size(\rho_1)-1} \prod_{i=1}^n (-1)^{size(\rho_{i+1})-size(\rho_i)} = (-1)^{n-1}$, and hence $weight(\rho) = \lim_{n \rightarrow \infty} (-1)^{size(\rho_1)-1} \prod_{i=1}^n (-1)^{size(\rho_{i+1})-size(\rho_i)} w(\rho_i, w_i, \rho_{i+1}) = (-1)^{n-1} |R|$. \square

Since by Lemma 20, a path ρ in $k\text{-dis}(\mathcal{A}_k)$ is final whenever any path ρ' in \mathcal{A}'_k with $\pi_{last}(\rho') = \rho$ is, this means $L_{k\text{-dis}(\mathcal{A}_k)} = L_{\mathcal{A}'_k}$. This proves the final result:

Theorem 23. *$k\text{-dis}(\mathcal{A}_k)$ is an IBA that accepts the same language as \mathcal{A}_k .*

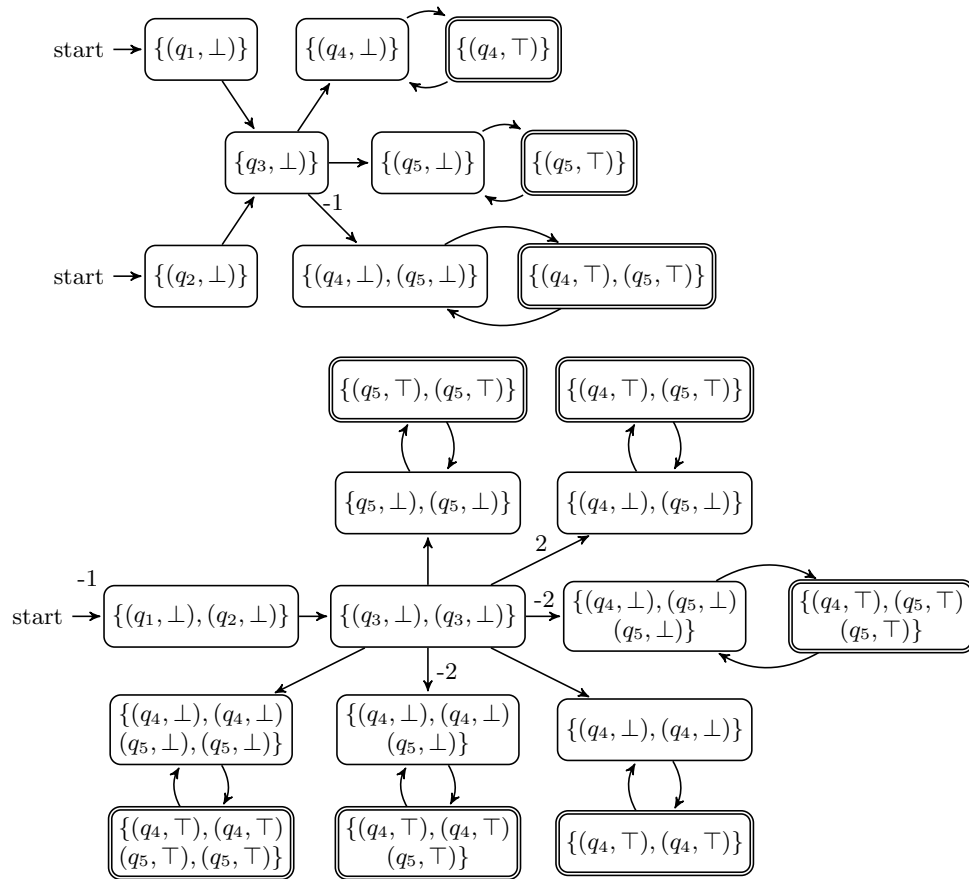


Fig. 3. The k -disambiguation of the automaton in Figure 2. The elements of $[k]^{Q \times \mathbb{B}}$ are represented by multisets. Unlabeled edges have weight 1.

Proof. We will prove equivalence between \mathcal{A}'_k and $k\text{-dis}(\mathcal{A}_k)$. Combined with Lemma 19, this gives us the required result. As before, we need to show that the edges of $k\text{-dis}(\mathcal{A}_k)$ over loops have weight 1, any word w has at most N final paths for some global bound N , and the sum of the weights of final paths over w is 1 if w is accepted by \mathcal{A}'_k and 0 otherwise.

Pick any loop in $k\text{-dis}(\mathcal{A}_k)$ and any a -edge from any \vec{r} to any \vec{r}' . Note that $w(\vec{r}, a, \vec{r}') > 0$ implies that $\text{size}(\vec{r}) \leq \text{size}(\vec{r}')$ and hence for any \vec{r}, \vec{r}' on that loop, $\text{size}(\vec{r}) = \text{size}(\vec{r}')$. Let $w_1 \dots w_m$ be such that there exists a path from an initial state to \vec{r} , and let $w_{m+1} \dots w_n$ be a word that traverses the loop starting with this edge from \vec{r} to \vec{r}' (hence, $w_{m+1} = a$). Finally let $w_{n+1} w_{n+2} \dots$ be a word such that $w_1 \dots w_m (w_{m+1} \dots w_n)^C w_{n+1} \dots$ has a final path that traverses the loop over \vec{r} at least C times, such a word exists since every state can reach a loop over a final state. Note that weights in $k\text{-dis}(\mathcal{A}_k)$ are integers. Then the absolute weight of this path is at least $w(\vec{r}, w_{m+1}, \vec{r}')^C$. Using Lemmas 22 and 20, then, we see that there exist at least $w(\vec{r}, w_{m+1}, \vec{r}')^C$ final paths over $w_1 \dots w_m (w_{m+1} \dots w_n)^C w_{n+1} \dots$. Since this is the case for any C , and the number of final paths over any word in \mathcal{A}'_k is bounded above, we must have that $w(\vec{r}, w_{m+1}, \vec{r}') = 1$ and hence $\Delta(a)_{\vec{r}, \vec{r}'} = 1$.

Now let w be any word. By Lemma 20 a path in \mathcal{A}'_k is final if and only if the path in $k\text{-dis}(\mathcal{A}_k)$ it maps to under π_{last} is final, and by Lemma 22 any path with nonzero weight (i.e. any path) has at least one path in \mathcal{A}'_k mapping to it. Hence we see that $k\text{-dis}(\mathcal{A}_k)$ has at most as many final paths over w as \mathcal{A}'_k does, and since the number of final paths in \mathcal{A}'_k is globally bounded, so is the number of final paths in $k\text{-dis}(\mathcal{A}_k)$.

Let w be any word and define the equivalence relation $=_\pi \subseteq Q'^\omega \times Q'^\omega$ as $\rho =_\pi \rho'$ iff $\pi_{last}(\rho) = \pi_{last}(\rho')$ (note that $\pi_{last}(\rho) \in ([k]^{Q \times \mathbb{B}})^\omega$). Note that this equivalence relation partitions the final paths of \mathcal{A}'_k over w , and for any ρ, ρ' with $\rho =_\pi \rho'$, $\text{weight}(\rho) = \text{weight}(\rho') = (-1)^{\lim_{i \rightarrow \infty} |\rho_i|}$. Each equivalence class can be represented by a path in $k\text{-dis}(\mathcal{A}_k)$. Let ρ' be the representative of the equivalence class containing ρ , and let R be the size of this equivalence class. By Lemma 22, $\text{weight}(\rho') = \text{weight}(\rho)R$. Hence, the sum of the weights of final paths in $k\text{-dis}(\mathcal{A}_k)$ is equal to the sum of the weights of all the equivalence classes of $=_\pi$ where the representative is a final path, which by Lemma 20 is precisely the sum of the weights of final paths in \mathcal{A}'_k .

Hence, the weight of a word w in $k\text{-dis}(\mathcal{A}_k)$ is equal to the weight of w in \mathcal{A}'_k . This means $k\text{-dis}(\mathcal{A}_k)$ is equivalent to \mathcal{A}'_k which is in turn equivalent to \mathcal{A}_k , concluding the proof. \square

Theorem 24. *Given a k -ambiguous automaton \mathcal{A}_k with n states, the disambiguation $k\text{-dis}(\mathcal{A}_k)$ has at most k^{2n} states. Moreover, $k\text{-dis}(\mathcal{A}_k)$ can be calculated using a PSPACE transducer.*

Proof. The number of states follows from the size of $[k]^{Q \times \mathbb{B}}$. Note that k is at most singly exponential in the size of the state set of the automaton, which follows from for instance [31, Theorem 2.1], or the fact that finite ambiguity implies the nonexistence of diamonds on a loop. This means that k^{2n} is also singly

exponential in n . Hence, we can iterate over every element in Q'' , each of which is a vector in $[k]^{Q \times \mathcal{B}}$ which can be represented in polynomial space. Given $\vec{r}, \vec{r}' \in Q''$ and $a \in \Sigma$, to calculate $\Delta''(a)_{\vec{r}, \vec{r}'}$ we need to calculate $(-1)^{\text{size}(\vec{r}') - \text{size}(\vec{r})}$ and $w(\vec{r}, a, \vec{r}')$. Clearly the former can be calculated in polynomial space. For the latter, notice that $w(\vec{r}, a, \vec{r}')$ can be doubly exponential, requiring an exponential representation. Hence, we cannot simply list the edges between states in \mathcal{A}'_k to calculate $w(\vec{r}, a, \vec{r}')$. However, we have the following lemma:

Lemma 25. *Let $w(\vec{r}, a, \vec{r}')$ as defined above, let $b'' = \perp$ if $\vec{r}_{q, \perp} = 0$ for every $q \in Q$ and $b'' = \top$ otherwise. For each $(q, b) \in Q \times \mathcal{B}$ let $C_{q, b}$ denote the set of those $\vec{r}'[q, b] \in [k]^{Q \times \mathcal{B}}$ such that*

$$\begin{aligned} \text{size}(\vec{r}'[q, b]) &\geq \vec{r}_{q, b} \quad \wedge \quad \forall (q', b') \in Q \times \mathcal{B} . \vec{r}'[q, b]_{q', b'} \leq \vec{r}_{q, b} \quad \wedge \\ \forall (q', b') \in Q \times \mathcal{B} . ((b' \neq (q \in F \vee b) \wedge b'') \vee q' \notin \delta(q, a)) &\rightarrow \vec{r}'[q, b]_{q', b'} = 0 . \end{aligned}$$

Let $\#succ(q, n, \vec{r}'[q, b]) = \sum_{j=0}^n (-1)^j \binom{n}{j} \left(\prod_{q', b'} \binom{n-j}{\vec{r}'[q, b]_{q', b'}} \right)$. Then

$$w(\vec{r}, a, \vec{r}') = \sum_{f: Q \times \mathcal{B} \rightarrow C_{q, b} \mid \sum_{q, b} f(q, b) = \vec{r}'} \prod_{q, b} \#succ(q, \vec{r}_{q, b}, f(q, b)).$$

Proof. We have that $w(\vec{r}, a, \vec{r}') = |C'|$, where C' is defined as in Lemma 21. We define the following equivalence relation: two elements $f, f' \in C'$ are equivalent if for every $(q, b) \in Q \times \mathcal{B}$ we have that $\sum_i [[f(q, b, i)]] = \sum_i [[f'(q, b, i)]]$, where for any set S , $[[S]]$ denotes the characteristic vector of S .

Let $\mathcal{G} := \{g : Q \times \mathcal{B} \rightarrow C_{q, b} \mid \sum_{q, b} g(q, b) = \vec{r}'\}$. Clearly the equivalence classes of C' can be characterised by elements of \mathcal{G} . We can view any function $f \in C'$ as a way of distributing non-empty successor sets over the numbered elements in \vec{r} in such a way that it adds up to \vec{r}' . Hence, for any $g \in \mathcal{G}$, the number of elements in the equivalence class of g is given by those functions that, for every $(q, b) \in Q \times \mathcal{B}$, distribute the elements in $g(q, b)$ to the $\vec{r}_{q, b}$ paths ending in (q, b) . These distributions are independent, and hence the number of elements in the equivalence class of g is the product over all (q, b) of the number of ways to distribute the elements in $g(q, b)$.

This number is captured by $\#succ(q, \vec{r}, g(q, b))$. We will explain $\#succ$ as a balls-and-bins problem. We need to assign a nonempty successor set to each of the $\vec{r}_{q, b}$ paths ending in (q, b) such that for each $(q', b') \in Q \times \mathcal{B}$ there are $g(q, b)_{q', b'}$ sets containing (q', b') . This is equivalent to dividing $\sum_{q', b'} g(q, b)_{q', b'}$ coloured balls, with colours in $Q \times \mathcal{B}$, over $\vec{r}_{q, b}$ bins in such a way that no bin has two balls of the same colour and no bin is empty. We can express the number of distributions where bins may be empty by distributing the balls per colour independently, this is given by $\prod_{q', b'} \binom{n}{g(q, b)_{q', b'}}$ where n is the number of bins (i.e. $\vec{r}_{q, b}$). Using the inclusion-exclusion principle, then, we can add the restriction that no bins may be empty by including/excluding sets of empty bins. This gives us the formula $\#succ(q, \vec{r}, g(q, b)) = \sum_{j=0}^n (-1)^j \binom{n}{j} \left(\prod_{q', b'} \binom{n-j}{\vec{r}_{q, b} - g(q, b)_{q', b'}} \right)$.

Hence, the number of elements in the equivalence class of g is given by $\prod_{q,b} \#succ(q, \vec{r}_{q,b}, g(q,b))$ and therefore we have that $w(\vec{r}, a, \vec{r}') = |C'| = \sum_{g \in \mathcal{G}} \prod_{q,b} \#succ(q, \vec{r}_{q,b}, g(q,b))$. \square

Using this lemma, we can enumerate every vector in $C_{q,b}$ for every (q,b) -pair, which is a polynomial combination of polynomially sized objects. We can then calculate $\#succ$ with a PSPACE transducer to find $w(\vec{r}, a, \vec{r}')$. Hence, we can construct $k\text{-dis}(\mathcal{A}_k)$ with a PSPACE transducer. \square

By Lemma 18, we have an $O(2^n)$ lower bound even if $k = n$. From Theorem 24 we already have a $2^{O(n \log n)}$ upper bound, leaving only a small gap.

When the ambiguity is comparatively low, we can do even better:

Theorem 26. *Given a k -ambiguous automaton \mathcal{A}_k with n states where $k = O(\log n)$, the disambiguation $k\text{-dis}(\mathcal{A}_k)$ has at most $(2n)^{O(\log n)}$ states. Moreover, $k\text{-dis}(\mathcal{A}_k)$ can be calculated using a POLYLOGSPACE transducer.*

Proof. While in general, elements in $[\log n]^{[n] \times \mathbb{B}}$ take $2n \log n$ space to represent, we know that for any $\vec{r} \in Q''$, $\sum_i \vec{r}_i \leq \log n$ and hence we can instead represent elements in Q'' as vectors in $([n] \times \mathbb{B})^{\log n}$, which take $\log(2n^{\log n}) = O((\log n)^2)$ space. Moreover, $w(\vec{r}, a, \vec{r}')$ is at most exponential in the ambiguity of \mathcal{A}_k and hence we can simply enumerate the successors of any $P \in Q'$ with $\pi_{last}(P) = \vec{r}$ in POLYLOGSPACE. Thus, we can calculate $k\text{-dis}(\mathcal{A}_k)$ in POLYLOGSPACE. \square

4 Model Checking IBA

In this section we will consider the problem of model checking IBAs against Markov chains. A *Markov chain* (MC) is a pair (S, P) where S is the finite state set, and $P \in [0, 1]^{S \times S}$ is a stochastic matrix specifying transition probabilities. Given an initial distribution ι , an MC \mathcal{M} induces a probability measure $\Pr_t^{\mathcal{M}}$ over infinite words. The model checking question asks, what is the probability of the language accepted by an IBA?

We show that model checking IBAs against MCs can be done in NC using a modified procedure for model checking UBAs from [3]. This is the main theorem:

Theorem 27. *Let \mathcal{M} be an MC and let \mathcal{A} be an IBA. The probability that a random word sampled from \mathcal{M} is in $L_{\mathcal{A}}$ can be computed in NC.*

In order to prove this, we will use the following properties of nonnegative matrices:

Theorem 28. *Let $M \in \mathbb{R}^{S \times S}$ be a nonnegative matrix. Then the following all hold:*

1. *The spectral radius $\rho(M)$ is an eigenvalue of A and there is a nonnegative eigenvector \vec{x} with $M\vec{x} = \rho(M)\vec{x}$. Such a vector \vec{x} is called dominant.*
2. *If $0 \leq M' \leq M$ then $\rho(M') \leq \rho(M)$.*

3. There is $C \subseteq S$ such that $M_{C,C}$ is strongly connected and $\rho(M_{C,C}) = \rho(M)$.

Theorem 29. Let $M \in \mathbb{R}^{S \times S}$ be a strongly connected nonnegative matrix. Then the following all hold:

1. There is an eigenvector \vec{x} with $M\vec{x} = \rho(M)\vec{x}$ such that \vec{x} is strictly positive in all its components.
2. The eigenspace associated with $\rho(M)$ is one-dimensional.
3. If $0 \leq M' < M$ (i.e. $M'_{i,j} < M_{i,j}$ for some $i, j \in S$) then $\rho(M') < \rho(M)$.
4. If $\vec{x} \geq 0$ and $M\vec{x} \leq \rho(M)\vec{x}$ then $M\vec{x} = \rho(M)\vec{x}$.

These results can all be found in [4, Chapter 2]. We will also need the following well-known fact about Markov chains.

Lemma 30. Let $\mathcal{M} = (S, P)$ be a Markov chain, and $\mathcal{L} \subseteq S^\omega$ an ω -regular language. Suppose $s_0 \in S$ such that $Pr_{s_0} > 0$. Then there exists $s_0 \dots s_n \in Paths_{s_0}(P)$ such that $Pr_{s_n}(\{w \in s_n S^\omega : s_0 \dots s_{n-1} w \in \mathcal{L}\}) = 1$.

This can be found for instance in [3, Lemma 16].

We will now outline the model checking procedure. As in [3] the algorithm will consist of solving a system of linear equations, split up in two parts. The first part is the basic linear system, which we will describe below, while the second part will add normalising equations in the form of cuts (or pseudo-cuts, if [18] is followed). Since the weights of the edges within SCCs are all on loops, and therefore all have weight 1, this part can be copied verbatim from Baier et al[3]. The difference is in the basic system of equations.

Given an automaton $\mathcal{A} = (Q, \Sigma, M, \alpha, F)$ and Markov chain $\mathcal{M} = (S, P)$, we will write $\mathcal{A}[q]$ (resp. $\mathcal{M}[s]$) to denote the weighted automaton (resp. Markov chain) starting in $q \in Q$ (resp. $s \in S$). W.l.o.g. we will assume that every $q \in Q$ is reachable from some q' with $\alpha(q') \neq 0$ and every $q \in Q$ can reach a loop over a final state. Note that while \mathcal{A} is an IBA, $\mathcal{A}[q]$ does not necessarily have to be. We will write $\mathbb{E}_{\mathcal{M}} L_{\mathcal{A}}$ for the expected weight of a word in \mathcal{A} for a random word generated by \mathcal{M} . Since the weight of a word in an IBA is either 0 or 1, we see that $\mathbb{E}_{\mathcal{M}} L_{\mathcal{A}} = Pr_{\mathcal{M}}(L_{\mathcal{A}})$ if \mathcal{A} is an IBA.

Lemma 31. Let $\mathcal{A} = (Q, \Sigma, M, \alpha, F)$ be an IBA w.r.t. Markov chain $\mathcal{M} = (S, P)$ with initial distribution ι . The following equations hold:

$$Pr_{\mathcal{M}}(L_{\mathcal{A}}) = \sum_{s \in S} \iota(s) \sum_{q \in Q} \alpha(q) \mathbb{E}_{\mathcal{M}[s]} L_{\mathcal{A}[q]} \quad (1)$$

$$\mathbb{E}_{\mathcal{M}[s]} L_{\mathcal{A}[q]} = \sum_{s' \in S} \sum_{q' \in Q} P_{s,s'} M(s)_{q,q'} \mathbb{E}_{\mathcal{M}[s']} L_{\mathcal{A}[q']} \quad (2)$$

Proof. The first equation holds because of the fact that \mathcal{A} is an IBA. Note that for any $q \in Q$, $|FinalPaths_{\mathcal{A}[q]}(w)| \leq N$ for all $w \in S^\omega$. For the second equation

we have the following:

$$\begin{aligned}
& \mathbb{E}_{\mathcal{M}[s]} L_{\mathcal{A}[q]} \\
&= \int_{w \in \text{Paths}(\mathcal{M}[s])} L_{\mathcal{A}[q]}(w) dPr(w) \\
&= \sum_{s' \in S} \int_{w \in \text{Paths}(\mathcal{M}[s]) \cap s' S^\omega} L_{\mathcal{A}[q]}(w) dPr(w) \quad \text{since the cylinder sets } sS^\omega \text{ partition } S^\omega \\
&= \sum_{s' \in S} P_{s,s'} \int_{w \in \text{Paths}(\mathcal{M}[s'])} L_{\mathcal{A}[q]}(sw) dPr(w) \quad \text{by definition of } Pr(w) \\
&= \sum_{s' \in S} P_{s,s'} \int_{w \in \text{Paths}(\mathcal{M}[s'])} \sum_{p \in \text{FinalPaths}_{\mathcal{A}[q]}(sw)} \text{weight}(p) dPr(w) \quad \text{definition of } L_{\mathcal{A}[q]} \\
&= \sum_{s' \in S} P_{s,s'} \int_{w \in \text{Paths}(\mathcal{M}[s'])} \sum_{q' \in Q} M(s)_{q,q'} \sum_{p \in \text{FinalPaths}_{\mathcal{A}[q']}(w)} \text{weight}(p) dPr(w) \\
&\quad \text{definition of } \text{FinalPaths} \\
&= \sum_{s' \in S} \sum_{q' \in Q} P_{s,s'} M(s)_{q,q'} \int_{w \in \text{Paths}(\mathcal{M}[s'])} \sum_{p \in \text{FinalPaths}_{\mathcal{A}[q']}(w)} \text{weight}(p) dPr(w) \\
&= \sum_{s' \in S} \sum_{q' \in Q} P_{s,s'} M(s)_{q,q'} \int_{w \in \text{Paths}(\mathcal{M}[s'])} L_{\mathcal{A}[q']}(w) dPr(w) \quad \text{by definition of } L \\
&= \sum_{s' \in S} \sum_{q' \in Q} P_{s,s'} M(s)_{q,q'} \mathbb{E}_{\mathcal{M}[s']} L_{\mathcal{A}[q']}
\end{aligned}$$

□

Let $\mathcal{M} = (S, P)$ be a Markov chain with state set S , and let ι be an initial distribution on S . Let $B \in \mathbb{R}^{(Q \times S) \times (Q \times S)}$ be the following matrix:

$$B_{\langle q,s \rangle, \langle q',s' \rangle} = P_{s,s'} M(s)_{q,q'}. \quad (3)$$

Define $\vec{z} \in \mathbb{R}^{Q \times S}$ by $\vec{z}_{q,s} = \mathbb{E}_{\mathcal{M}[s]} L_{\mathcal{A}[q]}$, i.e., the expected weight of a word for the Markov chain starting in s and the automaton starting in q . Similar to [3, Lemma 4] we have the following:

Lemma 32. *Let B and \vec{z} as defined above. We have that $\vec{z} = B\vec{z}$.*

Proof (of Lemma 32). Using Lemma 31:

$$\begin{aligned}
\vec{z}_{q,s} &= \mathbb{E}_{\mathcal{M}[s]} L_{\mathcal{A}[q]} \\
&= \sum_{s' \in S} \sum_{q' \in Q} P_{s,s'} M(s)_{q,q'} \mathbb{E}_{\mathcal{M}[s']} L_{\mathcal{A}[q']} \\
&= (B\vec{z})_{q,s}
\end{aligned}$$

□

W.l.o.g. we can assume that for every $\langle qs \rangle$ in B we can reach an accepting loop. We need to show that SCCs in B have a spectral radius of at most 1. In fact, we can give a probabilistic interpretation to values in $(B_{C,C})^n$, where $C \subseteq Q \times S$ is an SCC in B . This mirrors Proposition 6 in [3].

Lemma 33. Let $C \subseteq Q \times S$ and $\langle qs \rangle, \langle rt \rangle \in C$. Let $n \in \mathbb{N}$. Define $A := B_{C,C}$. Then $(A^n)_{\langle qs \rangle, \langle rt \rangle} = \mathbb{E}_{\mathcal{M}[s]} \text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w)$, where

$$\text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w) = \sum_{\langle q_0 s_0 \rangle \dots \langle q_n s_n \rangle \in \text{Paths}_{\langle qs \rangle, \langle rt \rangle}(A), w \in s_0 \dots s_n S^\omega} \prod_{0 \leq i < n} M(s_i)_{q_i, q_{i+1}}.$$

(Intuitively, $\text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w)$ is the weight of the prefixes of length n of paths over w in C starting in $\langle qs \rangle$ that reach $\langle rt \rangle$ at time n .)

Define

$$E_{\langle qs \rangle, \langle rt \rangle}^{C,n} := \{s_0 s_1 \dots \in s S^\omega \mid \exists q_1 \dots q_n. \langle q_0 s_0 \rangle \dots \langle q_n s_n \rangle \in \text{Paths}_{\langle qs \rangle, \langle rt \rangle}(A)\}.$$

In particular, if C is (a subset of) an SCC, then $(A^n)_{\langle qs \rangle, \langle rt \rangle} = \text{Pr}_{\mathcal{M}[s]}(E_{\langle qs \rangle, \langle rt \rangle}^{C,n})$ and hence $\rho(A) \leq 1$.

Proof.

$$\begin{aligned} & \mathbb{E}_{\mathcal{M}[s]} \text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w) \\ &= \int_{w \in \text{Paths}(\mathcal{M}[s_0])} \text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w) d \text{Pr}(w) \\ &= \sum_{s_0 \dots s_n \in S^n} \int_{w \in \text{Paths}(\mathcal{M}[s]) \cap s_0 \dots s_n S^\omega} \text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w) d \text{Pr}(w) \\ &= \sum_{s_0 \dots s_n \in S^n} \int_{w \in \text{Paths}(\mathcal{M}[s_n])} \prod_{0 \leq i < n} P_{s_i, s_{i+1}} \text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(s_0 \dots s_n w) d \text{Pr}(w) \\ &= \sum_{s_0 \dots s_n \in S^n} \int_{w \in \text{Paths}(\mathcal{M}[s_n])} \prod_{0 \leq i < n} P_{s_i, s_{i+1}} \\ & \quad \sum_{\langle q_0 s_0 \rangle \dots \langle q_n s_n \rangle \in \text{Paths}_{\langle qs \rangle, \langle rt \rangle}(A), w \in s_0 \dots s_n S^\omega} \prod_{0 \leq i < n} M(s_i)_{q_i, q_{i+1}} d \text{Pr}(w) \\ &= \sum_{s_0 \dots s_n \in S^n} \int_{w \in \text{Paths}(\mathcal{M}[s_n])} \\ & \quad \sum_{\langle q_0 s_0 \rangle \dots \langle q_n s_n \rangle \in \text{Paths}_{\langle qs \rangle, \langle rt \rangle}(A), w \in s_0 \dots s_n S^\omega} \prod_{0 \leq i < n} P_{s_i, s_{i+1}} M(s_i)_{q_i, q_{i+1}} d \text{Pr}(w) \\ &= \sum_{s_0 \dots s_n \in S^n} \sum_{\langle q_0 s_0 \rangle \dots \langle q_n s_n \rangle \in \text{Paths}_{\langle qs \rangle, \langle rt \rangle}(A)} \prod_{0 \leq i < n} P_{s_i, s_{i+1}} M(s_i)_{q_i, q_{i+1}} \\ &= \sum_{s_0 \dots s_n \in S^n} \sum_{\langle q_0 s_0 \rangle \dots \langle q_n s_n \rangle \in \text{Paths}_{\langle qs \rangle, \langle rt \rangle}(A)} \prod_{0 \leq i < n} A_{\langle q_i s_i \rangle, \langle q_{i+1} s_{i+1} \rangle} \\ &= (A^n)_{\langle qs \rangle, \langle rt \rangle} \end{aligned}$$

Note that since an accepting loop can be reached from every state in B , no SCC in B can have diamonds: otherwise, one could traverse a loop over that

diamond $\log N + 1$ times to get a word that has more than N final paths. Since any path that stays in an SCC has weight 1, and there are no diamonds in SCCs, we see that if C is (a subset of) an SCC, then $\mathbb{E}_{\mathcal{M}[s]} \text{weight}_{\langle qs \rangle, \langle rt \rangle}^{C,n}(w) = \text{Pr}_{\mathcal{M}[s]} \left(E_{\langle qs \rangle, \langle rt \rangle}^{C,n} \right)$ and hence $(A^n)_{\langle qs \rangle, \langle rt \rangle} = \text{Pr}_{\mathcal{M}[s]} \left(E_{\langle qs \rangle, \langle rt \rangle}^{C,n} \right)$. \square

The definitions of recurrent SCCs and cuts are completely equivalent to those in [3] and are copied here verbatim: a *recurrent SCC* is an SCC with spectral radius 1. We call a recurrent SCC D *accepting* if for some $\langle qt \rangle \in D$ we have $q \in F$. Let $D \subseteq Q \times S$ be an SCC of B . A set $\alpha \subseteq D$ is called a *fiber* if it can be written as $\alpha = A \times \{s\}$ for some $A \subseteq Q, s \in S$. Given such a fiber α and $t \in S$, if $M_{s,t} > 0$ we then define a fiber

$$\alpha \triangleright t := \{\langle q't \rangle \in D \mid q' \in \delta(q, s)\}$$

If $M_{s,t} = 0$ then $\alpha \triangleright t$ is left undefined. We extend this definition inductively by writing $\alpha \triangleright \epsilon = \alpha$ and $\alpha \triangleright wt = (\alpha \triangleright w) \triangleright t$ for $t \in S, w \in S^*$. If α is a singleton $\{d\}$ we may write $d \triangleright w$ for $\alpha \triangleright w$.

We call a fiber $\alpha \subseteq D$ a *cut* of D if (i) $\alpha = d \triangleright v$ for some $d \in D$ and $v \in S^*$, and (ii) $\alpha \triangleright w \neq \emptyset$ holds for all $w \in S^*$ such that $\alpha \triangleright w$ is defined. Clearly if α is a cut then so is $\alpha \triangleright w$ when the latter is defined. Given a cut $\alpha \subseteq D$, we call its characteristic vector $\vec{\mu} \in \{0, 1\}^D$ a *cut vector*.

Let D be a recurrent SCC. A vector $\vec{\mu} \in [0, 1]^D$ is called a *D-normaliser* if $\vec{\mu}^T \vec{z} = 1$.

We will need versions of [3, Lemma 8] and [3, Lemma 10] as well. Lemma 10.1 only relies on the probabilistic interpretation of submatrices in SCCs as shown in Lemma 33 and the nonnegativity of those matrices, and hence we can invoke its proof verbatim:

Lemma 34 (Lemma 10.1 in [3]). *Let $D \subseteq Q \times S$ be an SCC. Then D is recurrent if and only if it has a cut.*

Proof. Verbatim from the proof in [3]. \square

For our proof of [3, Lemma 8.1] we will first show something stronger:

Lemma 35. *Let D be a recurrent SCC, and let $C \subseteq Q \times S \setminus D$ be the set of states outside of D reachable from D . Then $\rho(B_{C,C}) < 1$.*

Proof. Towards a contradiction, assume $\rho(B_{C,C}) = 1$. Then by Theorem 28.3 there exists a recurrent SCC D' such that D' is reachable from D . We will create a word consisting of five parts such that \mathcal{A} has more than N final paths over that word. Since N is the global bound on the number of final paths in \mathcal{A} , this leads to a contradiction.

1. Firstly, pick $d = \langle qs_0 \rangle \in D$. Let $s_1 \dots s_i$ be such that $d \triangleright s_1 \dots s_i$ is a cut, such a word exists by Lemma 34.
2. Let $s_{i+1} \dots s_j$ be such that there exists a path from d to some $d' \in D'$ over $s_1 \dots s_j$.

3. Let $s_{j+1} \dots s_k$ be such that $d' \triangleright s_{j+1} \dots s_k$ is a cut. Again this exists by Lemma 34.
4. Let $s_{k+1} \dots s_l$ be such that $d \in d \triangleright s_1 \dots s_l$. This exists because $d \triangleright s_1 \dots s_i$ is a cut and for any cut c and any $s \in S$ we have that $c \triangleright s$ is also a cut.

Then consider the word $(s_1 \dots s_l)^{|D'|(N+1)}$. Because $d \in d \triangleright s_1 \dots s_l$ and we reach a cut in D' from d over $s_1 \dots s_l$, this means at least $|D'|(N+1)$ cuts in D' are reached after reading $(s_1 \dots s_l)^{|D'|(N+1)}$ from D . By the pigeonhole principle, there exists a $d' \in D'$ such that there are at least $N+1$ paths from d to d' over $(s_1 \dots s_l)^{|D'|(N+1)}$. Hence, if we pick $s_{l+1} \dots$ such that there is a final path from d' over $s_{l+1} \dots$, we see that there are more than N final paths from d over $(s_1 \dots s_l)^{|D'|(N+1)} s_{l+1} \dots$, contradicting the global bound on the number of final paths in \mathcal{A} . Thus, $\rho(D') < 1$ and therefore $\rho(B_{C,C}) < 1$. \square

Now we can show Lemma 8.1:

Lemma 36 (Lemma 8.1 in [3]). *Let D be a recurrent SCC, and let \vec{x} be any vector satisfying $\vec{x} = B\vec{x}$. Then $\vec{x}_D = B_{D,D}\vec{x}_D$.*

Proof. Let $\bar{D} = Q \times S \setminus D$ and let $C \subseteq \bar{D}$ be the set of states reachable from D . For all n we have:

$$\begin{aligned} \vec{x}_D &= (B^n \vec{x})_D \\ &= B^{n-1} B_{D,D} \vec{x}_D + B^{n-1} B_{D,\bar{D}} \vec{x}_{\bar{D}} \\ &= B^{n-1} B_{D,D} \vec{x}_D + B_{C,C}^{n-1} B_{D,\bar{D}} \vec{x}_{\bar{D}}. \end{aligned}$$

Since $\rho(B_{C,C}) < 1$ by Lemma 35, this second term goes to 0 as n goes to infinity. Hence, $\vec{x}_D = B_{D,D} \vec{x}_D$. \square

The proof of Lemma 8.2 in [3] relies on the following lemma:

Lemma 37. *Let D be a recurrent SCC in B , and let $\langle q, s \rangle \in D$. Then almost surely all final paths of $\mathcal{A}[q]$ are contained in D , and for any $\langle q', s' \rangle \notin D$ reachable from $\langle q, s \rangle$, $z_{\langle q', s' \rangle} = 0$.*

Proof. Suppose that with positive probability a word starting in s is generated such that $\mathcal{A}[q]$ has a final path outside D . We will construct a word consisting of three repeating parts that admits more than N final paths:

1. Let $s_0 \dots s_l \in S^*$ be such that $s_0 = s$ and $\langle q, s \rangle \triangleright s_1 \dots s_l$ is a cut containing $\langle q, s \rangle$. Such a word exists by Lemma 34.
2. The language of words that have a final path outside D is regular. The probability of generating a word with a final path outside D is bounded from above by the sum of probabilities of reaching some $\langle q', s' \rangle$ outside D times the probability of generating a word starting in s' over which there exists a final path from q' , and the probability of generating a word with a final path outside D is nonzero. Therefore, there must exist $\langle q', s' \rangle \notin D$ reachable from $\langle q, s \rangle$ such that with nonzero probability, a word starting in s' is generated

- over which $\mathcal{A}[q']$ has a final path. Let $s_l \dots s_{m'}$ be such that $s_{m'} = s'$ and $M(s_l \dots s_{m'})_{q,q'} > 0$. By Lemma 30 there exists a word $s_{m'} \dots s_m$ such that $Pr_{s_m}(\{w \in s_m S^\omega \mid \text{there exists a final path from } q' \text{ over } s_{m'} \dots s_m w\}) = 1$. Hence, $s_l \dots s_m$ is such that $Pr_{s_l}(s_l \dots s_m S^\omega) > 0$ and any word prefixed by $s_l \dots s_m$ almost surely has a final path outside D .
3. Let $\langle q'', s'' \rangle \in \langle q, s \rangle \triangleright s_0 \dots s_m$ (which exists since $\langle q, s \rangle \triangleright s_0 \dots s_l$ is a cut and by extension so is $\langle q, s \rangle \triangleright s_0 \dots s_m$). By strongly connectedness of D there exists $s_m \dots s_n \in S^*$ such that $s_m = s''$ and $\langle q, s \rangle \in \langle q'', s'' \rangle \triangleright s_{m+1} \dots s_n$.

Now consider the word $(s_0 \dots s_n)^{N+1}$. By construction, $\langle q, s \rangle \in \langle q, s \rangle \triangleright s_0 \dots s_n$ and there exists $\langle q', s' \rangle \notin D$ such that q reaches q' over $s_0 \dots s_{m-1}$ and $Pr_{s_m}(\{w \in s_m S^\omega \mid \text{there exists a final path from } q' \text{ over } s_{m'} \dots s_m w\}) = 1$. Hence, after every iteration of $s_0 \dots s_n$, the automaton can nondeterministically choose to stay in D or move to a set of states out of D after which it will almost surely have a final path. Thus, for almost any word prefixed with $(s_0 \dots s_n)^{N+1}$, there are at least $N+1$ final paths, which contradicts the fact that \mathcal{A} is an IBA. This shows that almost surely any final path from $\langle q, s \rangle \in D$ stays in D , and hence for any $\langle q', s' \rangle \notin D$ reachable from $\langle q, s \rangle$, $z_{\langle q', s' \rangle} = 0$. \square

With this we can invoke the proof of Lemma 8.2 verbatim:

Lemma 38 (Lemma 8.2 in [3]). *Let D be a recurrent SCC. For all $d \in D$, we have $\bar{z}_d > 0$ iff D is accepting.*

Proof. Verbatim from [3], relying only on Lemma 37 and the probabilistic interpretation in Lemma 33. \square

The proof of Lemma 10.2 in [3] only relies on Lemma 38 and hence can be invoked verbatim. This shows that cut vectors are D -normalisers.

Lemma 39 (Lemma 10.2 in [3]). *Let D be an accepting recurrent SCC and let $\bar{\mu}$ be a cut vector in D . Then $\bar{\mu}^\top \bar{z}_D = 1$.*

Proof. Verbatim. \square

Finally we can prove our version of Lemma 12:

Lemma 40 (Lemma 12 in [3]). *Let \mathcal{D}_+ be the set of accepting recurrent SCCs, and \mathcal{D}_0 the set of non-accepting recurrent SCCs. For each $D \in \mathcal{D}_+$, let $\bar{\mu}_D \in [0, 1]^D$ be a D -normalizer (which exists by Lemma 39). Then \bar{z} is the unique solution of the following linear system:*

$$\begin{aligned} \bar{\zeta} &= B\bar{\zeta} \\ \text{for all } D \in \mathcal{D}_+ : \bar{\mu}_D^\top \bar{\zeta}_D &= 1 \\ \text{for all } D \in \mathcal{D}_0 : \bar{\zeta}_D &= \vec{0} \end{aligned}$$

Proof. This proof is equivalent to the one in [3] and is included for completeness. The vector \bar{z} solves the system of equations by Lemma 32, Lemma 38, and the definition of a D -normalizer. To show uniqueness, let \bar{x} solve the system of equations. We show that $\bar{x} = \bar{z}$.

- Let $D \in \mathcal{D}_0$. Then $\vec{x}_D = \vec{0} = \vec{z}_D$.
- Let $D \in \mathcal{D}_+$. By Lemma 36, we have $\vec{x}_D = B_{D,D}\vec{x}_D$ and $\vec{z}_D = B_{D,D}\vec{z}_D$. By Theorem 29, the eigenspace of $B_{D,D}$ is one-dimensional, implying that \vec{x}_D is a scalar multiple of \vec{z}_D . We have $\vec{\mu}_D^\top \vec{x}_D = 1 = \vec{\mu}_D^\top \vec{z}_D$, hence $\vec{x}_D = \vec{z}_D$.
- Let $D := \bigcup \mathcal{D}_+ \cup \bigcup \mathcal{D}_0$ be the union of all recurrent SCCs, and let $\bar{D} := (Q \times S) \setminus D$. We have $\vec{x} - \vec{z} = B(\vec{x} - \vec{z})$. It follows

$$\begin{aligned} \vec{x}_{\bar{D}} - \vec{z}_{\bar{D}} &= B_{\bar{D},\bar{D}}(\vec{x}_{\bar{D}} - \vec{z}_{\bar{D}}) + B_{\bar{D},D}(\vec{x}_D - \vec{z}_D) \\ &= B_{\bar{D},\bar{D}}(\vec{x}_{\bar{D}} - \vec{z}_{\bar{D}}) + B_{\bar{D},D}(\vec{0}) \\ &= B_{\bar{D},\bar{D}}(\vec{x}_{\bar{D}} - \vec{z}_{\bar{D}}) \end{aligned}$$

by the previous two items. By Lemma 35, $\rho(B_{\bar{D},\bar{D}}) < 1$. Thus, $\vec{x}_{\bar{D}} = \vec{z}_{\bar{D}}$. \square

This enables us to prove Theorem 27:

Proof (of Theorem 27). It suffices to calculate D -normalisers and solve the system of equations described in Lemma 40. Solving a system of equations can be done in NC, and we can find a D -normaliser using [3, Lemma 22] in NC, proving the theorem. \square

Corollary 41. *Let \mathcal{M} be an MC and let \mathcal{A}_k be a k -ABA with n states, where $k = O(\log n)$. The probability that a random word sampled from \mathcal{M} is in $L_{\mathcal{A}}$ can be computed in POLYLOGSPACE .*

Proof. This combines Theorems 26 and 27. Since NC is contained in POLYLOGSPACE [23], this proves the corollary. \square

PSPACE-hardness of model checking k -ABAs against Markov chains can be shown easily from the *finite intersection problem* introduced by D. Kozen [19]. This problem asks, given a set of deterministic automata, whether the intersection of the languages is empty. By considering the union of the complements of the automata as a k -ABA we see that the probabilistic universality problem for k -ABAs is PSPACE-hard. Membership in PSPACE is shown by translating the k -ABA into an equivalent IBA and model checking this IBA against the Markov chain.

Theorem 42. *The model checking problem for k -ABAs is PSPACE-complete.*

Corollary 43. *Let \mathcal{M} be an MC and let \mathcal{A} be an NBA with n states. The probability that a word in $L_{\mathcal{A}}$ is accepted by \mathcal{M} can be computed in PSPACE.*

Proof. By Löding et al. [22], \mathcal{A} can be converted in a k -ABA \mathcal{A}_k with at most 3^n states where $k = n$. Hence, using Corollary 41, we can calculate $\Pr(L_{\mathcal{A}_k}) = \Pr(L_{\mathcal{A}})$ in $\text{POLYLOG}(2^n) = \text{POLY}(n)$ space. \square

Acknowledgements Mathieu Kubik, Tomasz Ponitka, and Joao Paulo Costa-longa contributed ideas to the proofs of Lemmas 8, 7, and 25, respectively.

References

1. Almagor, S., Boker, U., Kupferman, O.: What’s decidable about weighted automata? *Information and Computation* (2020)
2. Angluin, D., Antonopoulos, T., Fisman, D.: Strongly Unambiguous Büchi Automata Are Polynomially Predictable With Membership Queries. In: 28th EACSL Annual Conference on Computer Science Logic (CSL 2020) (2020)
3. Baier, C., Kiefer, S., Klein, J., Klüppelholz, S., Müller, D., Worrell, J.: Markov chains and unambiguous Büchi automata (extended version of a CAV’16 paper). arXiv:1605.00950 (2016), <http://arxiv.org/abs/1605.00950>
4. Berman, A., Plemmons, R.J.: Nonnegative matrices in the mathematical sciences. Academic Press (1979)
5. Berstel, J., Reutenauer, C.: Rational Series and Their Languages. Springer (1988)
6. Carlyle, J.W., Paz, A.: Realizations by stochastic finite automata. *Journal of Computer and System Sciences* **5**(1), 26–40 (1971)
7. Colcombet, T.: Unambiguity in automata theory. In: 17th International Workshop on Descriptive Complexity of Formal Systems (DCFS) (2015)
8. Droste, M., Kuich, W., Vogler, H.: Handbook of Weighted Automata. Springer Publishing Company, Incorporated, 1st edn. (2009)
9. Droste, M., Meinecke, I.: Weighted automata and regular expressions over valuation monoids. *International Journal of Foundations of Computer Science* (2011)
10. Fijalkow, N.: Undecidability results for probabilistic automata. *ACM SIGLOG News* **4**(4), 10–17 (2017)
11. Filiot, E., Gentilini, R., Raskin, J.F.: Finite-Valued Weighted Automata. In: 34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014) (2014)
12. Fliess, M.: Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées* **53**, 197–222 (1974)
13. Golomb, S.W., Goldstein, R.M., Hales, A.W., Welch, L.R.: Shift register sequences. Holden-Day series in information systems, Holden-Day, San Francisco (1967)
14. Holzer, M., Kutrib, M.: Nondeterministic descriptive complexity of regular languages. *International Journal of Foundations of Computer Science* (2003)
15. Indzhev, E., Kiefer, S.: On complementing unambiguous automata and graphs with many cliques and cocliques. arXiv preprint arXiv:2105.07470 (2021)
16. Jirásek, J., Jirásková, G., Sebej, J.: Operations on unambiguous finite automata. *International Journal of Foundations of Computer Science* **29**(5), 861–876 (2018)
17. Karmarkar, H., Joglekar, M., Chakraborty, S.: Improved upper and lower bounds for Büchi disambiguation. In: Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Proceedings (2013)
18. Kiefer, S., Widdershoven, C.: Efficient analysis of unambiguous automata using matrix semigroup techniques. arXiv preprint arXiv:1906.10093 (2019)
19. Kozen, D.: Lower bounds for natural proof systems. In: 18th Annual Symposium on Foundations of Computer Science (sfcs 1977). pp. 254–266. IEEE (1977)
20. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM Journal of Computing* **27**(4), 1073–1082 (1998)
21. Leung, H.: Descriptive complexity of NFA of different ambiguity. *International Journal of Foundations of Computer Science* **16**(5), 975–984 (2005)
22. Löding, C., Pirogov, A.: On finitely ambiguous Büchi automata. In: International Conference on Developments in Language Theory. pp. 503–515. Springer (2018)

23. Papadimitriou, C.M.: Computational complexity. Addison-Wesley, Reading, Massachusetts (1994)
24. Paz, A.: Introduction to probabilistic automata. Academic Press (2014)
25. Rabin, M.O.: Probabilistic automata. *Information and control* **6**(3), 230–245 (1963)
26. Raskin, M.: A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018) (2018)
27. Safra, S.: On the complexity of omega-automata. In: Proc. 29th IEEE Symp. Found. of Comp. Sci. pp. 319–327 (1988)
28. Schmidt, E.: Succinctness of descriptions of context-free, regular and finite languages. Ph.D. thesis, Cornell University, Ithaca, NY (1978)
29. Schützenberger, M.: On the definition of a family of automata. *Information and Control* **4**(2), 245–270 (1961)
30. Tzeng, W.G.: On path equivalence of nondeterministic finite automata. *Information Processing Letters* **58**(1), 43–46 (1996)
31. Weber, A., Seidl, H.: On the degree of ambiguity of finite automata. *Theoretical Computer Science* **88**(2), 325–349 (1991)