Probabilistic Model Checking via Families of Deterministic and Unambiguous Finite Automata

Christel Baier
Technische Universität Dresden, Germany

Sascha Klüppelholz
Technische Universität Dresden, Germany

Timm Spork
Technische Universität Dresden, Germany

Abstract

Families of deterministic finite automata (FDFA) have been introduced as a concise automaton model that characterizes ω-regular languages by processing their ultimately periodic words. FDFA are known to enjoy many good properties and can be exponentially more succinct than deterministic ω-automata with Rabin, Streett or parity acceptance. This paper addresses two main questions: (1) Are FDFA suitable for probabilistic model checking purposes? and (2) Is it possible to obtain an even more compact representation of ω-regular languages by allowing the components of an FDFA to be unambiguous instead of deterministic? Question (1) is answered in the affirmative by presenting the first polynomial-time algorithm for computing the probability that a discrete-time Markov chain satisfies an ω-regular property represented as an FDFA. Question (2) is motivated by the fact that unambiguous finite automata may require exponentially fewer states than deterministic ones. This paper introduces a model of families of unambiguous finite automata (FUFA) that captures the class of ω-regular languages. FUFA can be exponentially more succinct than both FDFA and unambiguous Büchi automata, and there is a single-exponential translation from linear temporal logic (LTL) to FUFA. This stands in contrast to a double-exponential lower bound for the translation from LTL to FDFA. Moreover, the polynomial-time probabilistic model checking algorithm for discrete-time Markov chains against FDFA-specifications is extended to the case where the property is represented by an FUFA with a deterministic leading automaton.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Verification by model checking

Keywords and phrases Families of Finite Automata, FDFA, Unambiguous Automata, Discrete-time Markov Chains, Probabilistic Model Checking, Verification

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.15

Related Version *Full Version*: `https://arxiv.org/abs/2606.21976` [5]

Funding This work was partly funded by the DFG grant 389792660 as part of TRR 248 (Foundations of Perspicuous Software Systems, see `https://perspicuous-computing.science`), the Cluster of Excellence EXC 2050/2 (CeTI, project ID 390696704, as part of Germany's Excellence Strategy), and by the BMFTR (Federal Ministry of Research, Technology and Space) in DAAD project 57616814 (SECAI, School of Embedded and Composite AI) as part of the program Konrad Zuse Schools of Excellence in Artificial Intelligence.

Acknowledgements We thank Dana Fisman for telling us about the double-exponential lower bound for the translation from LTL to FDFA, Jianlin Li for providing us with a copy of [23], and Robin Ziemek for many interesting discussions on the topic.

1 Introduction

Regular languages over infinite words, also called *ω-regular languages*, are widely used in fields such as verification and synthesis [31, 38, 3, 36]. There are many ways to represent ω-regular languages like, e.g., ω-automata, ω-regular expressions, or logics [18, 39].

Families of deterministic finite automata (*FDFA*) [2, 1] gained popularity in recent years as a concise description of ω-regular languages that is well-suited for automata learning [2, 24, 25, 7]. The central idea behind FDFA is to exploit the fact that ω-regular languages are uniquely characterized by their *ultimately periodic words*, i.e., the words in the language that are of the form $w = uv^\omega$ for finite words u, v, where v is nonempty [8, 9].

Formally, an FDFA $\mathcal{F} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in\mathcal{Q}})$ consists of a *leading automaton* $\mathcal{Q}$ and a set of *progress automata* $\mathcal{P}^q$, one for each state q of $\mathcal{Q}$. Here, $\mathcal{Q}$ and all of the $\mathcal{P}^q$ are *deterministic finite automata* (DFA). The inputs of $\mathcal{F}$ are *decompositions* (u, v) that represent ultimately periodic words xy^ω, i.e., that satisfy $uv^\omega = xy^\omega$. Intuitively, $\mathcal{F}$ accepts (u, v) if (1) v is accepted by the progress automaton $\mathcal{P}^q$, for q the state reached in $\mathcal{Q}$ after reading u, and (2) v closes a loop on q in $\mathcal{Q}$. It was shown in [2, 1] that *saturated* FDFA, which are FDFA in which either *all* decompositions of an ultimately periodic word that satisfy (2) are accepted or *none* of them is, characterize the set of ω-regular languages and enjoy many nice properties like constant space complementation and efficiently implementable Boolean operations and decision problems. Moreover, saturated FDFA can be exponentially more succinct than deterministic Rabin, Streett or parity automata [1], making them a promising automaton model for applications in the field of (probabilistic) model checking.

To the best of our knowledge, the only available works that discuss the use of FDFA in the context of model checking are [23, 26]. While the former gives a translation from the linear temporal logic (LTL) [32] to FDFA and shows how to use FDFA for the verification of transition systems against LTL-formulas, the latter proposes to model both the system under consideration and the property of interest as FDFA, and check if the system satisfies the property by deciding language containment between the two FDFA, which can be done efficiently [1]. Both [23, 26] do, however, not consider *probabilistic* model checking.

Given an ω-regular specification E and a discrete-time Markov chain (DTMC) $\mathcal{M}$, the *probabilistic model checking problem* asks to compute the probability that $\mathcal{M}$ generates a path that satisfies E [3]. Classical solutions to this problem translate the ω-regular specification into a deterministic ω-automaton $\mathcal{A}$ and compute the probability of reaching specific bottom strongly connected components (BSCCs) in a product of $\mathcal{M}$ and $\mathcal{A}$ [38, 13, 3]. Because of the succinctness of saturated FDFA compared to types of deterministic ω-automata typically used in probabilistic verification [1], it is of interest to investigate if saturated FDFA are a suitable model for the representation of ω-regular properties in probabilistic model checking.

We answer this question in the affirmative by providing the *first* probabilistic model checking algorithm for DTMCs $\mathcal{M}$ against ω-regular specifications represented as saturated FDFA $\mathcal{F} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in\mathcal{Q}})$. The algorithm is based on a classification and subsequent reachability analysis of specific *good* BSCCs in a product of $\mathcal{M}$ and $\mathcal{Q}$, and its runtime is polynomial in the size of $\mathcal{M}$ and the sizes of the components of $\mathcal{F}$.

By definition, all components of an FDFA are deterministic. It is, however, well-known that nondeterministic and unambiguous finite automata can require exponentially fewer states than minimal DFA for the same language [34, 35, 22]. Here, an automaton is unambiguous if every word has at most one accepting run [12]. Replacing the deterministic finite automata in an FDFA with nondeterministic and/or unambiguous finite automata can therefore yield up to exponential savings in the individual components of the resulting family of automata.

Following this observation, we introduce the novel notion of *families of unambiguous finite automata* (*FUFA*), which have the same structure as FDFA, but allow a nondeterministic leading automaton and unambiguous progress automata. The notion from the literature closest to FUFA are families of nondeterministic finite automata (FNFA) [7], which require a deterministic leading automaton but allow *nondeterministic* progress automata.

Similar to the case of FDFA, *saturated* FUFA characterize the set of ω-regular languages. Furthermore, FUFA can be exponentially more succinct than FDFA and unambiguous Büchi automata (UBA), and there is a single-exponential translation from LTL to FUFA. This stands in contrast to the case of saturated FDFA, where a double-exponential blow-up is inevitable [16, 10]. Additionally, it is shown how to combine the probabilistic model checking algorithm for DTMCs against FDFA-specifications developed in this paper with methods from [4] for probabilistic model checking of DTMCs against UBA-specifications to obtain a polynomial-time algorithm that computes the probability of a DTMC to generate an infinite path that satisfies an ω-regular specification represented as a *d*FUFA, which is an FUFA with a *deterministic* leading automaton. While the complexity of probabilistic model checking of DTMCs against general FUFA-specifications remains open, the corresponding polynomial-time algorithm for dFUFA is a first step in the direction of applying FUFA in probabilistic verification.

Main Contributions. In summary, the main contributions are

- a poly-time model checking algorithm for DTMC against FDFA-specifications (Section 3).
- the introduction of FUFA (families of unambiguous finite automata) as a new model to represent ω-regular languages that can be exponentially more succinct than FDFA and UBA, and for which there is a single-exponential translation from LTL (Section 4).
- a poly-time model checking algorithm for DTMC against dFUFA-specifications (Section 5).

Proofs can be found in the full version [5].

2 Preliminaries

Words and languages. A finite set Σ is called an *alphabet*. Let ε be the *empty word*, Σ^* the set of *finite words* over Σ, $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, and $\Sigma^{*+} = \Sigma^* \times \Sigma^+$. The *concatenation* of $w_1, w_2 \in \Sigma^*$ is $w_1 \cdot w_2$, where $w \cdot \varepsilon = w$, and we often write $w_1 w_2$ instead of $w_1 \cdot w_2$. For $k \in \mathbb{N}$, w^k is the k-fold repetition of w, where $w^0 = \varepsilon$. The *length* of $w = w_1 \dots w_n \in \Sigma^*$ is $|w| = 0$ if $w = \varepsilon$ and $|w| = n$ otherwise. Σ^ω is the set of *infinite* or *ω-words* over Σ. A *language* is a subset of Σ^*, and an *ω-language* is a subset of Σ^ω.

Automata over finite words. A *non-deterministic finite automaton* (*NFA*) is a 5-tuple $\mathcal{A} = (Q, \Sigma, q_{init}, \delta, F)$ with Q a finite set of *states*, Σ an *alphabet*, $q_{init} \in Q$ a unique *initial state*, $\delta \colon Q \times \Sigma \to 2^Q$ a *transition function* and $F \subseteq Q$ a set of *final states*. To emphasize the automaton $\mathcal{A}$ under consideration we may write, e.g., $F^{\mathcal{A}}$ instead of F. By slight abuse of notation we extend δ to finite words via $\delta \colon Q \times \Sigma^* \to 2^Q$ with $\delta(q, \varepsilon) = q$ and $\delta(q, aw) = \bigcup_{q' \in \delta(q,a)} \delta(q', w)$ for $q \in Q, a \in \Sigma$ and $w \in \Sigma^*$. A *run* of $\mathcal{A}$ on a word $w = a_1 \dots a_n \in \Sigma^*$ is a sequence of states $q_0 q_1 \dots q_n$ such that $q_0 = q_{init}$ and $q_{i+1} \in \delta(q_i, a_{i+1})$ for all $0 \leq i < n$. $\mathcal{A}$ *accepts* the word $w \in \Sigma^*$ if there is a run r of $\mathcal{A}$ on w that ends in a final state, and in this case r is *accepting*. $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w\}$ is the *language of* $\mathcal{A}$.

Given $w \in \Sigma^*$, let $\mathcal{A}(w) = \delta(q_{init}, w)$. The *size of* $\mathcal{A}$ is $|\mathcal{A}| = |Q|$. We often, by slight abuse of notation, identify Q with $\mathcal{A}$ and write $q \in \mathcal{A}$ instead of $q \in Q$. For $q \in Q$ let $\mathcal{A}_q$ be like $\mathcal{A}$, but with initial state q. $\mathcal{A}$ is *total* if $|\delta(q, a)| \geq 1$ for all $q \in Q$ and $a \in \Sigma$. Every automaton $\mathcal{A}$ can be made total by adding a rejecting sink state reached via all transitions for which $\delta(q, a) = \emptyset$ in $\mathcal{A}$. W.l.o.g. we assume all states of $\mathcal{A}$ to be reachable from q_{init}.

A *deterministic finite automaton* (*DFA*) is an NFA that satisfies $|\delta(q,a)| \leq 1$ for every $q \in Q$ and $a \in \Sigma$. An *unambiguous finite automaton* (*UFA*) is an NFA that has at most one accepting run on every $w \in \Sigma^*$. NFA, UFA and DFA all characterize the regular languages.

ω-Automata. A *nondeterministic Büchi automaton* (*NBA*) has the same structure as an NFA, but operates on ω-words. A *run* of an NBA $\mathcal{A}$ on $w = a_1 a_2 \ldots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 \ldots \in Q^\omega$ with $q_0 = q_{init}$ and $q_{i+1} \in \delta(q_i, a_{i+1})$ for all $i \in \mathbb{N}$. The run ρ is *accepted* by $\mathcal{A}$ if $\inf(\rho) \cap F \neq \emptyset$, where $\inf(\rho) = \{q \in Q \mid q \text{ is visited infinitely often along } \rho\}$, and in this case ρ is called *accepting*. The *language* of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is defined as for NFA. *Unambiguous Büchi automata* (*UBA*) are NBA in which every word $w \in \Sigma^\omega$ has at most one accepting run. Both NBA and UBA characterize the set of ω-regular languages.

Deterministic Rabin automata (*DRA*) are deterministic ω-automata that also characterize the set of ω-regular languages. Instead of final states, a DRA $\mathcal{A}$ has n so-called *Rabin pairs* (T_i, R_i) with $T_i, R_i \subseteq Q$. A word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if the unique run ρ of $\mathcal{A}$ on w satisfies $\inf(\rho) \cap T_i = \emptyset$ and $\inf(\rho) \cap R_i \neq \emptyset$ for some $1 \leq i \leq n$. The language $\mathcal{L}(\mathcal{A})$ of a DRA $\mathcal{A}$ is defined as for NBA. W.l.o.g. we assume all DRA and DFA to be total.

Ultimately periodic words and FDFA. A word $w \in \Sigma^\omega$ is *ultimately periodic* if $w = uv^\omega$ for a $(u,v) \in \Sigma^{*+}$. For every ω-regular language L, $\mathrm{UP}(L) = \{uv^\omega \in L \mid (u,v) \in \Sigma^{*+}\} \neq \emptyset$ [8]. Moreover, for ω-regular languages L, L' it holds that $L = L'$ iff $\mathrm{UP}(L) = \mathrm{UP}(L')$ [9].

A *family of deterministic finite automata* (*FDFA*) is a tuple $\mathcal{F} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q \in \mathcal{Q}})$. $\mathcal{Q}$ is the *leading automaton* and satisfies $F^{\mathcal{Q}} = \emptyset$, and each $\mathcal{P}^q$, $q \in \mathcal{Q}$, is a *progress automaton*. In an FDFA, $\mathcal{Q}$ and all $\mathcal{P}^q$ are DFA. The *size* of $\mathcal{F}$ is $|\mathcal{F}| = (|\mathcal{Q}|, \max_{q \in \mathcal{Q}} |\mathcal{P}^q|)$. FDFA process ultimately periodic words represented by *decompositions* $(u,v) \in \Sigma^{*+}$. A decomposition (u,v) is *normalized* (w.r.t. $\mathcal{Q}$) if $\mathcal{Q}(u) = \mathcal{Q}(uv)$, and $\mathcal{F}$ *accepts* (u,v) if it is normalized and $v \in \mathcal{L}(\mathcal{P}^q)$ for $q = \mathcal{Q}(u)$. Let $[\![\mathcal{F}]\!] = \{(u,v) \in \Sigma^{*+} \mid \mathcal{F} \text{ accepts } (u,v)\}$ and $\mathcal{L}(\mathcal{F}) = \{uv^\omega \mid \exists (x,y) \in \Sigma^{*+} \colon xy^\omega = uv^\omega \text{ and } (x,y) \in [\![\mathcal{F}]\!]\}$. $\mathcal{F}$ *accepts* $w = uv^\omega$ if $w \in \mathcal{L}(\mathcal{F})$.

An FDFA is *saturated* if for all $w = uv^\omega$ it accepts either every or none of the normalized decompositions of w. Saturated FDFA characterize the ω-regular languages [1, 7], i.e., for every saturated FDFA $\mathcal{F}$ there is a unique ω-regular language $\mathcal{L}_\omega(\mathcal{F})$ with $\mathrm{UP}(\mathcal{L}_\omega(\mathcal{F})) = \mathcal{L}(\mathcal{F})$, and for every ω-regular language L there is a saturated FDFA $\mathcal{F}_L$ with $\mathcal{L}(\mathcal{F}_L) = \mathrm{UP}(L)$.

Markov chains. We fix a countable set AP of *atomic propositions*. For a finite set S, let $\mathrm{Distr}(S) = \{\mu \colon S \to [0,1] \mid \sum_{s \in S} \mu(s) = 1\}$ contain the *probability distributions* on S.

A *discrete-time Markov chain* (*DTMC*) is a tuple $\mathcal{M} = (S, P, s_{init}, \ell)$ with S a finite set of *states*, $P \colon S \to \mathrm{Distr}(S)$ a *transition distribution function*, $s_{init} \in S$ a unique *initial state*, and $\ell \colon S \to 2^{AP}$ a *labeling function*. $P(s,s') = P(s)(s')$ denotes the probability to move from s to s' in one step, and $\mathrm{Succ}(s) = \{s' \in S \mid P(s,s') > 0\}$ is the set of *successors* of s. For $s \in S$, let $\mathcal{M}_s$ be like $\mathcal{M}$, but with initial state s. The *size* of $\mathcal{M}$ is $|\mathcal{M}| = |S|$.

A sequence $\pi = s_0 s_1 \ldots \in S^\omega$ is a (*infinite*) *path* of $\mathcal{M}$ if $s_0 = s_{init}$ and $s_{i+1} \in \mathrm{Succ}(s_i)$ for all $i \in \mathbb{N}$. $\pi[i] = s_i$ is the state at position i of π, and $\mathrm{tr}(\pi) = \ell(s_0)\ell(s_1) \ldots \in (2^{AP})^\omega$ is the *trace* of π. *Finite* paths $\pi = s_0 s_1 \ldots s_n \in S^{n+1}$ and their traces are defined analogously, and if π is finite we set $\mathrm{last}(\pi) = s_n$. $\mathrm{Paths}(\mathcal{M})$ is the set of all paths of $\mathcal{M}$, and $\mathrm{Paths}^*(\mathcal{M})$ is the set of all finite paths of $\mathcal{M}$. For $s \in S$, let $\mathrm{Paths}(s) = \mathrm{Paths}(\mathcal{M}_s)$ and $\mathrm{Paths}^*(s) = \mathrm{Paths}^*(\mathcal{M}_s)$.

$C \subseteq S$ is a *strongly connected component* (*SCC*) of $\mathcal{M}$ if all $s, s' \in C$ are reachable from one another via finite paths. An SCC C is *bottom* (*BSCC*) if $\mathrm{Succ}(s) \subseteq C$ for all $s \in C$.

For a DTMC $\mathcal{M}$ and a deterministic automaton $\mathcal{A}$ with alphabet 2^{AP}, the *product* $\mathcal{M} \otimes \mathcal{A}$ is a DTMC with state space $S \times Q$, initial state $\langle s_{init}, \delta(q_{init}, \ell(s_{init}))\rangle$, labeling function $\ell'(\langle s,q\rangle) = \ell(s)$ and transition probabilities $P'(\langle s,q\rangle, \langle s',q'\rangle) = P(s,s')$ if $q' = \delta(q, \ell(s'))$ and 0 otherwise. The DTMC $\mathcal{M} \times \mathcal{A}$ is build similar to $\mathcal{M} \otimes \mathcal{A}$, but with initial state $\langle s_{init}, q_{init}\rangle$.

We consider the standard probability measure $\Pr^{\mathcal{M}}$ on subsets of $\mathrm{Paths}(\mathcal{M})$, defined via *cylinder sets* $Cyl(\pi) = \{\sigma \in \mathrm{Paths}(\mathcal{M}) \mid \pi \text{ is a prefix of } \sigma\}$ for $\pi = s_0 s_1 \dots s_n \in \mathrm{Paths}^*(\mathcal{M})$. We abbreviate $\Pr^{\mathcal{M}}(Cyl(\pi))$ by $\Pr^{\mathcal{M}}(\pi)$ and the measure yields $\Pr^{\mathcal{M}}(\pi) = 0$ if $s_0 \neq s_{init}$ and $\Pr^{\mathcal{M}}(\pi) = \prod_{j=0}^{n-1} P(s_j, s_{j+1})$ otherwise. Given an ω-regular property E, $\Pr^{\mathcal{M}}(E)$ is the probability of $\mathcal{M}$ to generate a path with trace in E. For details on DTMCs see, e.g., [19, 3].

LTL. For $a \in AP$, formulas of the *linear temporal logic* (*LTL*) are formed w.r.t. the grammar

$$\varphi ::= true \mid a \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2.$$

Here, $\bigcirc$ is the *next*-operator, so $\sigma \in (2^{AP})^\omega$ satisfies $\bigcirc\varphi$ iff φ holds in $\sigma[1]$, and U is the *until*-operator, so σ satisfies $\varphi_1 \mathsf{U} \varphi_2$ iff, along σ, φ_1 holds until φ_2 is true. As syntactic sugar we define $\Diamond\varphi \equiv true \mathsf{U} \varphi$ which is satisfied by σ iff φ holds *eventually* along σ. Let $\mathrm{Words}(\varphi) = \{w \in (2^{AP})^\omega \mid w \text{ satisfies } \varphi\}$. For a DTMC $\mathcal{M}$, $\Pr^{\mathcal{M}}(\varphi)$ is the probability that $\mathcal{M}$ generates a path π with $\mathrm{tr}(\pi) \in \mathrm{Words}(\varphi)$. For details on LTL see, e.g., [32, 3].

3 Model Checking DTMCs against Saturated FDFA

Let $\mathcal{M}$ be a DTMC and E an ω-regular property. The *probabilistic model checking problem* for $\mathcal{M}$ and E asks to compute $\Pr^{\mathcal{M}}(E)$, i.e., the probability that $\mathcal{M}$ generates a path π with $\mathrm{tr}(\pi) \in E$. In this section we show how to efficiently solve the probabilistic model checking problem in the case that E is represented by a saturated FDFA $\mathcal{F} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q \in \mathcal{Q}})$.

▶ **Theorem 3.1.** *For a given DTMC $\mathcal{M}$ and a given saturated FDFA $\mathcal{F}$ with $\mathcal{L}_\omega(\mathcal{F}) = E$, the probability $\Pr^{\mathcal{M}}(E)$ can be computed in time polynomial in the sizes of $\mathcal{M}$ and $\mathcal{F}$.*

To prove Theorem 3.1 we present a polynomial-time probabilistic model checking algorithm for DTMCs $\mathcal{M}$ against saturated FDFA $\mathcal{F}$. The algorithm follows the standard approach for model checking $\mathcal{M}$ against deterministic ω-automata $\mathcal{A}$ [3], which boils down to (1) constructing the product $\mathcal{M} \otimes \mathcal{A}$, (2) computing the BSCCs $B_1, \dots, B_k$ of $\mathcal{M} \otimes \mathcal{A}$ that satisfy the acceptance condition of $\mathcal{A}$, and (3) computing the probability of reaching one of these BSCCs B_i in the product. When dealing with (saturated) FDFA instead of classical deterministic ω-automata, the challenge is that FDFA can only process the ultimately periodic words of a given ω-regular language. Hence, the decision in (2) if a BSCC of the product of $\mathcal{M}$ and (in this case) the leading automaton $\mathcal{Q}$ of $\mathcal{F}$ is accepting requires some extra care.

Intuitively, our algorithm works as follows: First, the set of BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ is computed. Afterwards, for each BSCC B of $\mathcal{M} \otimes \mathcal{Q}$, an arbitrary state $\langle s, q\rangle \in B$ is picked, and the product $I_B = \mathcal{M}_s \times \mathcal{Q}_q \times \mathcal{P}^q$ is analyzed for the existence of a BSCC that does not contain a state $\langle s, q, f\rangle$ with $f \in F^{\mathcal{P}^q}$. Importantly, the existence of such a *rejecting* BSCC in I_B does not depend on the specific choice of $\langle s, q\rangle \in B$ (Lemma 3.3), and checking if I_B has a rejecting BSCC can be done by a simple graph analysis in time linear in $|I_B|$ (Lemma 3.4). If I_B does not contain a rejecting BSCC then B is classified as *good*. In the end, $\Pr^{\mathcal{M}}(E)$ equals the probability of reaching a good BSCC in $\mathcal{M} \otimes \mathcal{Q}$ (Proposition 3.7). The majority of this section is devoted to the proof of correctness of this algorithm, and hence of Theorem 3.1.

The upcoming proofs make use of an auxiliary DRA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = E$ and exploit the fact that $\Pr^{\mathcal{M}}(E)$ equals the probability to reach an *accepting* BSCC in $\mathcal{M} \otimes \mathcal{A}$ [3]. Here, a BSCC B of $\mathcal{M} \otimes \mathcal{A}$ is accepting if there is a Rabin pair (T_i, R_i) of $\mathcal{A}$ with $B|_{\mathcal{A}} \cap T_i = \emptyset$ and $B|_{\mathcal{A}} \cap R_i \neq \emptyset$, where $B|_{\mathcal{A}} = \{q \in \mathcal{A} \mid \exists\, s \in \mathcal{M} \colon \langle s, q\rangle \in B\}$. Otherwise, B is *rejecting*.

The following lemma shows that if a BSCC B of $\mathcal{M} \otimes \mathcal{A}$ is accepting, all paths that reach B and visit all its states infinitely often satisfy E, while if B is rejecting all these paths violate E. Recall that for $\sigma \in \mathrm{Paths}(\mathcal{M} \otimes \mathcal{A})$, $\mathrm{tr}(\sigma)$ equals the trace of the path of $\mathcal{M}$ obtained from the projection of σ to its first components.

▶ **Lemma 3.2.** *Let $u \in \mathrm{Paths}^*(\mathcal{M} \otimes \mathcal{A})$ end in $\langle s, p\rangle \in B$ for some BSCC B of $\mathcal{M} \otimes \mathcal{A}$, and let $\pi \in \mathrm{Paths}^{\mathcal{M}\otimes\mathcal{A}}(\langle s, p\rangle)$ be such that, along π, every state of B is visited infinitely often.*
1. *If B is accepting then $\mathrm{tr}(u\pi) \in E$.*
2. *If B is rejecting then $\mathrm{tr}(u\pi) \notin E$.*

Proof Sketch. If B is accepting then there is a Rabin pair (T_i, R_i) with $B|_{\mathcal{A}} \cap T_i = \emptyset$ and $B|_{\mathcal{A}} \cap R_i \neq \emptyset$. Because all states of B are visited infinitely often along π, so is a state in R_i, while no state in T_i is visited at all. Thus, $\mathrm{tr}(u\pi) \in \mathcal{L}(\mathcal{A}) = E$. If B is rejecting then for every (T_i, R_i) we have $T_i \cap B|_{\mathcal{A}} \neq \emptyset$, so $\inf(u\pi) \cap T_i \neq \emptyset$, or $R_i \cap B|_{\mathcal{A}} = \emptyset$. Hence, $\mathrm{tr}(u\pi) \notin E$. ◀

Let $\langle s, q\rangle$ be a state in a BSCC of $\mathcal{M} \otimes \mathcal{Q}$. We define $I_{\langle s,q\rangle} = \mathcal{M}_s \times \mathcal{Q}_q \times \mathcal{P}^q$ and say that a state $\langle s, q, f\rangle$ in a BSCC of $I_{\langle s,q\rangle}$ is *accepting* if $f \in F^{\mathcal{P}^q}$. Note that any accepting state of $I_{\langle s,q\rangle}$ has s in its first and q in its second component. W.l.o.g. we assume that accepting states in $I_{\langle s,q\rangle}$ are labeled with a fresh atomic proposition *accept*. A BSCC B of $I_{\langle s,q\rangle}$ is *rejecting* if it does not contain an accepting state. It turns out that the specific choice of $\langle s, q\rangle$ in a BSCC of $\mathcal{M} \otimes \mathcal{Q}$ has no influence on the existence of rejecting BSCCs in $I_{\langle s,q\rangle}$.

▶ **Lemma 3.3.** *Let C be a BSCC of $\mathcal{M} \otimes \mathcal{Q}$ and $\langle s, q\rangle, \langle s', q'\rangle \in C$. If $I_{\langle s,q\rangle}$ has a rejecting BSCC then $I_{\langle s',q'\rangle}$ does as well.*

Proof Sketch. Let u, v, w be such that u reaches $\langle s', q'\rangle$ in $\mathcal{M} \otimes \mathcal{Q}$, v takes $\mathcal{M} \otimes \mathcal{Q}$ from $\langle s', q'\rangle$ to $\langle s, q\rangle$, and w reaches a rejecting BSCC in $I_{\langle s,q\rangle}$. Assume that $I_{\langle s',q'\rangle}$ does not have a rejecting BSCC. Then there is an x such that vwx reaches an accepting state $\langle s', q', f\rangle$ in $I_{\langle s',q'\rangle}$, and so $(\mathrm{tr}(u), \mathrm{tr}(vwx)) \in [\![\mathcal{F}]\!]$. However, $\mathcal{Q}(\mathrm{tr}(uv)) = q = \delta^{\mathcal{Q}}(q, \mathrm{tr}(wxv))$ and $\mathrm{tr}(wxv) \notin \mathcal{L}(\mathcal{P}^q)$, where the latter follows since a rejecting BSCC of $I_{\langle s,q\rangle}$ is reached upon w. Hence, $(\mathrm{tr}(uv), \mathrm{tr}(wxv)) \notin [\![\mathcal{F}]\!]$, contradicting the saturation of $\mathcal{F}$. ◀

Let B be a BSCC of $\mathcal{M}\otimes\mathcal{Q}$ and let $\langle s, q\rangle$ be an arbitrary state in B. We set $I_B = I_{\langle s,q\rangle}$ and say that B is *good* iff I_B does not have a rejecting BSCC. By Lemma 3.3, the classification of B as *good* is independent of the choice of $\langle s, q\rangle$. Deciding if B is good can be done efficiently.

▶ **Lemma 3.4.** *Deciding if a BSCC B of $\mathcal{M} \otimes \mathcal{Q}$ is good is possible in time linear in $|I_B|$.*

Proof Sketch. It holds that I_B has a rejecting BSCC iff $\Pr^{I_B}(\Diamond \mathit{accept}) < 1$, which is decidable in time linear in $|I_B|$, see, e.g., [3]. ◀

Recall that $\mathcal{A}$ is an auxiliary DRA with $\mathcal{L}(\mathcal{A}) = E = \mathcal{L}_\omega(\mathcal{F})$. The next lemma shows a direct connection between reaching good BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ and accepting BSCCs of $\mathcal{M} \otimes \mathcal{A}$.

▶ **Lemma 3.5.** *Let $u \in \mathrm{Paths}^*(\mathcal{M})$ such that the lifting of u to $\mathcal{M} \otimes \mathcal{A}$ reaches a BSCC $B_{\mathcal{A}}$, and the lifting of u to $\mathcal{M} \otimes \mathcal{Q}$ reaches a BSCC $B_{\mathcal{Q}}$. Then $B_{\mathcal{A}}$ is accepting iff $B_{\mathcal{Q}}$ is good.*

Proof Sketch. Let $B_{\mathcal{A}}$ be accepting and assume that $I_{B_{\mathcal{Q}}} = I_{\langle s,q\rangle}$ has a rejecting BSCC. Then there is a finite cycle v in $B_{\mathcal{A}}$ visiting all states of $B_{\mathcal{A}}$, while if followed in $I_{B_{\mathcal{Q}}}$ ends in a state $\langle s, q, r\rangle$ in the rejecting BSCC. Hence, $uv^\omega \in \mathcal{L}(\mathcal{A}) = E$ by Lemma 3.2, but $uv^\omega \notin \mathcal{L}_\omega(\mathcal{F}) = E$, a contradiction. The reverse direction can be shown similarly. ◀

The precondition of Lemma 3.5 requires that the finite path u is such that its lifting to both $\mathcal{M} \otimes \mathcal{Q}$ and $\mathcal{M} \otimes \mathcal{A}$ reaches a BSCC. However, the final model checking algorithm does not have access to the DRA $\mathcal{A}$, and hence has no information about the runs of $\mathcal{M} \otimes \mathcal{A}$ on u. It turns out that this is not a problem, since after reaching a BSCC in $\mathcal{M} \otimes \mathcal{Q}$ on u either all BSCCs of $\mathcal{M} \otimes \mathcal{A}$ reachable after u are accepting, or all of them are rejecting.

This is the case because of the following technical lemma, which states that all finite words reaching the same state in the leading automaton $\mathcal{Q}$ of an FDFA $\mathcal{F}$ for an ω-regular property E have the same *residual languages* w.r.t. E. For a given $x \in \Sigma^*$, the residual language of x w.r.t. E is $Res_x^E = \{z \in \Sigma^\omega \mid xz \in E\}$.

▶ **Lemma 3.6.** *Let $x, y \in \Sigma^*$ and let $\mathcal{Q}$ be the leading automaton of an FDFA $\mathcal{F}$ for E. If $\mathcal{Q}(x) = \mathcal{Q}(y)$ then $Res_x^E = Res_y^E$.*

Proof Sketch. Let $Q(x) = Q(y)$. Then $xuv^\omega \in E$ iff $yuv^\omega \in E$, and so $\mathrm{UP}(Res_x^E) = \mathrm{UP}(Res_y^E)$. Because Res_x^E and Res_y^E are ω-regular, this implies $Res_x^E = Res_y^E$ [9]. ◀

Let $u \in \mathrm{Paths}^*(\mathcal{M})$ be such that the lifting of u to $\mathcal{M} \otimes \mathcal{Q}$ reaches a state $\langle s, q\rangle$ in a BSCC $B_\mathcal{Q}$. Because $\mathcal{M} \otimes \mathcal{A}$ is a DTMC, almost surely a finite extension v of u is generated by $\mathcal{M}$ such that the lifting of uv to $\mathcal{M} \otimes \mathcal{A}$ ends in a BSCC $B_\mathcal{A}$. As $\mathcal{M} \otimes \mathcal{Q}$ is in $\langle s, q\rangle \in B_\mathcal{Q}$ after u, almost surely $\mathcal{M}$ generates an extension w of uv such that the lifting of uvw to $\mathcal{M} \otimes \mathcal{Q}$ also ends in $\langle s, q\rangle$. But then $Res_u^E = Res_{uvw}^E$ by Lemma 3.6, and so either almost surely a path with prefix u is in E (if $B_\mathcal{A}$ is accepting), or violates E (if $B_\mathcal{A}$ is rejecting).

Hence, after entering a BSCC $B_\mathcal{Q}$ of $\mathcal{M} \otimes \mathcal{Q}$ the probability of $\mathcal{M}$ to generate a path in E equals 1 (if $B_\mathcal{Q}$ is good), or it equals 0. Let *accBSCC* contain all accepting BSCCs of $\mathcal{M} \otimes \mathcal{A}$, and *goodBSCC* contain all good BSCCs of $\mathcal{M} \otimes \mathcal{Q}$. Because $\Pr^\mathcal{M}(E) = \Pr^{\mathcal{M}\otimes\mathcal{A}}(\Diamond\, accBSCC)$ [3], Lemma 3.5 implies the following result.

▶ **Proposition 3.7.** $\Pr^\mathcal{M}(E) = \Pr^{\mathcal{M}\otimes\mathcal{Q}}(\Diamond\, goodBSCC)$.

The following algorithm efficiently computes the satisfaction probability of a DTMC $\mathcal{M}$ against an ω-regular property E represented by a saturated FDFA $\mathcal{F} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in\mathcal{Q}})$:

1. Compute the set $\mathcal{B}$ of BSCCs of $\mathcal{M} \otimes \mathcal{Q}$.
2. For every $B \in \mathcal{B}$ pick an arbitrary $\langle s, q\rangle \in B$ and check if $I_B = I_{\langle s,q\rangle} = \mathcal{M}_s \times \mathcal{Q}_q \times \mathcal{P}^q$ contains a rejecting BSCC. If this is not the case, add B to the set *goodBSCC*.
3. Compute $\Pr^\mathcal{M}(E) = \Pr^{\mathcal{M}\otimes\mathcal{Q}}(\Diamond\, goodBSCC)$.

The correctness of the algorithm follows from Proposition 3.7. Regarding its complexity, the procedure boils down to obtaining the set of good BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ and a subsequent computation of reachability probabilities in $\mathcal{M} \otimes \mathcal{Q}$. Deciding if a BSCC B of $\mathcal{M} \otimes \mathcal{Q}$ is good can be done in time linear in the size of I_B (Lemma 3.4), while the reachability probability of *goodBSCC* in $\mathcal{M} \otimes \mathcal{Q}$ can be computed in time polynomial in $|\mathcal{M} \otimes \mathcal{Q}| = |\mathcal{M}| \cdot |\mathcal{Q}|$ by solving a system of linear equations [3]. Hence, the overall runtime of the algorithm is polynomial in the sizes of $\mathcal{M}$ and $\mathcal{F}$. This finishes the proof of Theorem 3.1.

To conclude this section we discuss how to obtain an FDFA for a given ω-regular property. In the literature, FDFA are mainly applied in learning algorithms for ω-regular languages [2, 24, 25, 7]. These algorithms naturally yield a way to compute FDFA representations for a given ω-regular specification. An orthogonal approach is the direct construction of saturated FDFA from, e.g., LTL formulas. The only direct translation from LTL to FDFA that we are aware of is presented in [23]. It takes inspiration from the works of Esparza et al. [14, 15] on the construction of ω-automata from LTL formulas and, in the worst case, incurs a double-exponential blow-up. Unfortunately, this blow-up is inevitable.

▶ **Proposition 3.8** ([16, 10])**.** *There is a family of LTL-formulas φ_n of length polynomial in n such that any saturated FDFA for φ_n has a leading automaton of size at least 2^{2^n}.*

Proof Sketch. Let $L'_n = \{x\#w\#y\$w \mid x, y \in \{0,1,\#\}^*, w \in \{0,1\}^n\}$ and $L_n = L'_n \cdot \#^\omega$. For each n there is an LTL formula φ_n of length quadratic in n with $\mathrm{Words}(\varphi_n) = L_n$ [20]. The leading automaton of any saturated FDFA for L_n must "recognize" if a prefix is in L'_n. But this requires a DFA with at least 2^{2^n} states [11]. ◀

Nevertheless, it was shown in [1] that saturated FDFA can be exponentially more succinct than deterministic Rabin, Streett or parity automata, which are types of ω-automata often used in probabilistic verification [3]. The probabilistic model checking procedure outlined in this section may therefore still reduce the total verification time significantly.

4 Families of Unambiguous Finite Automata

It is well-known that an NFA or a UFA for a given regular language L may require exponentially fewer states than a DFA for L [34]. Hence, allowing nondeterminism in the leading automaton and/or the progress automata of an FDFA might yield families of finite automata of significantly reduced size. Based on this observation, we define the novel notion of *families of unambiguous finite automata* (*FUFA*), which have leading NFA and progress UFA.

▶ **Definition 4.1.** *A* family of unambiguous finite automata (FUFA) *is a tuple* $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ *with* $A \subseteq \mathcal{Q}$*, where* $\mathcal{Q}$ *is a leading NFA and* $\mathcal{P}^q$*,* $q \in A$*, are progress UFA.*

Let $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ be an FUFA. In contrast to FDFA, it is not required in an FUFA that every state $q \in \mathcal{Q}$ has a corresponding progress automaton $\mathcal{P}^q$, but only those q contained in $A \subseteq \mathcal{Q}$. As every DFA is both a UFA and an NFA, all FDFA can be interpreted as FUFA in which A contains all states of $\mathcal{Q}$. On the other hand, not every FUFA is an FDFA, since the leading automaton and the progress automata of an FUFA can contain nondeterminism.

We extend the notion of (normalized) acceptance of decompositions $(u, v) \in \Sigma^{*+}$ from FDFA to FUFA. The main difference between the acceptance of (u, v) in an FDFA and an FUFA is that in the latter, acceptance of v by a progress automaton can only be defined if a state in A is reachable on u, i.e., if $A_u \coloneqq \mathcal{Q}(u) \cap A$ is nonempty. Moreover, the leading automaton of an FUFA can be nondeterministic, so $\mathcal{Q}(u)$ may contain multiple states and it is possible that (u, v) is normalized only w.r.t. some of them. Hence, accepted normalized decompositions (u, v) are required to satisfy the normalization condition w.r.t. at least one $q \in A_u$ for which $v \in \mathcal{L}(\mathcal{P}^q)$. These observations are combined in the following definition.

▶ **Definition 4.2.** *Let* $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ *be an FUFA. A decomposition* $(u, v) \in \Sigma^{*+}$ *is* q-normalized *if* $q \in A_u$ *and* $q \in \delta^{\mathcal{Q}}(q, v)$*, and* (u, v) *is* normalized *if it is* q*-normalized w.r.t. some* $q \in A$*. Moreover,* $\mathcal{U}$ accepts *the decomposition* (u, v) *if there is a* $q \in A$ *such that* (u, v) *is* q*-normalized and* $v \in \mathcal{L}(\mathcal{P}^q)$*. The sets* $[\![\mathcal{U}]\!]$*,* $\mathcal{L}(\mathcal{U})$ *and* $\mathcal{L}_\omega(\mathcal{U})$ *are defined as for FDFA.*

Similar to FDFA, we also require FUFA to be *saturated*. While in a saturated FDFA either every or no normalized decomposition of a word is accepted, in a saturated FUFA we again have to restrict the definition to those decompositions $(u, v) \in \Sigma^{*+}$ with $A_u \neq \emptyset$.

▶ **Definition 4.3.** *An FUFA* $\mathcal{U}$ *is saturated if for every ultimately periodic word* w *either every normalized decomposition of* w *is accepted by* $\mathcal{U}$*, or none of them is.*

Let $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ be a saturated FUFA, $w \in \Sigma^\omega$ an ultimately periodic word, and (u, v) a normalized decomposition of w. The central idea behind FUFA is to obtain an *unambiguous extension* of FDFA. The nondeterminism of $\mathcal{Q}$ and $\mathcal{P}^q$, $q \in A$, can, however, cause some undesired behavior when viewing FUFA under this premise. For instance, it is possible that there are $q_1, \ldots, q_n \in A_u$ such that (u, v) is q_i-normalized and $v \in \mathcal{L}(\mathcal{P}^{q_i})$ for all $1 \leq i \leq n$. This is not in the spirit of unambiguous automata, which have at most one accepting run for every word. Adjusted to FUFA this means that for every decomposition (u, v) of w there should be a *unique* $q \in A_u$ for which (u, v) is q-normalized and $v \in \mathcal{L}(\mathcal{P}^q)$. As another example, a saturated FDFA $\mathcal{F} = (\mathcal{Q}_\mathcal{F}, \{\mathcal{P}^q_\mathcal{F}\}_{q\in\mathcal{Q}_\mathcal{F}})$ satisfies that if $(u, v) \in [\![\mathcal{F}]\!]$

then also $(u, v^n) \in [\![\mathcal{F}]\!]$ for all n [7]. Hence, for $q = \mathcal{Q}_\mathcal{F}(u)$ it holds that $v^n \in \mathcal{L}(\mathcal{P}^q_\mathcal{F})$ for all $n \geq 1$. In saturated FUFA, however, decompositions (u, v^i) and (u, v^j) for $i \neq j$ can be normalized w.r.t. different q_i and q_j in A_u, and accepted via different progress automata $\mathcal{P}^{q_i}$ and $\mathcal{P}^{q_j}$. This distinguishes saturated FUFA significantly from saturated FDFA.

To bring saturated FUFA closer to saturated FDFA and unambiguous automata, we require saturated FUFA to be *decomposition-unambiguous* and *power-unambiguous*.

▶ **Definition 4.4.** *Let $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ be a saturated FUFA.*

1. *$\mathcal{U}$ is* decomposition-unambiguous *if for every $(u, v) \in \Sigma^{*+}$ there is at most one $q \in A_u$ such that (u, v) is q-normalized and $v \in \mathcal{L}(\mathcal{P}^q)$.*
2. *$\mathcal{U}$ is* power-unambiguous *if for all q-normalized decompositions $(u, v) \in [\![\mathcal{U}]\!]$ with $v \in \mathcal{L}(\mathcal{P}^q)$ and every $f \in \mathcal{P}^q(v) \cap F^{\mathcal{P}^q}$ it holds that $F^{\mathcal{P}^q} \cap \delta^{\mathcal{P}^q}(f, v^n) \neq \emptyset$ for all $n \geq 1$.*

Decomposition- and power-unambiguity imply several useful properties that are guaranteed in FUFA $\mathcal{U}$ that satisfy (one of) these definitions. For instance, decomposition-unambiguity of $\mathcal{U}$ yields that for every $(u, v) \in [\![\mathcal{U}]\!]$ there is a unique $q \in A_u$ such that (u, v) is q-normalized and $v \in \mathcal{L}(\mathcal{P}^q)$. If $\mathcal{U}$ is, on the other hand, power-unambiguous then no $f \in F^{\mathcal{P}^q}$ is terminal, and $v \in \mathcal{L}(\mathcal{P}^q)$ for v the period of a q-normalized decomposition (u, v) implies $v^n \in \mathcal{L}(\mathcal{P}^q)$ for all $n \geq 1$. In fact, power-unambiguity of $\mathcal{U}$ even implies that $f_{n+1} \in \delta^{\mathcal{P}^q}(f_n, v)$ for all $n \geq 1$, where f_i denotes the final state reached by the accepting run of $\mathcal{P}^q$ on v^i. To see that this is indeed the case consider, for $n \geq 1$, the word $v^{n\cdot(n+1)}$. Then $\mathcal{P}^q$ has an accepting run π_1 on $v^{n\cdot(n+1)}$ that is in f_n after v^n. Such a π_1 exists since $f_n \in \mathcal{P}^q(v^n) \cap F^{\mathcal{P}^q}$ and, as $\mathcal{U}$ is power-unambiguous, $F^{\mathcal{P}^q} \cap \delta^{\mathcal{P}^q}(f_n, v^{n\cdot(n+1)-n}) = F^{\mathcal{P}^q} \cap \delta^{\mathcal{P}^q}(f_n, (v^n)^n) \neq \emptyset$. Similarly, $\mathcal{P}^q$ has an accepting run π_2 on $v^{n\cdot(n+1)}$ that is in f_{n+1} after v^{n+1}. Because $\mathcal{P}^q$ is unambiguous, π_1 and π_2 coincide. Thus, the unique accepting run of $\mathcal{P}^q$ on $v^{n\cdot(n+1)}$ must be in f_n after v^n and in f_{n+1} after v^{n+1}, which is only possible if $f_{n+1} \in \delta^{\mathcal{P}^q}(f_n, v)$. As a direct consequence, it follows that if $\mathcal{U}$ is power-unambiguous then for all q-normalized $(u, v) \in [\![\mathcal{U}]\!]$ the accepting run of $\mathcal{P}^q$ on v^j is a prefix of the accepting run of $\mathcal{P}^q$ on v^k for all $j < k$.

The next lemma suggest that saturation as well as decomposition-unambiguity and power-unambiguity are natural requirements for unambiguous extensions of saturated FDFA.

▶ **Lemma 4.5.** *Let $\mathcal{F} = (\mathcal{Q}_\mathcal{F}, \{\mathcal{P}^q_\mathcal{F}\}_{q\in\mathcal{Q}_\mathcal{F}})$ be a saturated FDFA. Let $\mathcal{U} = (\mathcal{Q}_\mathcal{U}, \{\mathcal{P}^q_\mathcal{U}\}_{q\in A})$ with $A = \mathcal{Q}_\mathcal{U}$ be the FUFA obtained by interpreting $\mathcal{Q}_\mathcal{F}$ as an NFA and the progress automata $\mathcal{P}^q_\mathcal{F}$ as UFA. Then $\mathcal{U}$ is saturated, decomposition-unambiguous and power-unambiguous.*

If not mentioned otherwise, we from now on assume that all FUFA are saturated, decomposition-unambiguous and power-unambiguous.

4.1 Succinctness of FUFA compared to FDFA and ω-Automata

Similar to FDFA, the *size* of an FUFA $\mathcal{U}$ is $|\mathcal{U}| = (|\mathcal{Q}|, \max_{q\in A} |\mathcal{P}^q|)$. We compare the succinctness of FUFA to other types of ω-automata, starting with saturated FDFA. Like for the comparison of DFA and UFA [34], there are ω-regular languages for which representations as saturated FDFA require exponentially more states than representations as FUFA.

▶ **Proposition 4.6.** *There is a family of ω-regular languages $(L_n)_{n\geq 1}$ over a ternary alphabet such that every L_n can be represented by a saturated, decomposition- and power-unambiguous FUFA of size $(n+2, 2)$, while every saturated FDFA for L_n is of size at least $(2^n, 2)$.*

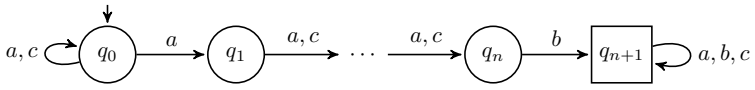

Figure 1 The leading automaton of an FUFA for the family of languages L_n used in the proof of Proposition 4.6. The states in A are depicted as rectangles. Adapted from [4].

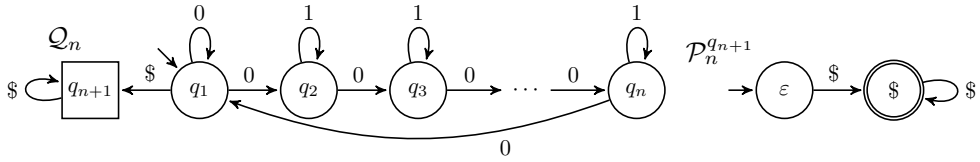

Figure 2 The FUFA $\mathcal{U}_n = (\mathcal{Q}_n, \mathcal{P}_n^{q_{n+1}})$ as used in the proof of Theorem 4.8.

Proof Sketch. Let $\Sigma = \{a, b, c\}$, $L'_n = (a + c)^* a (a + c)^{n-1} b$ and $L_n = L'_n \cdot \Sigma^\omega$. A saturated, decomposition- and power-unambiguous FUFA for L_n of size $(n{+}2, 2)$ has a leading automaton as shown in Figure 1 with $A = \{q_{n+1}\}$, and a progress automaton $\mathcal{P}^{q_{n+1}}$ accepting Σ^+. Since an ω-word is in L_n iff it has a prefix in L'_n, the leading automaton of any saturated FDFA for L_n must "recognize" L'_n. But any DFA for L'_n requires at least 2^n states [4]. ◀

In [4] it is argued that *every* deterministic ω-automaton accepting the language L_n used in the proof of Proposition 4.6 has at least 2^n states. This yields the following corollary.

▶ **Corollary 4.7.** *There is a family of ω-regular languages $(L_n)_{n\geq 1}$ over a ternary alphabet such that every L_n can be represented by a saturated, decomposition- and power-unambiguous FUFA of size $(n{+}2, 2)$, while every deterministic ω-automaton for L_n needs at least 2^n states.*

Next, we compare the succinctness of FUFA to that of unambiguous Büchi automata.

▶ **Theorem 4.8.** *There is a family of ω-regular languages $(L_n)_{n\geq 1}$ over a ternary alphabet such that every L_n can be represented by a saturated, decomposition- and power-unambiguous FUFA of size $(n{+}1, 2)$, while every UBA for L_n has at least $2^n{-}1$ states.*

Proof Sketch. Let $L'_n = (0 + (01^*)^{n-1}0)^*$ and $L_n = L'_n \cdot \$^\omega$. The FUFA $\mathcal{U}_n$ depicted in Figure 2 is saturated, decomposition- and power-unambiguous, and satisfies $\mathcal{L}_\omega(\mathcal{U}_n) = L_n$. Using techniques from [28] we can show that every UBA $\mathcal{B}_n$ for L_n has a substructure $\mathcal{A}_n$ that, if interpreted as a UFA, accepts L'_n. But then $\mathcal{A}_n$ requires at least $2^n{-}1$ states [21]. ◀

Because of the exponential lower bound for the translation from FUFA to UBA, the question arises if an exponential blow-up is also inevitable when translating saturated, decomposition- and power-unambiguous FUFA to NBA. It turns out that this is not the case.

▶ **Lemma 4.9.** *Let $\mathcal{U}$ be a saturated, decomposition- and power-unambiguous FUFA of size $|\mathcal{U}| = (n, k)$. Then there is an NBA $\mathcal{A}$ of size $\mathcal{O}(n^2k^3)$ such that $\mathrm{UP}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{U})$.*

Proof Sketch. We adjust the translation of saturated FDFA to NBA from [1, Thm. 5.8] to FUFA. For $q \in A$ and $f \in F^{\mathcal{P}^q}$, let $M_q = \{v \in \Sigma^* \mid q \in \mathcal{Q}(v)\}$ and $N_{q,f} = \{v \in \Sigma^+ \mid q \in \delta^{\mathcal{Q}}(q, v) \wedge f \in \mathcal{P}^q(v) \wedge f \in \delta^{\mathcal{P}^q}(f, v)\}$, as well as $L = \bigcup_{\{(q,f) \mid q \in A \wedge f \in F^{\mathcal{P}^q}\}} M_q \cdot N_{q,f}^\omega$. Then there is an NBA $\mathcal{A}$ for L of size $\mathcal{O}(n^2k^3)$ with $\mathrm{UP}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{U})$. ◀

As every saturated, decomposition- and power-unambiguous FUFA $\mathcal{U}$ can be translated into an NBA $\mathcal{A}$ with $\mathrm{UP}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{U})$, any such FUFA represents an ω-regular language. To show that also every ω-regular language can be represented by a saturated, decomposition- and power-unambiguous FUFA, we provide a polynomial translation from UBA to FUFA.

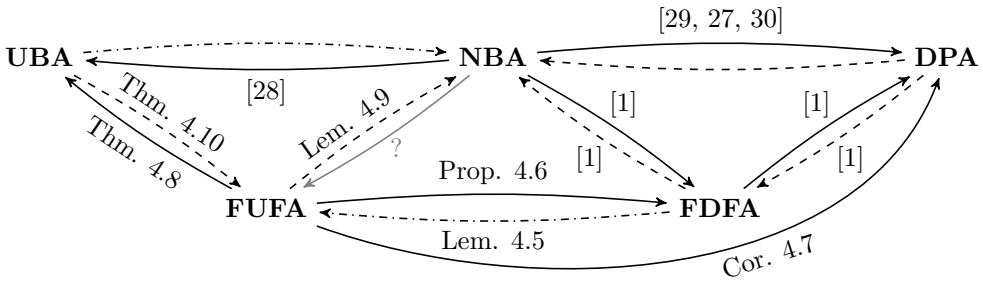

Figure 3 Overview of lower bounds for the state complexity of translations between FUFA, FDFA and different types of ω-automata. Solid arrows indicate an inevitable exponential blow-up, dashed arrows denote polynomial translations, and dash-dotted arrows are used for constant translations.

Given a UBA $\mathcal{A}$, let $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ be an FUFA whose leading automaton $\mathcal{Q}$ has the same structure as $\mathcal{A}$, but with $F^{\mathcal{Q}} = \emptyset$, and let $A = F^{\mathcal{A}}$. Moreover, for $q \in A$, let $\mathcal{P}^q$ be such that $\mathcal{L}(\mathcal{P}^q) = \{v \in \Sigma^+ \mid q \in \delta^{\mathcal{A}}(q, v)\}$. The automaton $\mathcal{P}^q$ is obtained from $\mathcal{A}_q$ by setting $F^{\mathcal{P}^q} = \{q'\}$ for a fresh copy q' of q and defining $\delta^{\mathcal{P}^q}$ such that if $p \neq q'$ then $\delta^{\mathcal{P}^q}(p, a)$ is like $\delta^{\mathcal{A}}(p, a)$, but with transitions to q replaced by transitions to q', while $\delta^{\mathcal{P}^q}(q', a)$ is like $\delta^{\mathcal{A}}(q, a)$, but with transitions to q replaced by self-loops on q'. The resulting FUFA $\mathcal{U}$ is saturated, decomposition- and power-unambiguous, and satisfies $\mathcal{L}(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{U})$.

▶ **Lemma 4.10.** *Let $\mathcal{A}$ be a UBA. Then there is a saturated, decomposition- and power-unambiguous FUFA $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ of size $|\mathcal{U}| = (|\mathcal{A}|, \mathcal{O}(|\mathcal{A}|))$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{U})$.*

The above translation from UBA $\mathcal{A}$ to FUFA $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q\in A})$ does not guarantee that the automaton obtained from $\mathcal{Q}$ by setting $F^{\mathcal{Q}} = A$ is a UFA. To see this let $\mathcal{A}$ have three states q_1, q_2, q_3, with q_1 the initial state, $F^{\mathcal{A}} = \{q_2, q_3\}$, and the only transitions $\delta^{\mathcal{A}}(q_1, a) = \{q_2, q_3\}, \delta^{\mathcal{A}}(q_2, b) = \{q_2\}$ and $\delta^{\mathcal{A}}(q_3, c) = \{q_3\}$. Then the FUFA constructed from $\mathcal{A}$ can move to both $q_2, q_3 \in A$ on a, i.e., $\mathcal{Q}$ is not unambiguous w.r.t. A.

▶ Remark 4.11. By Theorem 4.8 there are ω-regular languages L_n for which saturated, decomposition- and power-unambiguous FUFA of size $(n+2, 2)$ exist, while every UBA for L_n requires at least 2^n-1 states. The FUFA obtained from such a UBA then also has a leading automaton of size at least 2^n-1. Thus, the above translation is not optimal and can produce FUFA that are significantly larger than a minimal FUFA for the same language.

It is well-known that there is a single-exponential translation from LTL to UBA [38]. By Lemma 4.10, translating LTL to saturated, decomposition- and power-unambiguous FUFA is of similar complexity. This is in contrast to the double-exponential lower bound for the translation from LTL to saturated FDFA [16, 10] shown in Proposition 3.8.

▶ **Corollary 4.12.** *There is a single-exponential translation from LTL to saturated, power- and decomposition-unambiguous FUFA.*

The combination of Lemmas 4.9 and 4.10 yields the following result.

▶ **Theorem 4.13.** *Saturated, decomposition- and power-unambiguous FUFA are as expressive as the set of ω-regular languages.*

Figure 3 summarizes the succinctness results for FUFA compared to FDFA and other types of ω-automata. In the figure, **FUFA** stands for the class of saturated, decomposition- and power-unambiguous FUFA, and **FDFA** stands for the class of saturated FDFA. **DPA** stands

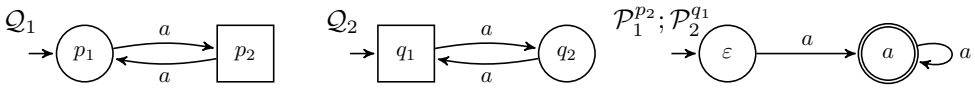

Figure 4 Two FUFA for $\{a^\omega\}$ without a decomposition of a^ω accepted by both.

for the class of *deterministic parity automata* (*DPA*), which is another type of deterministic ω-automata that characterizes the set of ω-regular languages. By first using a polynomial translation from DPA to saturated FDFA [1] and afterwards interpreting the resulting FDFA as an FUFA (Lemma 4.5), we obtain a polynomial translation from DPA to FUFA. Moreover, the exponential lower bound for translating FUFA to DPA does, by Corollary 4.7, also hold for all other types of deterministic ω-automata. The only missing lower bound is that for translating NBA to FUFA. We expect this translation to come with an inevitable exponential blow-up, but leave a formal proof of this claim for future work.

4.2 Boolean Operations and Decision Procedures for FUFA

Let $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q \in A})$ be an FUFA of size (n, k). The *membership problem* for $\mathcal{U}$ and a given $(u, v) \in \Sigma^{*+}$ asks if $(u, v) \in [\![\mathcal{U}]\!]$. An algorithm that solves this problem first computes A_u, then checks for all $q \in A_u$ if (u, v) is q-normalized, and afterwards decides if $v \in \mathcal{L}(\mathcal{P}^q)$ for any such q. All of these operations can be realized by running (slightly modified versions of) $\mathcal{Q}$ on u and some of the progress automata of $\mathcal{U}$ on v, yielding the following result.

▶ **Lemma 4.14.** *Given an FUFA $\mathcal{U}$ and a decomposition $(u, v) \in \Sigma^{*+}$, the membership problem for $\mathcal{U}$ and (u, v) can be solved in time polynomial in the sizes of $\mathcal{U}$, $|u|$ and $|v|$.*

Next, we consider the intersection operation for FUFA, i.e., we discuss how to obtain, from given FUFA $\mathcal{U}_1$ and $\mathcal{U}_2$, an FUFA $\mathcal{H}$ with $[\![\mathcal{H}]\!] = [\![\mathcal{U}_1]\!] \cap [\![\mathcal{U}_2]\!]$. Such an $\mathcal{H}$ can be constructed by taking as leading automaton the parallel product of $\mathcal{Q}_1$ and $\mathcal{Q}_2$, setting $A = A_1 \times A_2$, and defining $\mathcal{P}^{\langle q_1, q_2 \rangle}$ as the parallel product of $\mathcal{P}^{q_1}$ and $\mathcal{P}^{q_2}$ for all $\langle q_1, q_2 \rangle \in A$.

▶ **Lemma 4.15.** *Let $\mathcal{U}_i = (\mathcal{Q}_i, \{\mathcal{P}_i^q\}_{q \in A_i})$ for $i \in \{1, 2\}$ be two FUFA. Then there is an FUFA $\mathcal{H}$ of size $(|\mathcal{Q}_1| \cdot |\mathcal{Q}_2|, \max_{\langle q_1, q_2 \rangle \in A_1 \times A_2} |\mathcal{P}_1^{q_1}| \cdot |\mathcal{P}_2^{q_2}|)$ such that $[\![\mathcal{H}]\!] = [\![\mathcal{U}_1]\!] \cap [\![\mathcal{U}_2]\!]$. Moreover, $\mathcal{H}$ can be constructed such that saturation, decomposition-unambiguity and power-unambiguity are preserved, i.e., such that if both $\mathcal{U}_1$ and $\mathcal{U}_2$ have any of these properties, $\mathcal{H}$ does as well.*

Both the membership problem and the intersection operation described above consider (accepted) *decompositions*. Extending the results to ultimately periodic words $w \in \Sigma^\omega$, i.e., to deciding if $w \in \mathcal{L}(\mathcal{U})$ or constructing an FUFA $\mathcal{H}$ such that $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{U}_1) \cap \mathcal{L}(\mathcal{U}_2)$, is not straightforward. In the case of membership, an algorithm that decides if $w \in \mathcal{L}(\mathcal{U})$ has to find a normalized decomposition (u, v) of w that is accepted by $\mathcal{U}$. Without any information on, e.g., the form of the period of w or the length of its prefix, it is not clear how such a decomposition can be found algorithmically. Regarding intersection, situations in which $[\![\mathcal{U}_1]\!] \cap [\![\mathcal{U}_2]\!] = \emptyset$ but $\mathcal{L}(\mathcal{U}_1) \cap \mathcal{L}(\mathcal{U}_2) \neq \emptyset$ are challenging. An example for this is given in Figure 4, where $\mathcal{L}(\mathcal{U}_1) = \mathcal{L}(\mathcal{U}_2) = \{a^\omega\}$, but $[\![\mathcal{U}_1]\!] \cap [\![\mathcal{U}_2]\!] = \emptyset$. Because acceptance of FUFA is defined w.r.t. decompositions (u, v), it is not clear how to obtain an FUFA for $\mathcal{L}(\mathcal{U}_1) \cap \mathcal{L}(\mathcal{U}_2)$ from $\mathcal{U}_1$ and $\mathcal{U}_2$ in this case. We are not aware of any available methods that solve the corresponding problems for the easier case of (saturated) FDFA.

To conclude this section we discuss the challenges that arise in the search for the precise complexities of other decision problems and Boolean operations for FUFA.

▶ Remark 4.16. For saturated FDFA the operations of union and complementation can be done efficiently [1]. Since the precise complexity of these operations is, to the best of our knowledge, not yet known for UFA [33, 17], we leave the analysis of the complexity of the corresponding operations on FUFA for future work. Furthermore, deciding universality, language containment and language equality can be done efficiently for saturated FDFA [1]. While these problems are in P for UFA [35, 12], their precise complexities in the case of UBA are long-standing open problems [4]. Because of the polynomial-time translation from UBA to saturated, decomposition- and power-unambiguous FUFA (Lemma 4.10), the corresponding problems for (this type of) FUFA are at least as hard as for UBA.

5 Model Checking DTMCs against dFUFA-Specifications

In this section we take the first step towards an efficient probabilistic model checking algorithm that uses saturated, decomposition- and power-unambiguous FUFA for the representation of ω-regular properties. More precisely, we prove the existence of such an algorithm for the subclass of *deterministic* FUFA (*dFUFA*), which are FUFA with a deterministic leading automaton. The notion of dFUFA lies between FDFA and the families of nondeterministic finite automata (FNFA) defined in [7], which require deterministic leading automata but allow progress NFA. Since every FDFA can be interpreted as a dFUFA, the set of saturated, decomposition- and power-unambiguous dFUFA characterizes the set of ω-regular languages. Moreover, the determinism of the leading automaton $\mathcal{Q}$ of a dFUFA yields that $\mathcal{Q}(u)$ is a singleton for every $u \in \Sigma^*$. Therefore, dFUFA are always decomposition-unambiguous, and we usually do not explicitly mention decomposition-unambiguity in the context of dFUFA.

Let $\mathcal{M}$ be a DTMC, $\mathcal{U} = (\mathcal{Q}, \{\mathcal{P}^q\}_{q \in A})$ a saturated and power-unambiguous dFUFA, and $\mathcal{A}$ a DRA with $\mathcal{L}_\omega(\mathcal{U}) = E = \mathcal{L}(\mathcal{A})$. W.l.o.g. we assume that the alphabet of all automata considered in this section equals the set of states S of $\mathcal{M}$, i.e., that $\Sigma = S$. This assumption is common in the literature [6, 4] and ensures that the product of $\mathcal{M}$ with any progress UFA $\mathcal{P}^q$ remains unambiguous if interpreted as an automaton with final states in $S \times F^{\mathcal{P}^q}$.

Our goal is to compute $\Pr^{\mathcal{M}}(E)$ in time polynomial in the sizes of $\mathcal{M}$ and $\mathcal{U}$. We start by observing that, because the leading automaton $\mathcal{Q}$ of $\mathcal{U}$ is deterministic, the residual languages of words reaching the same state in $\mathcal{Q}$ coincide. This is similar to the case of saturated FDFA considered in Lemma 3.6 and can be proved analogously.

▶ **Corollary 5.1.** *Let $x, y \in \Sigma^*$ and let $\mathcal{Q}$ be the leading automaton of a dFUFA $\mathcal{U}$ for E. If $\mathcal{Q}(x) = \mathcal{Q}(y)$ then $Res_x^E = Res_y^E$.*

Because $\mathcal{Q}$ is deterministic, the product $\mathcal{M} \otimes \mathcal{Q}$ is again a DTMC, and so $\mathcal{M}$ almost surely generates a path whose lifting to $\mathcal{M} \otimes \mathcal{Q}$ reaches a BSCC [3]. As in the FDFA-setting, the computation of $\Pr^{\mathcal{M}}(E)$ boils down to the computation of reachability probabilities of specific BSCCs in $\mathcal{M} \otimes \mathcal{Q}$. Recall that, in contrast to FDFA, a (d)FUFA requires the existence of progress automata $\mathcal{P}^q$ only for states $q \in A$. A given BSCC of $\mathcal{M} \otimes \mathcal{Q}$ is, however, not guaranteed to contain any $\langle s, q \rangle$ with $q \in A$. Since the acceptance condition of (d)FUFA is only defined for decompositions $(u, v) \in \Sigma^{*+}$ with $A_u \neq \emptyset$, it is not immediately clear how to handle these BSCCs in a model checking algorithm. Fortunately, however, once a BSCC without a states $\langle s, q \rangle$ with $q \in A$ is entered, the probability of $\mathcal{M}$ to generate a path in E equals 0. Thus, these BSCCs do not play a role in the computation of $\Pr^{\mathcal{M}}(E)$.

▶ **Lemma 5.2.** *Let $u \in \mathrm{Paths}^*(\mathcal{M})$ such that the lifting of u to $\mathcal{M} \otimes \mathcal{Q}$ reaches a BSCC $B_{\mathcal{Q}}$ that does not contain a $\langle s, q \rangle \in B_{\mathcal{Q}}$ with $q \in A$. Then the probability of $\mathcal{M}$ to generate a path that satisfies E and has prefix u equals 0.*

Proof Sketch. Let $\mathcal{A}$ be a DRA with $\mathcal{L}(\mathcal{A}) = E$ and assume that an accepting BSCC $B_{\mathcal{A}}$ is reachable in $\mathcal{M} \otimes \mathcal{A}$ on some extension uv of u. Because $B_{\mathcal{A}}$ is a BSCC there is a z that extends uv and closes a cycle that visits all states of $B_{\mathcal{A}}$. Lemma 3.2 implies $uvz^\omega \in \mathcal{L}(\mathcal{A})$, so also $uvz^\omega \in \mathcal{L}(\mathcal{U})$. But then there must be a $\langle s', q' \rangle \in B_{\mathcal{Q}}$ with $q' \in A$, a contradiction. Hence, every BSCC reachable in $\mathcal{M} \otimes \mathcal{A}$ after u is rejecting, implying the claim. ◀

Let $\langle s, q \rangle$ be a state in a BSCC of $\mathcal{M} \otimes \mathcal{Q}$ with $q \in A$ and let again $I_{\langle s,q \rangle} = \mathcal{M}_s \times \mathcal{Q}_q \times \mathcal{P}^q$. Since $\mathcal{P}^q$ is a UFA, $I_{\langle s,q \rangle}$ is not guaranteed to be a DTMC. Additionally, let $\mathcal{B}(s, q)$ be a UBA with the same structure as $I_{\langle s,q \rangle}$ and $F^{\mathcal{B}(s,q)} = \{\langle s, q, f \rangle \mid f \in F^{\mathcal{P}^q}\}$. The UBA $\mathcal{B}(s, q)$ can be used to decide if BSCCs in $\mathcal{M} \otimes \mathcal{A}$ are accepting.

▶ **Lemma 5.3.** *Let $u \in \mathrm{Paths}^*(\mathcal{M})$ such that the lifting of u to $\mathcal{M} \otimes \mathcal{A}$ reaches a BSCC $B_{\mathcal{A}}$, and the lifting of u to $\mathcal{M} \otimes \mathcal{Q}$ reaches a BSCC $B_{\mathcal{Q}}$. Let $\langle s, q \rangle$ be the state reached on u in $B_{\mathcal{Q}}$, and assume that $q \in A$. Then $B_{\mathcal{A}}$ is accepting iff $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$.*

Proof Sketch. If $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) < 1$ there is $v \in \mathrm{Paths}^*(\mathcal{M}_s)$ such that (1) (u, v) is q-normalized, (2) $v \notin \mathcal{L}(\mathcal{P}^q)$ and (3) v closes a cycle in $B_{\mathcal{A}}$ that visits all states of $B_{\mathcal{A}}$. Combining (1)+(2) yields $uv^\omega \notin E = \mathcal{L}(\mathcal{A})$, so (3) and Lemma 3.2 imply that $B_{\mathcal{A}}$ is rejecting.

Now let $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$. There is $x \in \mathrm{Paths}^*(\mathcal{M}_s)$ closing a loop on $\langle s, q \rangle$ and such that the run of $\mathcal{M} \otimes \mathcal{A}$ on x starting in any $\langle s, a \rangle \in B_{\mathcal{A}}$ visits all states of $B_{\mathcal{A}}$. Because $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$ there is a y such that a $\langle s, q, f \rangle$ with $f \in F^{\mathcal{P}^q}$ is reachable in $I_{\langle s,q \rangle}$ upon xy. Thus, $(u, xy) \in [\![\mathcal{U}]\!]$ and $u(xy)^\omega \in E = \mathcal{L}(\mathcal{A})$, so $B_{\mathcal{A}}$ is accepting by Lemma 3.2. ◀

Similar to Lemma 3.3, the specific choice of $\langle s, q \rangle \in B_{\mathcal{Q}}$ with $q \in A$ does not matter when checking if $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$, as this is either the case for all such $\langle s, q \rangle$ or none of them.

▶ **Lemma 5.4.** *Let $B_{\mathcal{Q}}$ be a BSCC of $\mathcal{M} \otimes \mathcal{Q}$ and let $\langle s, q \rangle, \langle s', q' \rangle \in B_{\mathcal{Q}}$ with $q, q' \in A$. If $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$ then $\Pr^{\mathcal{M}_{s'}}(\mathcal{L}(\mathcal{B}(s', q'))) = 1$.*

Proof Sketch. Let u be such that its lifting to $\mathcal{M} \otimes \mathcal{Q}$ reaches $\langle s', q' \rangle$, while its lifting to $\mathcal{M} \otimes \mathcal{A}$ reaches a BSCC $B_{\mathcal{A}}$. Let v be an extension of u such that, upon uv, $\mathcal{M} \otimes \mathcal{Q}$ reaches $\langle s, q \rangle$. Since $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$, $B_{\mathcal{A}}$ is accepting by Lemma 5.3, and because $B_{\mathcal{A}}$ was already reached after u, the same lemma yields that $\Pr^{\mathcal{M}_{s'}}(\mathcal{L}(\mathcal{B}(s', q'))) = 1$. ◀

A BSCC B of $\mathcal{M} \otimes \mathcal{Q}$ is *good* if it contains a state $\langle s, q \rangle$ with $q \in A$ and such that $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$. Otherwise, B is *bad*. The classification of BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ into good and bad BSCCs is the key step of the algorithm. This is due to the following lemma.

▶ **Lemma 5.5.** *The paths of $\mathcal{M}$ whose lifting to $\mathcal{M} \otimes \mathcal{Q}$ reach a good BSCC almost surely satisfy E, and those whose lifting to $\mathcal{M} \otimes \mathcal{Q}$ reach a bad BSCC almost surely violate E.*

Proof Sketch. Let the lifting of $u \in \mathrm{Paths}^*(\mathcal{M})$ to $\mathcal{M} \otimes \mathcal{Q}$ reach a BSCC $B_{\mathcal{Q}}$. If $B_{\mathcal{Q}}$ is good then it follows from Lemma 5.3 that all BSCCs reachable in $\mathcal{M} \otimes \mathcal{A}$ after u are accepting. Otherwise, if $B_{\mathcal{Q}}$ is bad, then the combination of Lemmas 5.2 and 5.3 implies that all BSCCs reachable in $\mathcal{M} \otimes \mathcal{A}$ after u are rejecting. ◀

Since $\mathcal{M} \otimes \mathcal{Q}$ is a DTMC, almost surely $\mathcal{M}$ generates a path whose lifting to $\mathcal{M} \otimes \mathcal{Q}$ enters a BSCC $B_{\mathcal{Q}}$. It follows from Lemma 5.5 that if $B_{\mathcal{Q}}$ is good then almost surely a word in E will be generated, while if $B_{\mathcal{Q}}$ is bad this will almost surely not be the case. Hence, $\Pr^{\mathcal{M}}(E)$ equals the probability of reaching a good BSCC in $\mathcal{M} \otimes \mathcal{Q}$. This is formalized in the following proposition, where *goodBSCC* contains all good BSCCs of $\mathcal{M} \otimes \mathcal{Q}$.

▶ **Proposition 5.6.** $\Pr^{\mathcal{M}}(E) = \Pr^{\mathcal{M} \otimes \mathcal{Q}}(\Diamond \mathit{goodBSCC})$.

The probability of a DTMC $\mathcal{M}$ to generate a path satisfying an ω-regular property E represented by a saturated and power-unambiguous dFUFA can thus be computed as follows:

1. Compute the set of BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ that contain a state $\langle s, q \rangle$ with $q \in A$.
2. For each such BSCC B decide if B is good by picking any $\langle s, q \rangle \in B$ with $q \in A$ and checking if $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$. If this is the case, add B to $goodBSCC$.
3. Compute $\Pr^{\mathcal{M}}(E) = \Pr^{\mathcal{M} \otimes \mathcal{Q}}(\Diamond goodBSCC)$.

Correctness follows from Proposition 5.6 and the fact that either $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$ for all $\langle s, q \rangle$ with $q \in A$ in a BSCC of $\mathcal{M} \otimes \mathcal{Q}$, or none of them (Lemma 5.4). Furthermore, computing the BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ and the value $\Pr^{\mathcal{M} \otimes \mathcal{Q}}(\Diamond goodBSCC)$ is possible in time polynomial in the sizes of $\mathcal{M}$ and $\mathcal{Q}$ [3], and checking if $\Pr^{\mathcal{M}_s}(\mathcal{L}(\mathcal{B}(s, q))) = 1$ can be done in time polynomial in the sizes of $\mathcal{M}$ and $\mathcal{B}(s, q)$ [4]. Since the latter is necessary for at most $(|\mathcal{M}| \cdot |\mathcal{Q}|)$-many BSCCs, the overall runtime of the algorithm is polynomial in $|\mathcal{M}|$ and $|\mathcal{U}|$.

This yields an extension of Theorem 3.1 to saturated and power-unambiguous dFUFA.

▶ **Theorem 5.7.** *Let $\mathcal{M}$ be a DTMC and $\mathcal{U}$ a saturated and power-unambiguous dFUFA with $\mathcal{L}_\omega(\mathcal{U}) = E$. Then $\Pr^{\mathcal{M}}(E)$ can be computed in time polynomial in the sizes of $\mathcal{M}$ and $\mathcal{U}$.*

We conclude with a discussion on why replacing the UBA-methods used in the algorithm with approaches for model checking DTMCs against UFA-specifications is challenging.

The probabilistic model checking problem for a DTMC $\mathcal{M}$ against a UFA $\mathcal{A}$ asks to compute $\Pr^{\mathcal{M}}(\mathcal{L}(\mathcal{A}) \cdot \Sigma^\omega)$. This probability can be computed in polynomial time by solving a specific system of linear equations [6], provided that (1) the language of the UFA $\mathcal{A}$ is prefix-free or, more generally, that (2) for all $x, xy \in \mathcal{L}(\mathcal{A})$ the accepting run of $\mathcal{A}$ on x is a prefix of the accepting run of $\mathcal{A}$ on xy. For more details, see Appendix A. However, neither (1) nor (2) are guaranteed by the translation from UBA to FUFA given in Section 4, and so it is not clear how to construct an FUFA for a given ω-regular specification whose progress automata are suitable for the application of available UFA-based probabilistic model checking algorithms. To the best of our knowledge, it is not known whether model checking DTMCs against general UFA-specifications is possible in polynomial time. Moreover, the characterization of BSCCs of $\mathcal{M} \otimes \mathcal{Q}$ used in the model checking procedure proposed in this section depends on the almost sure generation of certain *infinite* paths of $\mathcal{M}$. Therefore, even if polynomial-time model checking algorithms for DTMCs against general UFA-specifications would be at our disposal, adjusting the algorithm such that it uses UFA- instead of UBA-techniques is non-trivial.

Lastly, we observe that the probabilistic model checking problem for DTMCs against FUFA-specifications is at least as hard as the corresponding problem for UFA-specifications. To see this, let $\mathcal{A}$ be a UFA over Σ. We can construct an FUFA $\mathcal{U}$ that accepts $\mathcal{L}(\mathcal{A}) \cdot \Sigma^\omega$ by choosing $\mathcal{Q}$ to be like $\mathcal{A}$, but with a fresh state f that is reachable on the full alphabet from every final state of $\mathcal{A}$ and closes a self-loop on Σ, setting $A = \{f\}$, and choosing $\mathcal{P}^f$ such that $\mathcal{L}(\mathcal{P}^f) = \Sigma^+$. Hence, a probabilistic model checking algorithm for DTMCs against FUFA-specifications would also yield a procedure for the verification of DTMCs against general UFA-specifications.

6 Conclusion

This paper presented the first polynomial-time probabilistic model checking algorithm for DTMCs against ω-regular specifications represented as saturated FDFA. Furthermore, FUFA were introduced as a variant of FDFA that allows nondeterministic leading automata and unambiguous progress automata. Saturated, decomposition- and power-unambiguous FUFA characterize the ω-regular languages and can be exponentially more succinct than saturated

FDFA and UBA. Additionally, there is a single-exponential translation from LTL to FUFA, which is in contrast to a double-exponential lower bound for the translation from LTL to FDFA. An extension of the probabilistic model checking algorithm for DTMCs against FDFA to dFUFA, i.e., to FUFA with a deterministic leading automaton, was also provided.

Finding a polynomial-time algorithm that solves the probabilistic model checking problem for DTMCs against general FUFA-specifications is challenging. In particular, the fact that the leading automaton of an FUFA can be ambiguous is both a blessing and a curse. While this ambiguity is key in the proof of Theorem 4.8 to obtain an exponential lower bound for the translation from FUFA to UBA, it also increases the complexity of algorithmic questions like probabilistic model checking for FUFA. In particular, the known PSPACE-hardness results for model checking DTMCs against NBA-specifications [37] suggest that there might not be an efficient probabilistic model checking algorithm against FUFA-specifications at all.

An analysis of the complexity of the probabilistic model checking problem against general FUFA-specifications is important future work. Moreover, the exact cost of translating NBA into FUFA and the question if dFUFA can be exponentially more succinct than saturated FDFA are open. An experimental evaluation of the proposed algorithms is also in order.

References

1 Dana Angluin, Udi Boker, and Dana Fisman. Families of DFAs as acceptors of ω-regular languages. *Logical Methods in Computer Science*, Volume 14, Issue 1, February 2018. doi:10.23638/LMCS-14(1:15)2018.

2 Dana Angluin and Dana Fisman. Learning regular omega languages. *Theoretical Computer Science*, 650:57–72, October 2016. doi:10.1016/j.tcs.2016.07.031.

3 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

4 Christel Baier, Stefan Kiefer, Joachim Klein, David Müller, and James Worrell. Markov chains and unambiguous automata. *Journal of Computer and System Sciences*, 136:113–134, September 2023. doi:10.1016/j.jcss.2023.03.005.

5 Christel Baier, Sascha Klüppelholz, and Timm Spork. Probabilistic model checking via families of deterministic and unambiguous finite automata, 2026. doi:10.48550/arXiv.2606.21976.

6 Michael Benedikt, Rastislav Lenhardt, and James Worrell. Model checking Markov chains against unambiguous Büchi automata. *CoRR*, abs/1405.4560, 2014. doi:10.48550/arXiv.1405.4560.

7 León Bohn, Yong Li, Christof Löding, and Sven Schewe. Saturation problems for families of automata. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 146:1–146:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2025.146.

8 J. Richard Büchi. Symposium on decision problems: On a decision method in restricted second order arithmetic. In Ernest Nagel, Patrick Suppes, and Alfred Tarski, editors, *Logic, Methodology and Philosophy of Science*, volume 44 of *Studies in Logic and the Foundations of Mathematics*, pages 1–11. Elsevier, 1966. doi:10.1016/S0049-237X(09)70564-6.

9 Hugues Calbrix, Maurice Nivat, and Andreas Podelski. Ultimately periodic words of rational ω-languages. In Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David Schmidt, editors, *Mathematical Foundations of Programming Semantics (MFPS 1993)*, volume 802 of *Lecture Notes in Computer Science (LNCS)*, pages 554–566, Berlin, Heidelberg, 1994. Springer. doi:10.1007/3-540-58027-1_27.

10 Balder ten Cate, Dana Fisman, Roi Ohayon, and Patrik Sestic. Characterizing LTL Formulas by Examples, April 2026. doi:10.48550/arXiv.2604.22097.

11 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981. doi:10.1145/322234.322243.

12 Thomas Colcombet. Unambiguity in automata theory. In Jeffrey Shallit and Alexander Okhotin, editors, *Descriptional Complexity of Formal Systems (DCFS 2015)*, volume 9118 of *Lecture Notes in Computer Science (LNCS)*, pages 3–18. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-19225-3_1.

13 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, July 1995. doi:10.1145/210332.210339.

14 Javier Esparza, Jan Křetínský, Jean-François Raskin, and Salomon Sickert. From LTL and limit-deterministic Büchi automata to deterministic parity automata. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2017)*, volume 10205 of *Lecture Notes in Computer Science (LNCS)*, pages 426–442, Berlin, Heidelberg, 2017. Springer. doi:10.1007/978-3-662-54577-5_25.

15 Javier Esparza, Jan Křetínský, and Salomon Sickert. A unified translation of linear temporal logic to ω-automata. *J. ACM*, 67(6), October 2020. doi:10.1145/3417995.

16 Dana Fisman. Personal communication, 2024.

17 Mika Göös, Stefan Kiefer, and Weiqiang Yuan. Lower bounds for unambiguous automata via communication complexity. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 126:1–126:13, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.126.

18 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science (LNCS)*. Springer, Berlin, Heidelberg, 2002. doi:10.1007/3-540-36387-4.

19 John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Undergraduate Texts in Mathematics (UTM). Springer New York, 1976.

20 Orna Kupferman and Moshe Y. Vardi. From linear time to branching time. *ACM Trans. Comput. Logic*, 6(2):273–294, April 2005. doi:10.1145/1055686.1055689.

21 Hing Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM Journal on Computing*, 27(4):1073–1082, August 1998. doi:10.1137/S0097539793252092.

22 Hing Leung. Descriptional complexity of NFA of different ambiguity. *International Journal of Foundations of Computer Science*, 16:975–984, January 2004. doi:10.1142/S0129054105003418.

23 Jianlin Li. Translating LTL to FDFA and FDFA model checking. Bachelor's thesis. Nanjing University of Aeronautics and Astronautics, 2018.

24 Yong Li, Yu-Fang Chen, Lijun Zhang, and Depeng Liu. A novel learning algorithm for Büchi automata based on family of DFAs and classification trees. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2017)*, volume 10205 of *Lecture Notes in Computer Science (LNCS)*, pages 208–226, Berlin, Heidelberg, 2017. Springer. doi:10.1007/978-3-662-54577-5_12.

25 Yong Li, Sven Schewe, and Qiyi Tang. A novel family of finite automata for recognizing and learning ω-regular languages. In Étienne André and Jun Sun, editors, *Automated Technology for Verification and Analysis*, volume 14215 of *Lecture Notes in Computer Science (LNCS)*, pages 53–73, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-45329-8_3.

26 Yong Li, Yih-Kuen Tsay, Andrea Turrini, Moshe Y. Vardi, and Lijun Zhang. Congruence relations for Büchi automata. In Marieke Huisman, Corina Păsăreanu, and Naijun Zhan, editors, *Formal Methods (FM 2021)*, volume 13047 of *Lecture Notes in Computer Science (LNCS)*, pages 465–482, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-90870-6_25.

27 Christof Löding. Optimal bounds for transformations of ω-automata. In C. Pandu Rangan, V. Raman, and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1999)*, volume 1738 of *Lecture Notes in Computer Science (LNCS)*, pages 97–109, Berlin, Heidelberg, 1999. Springer. doi:10.1007/3-540-46691-6_8.

28 Christof Löding and Anton Pirogov. On finitely ambiguous Büchi automata. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory (DLT 2018)*, volume 11088 of *Lecture Notes in Computer Science (LNCS)*, pages 503–515, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-98654-8_41.

29 Max Michel. Complementation is more difficult with automata on infinite words. *CNET, Paris*, 15, 1988.

30 N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 255–264, 2006. doi:10.1109/LICS.2006.28.

31 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1989)*, pages 179–190, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/75277.75293.

32 Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57, USA, 1977. IEEE Computer Society. doi:10.1109/SFCS.1977.32.

33 Mikhail Raskin. A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 138:1–138:11, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2018.138.

34 Erik Meineche Schmidt. *Succinctness of Descriptions of Context-Free, Regular, and Finite Languages*. PhD thesis, University of Aarhus, 1978.

35 R. E. Stearns and H. B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985. doi:10.1137/0214044.

36 Wolfgang Thomas. Facets of synthesis: Revisiting Church's problem. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures (FoSSaCS 2009)*, volume 5504 of *Lecture Notes in Computer Science (LNCS)*, pages 1–14, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-00596-1_1.

37 Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *26th Annual Symposium on Foundations of Computer Science (FOCS 1985)*, pages 327–338, 1985. doi:10.1109/SFCS.1985.12.

38 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science (LICS 1986)*, pages 332–344. IEEE Computer Society Press, June 1986.

39 Thomas Wilke. ω-Automata. *CoRR*, abs/1609.03062, 2016. doi:10.48550/arXiv.1609.03062.

A Model Checking DTMCs Against UFA with Prefix-dependent Languages

Let $\mathcal{M}$ be a DTMC and $\mathcal{A}$ a UFA. Moreover, let S be the set of states of $\mathcal{M}$, let Q be the set of vertices of $\mathcal{A}$, and let $\mathcal{L}(\mathcal{A})$ be the language accepted by $\mathcal{A}$. W.l.o.g. we assume that the alphabet of $\mathcal{A}$ coincides with S and that $\ell(s) = \{s\}$ for all $s \in S$, which can always be achieved by existential renaming [6]. The model checking procedure proposed in [6] to compute the probability that $\mathcal{M}$ generates a path with a prefix accepted by $\mathcal{A}$, i.e., to compute $\Pr^{\mathcal{M}}(\mathcal{L}(\mathcal{A}) \cdot S^{\omega})$, works as follows:

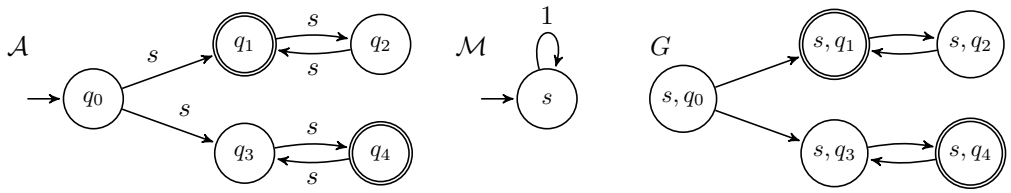

Figure 5 A UFA $\mathcal{A}$ for the language s^+ (left), a DTMC $\mathcal{M}$ (middle) and the directed graph G constructed according to the definitions given in [6] (right).

1. Construct a directed graph G with vertices in $V = S \times Q$ and where there is an edge from (s, q) to (s', q') iff $P(s, s') > 0$ and $q' \in \delta^{\mathcal{A}}(q, s')$.
2. Set $V^{acc} = \{(s, q) \mid q \in F^{\mathcal{A}}\}$, V^{dead} the set of all states in V that cannot reach V^{acc}, and $V^{?} = V \setminus (V^{acc} \cup V^{dead})$.
3. Solve the following system of linear equations:

$$\begin{aligned} x_{s,q} &= 1 && \text{if } (s,q) \in V^{acc} \\ x_{s,q} &= 0 && \text{if } (s,q) \in V^{dead} \\ x_{s,q} &= \sum_{s' \in S} \sum_{q' \in \delta^{\mathcal{A}}(q,s')} P(s,s') \cdot x_{s',q'} && \text{if } (s,q) \in V^{?} \end{aligned}$$

4. Return $\Pr^{\mathcal{M}}(\mathcal{L}(\mathcal{A}) \cdot S^{\omega}) = x_{s'_{init}, q_{init}}$, where s'_{init} is a (potentially fresh) state in $\mathcal{M}$ that transitions to the initial state of $\mathcal{M}$ with probability 1, and q_{init} is the initial state of $\mathcal{A}$.

Now consider the UFA $\mathcal{A}$ on the left of Figure 5 and the DTMC $\mathcal{M}$ in the middle of the figure. The (reachable fragment of the) directed graph G obtained from $\mathcal{M}$ and $\mathcal{A}$ is shown on the right of the figure. States represented as double-circles in G correspond to states in V^{acc}, and $V^{dead} = \emptyset$. Solving the system of linear equations from above yields

$$\begin{aligned} x_{s,q_0} &= \sum_{s' \in S} \sum_{q' \in \delta^{\mathcal{A}}(q_0,s')} P(s,s') \cdot x_{s',q'} = P(s,s) \cdot \underbrace{x_{s,q_1}}_{=1 \text{ as } (s,q_1) \in V^{acc}} + P(s,s) \cdot x_{s,q_3} \\ &= 1 + x_{s,q_3} = 1 + \sum_{s' \in S} \sum_{q' \in \delta^{\mathcal{A}}(q_3,s')} P(s,s') \cdot x_{s',q'} = 1 + \underbrace{x_{s,q_4}}_{=1 \text{ as } (s,q_4) \in V^{acc}} = 2. \end{aligned}$$

Since $\Pr^{\mathcal{M}}(\mathcal{L}(\mathcal{A}) \cdot S^{\omega})$ is a probability, it cannot coincide with $x_{s,q_0} = 2$ in this case.

The reason for the erroneous computation of x_{s,q_0} in the above example is that the system of linear equations counts paths of $\mathcal{M}$ starting with s multiple times. More precisely, in the directed graph G as in Figure 5, any infinite path that starts with s is already "accepted" when moving to $\langle s, q_1 \rangle \in V^{acc}$. However, there are two outgoing edges of q_0 on s, yielding a second transition of G from $\langle s, q_0 \rangle$ to $\langle s, q_3 \rangle$. To obtain x_{s,q_0}, the system of linear equations sums up the values of x_{s,q_1} and x_{s,q_3}, which both equal 1 since the DTMC $\mathcal{M}_s$ almost surely generates a path accepted by the automata $\mathcal{A}_{q_1}$ and $\mathcal{A}_{q_3}$. Hence, the probability of $\mathcal{M}$ to generate a path starting with s is counted twice, yielding $x_{s,q_0} = x_{s,q_1} + x_{s,q_3} = 2$.

This behavior cannot occur if the language of $\mathcal{A}$ is prefix-free or, more generally, if for all $x, xy \in \mathcal{L}(\mathcal{A})$ the accepting run of $\mathcal{A}$ on x is a prefix of the accepting run of $\mathcal{A}$ on xy. Given that (one of) these requirements are (is) satisfied, the algorithm of [6] correctly computes the desired probabilities. We are not aware of any results that allow the computation of satisfaction probabilities of DTMCs against arbitrary UFA-specifications in polynomial time.