Markov Chains and Unambiguous Büchi Automata

Christel Baier[1], Stefan Kiefer[2], Joachim Klein[1],
Sascha Klüppelholz[1], David Müller[1], and James Worrell[2]

[1] Technische Universität Dresden, Germany *
[2] University of Oxford, Great Britain **

Abstract. Unambiguous automata, i.e., nondeterministic automata with the restriction of having at most one accepting run over a word, have the potential to be used instead of deterministic automata in settings where nondeterministic automata can not be applied in general. In this paper, we provide a polynomially time-bounded algorithm for probabilistic model checking of discrete-time Markov chains against unambiguous Büchi automata specifications and report on our implementation and experiments.

1 Introduction

Unambiguity is a widely studied generalization of determinism with many important applications in automata-theoretic approaches, see e.g. [12,13]. A nondeterministic automaton is said to be unambiguous if each word has at most one accepting run. In this paper we consider unambiguous Büchi automata (UBA) over infinite words. Not only are UBA as expressive as the full class of nondeterministic Büchi automata (NBA) [2], they can also be exponentially more succinct than deterministic automata. For example, the language "eventually b occurs and a appears k steps before the first b" over the alphabet $\{a, b, c\}$ is recognizable by a UBA with $k+1$ states (see the UBA on the left of Fig. 1), while a deterministic automaton requires at least 2^k states, regardless of the acceptance condition, as it needs to store the positions of the a's among the last k input symbols. Languages of this type arise in a number of contexts, e.g., absence of unsolicited response in a communication protocol – if a message is received, then it has been sent in the recent past.

Furthermore, the NBA for linear temporal logic (LTL) formulas obtained by applying the classical closure algorithm of [40,39] are unambiguous. The

* The authors are supported by the DFG through the Collaborative Research Center SFB 912 – HAEC, the Excellence Initiative by the German Federal and State Governments (cluster of excellence cfAED and Institutional Strategy), the Research Training Groups QuantLA (GRK 1763) and RoSI (GRK 1907), and the DFG/NWO-project ROCKS.

** Kiefer is supported by a University Research Fellowship of the Royal Society. Worrell is supported by EPSRC grant EP/M012298/1.

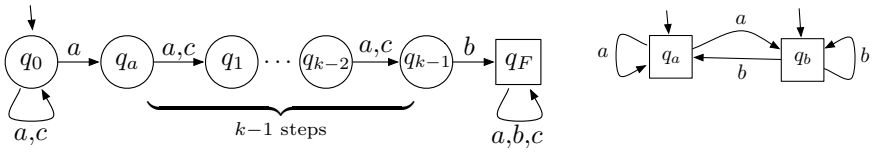

Fig. 1. Two UBA (where final states are depicted as boxes)

generated automata moreover enjoy the separation property: the languages of the states are pairwise disjoint. Thus, while the generation of deterministic ω-automata from LTL formulas involves a double exponential blow-up in the worst case, the translation of LTL formulas into separated UBA incurs only a single exponential blow-up. This fact has been observed by several authors, see e.g. [16,34], and recently adapted for LTL with step parameters [41,11].

These nice properties make UBA a potentially attractive alternative to deterministic ω-automata in those applications for which general nondeterministic automata are not suitable. However reasoning about UBA is surprisingly difficult. While many decision problems for unambiguous finite automata (UFA) are known to be solvable in polynomial time [37], the complexity of several fundamental problems for unambiguous automata over infinite words is unknown. This, for instance, applies to the universality problem, which is known to be in P for deterministic Büchi automata (DBA) and PSPACE-complete for NBA. However, the complexity of the universality problem for UBA is a long-standing open problem. Polynomial-time solutions are only known for separated UBA and other subclasses of UBA [9,26].

In the context of probabilistic model checking, UFA provide an elegant approach to compute the probability for a regular safety or co-safety property in finite-state Markov chains [7]. The use of separated UBA for a single exponential-time algorithm that computes the probability for an LTL formula in a Markov chain has been presented in [16]. However, separation is a rather strong condition and non-separated UBA (and even DBA) can be exponentially more succinct than separated UBA, see [9]. This motivates the design of algorithms that operate with general UBA rather than the subclass of separated UBA. Algorithms for the generation of (possibly non-separated) UBA from LTL formulas that are more compact than the separated UBA generated by the classical closure-algorithm have been realized in the tool `Tulip` [33,32] and the automata library `SPOT` [17].

The main theoretical contribution of this paper is a polynomial-time algorithm to compute the probability measure $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$ of the set of infinite paths generated by a finite-state Markov chain $\mathcal{M}$ that satisfy an ω-regular property given by a (not necessarily separated) UBA $\mathcal{U}$. The existence of such an algorithm has previously been claimed in [7,6,33] (see also [32]). However these previous works share a common fundamental error. Specifically they rely on the claim that if $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U})) > 0$ then there exists a state s of the Markov chain $\mathcal{M}$ and a state q of the automaton $\mathcal{U}$ such that q accepts almost all tra-

jectories emanating from s (see [7, Lemma 7.1], [6, Theorem 2], and [33, Section 3.3.1]). While this claim is true in case $\mathcal{U}$ is deterministic [14], it need not hold when $\mathcal{U}$ is merely unambiguous. Indeed, as we explain in Remark 3.1, a counterexample is obtained by taking $\mathcal{U}$ to be the automaton on the right in Fig. 1 and $\mathcal{M}$ the Markov chain that generates the uniform distribution on $\{a, b\}^\omega$. The long version of this paper [4] gives a more detailed analysis of the issue, decribing precisely the nature of the errors in the proofs of [7,33,6]. To the best of our knowledge these errors are not easily fixable, and the present paper takes a substantially different approach.

Our algorithm involves a two-phase method that first analyzes the strongly connected components (SCCs) of a graph obtained from the product of $\mathcal{M}$ and $\mathcal{U}$, and then computes the value $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$ using linear equation systems. The main challenge is the treatment of the individual SCCs. For a given SCC we have an equation system comprising a single variable and equation for each vertex (s, q), with s a state of $\mathcal{M}$ and q a state of $\mathcal{U}$. We use results in the spectral theory of non-negative matrices to argue that this equation system has a non-zero solution just in case the SCC makes a non-zero contribution to $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$. In order to compute the exact value of $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$ the key idea is to introduce an additional normalization equation. To obtain the latter we identify a pair (s, R), where s is a state of the Markov chain $\mathcal{M}$ and R a set of states of automaton $\mathcal{U}$ such that almost all paths starting in s have an accepting run in $\mathcal{U}$ when the states in R are declared to be initial. The crux of establishing a polynomial bound on the running time of our algorithm is to find such a pair (s, R) efficiently (in particular, without determinizing $\mathcal{U}$) by exploiting structural properties of unambiguous automata.

As a consequence of our main result, we obtain that the *almost universality* problem for UBA, which can be seen as probabilistic variant of the universality problem for UBA and which asks whether a given UBA accepts almost all infinite words, is solvable in polynomial time.

The second contribution of the paper is an implementation of the new algorithm as an extension of the model checker `PRISM`, using the automata library `SPOT` [17] for the generation of UBA from LTL formulas and the `COLT` library [25] for various linear algebra algorithms. We evaluate our approach using the bounded retransmission protocol case study from the PRISM benchmark suite [31] as well as specific aspects of our algorithm using particularily "challenging" UBA.

Outline. Section 2 summarizes our notations for Büchi automata and Markov chains. The theoretical contribution will be presented in Section 3. Section 4 reports on the implementation and experimental results. Section 5 contains concluding remarks. The full version of this paper [4] contains an appendix with the counterexamples for the previous approaches, proofs and further details on the implementation and results of experimental studies. Further information is available on the website [1] as well.

2 Preliminaries

We suppose the reader to be familiar with the basic notions of ω-automata and Markov chains, see e.g. [22,29]. In what follows, we provide a brief summary of our notations for languages and the uniform probability measure on infinite words, Büchi automata as well as Markov chains.

Prefixes, cylinder sets and uniform probability measure for infinite words. Throughout the document, we suppose Σ is a finite alphabet with two or more elements. If $w = a_1\, a_2\, a_3 \ldots \in \Sigma^\omega$ is an infinite word then $\mathit{Pref}(w)$ denotes the set of finite prefixes of w, i.e., $\mathit{Pref}(w)$ consists of the empty word and all finite words $a_1\, a_2 \ldots a_n$ where $n \geqslant 1$. Given a finite word $x = a_1\, a_2 \ldots a_n \in \Sigma^*$, the cylinder set of x, denoted $\mathit{Cyl}(x)$, is the set of infinite words $w \in \Sigma^\omega$ such that $x \in \mathit{Pref}(w)$. The set Σ^ω of infinite words over Σ is supposed to be equipped with the σ-algebra generated by the cylinder sets of the finite words and the probability measure given by $\Pr\big(\, \mathit{Cyl}(a_1\, a_2 \ldots a_n)\,\big) \;=\; 1/|\Sigma|^n$ where $a_1, \ldots, a_n \in \Sigma$. Note that all ω-regular languages over Σ are measurable. We often make use of the following lemma (see [4] for its proof):

Lemma 2.1. *If $L \subseteq \Sigma^\omega$ is ω-regular and $\Pr(L) > 0$ then there exists $x \in \Sigma^*$ such that $\Pr\big\{\, w \in \Sigma^\omega \,:\, xw \in L \,\big\} = 1$.*

Büchi automata. A nondeterministic Büchi automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, Σ denotes the alphabet, $\delta : Q \times \Sigma \to 2^Q$ denotes the transition function, and F is a set of accepting states. We extend the transition function $\delta : 2^Q \times \Sigma \to 2^Q$ in the standard way for subsets of Q and finite words over Σ. Given states $q, p \in Q$ and a finite word $x = a_1\, a_2 \ldots a_n \in \Sigma^*$ then a run for x from q to p is a sequence $q_0\, q_1 \ldots q_n \in Q^+$ with $q_0 = q$, $q_n = p$ and $q_{i+1} \in \delta(q_i, a_{i+1})$ for $0 \leqslant i < n$. A run in $\mathcal{A}$ for an infinite word $w = a_1\, a_2\, a_3 \ldots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0\, q_1 \ldots \in Q^\omega$ such that $q_{i+1} \in \delta(q_i, a_{i+1})$ for all $i \in \mathbb{N}$ and $q_0 \in Q_0$. Run ρ is called accepting, if $q_i \in F$ for infinitely many $i \in \mathbb{N}$. The language $\mathcal{L}_\omega(\mathcal{A})$ of accepted words consists of all infinite words $w \in \Sigma^\omega$ that have at least one accepting run. If $R \subseteq Q$ then $\mathcal{A}[R]$ denotes the automaton $\mathcal{A}$ with R as set of initial states. For $q \in Q$, $\mathcal{A}[q] = \mathcal{A}[\{q\}]$. If $\mathcal{A}$ is understood from the context, then we write $\mathcal{L}_\omega(R)$ rather than $\mathcal{L}_\omega(\,\mathcal{A}[R]\,)$ and $\mathcal{L}_\omega(q)$ rather than $\mathcal{L}_\omega(\,\mathcal{A}[q]\,)$. $\mathcal{A}$ is called deterministic if Q_0 is a singleton and $|\delta(q,a)| \leqslant 1$ for all states q and symbols $a \in \Sigma$ and unambiguous if each word $w \in \Sigma^\omega$ has at most one accepting run in $\mathcal{A}$. Clearly, each deterministic automaton is unambiguous. We use the shortform notations NBA, DBA and UBA for nondeterministic, deterministic and unambiguous Büchi automata, respectively.

Markov chains. In this paper we only consider finite-state discrete-time Markov chains. Formally, a Markov chain is a triple $\mathcal{M} = (S, P, \iota)$ where S is a finite set of states, $P : S \times S \to [0,1]$ is the transition probability function satisfying $\sum_{s' \in S} P(s,s') = 1$ for all states $s \in S$ and ι an initial distribution on S. We write

$\Pr^{\mathcal{M}}$ to denote the standard probability measure on the infinite paths of $\mathcal{M}$. For $s \in S$, the notation $\Pr_s^{\mathcal{M}}$ will be used for $\Pr^{\mathcal{M}_s}$ where $\mathcal{M}_s = (S, P, \mathrm{Dirac}[s])$ and $\mathrm{Dirac}[s] : S \to [0,1]$ denotes the Dirac distribution that assigns probability 1 to state s and 0 to all other states. If $L \subseteq S^\omega$ is measurable then $\Pr^{\mathcal{M}}(L)$ is a shortform notation for the probability for $\mathcal{M}$ to generate an infinite path π with $\pi \in L$.

Occasionally, we also consider Markov chains with transition labels in some alphabet Σ. These are defined as triples $\mathcal{M} = (S, P, \iota)$ where S and ι are as above and the transition probability function is of the type $P : S \times \Sigma \times S \to [0,1]$ such that $\sum_{(a,s') \in \Sigma \times S} P(s,a,s') = 1$ for all states $s \in S$. If $L \subseteq \Sigma^\omega$ is measurable then $\Pr^{\mathcal{M}}(L)$ denotes the probability measure of the set of infinite paths π where the projection to the transition labels constitutes a word in L. Furthermore, if $\mathcal{M}[\Sigma] = (S, P, \iota)$ is a transition-labeled Markov chain where $S = \{s\}$ is a singleton and $P(s,a,s) = 1/|\Sigma|$ for all symbols $a \in \Sigma$, then $\Pr^{\mathcal{M}[\Sigma]}(L) = \Pr(L)$ for all measurable languages L.

3 Analysis of Markov chains against UBA-specifications

The task of the *probabilistic model-checking problem* for a given Markov chain $\mathcal{M}$ and NBA $\mathcal{A}$ is to compute $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{A}))$ where $\mathcal{M}$ is either a plain Markov chain and the alphabet of $\mathcal{A}$ is the state space of $\mathcal{M}$ or the transitions of $\mathcal{M}$ are labeled by symbols of the alphabet of $\mathcal{A}$. The *positive model-checking problem* for $\mathcal{M}$ and $\mathcal{A}$ asks whether $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{A})) > 0$. Likewise, the *almost-sure model-checking problem* for $\mathcal{M}$ and $\mathcal{A}$ denotes the task to check whether $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{A})) = 1$. While the positive and the almost-sure probabilistic model-checking problems for Markov chains and NBA are both known to be PSPACE-complete [38,14], the analysis of Markov chains against UBA-specification can be carried out efficiently as stated in the following theorem:

Theorem 3.1. *Given a Markov chain $\mathcal{M}$ and a UBA $\mathcal{U}$, the value $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$ is computable in time polynomial in the sizes of $\mathcal{M}$ and $\mathcal{U}$.*

Remark 3.1. The statement of Theorem 3.1 has already been presented in [5] (see also [33,6]). However, the presented algorithm to compute $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$ is flawed. This approach, rephrased for the special case where the task is to compute $\Pr(\mathcal{L}_\omega(\mathcal{U}))$ for a given positive UBA $\mathcal{U}$ (which means a UBA where $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$) relies on the mistaken belief that there is at least one state q in $\mathcal{U}$ such that $\Pr(\mathcal{L}_\omega(\mathcal{U}[q])) = 1$. However, such states need not exist. To illustrate this, we consider the UBA $\mathcal{U}$ with two states q_a and q_b and $\delta(q_a, a) = \delta(q_b, b) = \{q_a, q_b\}$ and $\delta(q_a, b) = \delta(q_b, a) = \varnothing$. (See the UBA on the right of Fig. 1.) Both states are initial and final. Clearly, $\mathcal{L}_\omega(\mathcal{U}[q_a]) = a\Sigma^\omega$ and $\mathcal{L}_\omega(\mathcal{U}[q_b]) = b\Sigma^\omega$. Thus, $\mathcal{U}$ is universal and $\Pr(\mathcal{L}_\omega(\mathcal{U})) = 1$, while $\Pr(\mathcal{L}_\omega(\mathcal{U}[q_a])) = \Pr(\mathcal{L}_\omega(\mathcal{U}[q_b])) = \frac{1}{2}$.

Outline of Section 3. The remainder of Section 3 is devoted to the proof of Theorem 3.1. We first assume that the Markov chain $\mathcal{M}$ generates all words according

to a uniform distribution and explain how to compute the value $\Pr(\mathcal{L}_\omega(\mathcal{U}))$ for a given UBA $\mathcal{U}$ in polynomial time. For this, we first address the case of strongly connected UBA (Section 3.1) and then lift the result to the general case (Section 3.2). The central idea of the algorithm relies on the observation that each positive, strongly connected UBA has "recurrent sets" of states, called *cuts*. We exploit structural properties of unambiguous automata for the efficient construction of a cut and show how to compute the values $\Pr(\mathcal{L}_\omega(\mathcal{U}[q]))$ for the states of $\mathcal{U}$ by a linear equation system with one equation per state and one equation for the generated cut. Furthermore, positivity of a UBA $\mathcal{U}$ (i.e., $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$) is shown to be equivalent to the existence of a positive solution of the system of linear equations for the states. Finally, we explain how to adapt these techniques to general Markov chains (Section 3.3).

3.1 Strongly connected UBA

We start with some general observations about strongly connected Büchi automata under the probabilistic semantics. For this, we suppose $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ is a strongly connected NBA where Q_0 and F are nonempty. Clearly, $\mathcal{L}_\omega(q) \neq \varnothing$ for all states q and

$$\begin{aligned} \Pr(\mathcal{L}_\omega(\mathcal{A})) > 0 \quad &\text{iff} \quad \Pr(\mathcal{L}_\omega(q)) > 0 \text{ for some state } q \\ &\text{iff} \quad \Pr(\mathcal{L}_\omega(q)) > 0 \text{ for all states } q \end{aligned}$$

Moreover, almost all words $w \in \Sigma^\omega \setminus \mathcal{L}_\omega(\mathcal{A})$ have a finite prefix x with $\delta(Q_0, x) = \varnothing$ (for the proof see [4]):

Lemma 3.1 (Measure of strongly connected NBA). *For each strongly connected NBA $\mathcal{A}$ with at least one final state, we have:*

$$\Pr(\mathcal{L}_\omega(\mathcal{A})) \quad = \quad 1 - \Pr\big\{\, w \in \Sigma^\omega \,:\, w \text{ has a finite prefix } x \text{ with } \delta(Q_0, x) = \varnothing \,\big\}$$

In particular, $\mathcal{A}$ is almost universal if and only if $\delta(Q_0, x) \neq \varnothing$ for all finite words $x \in \Sigma^*$. This observation will be crucial at several places in the soundness proof of our algorithm for UBA, but can also be used to establish PSPACE-hardness of the positivity (probabilistic nonemptiness) and almost universality problem for strongly connected NBA, see [4]. For computing $\Pr(\mathcal{L}_\omega(\mathcal{U}))$ given a UBA $\mathcal{U}$, it suffices to compute the values $\Pr(\mathcal{L}_\omega(q))$ for the (initial) states of $\mathcal{U}$ as we have

$$\Pr(\mathcal{L}_\omega(\mathcal{U})) \quad = \quad \sum_{q \in Q_0} \Pr(\mathcal{L}_\omega(q))$$

Furthermore, in each strongly connected UBA, the accepting runs of almost all words $w \in \mathcal{L}_\omega(\mathcal{U})$ visit each state of $\mathcal{U}$ infinitely often (see [4]).

Deciding positivity for strongly connected UBA. The following lemma provides a criterion to check positivity of a strongly connected UBA in polynomial time using standard linear algebra techniques.

Lemma 3.2. *Let $\mathcal{U}$ be a strongly connected UBA with at least one initial and one final state, and*

$$(*) \qquad \zeta_q \;=\; \frac{1}{|\Sigma|} \cdot \sum_{a \in \Sigma} \sum_{p \in \delta(q,a)} \zeta_p \qquad \text{for all } q \in Q$$

Then, the following statements are equivalent:

(1) $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$,
(2) *the linear equation system (*) has a strictly positive solution, i.e., a solution $(\zeta_q^*)_{q \in Q}$ with $\zeta_q^* > 0$ for all $q \in Q$,*
(3) *the linear equation system (*) has a non-zero solution.*

Given the strongly connected UBA $\mathcal{U}$ with at least one final state, we define a matrix $M \in [0,1]^{Q \times Q}$ by $M_{p,q} = \frac{1}{|\Sigma|}|\{a \in \Sigma : q \in \delta(p,a)\}|$ for all $p, q \in Q$. Since $\mathcal{U}$ is strongly connected, M is irreducible. Write $\rho(M)$ for the spectral radius of M. We will use the following Lemma in the proof of Lemma 3.2.

Lemma 3.3. *We have $\rho(M) \leq 1$. Moreover $\rho(M) = 1$ if and only if $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$.*

Proof. For $p, q \in Q$ and $n \in \mathbb{N}$, let $E_{p,n,q} \subseteq \Sigma^\omega$ denote the event of all words $w = a_1 a_2 \ldots$ such that $q \in \delta(p, a_1 a_2 \ldots a_n)$. Its probability under the uniform distribution on Σ^ω is an entry in the n-th power of M:

$$\Pr(E_{p,n,q}) = (M^n)_{p,q} \tag{1}$$

In particular, $M^n_{p,q} \leq 1$ for all n. From the boundedness of M^n it follows (e.g., by [24, Corollary 8.1.33]) that $\rho(M) \leq 1$. The same result implies that

$$\begin{aligned} \rho(M) = 1 \quad &\iff \quad \limsup_{n \to \infty} (M^n)_{p,q} > 0 \quad \text{for all } p, q \in Q \\ &\iff \quad \limsup_{n \to \infty} (M^n)_{p,q} > 0 \quad \text{for some } p, q \in Q \end{aligned} \tag{2}$$

For the rest of the proof, fix some state $p \in Q$. By the observations from the beginning of Section 3.1 it suffices to show that $\Pr(\mathcal{L}_\omega(p)) > 0$ if and only if $\rho(M) = 1$. To this end, consider the event $E_{p,n} := \bigcup_{q \in Q} E_{p,n,q}$. Notice that $(E_{p,n})_{n \in \mathbb{N}}$ forms a decreasing family of sets. We have:

$$\begin{aligned} \Pr(\mathcal{L}_\omega(p)) &= \lim_{n \to \infty} \Pr(E_{p,n}) && \text{by Lemma 3.1} \\ &= \lim_{n \to \infty} \Pr\left(\bigcup_{q \in Q} E_{p,n,q} \right) && \text{definition of } E_{p,n} \end{aligned} \tag{3}$$

Assuming that $\rho(M) = 1$, we show that $\Pr(\mathcal{L}_\omega(p)) > 0$. Let $q \in Q$. We have:

$$\begin{aligned} \Pr(\mathcal{L}_\omega(p)) &\geq \limsup_{n \to \infty} \Pr(E_{p,n,q}) && \text{by (3)} \\ &= \limsup_{n \to \infty} (M^n)_{p,q} && \text{by (1)} \\ &> 0 && \text{by (2)} \end{aligned}$$

Conversely, assuming that $\rho(M) < 1$, we show that $\Pr(\mathcal{L}_\omega(p)) = 0$.

$$\begin{aligned}
\Pr(\mathcal{L}_\omega(p)) &= \lim_{n\to\infty} \Pr\left(\bigcup_{q\in Q} E_{p,n,q}\right) && \text{by (3)}\\
&\leq \limsup_{n\to\infty} \sum_{q\in Q} \Pr(E_{p,n,q}) && \text{union bound}\\
&= \limsup_{n\to\infty} \sum_{q\in Q} (M^n)_{p,q} && \text{by (1)}\\
&= 0 && \text{by (2)}
\end{aligned}$$

This concludes the proof. □

Proof (of Lemma 3.2). "(1) $\Longrightarrow$ (2)": Suppose $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$. Define the vector $(\zeta^*_q)_{q\in Q}$ with $\zeta^*_q = \Pr(\mathcal{L}_\omega(q))$. It holds that

$$\mathcal{L}_\omega(q) = \bigcup_{a\in\Sigma} \bigcup_{p\in\delta(q,a)} \{aw : w \in \mathcal{L}_\omega(p)\}$$

Since $\mathcal{U}$ is unambiguous, the sets $\{aw : w \in \mathcal{L}_\omega(p)\}$ are pairwise disjoint. So, the vector $(\zeta^*_q)_{q\in Q}$ is a solution to the equation system.

As $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$ and $\mathcal{U}$ is strongly connected, the observation at the beginning of Section 3.1 yields that $\Pr(\mathcal{L}_\omega(q)) > 0$ for all states q. Thus, the vector $(\zeta^*_q)_{q\in Q}$ is strictly positive.

"(2) $\Longrightarrow$ (3)" holds trivially.

"(3) $\Longrightarrow$ (1)": Suppose ζ^* is a non-zero solution of the linear equation system. Then, $M\zeta^* = \zeta^*$. Thus, 1 is an eigenvalue of M. This yields $\rho(M) \geqslant 1$. But then $\rho(M) = 1$ and $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$ by Lemma 3.3. □

Computing pure cuts for positive, strongly connected UBA. The key observation to compute the values $\Pr(\mathcal{L}_\omega(q))$ for the states q of a positive, strongly connected UBA $\mathcal{U}$ is the existence of so-called cuts. These are sets C of states with pairwise disjoint languages such that almost all words have an accepting run starting in some state $q \in C$. More precisely:

Definition 3.1 ((Pure) cut). *Let $\mathcal{U}$ be a UBA and $C \subseteq Q$. C is called a* cut *for $\mathcal{U}$ if $\mathcal{L}_\omega(q) \cap \mathcal{L}_\omega(p) = \varnothing$ for all $p, q \in C$ with $p \neq q$ and $\mathcal{U}[C]$ is almost universal. A cut is called* pure *if it has the form $\delta(q, z)$ for some state q and some finite word $z \in \Sigma^*$.*

Obviously, $\mathcal{U}$ is almost universal iff Q_0 is a cut. If $q \in Q$ and K_q denotes the set of finite words $z \in \Sigma^*$ such that $\delta(q, z)$ is a cut then $\Pr(\mathcal{L}_\omega(q))$ equals the probability measure of the language L_q consisting of all infinite words $w \in \Sigma^\omega$ that have a prefix in K_q, see [4].

Lemma 3.4 (Characterization of pure cuts). *Let $\mathcal{U}$ be a strongly connected UBA. For all $q \in Q$ and $z \in \Sigma^*$ we have: $\delta(q,z)$ is a cut iff $\delta(q,zy) \neq \varnothing$ for each word $y \in \Sigma^*$. Furthermore, if $\mathcal{U}$ is positive then for each cut C:*

$$\begin{aligned} & C \text{ is pure, i.e., } C = \delta(q,z) \text{ for some state-word pair } (q,z) \in Q \times \Sigma^* \\ \textit{iff} \quad & \text{for each state } q \in Q \text{ there is some word } z \in \Sigma^* \text{ with } C = \delta(q,z) \\ \textit{iff} \quad & \text{for each cut } C' \text{ there is some word } y \in \Sigma^* \text{ with } C = \delta(C',y) \end{aligned}$$

The proof of Lemma 3.4 is provided in [4]. By Lemma 2.1 and Lemma 3.4 we get:

Corollary 3.1. *If $\mathcal{U}$ is a strongly connected UBA then $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$ iff $\mathcal{U}$ has a pure cut.*

For the rest of Section 3.1, we suppose that $\mathcal{U}$ is positive and strongly connected. The second part of Lemma 3.4 yields that the pure cuts constitute a bottom strongly connected component of the automaton obtained from $\mathcal{U}$ using the standard powerset construction. The goal is now to design an efficient (polynomially time-bounded) algorithm for the generation of a pure cut. For this, we observe that if $q,p \in Q$, $q \neq p$, then $\{q,p\} \subseteq C$ for some pure cut C iff there exists a word y such that $\{q,p\} \subseteq \delta(q,y)$, see [4].

Definition 3.2 (Extension). *A word $y \in \Sigma^*$ is an extension for a state-word pair $(q,z) \in Q \times \Sigma^*$ iff there exists a state $p \in Q$ such that $q \neq p$, $\delta(p,z) \neq \varnothing$ and $\{q,p\} \subseteq \delta(q,y)$.*

It is easy to see that if y is an extension of (q,z), then $\delta(q,yz)$ is a proper superset of $\delta(q,z)$ (see [4]). Furthermore, for all state-word pairs $(q,z) \in Q \times \Sigma^*$ (see [4]):

$$\delta(q,z) \text{ is a cut} \quad \text{iff} \quad \text{there is no extension for } (q,z)$$

These observations lead to the following algorithm for the construction of a pure cut. We pick an arbitrary state q in the UBA and start with the empty word $z_0 = \varepsilon$. The algorithm iteratively seeks for an extension for the state-word pair (q,z_i). If an extension y_i for (q,z_i) has been found then we switch to the word $z_{i+1} = y_i z_i$. If no extension exists then (q,z_i) is a pure cut. In this way, the algorithm generates an increasing sequence of subsets of Q,

$$\delta(q,z_0) \;\subsetneq\; \delta(q,z_1) \;\subsetneq\; \delta(q,z_2) \;\subsetneq\; \ldots \;\subsetneq\; \delta(q,z_k),$$

which terminates after at most $|Q|$ steps and yields a pure cut $\delta(q,z_k)$.

It remains to explain an efficient realization of the search for an extension of the state-word pairs (q,z_i). The idea is to store the sets $Q_i[p] = \delta(p,z_i)$ for all states p. The sets $Q_i[p]$ can be computed iteratively by:

$$Q_0[p] = \{p\} \quad \text{and} \quad Q_{i+1}[p] \;=\; \bigcup_{r \in \delta(p,y_i)} Q_i[r]$$

To check whether (q, z_i) has an extension we apply standard techniques for the intersection problem for the languages $H_{q,q} = \{y \in \Sigma^* : q \in \delta(q, y)\}$ and $H_{q,F_i} = \{y \in \Sigma^* : \delta(q, y) \cap F_i \neq \varnothing\}$ where $F_i = \{p \in Q \setminus \{q\} : Q_i[p] \neq \varnothing\}$. Then, for each word $y \in \Sigma^*$ we have: $y \in H_{q,q} \cap H_{q,F_i}$ if and only if y is an extension of (q, z_i). The languages $H_{q,q}$ and H_{q,F_i} are recognized by the NFA $\mathcal{U}_{q,q} = (Q, \Sigma, \delta, q, q)$ and $\mathcal{U}_{q,F_i} = (Q, \Sigma, \delta, q, F_i)$. Thus, to check the existence of an extension and to compute an extension y (if existent) where the word y has length at most $|Q|^2$, we may run an emptiness check for the product-NFA $\mathcal{U}[q, q] \otimes \mathcal{U}[q, F_i]$. We conclude:

Corollary 3.2. *Given a positive, strongly connected UBA $\mathcal{U}$, a pure cut can be computed in time polynomial in the size of $\mathcal{U}$.*

Computing the measure of positive, strongly connected UBA. We suppose that $\mathcal{U} = (Q, \Sigma, \delta, Q_0, F)$ is a positive, strongly connected UBA and C is a cut. (C might be a pure cut that has been computed by the techniques explained above. However, in Theorem 3.2 C can be any cut.) Consider the linear equation system of Lemma 3.2 with variables ζ_q for all states $q \in Q$ and add the constraint that the variables ζ_q for $q \in C$ sum up to 1.

Theorem 3.2. *Let $\mathcal{U}$ be a positive, strongly connected UBA and C a cut. Then, the probability vector $(\Pr(\mathcal{L}_\omega(q)))_{q \in Q}$ is the unique solution of the following linear equation system:*

$$(1) \quad \zeta_q = \frac{1}{|\Sigma|} \cdot \sum_{a \in \Sigma} \sum_{p \in \delta(q,a)} \zeta_p \qquad \text{for all states } q \in Q$$

$$(2) \quad \sum_{q \in C} \zeta_q = 1$$

Proof. Let $n = |Q|$. Define a matrix $M \in [0, 1]^{Q \times Q}$ by $M_{q,p} = |\{a \in \Sigma : p \in \delta(q, a)\}|/|\Sigma|$ for all $q, p \in Q$. Then, the n equations (1) can be written as $\zeta = M\zeta$, where $\zeta = (\zeta_q)_{q \in Q}$ is a vector of n variables. It is easy to see that the values $\zeta^*_q = \Pr(\mathcal{L}_\omega(q))$ for $q \in Q$ satisfy the equations (1). That is, defining $\zeta^* = (\zeta^*_q)_{q \in Q}$ we have $\zeta^* = M\zeta^*$. By the definition of a cut, those values also satisfy equation (2).

It remains to show uniqueness. We employ Perron-Frobenius theory as follows. Since $\zeta^* = M\zeta^*$, the vector ζ^* is an eigenvector of M with eigenvalue 1. Since ζ^* is strictly positive (i.e., positive in all components), it follows from [8, Corollary 2.1.12] that $\rho = 1$ for the spectral radius ρ of M. Since $\mathcal{U}$ is strongly connected, matrix M is irreducible. By [8, Theorem 2.1.4 (b)] the spectral radius $\rho = 1$ is a simple eigenvalue of M, i.e., all solutions of $\zeta = M\zeta$ are scalar multiples of ζ^*. Among those multiples, only ζ^* satisfies equation (2). Uniqueness follows. ∎

Together with the criterion of Lemma 3.2 to check whether a given strongly connected UBA is positive, we obtain a polynomially time-bounded computation scheme for the values $\Pr(\mathcal{L}_\omega(q))$ for the states q of a given strongly connected UBA. The next section shows how to lift these results for arbitrary UBA.

3.2 Computing the measure of arbitrary UBA

In what follows, let $\mathcal{U} = (Q, \Sigma, \delta, Q_0, F)$ be a (possibly not strongly connected) UBA. We assume that all states are reachable from Q_0 and that F is reachable from all states. Thus, $\mathcal{L}_\omega(q) \neq \varnothing$ for all states q.

Let $\mathcal{C}$ be a strongly connected component (SCC) of $\mathcal{U}$. $\mathcal{C}$ is called non-trivial if $\mathcal{C}$ viewed as a direct graph contains at least one edge, i.e., if $\mathcal{C}$ is cyclic. $\mathcal{C}$ is called bottom if $\delta(q,a) \subseteq \mathcal{C}$ for all $q \in \mathcal{C}$ and all $a \in \Sigma$. We define Q_{BSCC} to be the set of all states $q \in Q$ that belong to some bottom SCC (BSCC) of $\mathcal{U}$. If $\mathcal{C}$ is a non-trivial SCC of $\mathcal{U}$ and $p \in \mathcal{C}$ then the sub-NBA

$$\mathcal{U}\big|_{\mathcal{C},p} \quad = \quad (\mathcal{C}, \Sigma, \delta|_{\mathcal{C}}, \{p\}, \mathcal{C} \cap F)$$

of $\mathcal{U}$ with state space $\mathcal{C}$, initial state p and the transition function $\delta|_{\mathcal{C}}$ given by $\delta|_{\mathcal{C}}(q,a) = \delta(q,a) \cap \mathcal{C}$ is strongly connected and unambiguous. Let L_p be the accepted language, i.e., $L_p = \mathcal{L}_\omega(\mathcal{U}\big|_{\mathcal{C},p})$. The values $\Pr(L_p)$, $p \in \mathcal{C}$, can be computed using the techniques for strongly connected UBA presented in Section 3.1. A non-trivial SCC $\mathcal{C}$ is said to be positive if $\Pr(L_p) > 0$ for all/some state(s) p in $\mathcal{C}$.

Lemma 3.5. *Let $\mathcal{C}$ be a non-trivial SCC of $\mathcal{U}$. If $\mathcal{C}$ is positive then $\Pr(\mathcal{L}_\omega(q)) = 0$ for all states $q \in Q \setminus \mathcal{C}$ that are reachable from $\mathcal{C}$.*

The proof of Lemma 3.5 can found in [4]. We perform the following preprocessing, exploiting Lemma 3.5. As before, for any $p \in Q$ we write $\mathcal{L}_\omega(p)$ for $\mathcal{L}_\omega(\mathcal{U}[p]\,)$, and call p zero if $\Pr(\mathcal{L}_\omega(p)) = 0$. First we remove all states that are not reachable from any initial state. Then we run standard graph algorithms to compute the directed acyclic graph (DAG) of SCCs of $\mathcal{U}$. By processing the DAG bottom-up we can remove all zero states by running the following loop: If all BSCCs are marked (initially, all SCCs are unmarked) then exit the loop; otherwise pick an unmarked BSCC $\mathcal{C}$.

- If $\mathcal{C}$ is trivial or does not contain any final state then we remove it: more precisely, we remove it from the DAG of SCCs, and we modify $\mathcal{U}$ by deleting all transitions $p \xrightarrow{a} q$ where $q \in \mathcal{C}$.
- Otherwise, $\mathcal{C}$ is a non-trivial BSCC with at least one final state. We check whether $\mathcal{C}$ is positive by applying the techniques of Section 3.1. If it is positive, we mark it; otherwise we remove it as described above.

Note that this loop does not change $\Pr(\mathcal{L}_\omega(p))$ for any state p.

Let Q_{BSCC} denote the set of states in $\mathcal{U}$ that belong to some BSCC. The values $\Pr(\,\mathcal{L}_\omega(p)\,)$ for the states $p \in Q_{BSCC}$ can be computed using the techniques of Section 3.1. The remaining task is to compute the values $\Pr(\,\mathcal{L}_\omega(q)\,)$ for the states $q \in Q \setminus Q_{BSCC}$. For $q \in Q \setminus Q_{BSCC}$, let $\beta_q = 0$ if $\delta(q,a) \cap Q_{BSCC} = \varnothing$ for all $a \in \Sigma$. Otherwise:

$$\beta_q \quad = \quad \frac{1}{|\Sigma|} \cdot \sum_{a \in \Sigma} \; \sum_{p \in \delta(q,a) \cap Q_{BSCC}} \Pr(\,\mathcal{L}_\omega(p)\,)$$

In [4] we show:

Theorem 3.3. *If all BSCCs of $\mathcal{U}$ are non-trivial and positive, then the linear equation system*

$$\zeta_q \;=\; \frac{1}{|\Sigma|} \cdot \sum_{a \in \Sigma} \sum_{\substack{r \,\in\, \delta(q,\,a) \\ r \,\notin\, Q_{BSCC}}} \zeta_r \;+\; \beta_q \qquad \text{for } q \in Q \setminus Q_{BSCC}$$

has a unique solution, namely $\zeta_q^* = \Pr(\,\mathcal{L}_\omega(q)\,)$.

This yields that the value $\Pr(\mathcal{L}_\omega(\mathcal{U}))$ for given UBA $\mathcal{U}$ is computable in polynomial time.

Remark 3.2. For the special case where $\delta(q,a) = \{q\}$ for all $q \in F$ and $a \in \Sigma$, the language of $\mathcal{U}$ is a co-safety property and $\Pr(\mathcal{L}_\omega(q)) = 1$ if $q \in F = Q_{BSCC}$, when we assume that all BSCCs are non-trivial and positive. In this case, the linear equation system in Theorem 3.3 coincides with the linear equation system presented in [7] for computing the probability measure of the language of $\mathcal{U}$ viewed as an UFA.

Remark 3.3. As a consequence of our results, the positivity problem ("does $\Pr(\mathcal{L}_\omega(\mathcal{U})) > 0$ hold?") and the almost universality problem ("does $\Pr(\mathcal{L}_\omega(\mathcal{U})) = 1$ hold?") for UBA are solvable in polynomial time. This should be contrasted with the standard (non-probabilistic) semantics of UBA and the corresponding results for NBA. The non-emptiness problem for UBA is in P (this already holds for NBA), while the complexity-theoretic status of the universality problem for UBA is a long-standing open problem. For standard NBA, it is well known that the non-emptiness problem is in P and the universality problem is PSPACE-complete. However, the picture changes when switching to NBA with the probabilistic semantics as both the positivity problem and the almost universality problem for NBA are PSPACE-complete, even for strongly connected NBA (see [4]).

3.3 Probabilistic model checking of Markov chains against UBA

To complete the proof of Theorem 3.1, we show how the results of the previous section can be adapted to compute the value $\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U}))$ for a Markov chain $\mathcal{M} = (S, P, \iota)$ and a UBA $\mathcal{U} = (Q, \Sigma, \delta, Q_0, F)$ with alphabet $\Sigma = S$. [3] The necessary adaptions to the proofs are detailed in [4].

If $\mathcal{A}$ is an NBA over the alphabet S and $s \in S$, then $\Pr_s^{\mathcal{M}}(\mathcal{A})$ denotes the probability $\Pr_s^{\mathcal{M}}(\mathit{\Pi})$ with $\mathit{\Pi}$ being the set of infinite paths $\pi = s_0 s_1 \ldots \in S^\omega$

[3] In practice, e.g., when the UBA is obtained from an LTL formula, the alphabet of the UBA is often defined as $\Sigma = 2^{AP}$ over a set of atomic propositions AP and the Markov chain is equipped with a labeling function from states to the atomic propositions that hold in each state. Clearly, unambiguity w.r.t. the alphabet 2^{AP} implies unambiguity w.r.t. the alphabet S when switching from the original transition function $\delta : Q \times 2^{AP} \to 2^Q$ to the transition function $\delta_S : Q \times S \to 2^Q$ given by $\delta_S(q,s) = \delta(q, L(s))$, where $L : S \to 2^{AP}$ denotes the labeling function of $\mathcal{M}$.

starting with $s_0 = s$ and such that $s_1 s_2 \ldots \in \mathcal{L}_\omega(\mathcal{A})$. Our algorithm relies on the observation that

$$\Pr^{\mathcal{M}}(\mathcal{L}_\omega(\mathcal{U})) \;=\; \sum_{s \in S} \iota(s) \cdot \Pr_s^{\mathcal{M}}(\,\mathcal{U}[\delta(Q_0, s)]\,)$$

As the languages of the UBA $\mathcal{U}[q]$ for $q \in \delta(Q_0, s)$ are pairwise distinct (by the unambiguity of $\mathcal{U}$), we have $\Pr_s^{\mathcal{M}}(\mathcal{U}[\delta(Q_0, s)]) \;=\; \sum_{q \in \delta(Q_0,s)} \Pr_s^{\mathcal{M}}(\mathcal{U}[q])$.

Thus, the task is to compute the values $\Pr_s^{\mathcal{M}}(\mathcal{U}[q])$ for $s \in S$ and $q \in Q$. As a first step, we build a UBA $\mathcal{P} = \mathcal{M} \otimes \mathcal{U}$ that arises from the synchronous product of the UBA $\mathcal{U}$ with the underlying graph of the Markov chain $\mathcal{M}$. Formally, $\mathcal{P} = (S \times Q, \Sigma, \Delta, Q_0', S \times F)$ where Q_0' consists of all pairs $\langle s, q\rangle \in S \times Q$ where $\iota(s) > 0$ and $q \in \delta(Q_0, s)$. Let $s, t \in S$ and $q \in Q$. If $P(s,t) = 0$ then $\Delta(\langle s, q\rangle, t) = \varnothing$, while for $P(s,t) > 0$, the set $\Delta(\langle s, q\rangle, t)$ consists of all pairs $\langle t, p\rangle$ where $p \in \delta(q, s)$. We are only concerned with the reachable fragment of the product.

Given that $\mathcal{M}$ viewed as an automaton over the alphabet S behaves deterministically and we started with an unambiguous automaton $\mathcal{U}$, the product $\mathcal{P}$ is unambiguous as well. Let $\mathcal{P}[s, q]$ denote the UBA resulting from $\mathcal{P}$ by declaring $\langle s, q\rangle$ to be initial. It is easy to see that $\Pr_s^{\mathcal{M}}(\,\mathcal{P}[s, q]\,) = \Pr_s^{\mathcal{M}}(\,\mathcal{U}[q]\,)$ for all states $\langle s, q\rangle$ of $\mathcal{P}$, as the product construction only removes transitions in $\mathcal{U}$ that can not occur in the Markov chain. Our goal is thus to compute the values $\Pr_s^{\mathcal{M}}(\mathcal{P}[s, q])$. For this, we remove all states $\langle s, q\rangle$ from $\mathcal{P}$ that can not reach a state in $S \times F$. Then, we determine the non-trivial SCCs of $\mathcal{P}$ and, for each such SCC $\mathcal{C}$, we analyze the sub-UBA $\mathcal{P}\big|_{\mathcal{C}}$ obtained by restricting to the states in $\mathcal{C}$. An SCC $\mathcal{C}$ of $\mathcal{P}$ is called positive if $\Pr_s^{\mathcal{M}}(\mathcal{P}\big|_{\mathcal{C}}[s, q]) > 0$ for all/any $\langle s, q\rangle \in \mathcal{C}$. As in Lemma 3.5, one can show that the probabilities after leaving a positive SCC $\mathcal{C}$ are zero. Thus, we treat the SCCs in a bottom-up manner as in Section 3.2, starting with the BSCCs and removing them if they are non-positive. Clearly, if a BSCC $\mathcal{C}$ of $\mathcal{P}$ does not contain a final state or is trivial, then $\mathcal{C}$ is not positive. Analogously to Lemma 3.2, a non-trivial BSCC $\mathcal{C}$ in $\mathcal{P}$ containing at least one final state is positive if and only if the linear equation system

$$(*) \qquad \zeta_{s,q} \;=\; \sum_{t \in \mathrm{Post}(s)} \; \sum_{p \in \delta_{\mathcal{C}}(q,t)} P(s,t) \cdot \zeta_{t,p} \quad \text{for all } \langle s, q\rangle \in \mathcal{C}$$

has a strictly positive solution if and only if $(*)$ has a non-zero solution. Here, $\mathrm{Post}(s) = \{t \in S : P(s,t) > 0\}$ denotes the set of successors of state s in $\mathcal{M}$ and $\delta_{\mathcal{C}}(q,t) = \{p \in \delta(q,t) : \langle t, p\rangle \in \mathcal{C}\}$.

We now explain how to adapt the cut-based approach of Section 3.1 for computing the probabilities in a positive BSCC $\mathcal{C}$ of $\mathcal{P}$. For $\langle s, q\rangle \in \mathcal{C}$ and $t \in S$, let $\Delta_{\mathcal{C}}(\langle s, q\rangle, t) = \Delta(\langle s, q\rangle, t) \cap \mathcal{C}$. A pure cut in $\mathcal{C}$ denotes a set $C \subseteq \mathcal{C}$ such that $\Pr_s^{\mathcal{M}}(\mathcal{P}[C]) = 1$ and $C = \Delta_{\mathcal{C}}(\langle s, q\rangle, z)$ for some $\langle s, q\rangle \in \mathcal{C}$ and some finite word $z \in S^*$ such that $s\,z$ is a cycle in $\mathcal{M}$. (In particular, the last symbol of z is s, and therefore $C \subseteq \{\langle s, p\rangle \in \mathcal{C} : p \in Q\}$.) To compute a pure cut in $\mathcal{C}$, we pick an arbitrary state $\langle s, q\rangle$ in $\mathcal{C}$ and successively generate path fragments

$z_0, z_1, \ldots, z_k \in S^*$ in $\mathcal{M}$ by adding prefixes. More precisely, $z_0 = \varepsilon$ and z_{i+1} has the form yz_i for some $y \in S^+$ such that (1) $s\,y$ is a cycle in $\mathcal{M}$ and (2) there exists a state $p \in Q \setminus \{q\}$ in $\mathcal{U}$ with $\Delta_{\mathcal{C}}(\langle s,p\rangle, z_i) \neq \varnothing$ and $\{\langle s,q\rangle, \langle s,p\rangle\} \subseteq \Delta_{\mathcal{C}}(\langle s,q\rangle, y)$. Each such word y is called an extension of $(\langle s,q\rangle, z_i)$, and $\Delta_{\mathcal{C}}(\langle s,q\rangle, z_{i+1}) = \Delta_{\mathcal{C}}(\langle s,q\rangle, yz_i)$ is a proper superset of $\Delta_{\mathcal{C}}(\langle s,q\rangle, z_i)$. The set $C = \Delta_{\mathcal{C}}(\langle s,q\rangle, z)$ is a pure cut if and only if $(\langle s,q\rangle, z_i)$ has no extension. The search for an extension can be realized efficiently using a technique similar to the one presented in Section 3.1. Thus, after at most $\min\{|\mathcal{C}|, |Q|\}$ iterations, we obtain a pure cut C.

Having computed a pure cut C of $\mathcal{C}$, the values $\Pr_s^{\mathcal{M}}(\mathcal{P}[s,q])$ for $\langle s,q\rangle \in \mathcal{C}$ are then computable as the unique solution of the linear equation system consisting of equations (*) and the additional equation $\sum_{\langle s,q\rangle \in C} \zeta_{s,q} = 1$.

In this way we adapt Theorem 3.2 to obtain the values $\Pr_s^{\mathcal{M}}(\mathcal{P}[s,q])$ for the states $\langle s,q\rangle$ belonging to some positive BSCC of $\mathcal{P}$. It remains to explain how to adapt the equation system of Theorem 3.3. Let Q_{BSCC} be the set of BSCC states of $\mathcal{P}$ and $Q_?$ be the states of $\mathcal{P}$ not contained in Q_{BSCC}. For $\langle s,q\rangle \in Q_?$, let $\beta_{s,q} = 0$ if $\Delta(\langle s,q\rangle, t) \cap Q_{BSCC} = \varnothing$ for all $t \in S$. Otherwise:

$$\beta_{s,q} \;=\; \sum_{t \in \mathrm{Post}(s)} \; \sum_{\substack{p \,\in\, \delta(q,t) \text{ s.t.} \\ \langle t,p\rangle \,\in\, Q_{BSCC}}} P(s,t) \cdot \Pr_t^{\mathcal{M}}(\,\mathcal{P}[t,p]\,)$$

Then, the vector $(\Pr_s^{\mathcal{M}}(\mathcal{P}[s,q]))_{\langle s,q\rangle \in Q_?}$ is the unique solution of the linear equation system

$$\zeta_{s,q} \;=\; \sum_{t \in \mathrm{Post}(s)} \; \sum_{\substack{p \,\in\, \delta(q,t) \text{ s.t.} \\ \langle t,p\rangle \,\notin\, Q_{BSCC}}} P(s,t) \cdot \zeta_{t,p} \;\;+\;\; \beta_{s,q} \qquad \text{for } \langle s,q\rangle \in Q_?$$

This completes the proof of Theorem 3.1.

4 Implementation and Experiments

We have implemented a probabilistic model checking procedure for Markov chains and UBA specifications using the algorithm detailed in Section 3 as an extension to the probabilistic model checker `PRISM` [30,35]. [4] Our implementation is based on the explicit engine of `PRISM`, where the Markov chain is represented explicitly. An implementation for the symbolic, MTBDD-based engines of `PRISM` is planned as future work.

Our implementation supports UBA-based model checking for handling the LTL fragment of `PRISM`'s PCTL*-like specification language as well as direct verification against a path specification given by a UBA provided in the HOA format [3]. For LTL formulas, we rely on external LTL-to-UBA translators. For the purpose of the benchmarks we employ the `ltl2tgba` tool from `SPOT` [18] to generate UBA for a given LTL formula.

[4] More details are available at [1]. All experiments were carried out on a computer with two Intel E5-2680 8-core CPUs at 2.70 GHz with 384GB of RAM running Linux.

For the linear algebra parts of the algorithms, we rely on the `COLT` library [25]. We considered two different variants for the SCC computations as detailed in [4]. The first variant relies on `COLT` to perform a QR decomposition of the matrix for the SCC to compute the rank, which allows for deciding the positivity of the SCC. The second approach relies on a variant of the power iteration method for iteratively computing an eigenvector. This method has the benefit that, in addition to deciding the positivity, the computed eigenvector can be directly used to compute the values for a positive SCC, once a cut has been found. (As the proof of Theorem 3.2 shows: $\Pr(\mathcal{L}_\omega(q)) = \zeta^*_q / \sum_{p \in C} \zeta^*_p$ if ζ^* is an eigenvector of the matrix M for eigenvalue 1.) We have evaluated the performance and scalability of the cut generation algorithm together with both approaches for treating SCCs with selected automata specifications that are challenging for our UBA-based model checking approach, see [4]. As the power iteration method performed better, our benchmark results presented in this section use this method for the SCC handling.

We report here on benchmarks using the bounded retransmission protocol (BRP) case study of the `PRISM` benchmark suite [31]. The model from the benchmark suite covers a single message transmission, retrying for a bounded number of times in case of an error. We have slightly modified the model to allow the transmission of an infinite number of messages by restarting the protocol once a message has been successfully delivered or the bound for retransmissions has been reached. We consider the LTL property

$$\varphi^k = (\neg \texttt{sender\_ok})\, \mathcal{U}\, \big((\texttt{retransmit} \wedge (\neg \texttt{sender\_ok}\, \mathcal{U}^{=k}\, \texttt{sender\_ok})\big),$$

ensuring that k steps before an acknowledgment the message was retransmitted. To remove the effect of selecting specific tools for the LTL to automaton translation (`ltl2tgba` for UBA, the Java-based `PRISM` reimplementation of `ltl2dstar` [28] to obtain a deterministic Rabin automaton (DRA) for the standard `PRISM` approach), we also consider direct model checking against automata specifications. As the language of φ^k is equivalent to the UBA depicted in Figure 1 (on the left) when $a = \texttt{retransmit} \wedge \neg\texttt{sender\_ok}$, $b = \texttt{sender\_ok}$ and $c = \neg\texttt{retransmit} \wedge \neg\texttt{send\_ok}$, we use this automaton and the minimal DBA for the language (this case is denoted by $\mathcal{A}$). We additionally consider the UBA and DBA obtained by replacing the self-loop in the last state with a switch back to the initial state (denoted by $\mathcal{B}$), i.e., roughly applying the ω-operator to $\mathcal{A}$.

Table 1 shows results for selected k (with a timeout 30 minutes), demonstrating that for this case study and properties our UBA-based implementation is generally competitive with the standard approach of `PRISM` relying on deterministic automata. For φ and $\mathcal{A}$, our implementation detects that the UBA has a special shape where all final states have a true-self loop, which allows for skipping the SCC handling. Without this optimization, t_{Cut} and t_{Pos} are in the sub-second range for φ and $\mathcal{A}$ for all considered k. At a certain point, the implementation of the standard approach in `PRISM` becomes unsuccessful, either due to timeouts in the DRA construction (φ: $k \geq 10$) or `PRISM` size limitations in the deterministic product construction (φ: $k \geq 8$, $\mathcal{A}/\mathcal{B}$: $k \geq 16$). For $k \geq 18$,

Table 1. Statistics for DBA/DRA- and UBA-based model checking of the BRP case study (parameters $N = 16$, $MAX = 128$), a DTMC with 29358 states, depicting the number of states for the automata and the product and the time for model checking (t_{MC}). For φ, t_{MC} includes the translation to the automaton, for $\mathcal{B}$ the time for checking positivity (t_{Pos}) and cut generation (t_{Cut}) are included in t_{MC}. The mark $-$ stands for "not available" or timeout (30 minutes).

	PRISM standard			PRISM UBA				
	DRA	product	t_{MC}	UBA	product	t_{MC}	t_{Pos}	t_{Cut}
$k = 4$, φ	118	62,162	0.8 s	6	34,118	0.6 s		
$\mathcal{A}$	33	61,025	0.8 s	6	34,118	0.5 s		
$\mathcal{B}$	33	75,026	0.7 s	6	68,474	1.9 s	1.1 s	<0.1 s
$k = 6$, φ	4,596	72,313	3.2 s	8	36,164	0.9 s		
$\mathcal{A}$	129	62,428	1.1 s	8	36,164	0.9 s		
$\mathcal{B}$	129	97,754	1.1 s	8	99,460	3.1 s	1.5 s	<0.1 s
$k = 8$, φ	297,204	–	–	10	38,207	0.8 s		
$\mathcal{A}$	513	64,715	1.1 s	10	38,207	0.7 s		
$\mathcal{B}$	513	134,943	1.3 s	10	136,427	4.5 s	2.5 s	<0.1 s
$k = 14$, φ	–	–	–	16	44,340	12.8 s	0.0 s	0.0 s
$\mathcal{A}$	32,769	83,845	5.3 s	16	44,340	1.0 s		
$\mathcal{B}$	32,769	444,653	6.0 s	16	246,346	10.2 s	6.5 s	<0.1 s
$k = 16$, φ	–	–	–	18	46,390	115.0 s		
$\mathcal{A}$	131,073	–	–	18	46,390	1.0 s		
$\mathcal{B}$	131,073	–	–	18	282,699	12.3 s	8.6 s	<0.1 s
$k = 48$, $\mathcal{A}$	–	–	–	50	79,206	1.8 s		
$\mathcal{B}$	–	–	–	50	843,414	88.4 s	71.1 s	<0.1 s

`ltl2tgba` was unable to construct the UBA for φ within the given time limit, for $k = 16$, 114.4 s of the 115.0 s were spent on constructing the UBA. As can be seen, using the UBA approach we were able to successfully scale the parameter k beyond 48 when dealing directly with the automata-based specifications ($\mathcal{A}/\mathcal{B}$) and within reasonable time required for model checking.

5 Conclusion

The main contribution of the paper is a polynomial-time algorithm for the quantitative analysis of Markov chains against UBA-specifications, and an implementation thereof. This yields a single exponential-time algorithm for the probabilistic model-checking problem for Markov chains and LTL formulas, and thus an alternative to the double exponential-time classical approach with deterministic automata that has been implemented in PRISM and other tools. Other single exponential algorithms for Markov chains and LTL are known, such as the automata-less method of [14] and the approaches with weak alternating

automata [10] or separated UBA [16]. To the best of our knowledge, no implementations of these algorithms are available.[5]

The efficiency of the proposed UBA-based analysis of Markov chains against LTL-specifications crucially depends on sophisticated techniques for the generation of UBA from LTL formulas. Compared to the numerous approaches for the generation of compact nondeterministic or deterministic automata, research on for efficient LTL-to-UBA translators is rare. The tool `Tulip` [33] uses a variant of the LTL-to-NBA algorithm by Gerth et al. [21] for the direct construction of UBA from LTL formulas, while `SPOT`'s LTL-to-UBA generator relies on an adaption of the Couvreur approach [15]. A comparison of NBA versus UBA sizes for LTL benchmark formulas from [20,36,19] (see [4]) using `SPOT` suggests that requiring unambiguity does not necessarily lead to a major increase in NBA size. An alternative to the direct translation of LTL formulas into UBA are standard LTL-to-NBA translators combined with disambiguation approaches for NBA (e.g. of [27]). However, we are not aware of tool support for these techniques.

Besides the design of efficient LTL-to-UBA translators that exploit the additional flexibility of unambiguous automata compared to deterministic ones, our future work will include a symbolic implementation of our algorithm and more experiments to evaluate the UBA-based approach against the classical approach with deterministic automata (e.g. realized in `PRISM` [30,35] and `IscasMC` [23] and using state-of-the-art generators for deterministic automata such as `Rabinizer`) or other single exponential-time algorithms [14,16,10], addressing the complex interplay between automata sizes, automata generation time, size of the (reachable fragment of the) product and the cost of the analysis algorithms that all influence the overall model checking time.

References

1. `http://wwwtcs.inf.tu-dresden.de/ALGI/PUB/CAV16/`. Website with additional material for this paper.
2. André Arnold. Deterministic and non ambiguous rational omega-languages. In *Automata on Infinite Words, Ecole de Printemps d'Informatique Théorique, Le Mont Dore, May 1984*, volume 192 of *Lecture Notes in Computer Science*, pages 18–27, 1985.
3. Tomás Babiak, Frantisek Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Kretínský, David Müller, David Parker, and Jan Strejcek. The Hanoi omega-automata format. In *27th International Conference on Computer Aided Verification (CAV)*, volume 9206 of *Lecture Notes in Computer Science*, pages 479–486, 2015.
4. Christel Baier, Stefan Kiefer, Joachim Klein, Sascha Klüppelholz, David Müller, and James Worrell. Markov chains and unambiguous Büchi automata (full paper). arXiv:????.?????, 2016. reference to be updated.

[5] The paper [16] reports on experiments with a prototype implementation, but this implementation seems not to be available anymore. As briefly explained in [4], our algorithm can be seen as a generalization of the approach of [16] with separated UBA. The tool Tulip [33] also has an engine for analysis of Markov chains against UBA-specifications, but it relies on the flawed algorithm of [7].

5. Michael Benedikt, Rastislav Lenhardt, and James Worrell. LTL model checking of interval Markov chains. In *19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 7795 of *Lecture Notes in Computer Science*, pages 32–46, 2013.
6. Michael Benedikt, Rastislav Lenhardt, and James Worrell. Model checking Markov chains against unambiguous Büchi automata. *CoRR*, abs/1405.4560, 2014.
7. Michael Benedikt, Ratislav Lenhardt, and James Worrell. Two variable vs. linear temporal logic in model checking and games. *Logical Methods in Computer Science*, 9(2), 2013.
8. Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*. Academic Press, 1979.
9. Nicolas Bousquet and Christof Löding. Equivalence and inclusion problem for strongly unambiguous Büchi automata. In *4th International Conference on Language and Automata Theory and Applications (LATA)*, volume 6031 of *Lecture Notes in Computer Science*, pages 118–129, 2010.
10. Doron Bustan, Sasha Rubin, and Moshe Y. Vardi. Verifying ω-regular properties of Markov chains. In *16th International Conference on Computer Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 189–201, 2004.
11. Souymodip Chakraborty and Joost-Pieter Katoen. Parametric LTL on Markov chains. In *8th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science*, volume 8705 of *Lecture Notes in Computer Science*, pages 207–221, 2014.
12. Thomas Colcombet. Forms of determinism for automata (invited talk). In *29th International Symposium on Theoretical Aspects of Computer Science, (STACS)*, volume 14 of *LIPIcs*, pages 1–23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
13. Thomas Colcombet. Unambiguity in automata theory. In *17th International Workshop on Descriptional Complexity of Formal Systems (DCFS)*, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18, 2015.
14. Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
15. Jean-Michel Couvreur. On-the-fly verification of linear temporal logic. In *World Congress on Formal Methods in the Development of Computing Systems (FM)*, volume 1708 of *Lecture Notes in Computer Science*, pages 253–271, 1999.
16. Jean-Michel Couvreur, Nasser Saheb, and Grégoire Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *10th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 2850 of *Lecture Notes in Computer Science*, pages 361–375, 2003.
17. Alexandre Duret-Lutz. Manipulating LTL formulas using Spot 1.0. In *11th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 8172 of *Lecture Notes in Computer Science*, pages 442–445, 2013.
18. Alexandre Duret-Lutz. LTL translation improvements in Spot 1.0. *International Journal of Critical Computer-Based Systems*, 5(1/2):31–54, 2014.
19. Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *21th International Conference on Software Engineering (ICSE)*, pages 411–420. ACM, 1999.
20. Kousha Etessami and Gerard Holzmann. Optimizing Büchi automata. In *11th International Conference on Concurrency Theory (CONCUR)*, volume 1877 of *Lecture Notes in Computer Science*, pages 153–167, 2000.

21. Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Fifteenth IFIP WG6.1 International Symposium on Protocol Specification (PSTV)*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.
22. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*, 2002.
23. Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. ISCASMC: A web-based probabilistic model checker. In *19th International Symposium on Formal Methods (FM)*, volume 8442 of *Lecture Notes in Computer Science*, pages 312–317, 2014.
24. Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, 2nd edition, 2013.
25. Wolfgang Hoschek. The colt distribution: Open source libraries for high performance scientific and technical computing in java. cern, geneva, 2004.
26. Dimitri Isaak and Christof Löding. Efficient inclusion testing for simple classes of unambiguous ω-automata. *Information Processing Letters*, 112(14-15):578–582, 2012.
27. Detlef Kähler and Thomas Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *Lecture Notes in Computer Science*, pages 724–735, 2008.
28. Joachim Klein and Christel Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theoretical Computer Science*, 363(2):182–195, 2006.
29. Vidyadhar G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.
30. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *23rd International Conference on Computer Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591, 2011.
31. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *9th International Conference on Quantitative Evaluation of SysTems (QEST)*, pages 203–204. IEEE Computer Society, 2012.
32. Rastislav Lenhardt. Tulip: Model checking probabilistic systems using expectation maximisation algorithm. In *10th International Conference on Quantitative Evaluation of Systems (QEST)*, volume 8054 of *Lecture Notes in Computer Science*, pages 155–159, 2013.
33. Rastislav Lenhardt. *Two Variable and Linear Temporal Logic in Model Checking and Games*. PhD thesis, University of Oxford, 2013.
34. Andreas Morgenstern. *Symbolic Controller Synthesis for LTL Specifications*. PhD thesis, Technische Universität Kaiserslautern, 2010.
35. The PRISM model checker. `http://www.prismmodelchecker.org/`.
36. Fabio Somenzi and Roderick Bloem. Efficient Büchi automata from LTL formulae. In *12th International Conference on Computer Aided Verification (CAV)*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263, 2000.
37. Richard E. Stearns and Harry B. Hunt. On the equivalence and containment problem for unambiguous regular expressions, grammars, and automata. *SIAM Journal on Computing*, pages 598–611, 1985.

38. Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 327–338. IEEE Computer Society, 1985.
39. Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *1st Symposium on Logic in Computer Science (LICS)*, pages 332–344. IEEE Computer Society Press, 1986.
40. Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 185–194. IEEE Computer Society, 1983.
41. Martin Zimmermann. Optimal bounds in parametric LTL games. *Theoretical Computer Science*, 493:30–45, 2013.