Efficient Analysis of Unambiguous Automata Using Matrix Semigroup Techniques

Stefan Kiefer
University of Oxford, UK

Cas Widdershoven
University of Oxford, UK

Abstract

We introduce a novel technique to analyse unambiguous Büchi automata quantitatively, and apply this to the model checking problem. It is based on linear-algebra arguments that originate from the analysis of matrix semigroups with constant spectral radius. This method can replace a combinatorial procedure that dominates the computational complexity of the existing procedure by Baier et al. We analyse the complexity in detail, showing that, in terms of the set Q of states of the automaton, the new algorithm runs in time $O(|Q|^4)$, improving on an efficient implementation of the combinatorial algorithm by a factor of $|Q|$.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Design and analysis of algorithms

Keywords and phrases Algorithms, Automata, Markov Chains, Matrix Semigroups

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.82

Related Version A full version of the paper is available at `https://arxiv.org/abs/1906.10093`.

Funding *Stefan Kiefer*: Work supported by a Royal Society University Research Fellowship.

1 Introduction

Given a finite automaton $\mathcal{A}$, what is the proportion of words accepted by it? This question is natural but imprecise: there are infinitely many words and the proportion of accepted words may depend on the word length. One may consider the sequence $d_0, d_1, \ldots$ where d_i is the proportion of length-i words accepted by $\mathcal{A}$, i.e., $d_i = \frac{|L(\mathcal{A}) \cap \Sigma^i|}{|\Sigma|^i}$. The sequence does not necessarily converge, but one may study, e.g., possible limits and accumulation points [5].

Alternatively, one can specify a probability distribution on words, e.g., with a Markov chain, and ask for the probability that a word is accepted by $\mathcal{A}$. For instance, if $\Sigma = \{a, b\}$, one may generate a random word, letter by letter, by outputting a, b with probability $1/3$ each, and ending the word with probability $1/3$. For an NFA $\mathcal{A}$, determining whether the probability of generating an accepted word is 1 is equivalent to universality (is $L(\mathcal{A}) = \Sigma^*$?), a PSPACE-complete problem. However, if $\mathcal{A}$ is unambiguous, i.e., every accepted word has exactly one accepting path, then one can compute the probability of generating an accepted word in polynomial time by solving a linear system of equations. Unambiguousness allows us to express the probability of a union as the sum of probabilities:

▶ **Example 1.** Consider the unambiguous automaton $\mathcal{A}$ in Figure 1 (left). If we generate a random word over $\{a, b\}$ according to the process described above, we have the following linear system for the vector $\vec{z}$ where $\vec{z}_q$ is, for each $q \in \{q_0, q_1, q_2, q_3\}$, the probability that the word is accepted when q is taken as initial state:

$$\vec{z}_{q_0} = \tfrac{1}{3}\vec{z}_{q_1} + \tfrac{1}{3} \qquad \vec{z}_{q_1} = \tfrac{1}{3}\vec{z}_{q_0} + \tfrac{1}{3}(\vec{z}_{q_1} + \vec{z}_{q_3})$$
$$\vec{z}_{q_2} = \tfrac{1}{3}\vec{z}_{q_3} + \tfrac{1}{3}(\vec{z}_{q_0} + \vec{z}_{q_2}) \qquad \vec{z}_{q_3} = \tfrac{1}{3}\vec{z}_{q_2}$$

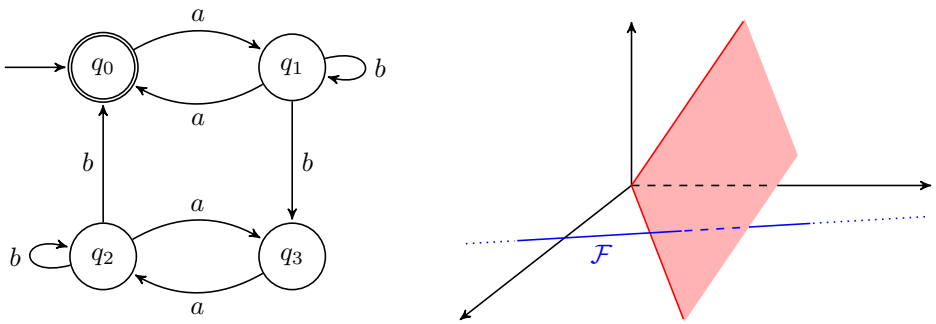

Figure 1 Left: unambiguous automaton $\mathcal{A}$. Right: visualisation of the affine space $\mathcal{F}$ (blue) and the vector space spanned by (pseudo-)cuts (red); these spaces are orthogonal.

The constant term in the equation for $\vec{z}_{q_0}$ reflects the fact that q_0 is accepting. The (other) coefficients $\frac{1}{3}$ correspond to the production of either a or b. The linear system has a unique solution.

One may view an NFA $\mathcal{A}$ as a Büchi automaton, so that its language $L(\mathcal{A}) \subseteq \Sigma^\omega$ is the set of those infinite words that have an accepting run in $\mathcal{A}$, i.e., a run that visits accepting states infinitely often. There is a natural notion of an infinite random word over Σ: in each step sample a letter from Σ uniformly at random, e.g., if $\Sigma = \{a, b\}$ then choose a and b with probability $1/2$ each. Perhaps more significantly, model checking Markov chains against Büchi automata, i.e., computing the probability that the random word generated by the Markov chain is accepted by the automaton, is a key problem in the verification of probabilistic systems. Unfortunately, like the aforementioned problem on finite words, it is also PSPACE-complete [8]. However, if the Büchi automaton is unambiguous, i.e., every accepted (infinite) word has exactly one accepting path, then one can compute the probability of generating an accepted word in polynomial time [2], both in the given Büchi automaton and in a given (discrete-time, finite-state) Markov chain. Since LTL specifications can be converted to unambiguous Büchi automata with a single-exponential blow-up, this leads to an LTL model-checking algorithm with single-exponential runtime, which is optimal. The polynomial-time algorithm from [2] for unambiguous Büchi automata is more involved than in the finite-word case.

▶ **Example 2.** In the following we view the automaton $\mathcal{A}$ from Figure 1 as an (unambiguous) Büchi automaton. If we generate a random word over $\{a, b\}$ according to the process described above, then the vector $\vec{z}$ where for each $q \in \{q_0, q_1, q_2, q_3\}$, $\vec{z}_q$ is the probability that the word is accepted when q is taken as initial state, is a solution to the following linear system:

$$\begin{aligned} \vec{z}_{q_0} &= \tfrac{1}{2}\vec{z}_{q_1} & \qquad \vec{z}_{q_1} &= \tfrac{1}{2}\vec{z}_{q_0} + \tfrac{1}{2}(\vec{z}_{q_1} + \vec{z}_{q_3}) \\ \vec{z}_{q_2} &= \tfrac{1}{2}\vec{z}_{q_3} + \tfrac{1}{2}(\vec{z}_{q_0} + \vec{z}_{q_2}) & \qquad \vec{z}_{q_3} &= \tfrac{1}{2}\vec{z}_{q_2} \end{aligned}$$

However, this linear system has multiple solutions: indeed, any scalar multiple $(1, 2, 2, 1)^\top$ is a solution.

In order to make such a linear system uniquely solvable, one needs to add further equations, and finding these further equations is where the real challenge lies. Assuming that the state space Q of $\mathcal{A}$ is strongly connected and the Markov chain generates letters uniformly at random as described above, a single additional equation $\vec{\mu}^\top \vec{z} = 1$ suffices (this can be shown with Perron-Frobenius theory: the eigenspace for the dominant eigenvalue of a nonnegative

irreducible matrix is one-dimensional). We call such a vector $\vec{\mu} \in \mathbb{R}^Q$ a *normaliser*. The aim of this paper is to use a novel, linear-algebra based technique to compute normalisers more efficiently.

The suggestion in [2] was to take as normaliser the characteristic vector $[\![c]\!] \in \{0,1\}^Q$ of a so-called *cut* $c \subseteq Q$. To define this, let us write $\delta(q,w)$ for the set of states reachable from a state $q \in Q$ via the word $w \in \Sigma^*$. A *cut* is a set of states of the form $c = \delta(q,w)$ such that $\delta(q,wx) \neq \emptyset$ holds for all $x \in \Sigma^*$. If a cut does not exist or if $\mathcal{A}$ does not have accepting states, then we have $\vec{z} = \vec{0}$.

▶ **Example 3.** In the automaton $\mathcal{A}$ from Figure 1, we have a cut $c = \delta(q_0, aba) = \{q_0, q_2\}$. Hence its characteristic vector $\vec{\mu} = (1,0,1,0)^\top$ is a normaliser, allowing us to add the equation $\vec{\mu}^\top \vec{z} = \vec{z}_{q_0} + \vec{z}_{q_2} = 1$. Now the system is uniquely solvable: $\vec{z} = \frac{1}{3}(1,2,2,1)^\top$. The equation $\vec{z}_{q_0} + \vec{z}_{q_2} = 1$ is valid by an ergodicity argument: intuitively, given a finite word that leads to q_0 *and* q_2, a random infinite continuation will almost surely enable an accepting run. For instance, $\vec{z}_{q_0} = \frac{1}{3}$ is the probability that a random infinite word over $\{a,b\}$ has an odd number of as before the first b. (This holds despite the fact that the word $abbb\ldots$ is not accepted from q_0.)

In Proposition 14 we show that an efficient implementation of the algorithm from [2] for computing a cut runs in time $O(|Q|^5)$. Our goal is to find a normaliser $\vec{\mu}$ more efficiently.

The general idea is to move from a combinatorial problem, namely computing a set $c \subseteq Q$, to a continuous problem, namely computing a vector $\vec{\mu} \in \mathbb{R}^Q$. To illustrate this, note that since we can choose as $\vec{\mu}$ the characteristic vector of an arbitrary cut, we may also choose a convex combination of such vectors, leading to a normaliser $\vec{\mu}$ with entries other than 0 or 1.

The technical key ideas of this paper draw on the observation that for unambiguous automata with cuts, the transition matrices generate a semigroup of matrices whose spectral radii are all 1. (The spectral radius of a matrix is the largest absolute value of its eigenvalues.) This observation enables us to adopt techniques that have recently been devised by Protasov and Voynov [16] for the analysis of matrix semigroups with constant spectral radius. To the best of the authors' knowledge, such semigroups have not previously been connected to unambiguous automata. This transfer is the main contribution of this paper.

To sketch the gist of this technique, for any $a \in \Sigma$ write $M(a) \in \{0,1\}^{Q \times Q}$ for the transition matrix of the unambiguous automaton $\mathcal{A}$, define the average matrix $\overline{M} = \frac{1}{|\Sigma|}\sum_{a\in\Sigma} M(a)$, and let $\vec{y} = \overline{M}\vec{y} \in \mathbb{R}^Q$ be an eigenvector with eigenvalue 1 (the matrix $\overline{M}$ has such an eigenvector if $\mathcal{A}$ has a cut). Since the matrix semigroup, $\mathcal{S} \subseteq \{0,1\}^{Q\times Q}$, generated by the transition matrices $M(a)$ has constant spectral radius, it follows from [16] that one can efficiently compute an affine space $\mathcal{F} \subseteq \mathbb{R}^Q$ with $\vec{y} \in \mathcal{F}$ and $\vec{0} \notin \mathcal{F}$ such that for any $\vec{v} \in \mathcal{F}$ and any $M \in \mathcal{S}$ we have $M\vec{v} \in \mathcal{F}$. Using the fact that $\delta(q,wx)$ is a cut (for all $x \in \Sigma^*$) whenever $\delta(q,w)$ is a cut, one can show that all characteristic vectors of cuts have the same scalar product with all $\vec{v} \in \mathcal{F}$, i.e., all characteristic vectors of cuts are in the vector space orthogonal to $\mathcal{F}$. Indeed, we choose as normaliser $\vec{\mu}$ a vector that is orthogonal to $\mathcal{F}$. This linear-algebra computation can be carried out in time $O(|Q|^3)$. In the visualisation on the right of Figure 1, the characteristic vectors of cuts lie in the plane shaded in red, which is orthogonal to straight line $\mathcal{F}$ (blue).

▶ **Example 4.** In the automaton $\mathcal{A}$ from Figure 1, the vector $\vec{y} = (1,2,2,1)^\top$ satisfies $\overline{M}\vec{y} = \vec{y}$ where $\overline{M} = \frac{1}{2}(M(a) + M(b))$. The affine space $\mathcal{F} := \{\vec{y} + s(1,-1,-1,1)^\top \mid s \in \mathbb{R}\}$ has the mentioned closure properties, i.e., $M(a)\mathcal{F} \subseteq \mathcal{F}$ and $M(b)\mathcal{F} \subseteq \mathcal{F}$. Note that the vector $\vec{\mu}$ from Example 3 is indeed orthogonal to $\mathcal{F}$, i.e., $\vec{\mu}^\top(1,-1,-1,1)^\top = 0$.

However, to ensure that $\vec{\mu}$ is a valid normaliser, we need to restrict it further. To this end, we compute, for some state $q \in Q$, the set $Co(q) \subseteq Q$ of *co-reachable* states, i.e., states $r \in Q$ such that $\delta(q, w) \supseteq \{q, r\}$ holds for some $w \in \Sigma^*$. This requires a combinatorial algorithm, which is similar to a straightforward algorithm that would verify the unambiguousness of $\mathcal{A}$. Its runtime is quadratic in the number of transitions of $\mathcal{A}$, i.e., $O(|Q|^4)$ in the worst case. Then we restrict $\vec{\mu}$ such that $\vec{\mu}_q = 1$ and μ is non-zero only in entries that correspond to $Co(q)$. In the visualisation on the right of Figure 1, restricting some components of $\vec{\mu}$ to be 0 corresponds to the vectors in the shaded (red) plane that lie on the plane described by $q' = 0$ for all $q' \in Q \setminus Co(q)$.

▶ **Example 5.** We have $Co(q_0) = \{q_0, q_2\}$. So we restrict $\vec{\mu}$ to be of the form $(1, 0, x, 0)^\top$. Together with the equation $\vec{\mu}^\top(1, -1, -1, 1)^\top = 0$ this implies $\vec{\mu} = (1, 0, 1, 0)^\top$. The point is that, although this is the same vector computed via a cut in Example 3, the linear-algebra based computation of $\vec{\mu}$ is more efficient.

In the rest of the paper we analyse the general case of model checking a given Markov chain against a given unambiguous Büchi automaton. The efficiency gain we aim for with our technique can only be with respect to the automaton, not the Markov chain; nevertheless, we analyse in detail the runtime in terms of the numbers of states and transitions in both the automaton and the Markov chain. The main results are developed in Section 3. In Section 3.1 we describe the general approach from [2, 3]. In Section 3.2 we analyse the runtime of an efficient implementation of the algorithm from [2, 3] for computing a cut. Our main contribution lies in Section 3.3, where we develop a new approach for computing a normaliser, based on the mentioned spectral properties of the transition matrices in unambiguous automata. We close in Section 4 with a discussion. The full version of this paper [12] contains an appendix with proofs.

2 Preliminaries

We assume the reader to be familiar with basic notions of finite automata over infinite words and Markov chains, see, e.g., [9, 13]. In the following we provide a brief summary of our notation and a few facts related to linear algebra.

Finite automata. A *Büchi automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where Q is the finite set of states, $Q_0 \subseteq Q$ is the set of initial states, Σ is the finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, and $F \subseteq Q$ is the set of accepting states. We extend the transition function to $\delta : Q \times \Sigma^* \to 2^Q$ and to $\delta : 2^Q \times \Sigma^* \to 2^Q$ in the standard way. For $q \in Q$ we write $\mathcal{A}_q$ for the automaton obtained from $\mathcal{A}$ by making q the only initial state.

Given states $q, r \in Q$ and a finite word $w = a_0 a_1 \cdots a_{n-1} \in \Sigma^*$, a *run* for w from q to r is a sequence $q_0 q_1 \cdots q_n \in Q^{n+1}$ with $q_0 = q$, $q_n = r$ and $q_{i+1} \in \delta(q_i, a_i)$ for $i \in \{0, \ldots, n-1\}$. A *run* in $\mathcal{A}$ for an infinite word $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0 q_1 \cdots \in Q^\omega$ such that $q_0 \in Q_0$ and $q_{i+1} \in \delta(q_i, a_i)$ for all $i \in \mathbb{N}$. Run ρ is called *accepting* if $\inf(\rho) \cap F \neq \emptyset$ where $\inf(\rho) \subseteq Q$ is the set of states that occur infinitely often in ρ. The *language* $\mathcal{L}(\mathcal{A})$ of accepted words consists of all infinite words $w \in \Sigma^\omega$ that have at least one accepting run. $\mathcal{A}$ is called *unambiguous* if each word $w \in \Sigma^\omega$ has at most one accepting run. We use the acronym UBA for unambiguous Büchi automaton.

We define $|\delta| := |\{(q, r) \mid \exists\, a \in \Sigma : r \in \delta(q, a)\}|$, i.e., $|\delta| \leq |Q|^2$ is the number of transitions in $\mathcal{A}$ when allowing for multiple labels per transition. In [12, Appendix A] we give an example that shows that the number of transitions can be quadratic in $|Q|$, even for UBAs with a

strongly connected state space. We assume $|Q| \leq |\delta|$, as states without outgoing transitions can be removed. In this paper, Σ may be a large set (of states in a Markov chain), so it is imperative to allow for multiple labels per transition. We use a lookup table to check in constant time whether $r \in \delta(q, a)$ holds for given r, q, a.

A *diamond* is given by two states $q, r \in Q$ and a finite word w such that there exist at least two distinct runs for w from q to r. One can remove diamonds (see [12, Appendix B.1]):

▶ **Lemma 6.** *Given a UBA, one can compute in time $O(|\delta|^2|\Sigma|)$ a UBA of at most the same size, with the same language and without diamonds.*

For the rest of the paper, we assume that UBAs do not have diamonds.

Vectors and matrices. We consider vectors and square matrices indexed by a finite set S. We write (column) vectors $\vec{v} \in \mathbb{R}^S$ with arrows on top, and $\vec{v}^\top$ for the transpose (a row vector) of $\vec{v}$. The zero vector and the all-ones vector are denoted by $\vec{0}$ and $\vec{1}$, respectively. For a set $T \subseteq S$ we write $[\![T]\!] \in \{0,1\}^S$ for the characteristic vector of T, i.e., $[\![T]\!]_s = 1$ if $s \in T$ and $[\![T]\!]_s = 0$ otherwise. A matrix $M \in [0,1]^{S \times S}$ is called *stochastic* if $M\vec{1} = \vec{1}$, i.e., if every row of M sums to one. For a set $U \subseteq S$ we write $\vec{v}_U \in \mathbb{R}^U$ for the restriction of $\vec{v}$ to U. Similarly, for $T, U \subseteq S$ we write $M_{T,U}$ for the submatrix of M obtained by deleting the rows not indexed by T and the columns not indexed by U. The (directed) *graph* of a nonnegative matrix $M \in \mathbb{R}^{S \times S}$ has vertices $s \in S$ and edges (s,t) if $M_{s,t} > 0$. We may implicitly associate M with its graph and speak about graph-theoretic concepts such as reachability and strongly connected components (SCCs) in M.

Markov chains. A (finite-state discrete-time) *Markov chain* is a pair $\mathcal{M} = (S, M)$ where S is the finite set of states, and $M \in [0,1]^{S \times S}$ is a stochastic matrix that specifies transition probabilities. An *initial distribution* is a function $\iota : S \to [0,1]$ satisfying $\sum_{s \in S} \iota(s) = 1$. Such a distribution induces a probability measure $\Pr_\iota^{\mathcal{M}}$ on the measurable subsets of S^ω in the standard way, see for instance [1, chapter 10.1, page 758]. If ι is concentrated on a single state s, we may write $\Pr_s^{\mathcal{M}}$ for $\Pr_\iota^{\mathcal{M}}$. We write E for the set of edges in the graph of M. Note that $|S| \leq |E| \leq |S|^2$, as M is stochastic.

Solving linear systems. Let $\kappa \in [2,3]$ be such that one can multiply two $n \times n$-matrices in time $O(n^\kappa)$ (in other literature, κ is often denoted by ω). We assume that arithmetic operations cost constant time. One can choose $\kappa = 2.4$, see [14] for a recent result. One can check whether an $n \times n$ matrix is invertible in time $O(n^\kappa)$ [6]. Finally, one can solve a linear system with n equations using the Moore-Penrose pseudo-inverse [11] in time $O(n^\kappa)$ [15].

Spectral theory. The *spectral radius* of a matrix $M \in \mathbb{R}^{S \times S}$, denoted $\rho(M)$, is the largest absolute value of the eigenvalues of M. By the Perron-Frobenius theorem [4, Theorems 2.1.1, 2.1.4], if M is nonnegative then the spectral radius $\rho(M)$ is an eigenvalue of M and there is a nonnegative eigenvector $\vec{x}$ with $M\vec{x} = \rho(M)\vec{x}$. Such a vector $\vec{x}$ is called *dominant*. Further, if M is nonnegative and strongly connected then $\vec{x}$ is strictly positive in all components and the eigenspace associated with $\rho(M)$ is one-dimensional.

3 Algorithms

Given a Markov chain $\mathcal{M}$, an initial distribution ι, and a Büchi automaton $\mathcal{A}$ whose alphabet is the state space of $\mathcal{M}$, the *probabilistic model-checking problem* is to compute $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$.

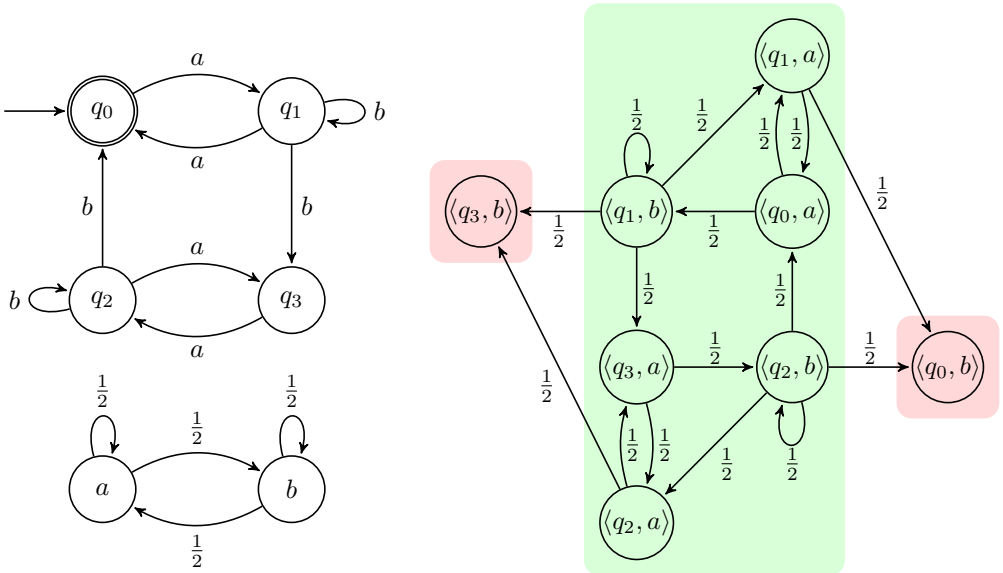

Figure 2 The UBA from Figure 1 and the Markov chain $\mathcal{M}$ on the left, and their product, B, on the right. The (single) accepting recurrent SCC is shaded green, and the two other SCCs are shaded red.

This problem is PSPACE-complete [8, 7], but solvable in polynomial time if $\mathcal{A}$ is deterministic. For UBAs a polynomial-time algorithm was described in [2, 3]. In this paper we obtain a faster algorithm (recall that E is the set of transitions in the Markov chain):

▶ **Theorem 7.** *Given a Markov chain $\mathcal{M} = (S, M)$, an initial distribution ι, and a UBA $\mathcal{A} = (Q, S, \delta, Q_0, F)$, one can compute $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$ in time $O(|Q|^\kappa|S|^\kappa + |Q|^3|E| + |\delta|^2|E|)$.*

Before we prove this theorem in Section 3.3, we describe the algorithm from [2, 3] and analyse the runtime of an efficient implementation.

3.1 The Basic Linear System

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

$$B_{\langle q,s\rangle,\langle q',s'\rangle} = \begin{cases} M_{s,s'} & \text{if } q' \in \delta(q,s) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Define $\vec{z} \in \mathbb{R}^{Q\times S}$ by $\vec{z}_{\langle q,s\rangle} = \Pr_s^{\mathcal{M}}(\mathcal{L}(\mathcal{A}_q))$. Then $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A})) = \sum_{q\in Q_0}\sum_{s\in S}\iota(s)\vec{z}_{\langle q,s\rangle}$. Lemma 4 in [3] implies that $\vec{z} = B\vec{z}$.

▶ **Example 8.** Consider the UBA $\mathcal{A}$ from Figure 1 and the two-state Markov chain $\mathcal{M}$ shown on the left of Figure 2. The weighted graph on the right of Figure 2 represents the matrix B, obtained from $\mathcal{A}$ and $\mathcal{M}$ according to Equation (1). It is natural to think of B as a product of $\mathcal{A}$ and $\mathcal{M}$. Notice that B is not stochastic: the sum of the entries in each row (equivalently, the total outgoing transition weight of a graph node) is not always one.

Although $\vec{z}$ is a solution the system of equations $\vec{\zeta} = B\vec{\zeta}$, this system does not uniquely identify $\vec{z}$. Indeed, any scalar multiple of $\vec{z}$ is a solution for these equations. To uniquely identify $\vec{z}$ by a system of linear equations, we need to analyse the SCCs of B.

All SCCs D satisfy $\rho(D) \leq 1$, see [3, Proposition 7]. An SCC D of B is called *recurrent* if $\rho(B_{D,D}) = 1$. It is called *accepting* if there is $\langle q, s\rangle \in D$ with $q \in F$.

▶ **Example 9.** The matrix B from Figure 2 has three SCCs, namely the two singleton sets $\{\langle q_0, b\rangle\}$ and $\{\langle q_3, b\rangle\}$, and $D = \{\langle q_0, a\rangle, \langle q_1, a\rangle, \langle q_1, b\rangle, \langle q_2, a\rangle, \langle q_2, b\rangle, \langle q_3, a\rangle\}$. Only D is recurrent; indeed, $\vec{y} = (\vec{y}_{\langle q_0,a\rangle}, \vec{y}_{\langle q_1,a\rangle}, \vec{y}_{\langle q_1,b\rangle}, \vec{y}_{\langle q_2,a\rangle}, \vec{y}_{\langle q_2,b\rangle}, \vec{y}_{\langle q_3,a\rangle})^\top = (2, 1, 3, 1, 3, 2)^\top$ is a dominant eigenvector with $B_{D,D}\vec{y} = \vec{y}$. Since q_0 is accepting, D is accepting recurrent.

Denote the set of accepting recurrent SCCs by $\mathcal{D}_+$ and the set of non-accepting recurrent SCCs by $\mathcal{D}_0$. By [3, Lemma 8], for $D \in \mathcal{D}_+$ we have $\vec{z}_d > 0$ for all $d \in D$, and for $D \in \mathcal{D}_0$ we have $\vec{z}_D = \vec{0}$. Hence, for $D \in \mathcal{D}_+$, there exists a *D-normaliser*, i.e., a vector $\vec{\mu} \in \mathbb{R}^D$ such that $\vec{\mu}^\top \vec{z}_D = 1$. This gives us a system of linear equations that identifies $\vec{z}$ uniquely [3]:

▶ **Lemma 10** (Lemma 12 in [3]). *Let $\mathcal{D}_+$ be the set of accepting recurrent SCCs, and $\mathcal{D}_0$ the set of non-accepting recurrent SCCs. For each $D \in \mathcal{D}_+$ let $\vec{\mu}(D)$ be a D-normaliser. Then $\vec{z}$ is the unique solution of the following linear system:*

$$\begin{aligned} \vec{\zeta} &= B\vec{\zeta} \\ \text{for all } D \in \mathcal{D}_+ : \quad \vec{\mu}(D)^\top \vec{\zeta}_D &= 1 \\ \text{for all } D \in \mathcal{D}_0 : \quad \vec{\zeta}_D &= \vec{0} \end{aligned} \tag{2}$$

Uniqueness follows from the fact that the system $\vec{\zeta} = B\vec{\zeta}$ describes the eigenspace of the dominant eigenvalue (here, 1) of a nonnegative strongly connected matrix (here, B), and such eigenspaces are one-dimensional. This leads to the following result:

▶ **Proposition 11.** *Suppose N is the runtime of an algorithm to calculate a normaliser for each accepting recurrent SCC. Then one can compute $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$ in time $O(|Q|^\kappa |S|^\kappa) + N$.*

Proof. Lemma 10 implies correctness of the following procedure to calculate $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$:

1. Set up the matrix B from Equation (1).
2. Compute the SCCs of B.
3. For each SCC C, check whether C is recurrent.
4. For each accepting recurrent SCC D, compute its D-normaliser $\vec{\mu}(D)$.
5. Compute $\vec{z}$ by solving the linear system (2) in Lemma 10.
6. Compute $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A})) = \sum_{s\in S} \sum_{q\in Q_0} \iota(s)\vec{z}_{q,s}$.

One can set up B in time $O(|Q|^2|S|^2)$. Using Tarjan's algorithm one can compute the SCCs of B in time linear in the vertices and edges of B, hence in $O(|Q|^2|S|^2)$ [17]. One can find those SCCs D which are recurrent in time $O(|Q|^\kappa|S|^\kappa)$ by checking if $I - B_{D,D}$ is invertible. The linear system (2) has $O(|Q||S|)$ equations, and thus can be solved in time $O(|Q|^\kappa|S|^\kappa)$. Hence the total runtime is $O(|Q|^\kappa|S|^\kappa) + N$. ◀

In Section 3.2 we describe the combinatorial, *cut* based, approach from [2, 3] to calculating D-normalisers and analyse its complexity. In Section 3.3 we describe a novel linear-algebra based approach, which is faster in terms of the automaton.

3.2 Calculating D-Normalisers Using Cuts

For the remainder of the paper, let D be an accepting recurrent SCC. A *fibre over $s \in S$* is a subset of D of the form $\alpha \times \{s\}$ for some $\alpha \subseteq Q$. Given a fibre $f = \alpha \times \{s\}$ and a state $s' \in S$, if $M_{s,s'} > 0$ we define the fibre $f \triangleright s'$ as follows:

$$f \triangleright s' := \{\langle q, s'\rangle \mid q \in \delta(\alpha, s)\} \cap D.$$

If $M_{s,s'} = 0$, then $f \triangleright s'$ is undefined, and for $w \in S^*$ we define $f \triangleright w = f$ if $w = \varepsilon$ and $f \triangleright ws' = (f \triangleright w) \triangleright s'$. If $f = \{d\}$ for some $d \in D$ we may write $d \triangleright s'$ for $f \triangleright s'$.

We call a fibre c a *cut* if $c = d \triangleright v$ for some $v \in S^*$ and $d \in D$, and $c \triangleright w \neq \emptyset$ for all $w \in S^*$ whenever $c \triangleright w$ is defined. Note that if c is a cut then so is $c \triangleright w$ whenever it is defined. Given a cut $c \subseteq D$ we call its characteristic vector $[\![c]\!] \in \{0,1\}^D$ a *cut vector*. In the example in Figure 2, it is easy to see that $\{\langle q_1, b\rangle\} = \langle q_0, a\rangle \triangleright b$ is a cut.

▶ **Lemma 12** (Lemma 10 in [3]). *There exists a cut. Any cut vector $\vec{\mu}$ is a normaliser, i.e., $\vec{\mu}^\top \vec{z}_D = 1$.*

Loosely speaking, $\vec{\mu}^\top \vec{z}_D \leq 1$ follows from unambiguousness, and $\vec{\mu}^\top \vec{z}_D \not< 1$ follows from an ergodicity argument (intuitively, all states in the cut are almost surely visited infinitely often). The following lemma is the basis for the cut computation algorithm in [2, 3]:

▶ **Lemma 13** (Lemma 17 in [3]). *Let $D \subseteq Q \times S$ be a recurrent SCC. Let $d \in D$. Suppose $w \in S^*$ is such that $d \triangleright w \ni d$ is not a cut. Then there are $v \in S^*$ and $e \neq d$ with $d \triangleright v \supseteq \{d, e\}$ and $e \triangleright w \neq \emptyset$. For any such e, $d \triangleright w \cap e \triangleright w = \emptyset$. Hence $d \triangleright vw \supseteq \{d, e\} \triangleright w \supsetneq d \triangleright w$.*

This suggests a way of generating an increasing sequence of fibres, culminating in a cut. We prove the following proposition:

▶ **Proposition 14.** *Let $D \subseteq Q \times S$ be a recurrent SCC. Denote by T the set of edges in $B_{D,D}$. One can compute a cut in time $O(|Q|^2|\delta||D| + |\delta||T|)$.*

Define, for some $d = \langle q, s\rangle \in D$, its *co-reachability* set $Co(d) \subseteq D$: it consists of those $e \in D$ such that there exists a word w with $\{d, e\} \subseteq d \triangleright w$. Note that $Co(d)$ is a fibre over s. In the example of Figure 2 we have that $Co(\langle q_0, a\rangle) = \{\langle q_0, a\rangle, \langle q_2, a\rangle\}$, with $\{\langle q_0, a\rangle, \langle q_2, a\rangle\} \in \langle q_0, a\rangle \triangleright ba$. The following lemma (proof in [12, Appendix B.2]) gives a bound on the time to compute $Co(d)$:

▶ **Lemma 15.** *One can compute $Co(d)$ in time $O(|Q||D|+|\delta||T|)$. Moreover, one can compute in time $O(|Q|^2|D| + |\delta||T|)$ a list $(CoPath(d)(e))_{e \in Co(d)}$ such that $CoPath(d)(e) \in S^*$ and $\{d, e\} \subseteq d \triangleright CoPath(d)(e)$ and $|CoPath(d)(e)| \leq |Q||D|$.*

The lemma is used in the proof of Proposition 14:

Proof sketch of Proposition 14. Starting from a singleton fibre $\{d\}$, where $d = \langle q, s\rangle \in D$ is chosen arbitrarily, we keep looking for words $v \in S^*$ that have the properties described in Lemma 13 to generate larger fibres $d \triangleright w$:

1. $w := \varepsilon$ (the empty word)
2. while $\exists\, v \in S^*$ and $\exists\, e \neq d$ such that $d \triangleright v \supseteq \{d, e\}$ and $e \triangleright w \neq \emptyset$:
   $w := vw$
3. return $d \triangleright w$.

By [2, Lemma 18] the algorithm returns a cut. In every loop iteration the fibre $d \triangleright w$ increases, so the loop terminates after at most $|Q|$ iterations. For efficiency we calculate $Co(d)$ and $CoPath(d)$ using Lemma 15, and we use dynamic programming to maintain the set, *Survives*, of those $e \in D$ for which $e \triangleright w \neq \emptyset$ holds. Whenever a prefix v is added to w, we update *Survives* by processing v backwards. This leads to the following algorithm:

1. Calculate $Co(d)$ and $CoPath(d)$ using Lemma 15
2. $w := \varepsilon$; *Survives* $:= (Q \times \{s\}) \cap D$

3. while $\exists\, e \in \mathit{Co}(d) \setminus \{d\}$ such that $e \in \mathit{Survives}$:
$v_0 = s$; $v_1 \dots v_n := \mathit{CoPath}(d)(e)$
for $i = n, n-1, \dots, 1$:
$\mathit{Survives} := \{\langle p, v_{i-1}\rangle \in D \mid (\delta(p, v_{i-1}) \times \{v_i\}) \cap \mathit{Survives} \neq \emptyset\}$
$w := v_1 \dots v_n w$
4. return $d \triangleright w$

The runtime analysis is in [12, Appendix B.2]. ◀

▶ **Example 16.** Letting $d = \langle q_0, a\rangle$ and $e = \langle q_2, a\rangle$ we have $\mathit{Co}(d) = \{d, e\}$ with $\mathit{CoPath}(d)(d) = \varepsilon$ and $\mathit{CoPath}(d)(e) = baa$. Initially we have $\mathit{Survives} = Q \times \{a\}$. In the first iteration the algorithm can only pick e. The inner loop updates *Survives* first to $\{q_0, q_1, q_2, q_3\} \times \{a\}$ (i.e., to itself), then to $\{q_1, q_2\} \times \{b\}$, and finally to $\{q_0, q_3\} \times \{a\}$. Now $(\mathit{Co}(d) \setminus d) \cap \mathit{Survives}$ is empty and the loop terminates. The algorithm returns the cut $d \triangleright baa = \{d, e\}$.

Applying Proposition 14 to the general procedure (Proposition 11) leads to the following result on the combinatorial approach:

▶ **Theorem 17.** *Given a Markov chain $\mathcal{M} = (S, M)$, an initial distribution ι, and a UBA $\mathcal{A} = (Q, S, \delta, Q_0, F)$, one can compute $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$ in time $O(|Q|^\kappa |S|^\kappa + |Q|^3|\delta||S| + |\delta|^2|E|)$.*

3.3 Calculating D-Normalisers Using Linear Algebra

Recall that D is an accepting recurrent SCC. For $t \in S$ define the matrix $\Delta(t) \in \{0,1\}^{D \times D}$ as follows:

$$\Delta(t)_{\langle q,s\rangle,\langle q',s'\rangle} := \begin{cases} 1 & \text{if } s' = t,\ M_{s,t} > 0, \text{ and } q' \in \delta(q,s) \\ 0 & \text{otherwise} \end{cases}$$

Note that the graph of $\Delta(t)$ contains exactly the edges of the graph of $B_{D,D}$ that end in vertices in $Q \times \{t\}$. If $M_{s,t} > 0$ holds for all pairs (s,t), then the matrices $(\Delta(t))_{t \in S}$ generate a semigroup of matrices, all of which have spectral radius 1. Such semigroups were recently studied by Protasov and Voynov [16]. Specifically, Theorem 5 in [16] shows that there exists an affine subspace $\mathcal{F}$ of $\mathbb{R}^D$ which excludes $\vec{0}$ and is invariant under multiplication by matrices from the semigroup. Moreover, they provide a way to compute this affine subspace efficiently. One can show that cut vectors are orthogonal to $\mathcal{F}$. The key idea of our contribution is to generalise cut vectors to *pseudo-cuts*, which are vectors $\vec{\mu} \in \mathbb{R}^D$ that are orthogonal to $\mathcal{F}$. We will show (in Lemma 20 below) how to derive a D-normaliser based on a pseudo-cut that is non-zero only in components that are in a co-reachability set $\mathit{Co}(d)$ (from Lemma 15).

If $M_{s,t} = 0$ holds for some s, t (which will often be the case in model checking), then $\Delta(s)\Delta(t)$ is the zero matrix, which has spectral radius 0, not 1. Therefore, the results of [16] are not directly applicable and we have to move away from matrix semigroups. In the following we re-develop and generalise parts of the theory of [16] so that the paper is self-contained and products of $\Delta(s)\Delta(t)$ with $M_{s,t} = 0$ are not considered.

Let $w = s_1 s_2 \dots s_n \in S^*$. Define $\Delta(w) = \Delta(s_1)\Delta(s_2)\cdots\Delta(s_n)$. We say w is *enabled* if $M_{s_i,s_{i+1}} > 0$ holds for all $i \in \{1, \dots, n-1\}$. If $f \subseteq D$ is a fibre over s such that sw is enabled, we have $[\![f \triangleright w]\!]^\top = [\![f]\!]^\top \Delta(w)$. We overload the term *fibre over s* to describe any vector $\vec{\mu} \in \mathbb{R}^D$ such that $\vec{\mu}_{\langle q,s'\rangle} = 0$ whenever $s' \neq s$. We define *pseudo-cuts over s* to be fibres $\vec{\mu}$ over s such that $\vec{\mu}^\top \Delta(w) \vec{z} = \vec{\mu}^\top \vec{z}$ holds for all $w \in S^*$ such that sw is enabled. Let $c \subseteq Q \times \{s\}$ be a cut with sw enabled. Then $c \triangleright w$ is a cut, and $[\![c]\!]^\top \Delta(w) \vec{z} = 1 = [\![c]\!]^\top \vec{z}$ holds by Lemma 12. It follows that cut vectors are pseudo-cuts.

▶ **Example 18.** Since $c = \{\langle q_0, a\rangle, \langle q_2, a\rangle\}$ from Example 16 is a cut, $\lceil c \rceil$ is a pseudo-cut over a. Pseudo-cuts do not need to be combinations of cut vectors: although the fibre $f = \{\langle q_0, a\rangle, \langle q_1, a\rangle\}$ is not a cut, $\lceil f \rceil$ is a pseudo-cut over a.

Fix some $d = \langle q, s\rangle \in D$. Recall that $Co(d)$ consists of those $e \in D$ such that there exists a word w with $\{d, e\} \subseteq d \triangleright w$. We define *$Co(d)$-pseudo-cuts* to be pseudo-cuts $\vec{\mu}$ over s such that $\vec{\mu}_d \neq 0$ and $\vec{\mu}_e = 0$ holds for all $e \notin Co(d)$.

▶ **Example 19.** Any cut vector is a $Co(d)$-pseudo-cut for some $d \in D$, by definition, and so are scalar multiples of cut vectors. The vector $\lceil f \rceil$ in Example 18, however, is not a $Co(d)$-pseudo-cut, since $\langle q_1, a\rangle \notin Co(\langle q_0, a\rangle)$ and $\langle q_0, a\rangle \notin Co(\langle q_1, a\rangle)$.

From a $Co(d)$-pseudo-cut we can easily derive a D-normaliser:

▶ **Lemma 20.** *Let $\vec{\mu} \in \mathbb{R}^D$ be a $Co(d)$-pseudo-cut. Then $\frac{1}{\vec{\mu}_d}\vec{\mu}$ is a D-normaliser.*

Proof. Let w be an enabled word in M such that $d \triangleright w$ is a cut containing d. Such a word exists (see the proof sketch of Proposition 14). Since $(\lceil d \rceil^\top \Delta(w))^\top = \lceil d \triangleright w \rceil$ is a D-normaliser (by Lemma 12), it suffices to prove that $\frac{1}{\vec{\mu}_d}\vec{\mu}^\top \vec{z} = \lceil d \rceil^\top \Delta(w)\vec{z}$.

We can write $\vec{\mu}$ as $\sum_{d' \in Co(d)} \vec{\mu}_{d'} \lceil d' \rceil$, so $\vec{\mu}^\top \Delta(w) = \sum_{d' \in Co(d)} \vec{\mu}_{d'} \lceil d' \rceil^\top \Delta(w)$. For any $d' \in Co(d) \setminus \{d\}$, let w' be such that $\{d, d'\} \subseteq d \triangleright w'$. Now we see that $d' \in d \triangleright ww'$, and since $d \triangleright w$ is a cut so are $d \triangleright ww'$ and $d \triangleright ww'w$. Thus,

$$\lceil d \rceil^\top \Delta(w)\vec{z} \;=\; \lceil d \rceil^\top \Delta(ww'w)\vec{z} \;\geq\; \lceil d \rceil^\top \Delta(w)\vec{z} + \lceil d' \rceil^\top \Delta(w)\vec{z},$$

which implies $\lceil d' \rceil^\top \Delta(w)\vec{z} = 0$ for every $d' \in Co(d) \setminus \{d\}$. This means that

$$\vec{\mu}^\top \Delta(w)\vec{z} = \sum_{d' \in Co(d)} \vec{\mu}_{d'} \lceil d' \rceil^\top \Delta(w)\vec{z} = \vec{\mu}_d \lceil d \rceil^\top \Delta(w)\vec{z}.$$

Since $\vec{\mu}$ is a pseudo-cut, this implies that $\frac{1}{\vec{\mu}_d}\vec{\mu}^\top \vec{z} = \frac{1}{\vec{\mu}_d}\vec{\mu}^\top \Delta(w)\vec{z} = \lceil d \rceil^\top \Delta(w)\vec{z}$. ◀

By Lemma 20, to find a D-normaliser it suffices to find a $Co(d)$-pseudo-cut. Fix a dominant eigenvector $\vec{y}$ of $B_{D,D}$ so that $\vec{y}$ is strictly positive in all components. One can compute such $\vec{y}$ in time $O(|D|^\kappa)$. By [2, Lemma 8] the vector $\vec{z}_D$ is also a dominant eigenvector of $B_{D,D}$, hence $\vec{y}$ and $\vec{z}_D$ (the latter of which is yet unknown) are scalar multiples. In order to compute a $Co(d)$-pseudo-cut, we compute a basis for the space spanned by $\Delta(w)\vec{y}$ for all enabled words w. We use a technique similar to the one employed by Tzeng in [18] for checking equivalence of probabilistic automata. To make this more efficient, we compute separate basis vectors for each $s \in S$. Define $\Delta'(t) \in \{0,1\}^{D \times D}$ as $\Delta'(t)_{\langle q_1, s_1\rangle, \langle q_2, s_2\rangle} = 1$ if $q_1 = q_2$ and $s_1 = s_2 = t$ and 0 otherwise. Note that $\Delta(s)\Delta'(s) = \Delta(s)$ holds for all $s \in S$.

▶ **Lemma 21.** *Suppose $\vec{y} = B_{D,D}\vec{y}$ is given. Denote by $V(s) \subseteq \mathbb{R}^D$ the vector space spanned by the vectors $\Delta'(s)\Delta(w)\vec{y}$ for $w \in S^*$ and $s \in S$. Let $Q_{D,t} = (Q \times \{t\}) \cap D$ and let $E(t) = \{(s,t) \mid M_{s,t} > 0\}$ be the set of edges in M that end in t. One can compute a basis $R(s)$ of $V(s)$ for all $s \in S$ in time $O(|Q|^2 \sum_{t \in S} |Q_{D,t}||E(t)|)$, where for each $\vec{r} \in R(s)$ we have $\vec{r} = \Delta'(s)\Delta(w)\vec{y}$ for some enabled word sw.*

Proof sketch. Fix an arbitrary total order $<_S$ on S. We define a total order $\ll_S$ on S^* as the "shortlex" order but with words read from right to left. That is, the empty word ε is the smallest element, and for $v, w \in S^*$ and $s, t \in S$, we have $vs \ll_S wt$ if (1) $|vs| < |wt|$ or (2) $|vs| = |wt|$ and $s <_S t$ or (3) $|vs| = |wt|$ and $s = t$ and $v \ll_S w$.

We use a technique similar to the one by Tzeng in [18]. At every step in the algorithm, *worklist* is a set of pairs $(sw, \Delta'(s)\Delta(w)\vec{y})$. We write $\min_{\ll_S}(\mathit{worklist})$ to denote the pair in *worklist* where sw is minimal with respect to $\ll_S$.

1. for each $s \in S$, let $R(s) := \{\Delta'(s)\vec{y}\}$ and $R(s)_\perp := \{\Delta'(s)\vec{y}\}$
2. *worklist* $:= \{(st, \Delta'(s)\Delta(t)\vec{y}) \mid M_{s,t} > 0\}$
3. while *worklist* $\neq \emptyset$:
   $(tw, \vec{u}) := \min_{\ll_S}(\textit{worklist})$; *worklist* $:=$ *worklist* $\setminus \{(tw, \vec{u})\}$
   Using the Gram-Schmidt process[1], let $\vec{u}_\perp$ be the orthogonalisation of $\vec{u}$ against $R_\perp(t)$
   if $\vec{u}_\perp \neq \vec{0}$, i.e., if $\vec{u}$ is linearly independent of $R_\perp(t)$:
      $R(t) := R(t) \cup \{\vec{u}\}$ and $R_\perp(t) := R_\perp(t) \cup \{\vec{u}_\perp\}$
      worklist $:=$ *worklist* $\cup \{(stw, \Delta'(s)\Delta(t)\vec{u}) \mid M_{s,t} > 0\}$
4. return $R(s)$ for all $s \in S$

At any point and for all $s \in S$, the sets $R(s)$ and $R(s)_\perp$ span the same vector space, and this space is a subspace of $V(s)$. The sets $R(s)$ and $R(s)_\perp$ consist of linearly independent fibres over s, and these fibres are possibly nonzero only in the $Q_{D,s}$-components. Hence $|\bigcup_{s\in S} R(s)| \leq |D|$ and thus there are at most $|D|$ iterations of the while loop that increase *worklist*. At every iteration where $\vec{u}$ is dependent on $R(t)_\perp$ the set *worklist* decreases by one, and therefore the algorithm terminates. In [12, Appendix B.3] we prove that in the end we have that $R(s)$ spans $V(s)$, and we analyse the runtime. ◀

▶ **Example 22.** Let us return to our running example. We see that the vector $\vec{y} = (\vec{y}_{\langle q_0,a\rangle}, \vec{y}_{\langle q_1,a\rangle}, \vec{y}_{\langle q_1,b\rangle}, \vec{y}_{\langle q_2,a\rangle}, \vec{y}_{\langle q_2,b\rangle}, \vec{y}_{\langle q_3,a\rangle})^\top = (2,1,3,1,3,2)^\top$ is a dominant eigenvector of $B_{D,D}$. Fix the order $a <_S b$. Step 1 initialises $R(a)$ to $\{\Delta'(a)\vec{y}\}$ and $R(b)$ to $\{\Delta'(b)\vec{y}\}$, where $\Delta'(a)\vec{y} = (2,1,0,1,0,2)^\top$ and $\Delta'(b)\vec{y} = (0,0,3,0,3,0)^\top$. Step 2 computes $\Delta'(a)\Delta(a)\vec{y} = (1,2,0,2,0,1)^\top$, which is linearly independent of $\Delta'(a)\vec{y}$. However, $\Delta'(b)\Delta(a)\vec{y} = (0,0,3,0,3,0)^\top = \Delta'(b)\vec{y}$. Also, $\Delta'(a)\Delta(b)\vec{y} = (3,0,0,0,0,3)^\top = 2\Delta'(a)\vec{y} - \Delta'(a)\Delta(a)\vec{y}$ and $\Delta'(b)\Delta(b)\vec{y} = (0,0,3,0,3,0)^\top = \Delta'(b)\vec{y}$. One can check that $\Delta'(a)\Delta(aa)\vec{y} = \Delta'(a)\vec{y}$ and $\Delta'(b)\Delta(aa)\vec{y} = \Delta'(b)\vec{y}$. Hence the algorithm returns $R(a) = \{(2,1,0,1,0,2)^\top, (1,2,0,2,0,1)^\top\}$ and $R(b) = \{(0,0,3,0,3,0)^\top\}$.

Fix $d = \langle q, s\rangle \in D$ for the rest of the paper. The following lemma characterises $Co(d)$-pseudo-cuts in a way that is efficiently computable:

▶ **Lemma 23.** *A vector $\vec{\mu} \in \mathbb{R}^D$ with $\vec{\mu}_d = 1$ and $\vec{\mu}_e = 0$ for all $e \notin Co(d)$ is a $Co(d)$-pseudo-cut if and only if $\vec{\mu}^\top \vec{r} = \vec{\mu}^\top \vec{y}$ holds for all $\vec{r} \in R(s)$.*

For an intuition of the proof, consider the affine space, $\mathcal{F} \subseteq \mathbb{R}^D$, affinely spanned by those $\Delta'(s)\Delta(w)\vec{y}$ for which sw is enabled. This affine space was alluded to in the beginning of this subsection and is visualised as a blue straight line on the right of Figure 1. The shaded plane in this figure is the vector space of pseudo-cuts over s. This space is orthogonal to $\mathcal{F}$. The following lemma says that $\mathcal{F}$ is affinely spanned by the points in $R(s)$. This strengthens the property of $R(s)$ in Lemma 21 where $R(s)$ was defined to span a *vector* space.

▶ **Lemma 24.** *Let $w \in S^*$ be such that sw is enabled. By the definition of $R(s)$ there are $\gamma_{\vec{r}} \in \mathbb{R}^D$ for each $\vec{r} \in R(s)$ such that $\Delta'(s)\Delta(w)\vec{y} = \sum_{\vec{r}\in R(s)} \gamma_{\vec{r}}\vec{r}$. We have $\sum_{\vec{r}\in R(s)} \gamma_{\vec{r}} = 1$.*

Proof. Let c be a cut containing d. Since $R(s)$ is a basis, for any $\vec{r} = \Delta'(s)\Delta(w_{\vec{r}})\vec{y} \in R(s)$ the word $sw_{\vec{r}}$ is enabled. Therefore, $c \triangleright w_{\vec{r}}$ is a cut and by Lemma 12 we have $\lceil c \triangleright w_{\vec{r}} \rceil^\top \vec{y} = \lceil c \rceil^\top \vec{y}$.

[1] For good numerical stability, one should use the so-called Modified Gram-Schmidt process [10, Chapter 19].

Hence $[c]^\top \vec{r} = [c]^\top \Delta'(s)\Delta(w_{\vec{r}})\vec{y} = [c]^\top \Delta(w_{\vec{r}})\vec{y} = [c \triangleright w_{\vec{r}}]^\top \vec{y} = [c]^\top \vec{y}$. Moreover, we have:

$$\begin{aligned}
[c]^\top \vec{y} &= [c]^\top \Delta(w)\vec{y} && \text{since } sw \text{ is enabled and by Lemma 12}\\
&= [c]^\top \Delta'(s)\Delta(w)\vec{y} && \text{since } [c] \text{ is a fibre over } s\\
&= [c]^\top \sum_{\vec{r}\in R(s)} \gamma_{\vec{r}}\vec{r} && \text{by the definition of } \gamma_{\vec{r}}\\
&= [c]^\top \vec{y} \sum_{\vec{r}\in R(s)} \gamma_{\vec{r}} && \text{as argued above.}
\end{aligned}$$

Therefore, $\sum_{\vec{r}\in R(s)} \gamma_{\vec{r}} = 1$. ◀

Now we can prove Lemma 23:

Proof of Lemma 23. For the "if" direction, let w be such that sw is enabled, and it suffices to show that $\vec{\mu}^\top \Delta(w)\vec{y} = \vec{\mu}^\top \vec{y}$. By Lemma 24 there are $\gamma_{\vec{r}}$ such that $\Delta'(s)\Delta(w)\vec{y} = \sum_{\vec{r}\in R(s)} \gamma_{\vec{r}}\vec{r}$ and $\sum_{\vec{r}\in R(s)} \gamma_{\vec{r}} = 1$. We have:

$$\vec{\mu}^\top \Delta(w)\vec{y} \;=\; \vec{\mu}^\top \Delta'(s)\Delta(w)\vec{y} \;=\; \sum_{\vec{r}\in R(s)} \gamma_{\vec{r}}\vec{\mu}^\top \vec{r} \;=\; \sum_{\vec{r}\in R(s)} \gamma_{\vec{r}}\vec{\mu}^\top \vec{y} \;=\; \vec{\mu}^\top \vec{y}\,,$$

where the last equality is from Lemma 24.

For the "only if" direction, suppose $\vec{\mu}$ is a $Co(d)$-pseudo-cut. Let $\vec{r} = \Delta'(s)\Delta(w_{\vec{r}})\vec{y} \in R(s)$. Then $sw_{\vec{r}}$ is enabled and $\vec{\mu}^\top \vec{r} = \vec{\mu}^\top \Delta'(s)\Delta(w_{\vec{r}})\vec{y} = \vec{\mu}^\top \Delta(w_{\vec{r}})\vec{y} = \vec{\mu}^\top \vec{y}$. ◀

▶ **Example 25.** In Example 16 we derived that $\vec{y} = (2,1,3,1,3,2)^\top$ and $R(a) = \{(2,1,0,1,0,2)^\top, (1,2,0,2,0,1)^\top\}$. The cut vector $\vec{\mu} = (1,0,0,1,0,0)^\top$ from Example 18 satisfies $\vec{\mu}^\top \vec{r} = 3 = \vec{\mu}^\top \vec{y}$ for both $\vec{r} \in R(a)$.

Using Lemmas 15, 21 and 23 we obtain:

▶ **Proposition 26.** *Let $D \subseteq Q \times S$ be a recurrent SCC. Denote by T_D the set of edges of $B_{D,D}$. For $t \in S$, let $E(t)$ denote the set of edges of M that end in t, and let $Q_{D,t} = (Q \times \{t\}) \cap D$. Let $d = \langle q, s\rangle \in D$. One can compute a $Co(d)$-pseudo-cut in time $O(|D|^\kappa + |Q||D| + |\delta||T_D| + |Q|^2 \sum_{t\in S} |Q_{D,t}||E(t)|)$.*

Now our main result follows, which we restate here:

▶ **Theorem 7.** *Given a Markov chain $\mathcal{M} = (S, M)$, an initial distribution ι, and a UBA $\mathcal{A} = (Q, S, \delta, Q_0, F)$, one can compute $\Pr_\iota^{\mathcal{M}}(\mathcal{L}(\mathcal{A}))$ in time $O(|Q|^\kappa |S|^\kappa + |Q|^3|E| + |\delta|^2|E|)$.*

4 Discussion

We have analysed two algorithms for computing normalisers: the cut-based one by Baier et al. [2, 3], and a new one, which draws from techniques by Protasov and Voynov [16] for the analysis of matrix semigroups. The first approach is purely combinatorial, and in terms of the automaton, an efficient implementation runs in time $O(|Q|^3|\delta| + |\delta|^2) = O(|Q|^3|\delta|)$ (Proposition 14).

The second approach combines a linear-algebra component to compute $R(s)$ with a combinatorial algorithm to compute the co-reachability set $Co(d)$. In terms of the automaton, the linear-algebra component runs in time $O(|Q|^3)$ (Lemma 21), while the combinatorial part runs in time $O(|\delta|^2)$, leading to an overall runtime of $O(|Q|^3 + |\delta|^2)$. Note that for all $r \in [1,2]$, if $|\delta| = \Theta(|Q|^r)$ then the second approach is faster by at least a factor of $|Q|$.

Although it is not the main focus of this paper, we have analysed also the model-checking problem, where a non-trivial Markov chain is part of the input. The purely combinatorial algorithm runs in time $O(|Q|^\kappa|S|^\kappa + |Q|^3|\delta||S| + |\delta|^2|E|)$, and the linear-algebra based algorithm in time $O(|Q|^\kappa|S|^\kappa + |Q|^3|E| + |\delta|^2|E|)$. There are cases in which the latter is asymptotically worse, but not if $\kappa = 3$ (i.e., solving linear systems in a normal way such as Gaussian elimination) or if $|E|$ is $O(|S|)$.

It is perhaps unsurprising that a factor of $|\delta|^2$ from the computation of $Co(d)$ occurs in the runtime, as it also occurs when one merely verifies the unambiguousness of the automaton, by searching the product of the automaton with itself. Can the factor $|\delta|^2$ (which may be quartic in $|Q|$) be avoided?

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

2 Christel Baier, Stefan Kiefer, Joachim Klein, Sascha Klüppelholz, David Müller, and James Worrell. Markov chains and unambiguous Büchi automata. In *Proceedings of Computer Aided Verification (CAV)*, volume 9779 of *LNCS*, pages 23–42, 2016.

3 Christel Baier, Stefan Kiefer, Joachim Klein, Sascha Klüppelholz, David Müller, and James Worrell. Markov chains and unambiguous automata. Draft journal submission. Available at https://arxiv.org/abs/1605.00950, 2019.

4 Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.

5 Manuel Bodirsky, Tobias Gärtner, Timo von Oertzen, and Jan Schwinghammer. Efficiently Computing the Density of Regular Languages. In *LATIN 2004: Theoretical Informatics*, pages 262–270. Springer, 2004.

6 James R. Bunch and John E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28:231–236, 1974.

7 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. Springer, 2004.

8 Costas Courcoubetis and Mihalis Yannakakis. The Complexity of Probabilistic Verification. *Journal of the ACM*, 42(4):857–907, 1995.

9 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*. Springer, 2002.

10 Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, second edition, 2002.

11 M. James. The generalised inverse. *The Mathematical Gazette*, 62:109–114, 1978.

12 Stefan Kiefer and Cas Widdershoven. Efficient Analysis of Unambiguous Automata Using Matrix Semigroup Techniques (full version). arXiv:1906.10093, 2016. URL: `http://arxiv.org/abs/1906.10093`.

13 Vidyadhar G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.

14 François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC'14, pages 296–303. ACM, 2014.

15 Marko D. Petković and Predrag S. Stanimirović. Generalised matrix inversion is not harder than matrix multiplication. *Journal of Computational and Applied Mathematics*, 230:270–282, 2009.

16 V.Yu. Protasov and A.S. Voynov. Matrix semigroups with constant spectral radius. *Linear Algebra and its Applications*, 513:376–408, 2017.

17 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

18 Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.