


Determinisation and Unambiguisation of Polynomially-Ambiguous Rational Weighted Automata

Ismaël Jecker

University of Franche-Comté, Besançon, France

Filip Mazowiecki 

University of Warsaw, Poland

David Purser 

University of Liverpool, UK

Abstract

We study the determinisation and unambiguisation problems of weighted automata over the rational field: Given a weighted automaton, can we determine whether there exists an equivalent deterministic, respectively unambiguous, weighted automaton? Recent results by Bell and Smertnig show that the problem is decidable, however they do not provide any complexity bounds. We show that both problems are **in PSPACE for polynomially-ambiguous weighted automata**.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Weighted automata, determinisation problem, finite & polynomial ambiguity

1 Introduction

Weighted automata are a popular model of computation assigning values to words [13]. The domain of these values has a semiring structure (a structure with addition and product that can define matrix multiplication). Two popular domains are: the field of rationals; and the min-plus semiring over naturals extended with $+\infty$ (also known as the tropical semiring). Depending on the domain the values have different interpretation. The most intuitive setting, called *probabilistic automata* [24], is when the words are assigned their probability of being accepted (this is a special case of the rational field domain).

We investigate the *determinisation problem* (also known as the sequentiality problem): given a weighted automaton decide if there is an equivalent deterministic weighted automaton. If it is the case we say that the automaton is determinisable. The problem can be stated for any semiring, but it is very different depending on the choice of the domain (see the survey [22]). For example, if the semiring is the Boolean semiring then weighted automata coincide with finite automata and the problem trivialises as every automaton can be determinised. Recently, the problem has been shown to be decidable for the rational field, but no complexity bounds were provided [3]. For the tropical semiring determinisation remains an intriguing open problem: decidability is known only for weighted automata with bounded ambiguity [20, 22].

Classes with bounded ambiguity can be defined for any automata model (not only weighted automata). The simplest and most studied class is the class of *unambiguous automata*, where every word has at most one accepting run (but the automaton does not have to be deterministic). This class has received a lot of attention for many automata models (see e.g. the survey [8]). Unambiguous weighted automata are mathematically an elegant class of functions, as they capture functions that use only the semiring product. A lot of research on the determinisation problem focused on the subproblem when the input automaton is unambiguous [23, 19]. In this case automata that can be determinised are characterised by forbidding a simple pattern in the automaton, called *twins property*, which can be detected in polynomial time. Due to these result papers often focus on the *unambiguisation problem*, i.e.

whether there exist an equivalent unambiguous automaton. The mentioned work deals with the tropical semiring and its variants. A similar characterisation also works over the rational fields, which we describe in Appendix A. Thus for unambiguous weighted automata over rationals the determinisation problem is decidable in polynomial time, which we consider a folklore result.

Other popular classes of bounded ambiguity are: *finitely-ambiguous automata* and *polynomially-ambiguous automata*, where the number of accepting runs is bounded by a constant and a polynomial in the size of input word, respectively. Both classes are characterised by forbidding simple patterns that can be detected in polynomial time [30]. The decidability results of determinisation over the tropical semiring are for precisely these two classes [20, 22]. In general bounded ambiguous classes of automata are well-known restrictions studied also in other areas [25, 1, 9].

We focus on the rational field, which recently gained more attention. For simplicity of explanation we present it now assuming weighted automata are over integers. In 2021 Bell and Smertnig [2] proved Reutenauer’s conjecture [26]. It states that a weighted automaton is unambiguisable if and only if the image of the automaton is a set of integers with finitely many prime divisors.¹ One implication is straightforward, as unambiguous weighted automata can only output values obtained as a product of its weights. The other implication is the core of the paper [2]. An important step is to compute the *linear-hull* of the input automaton, which boils down to computing the linear Zariski closure of the semigroup generated by the input matrices. It turns out that the input automaton is unambiguisable if and only if its linear hull is unambiguous, reducing the unambiguisation problem to computing the linear hull. The fact that it is computable essentially follows from computability of the Zariski closure (not linear) [16]. In a recent paper [3] Bell and Smertnig focus on computing directly the linear Zariski closure, however, still no complexity upper bound is known.

Our contribution

Following the work of Bell and Smertnig we study the determinisation and unambiguisation problems over the rational field. Our result is the following

► **Theorem 1.** *Unambiguisable and determinisable are decidable in polynomial space for polynomially-ambiguous weighted automata.*

A detailed overview of our approach is summarised in Section 3, after introducing the necessary technical preliminaries. Roughly speaking, our approach is to reduce the problems to patterns that behave like in the unary alphabet, i.e. by pumping the same infix. Over the unary alphabet the problem boils down to analysing simple properties of linear recursive sequences (see [1] or [21]). For polynomially-ambiguous weighted automata we crisply characterise the automata which can be determinised/unambiguised through a notion we call pumpability. Crucially, our notion of pumpability can be decided using a standard zeroness test for weighted automata — the automaton we require to do this is of exponential size, but combining this with an NC^2 algorithm for zeroness we obtain polynomial space. Our proofs are surprising for two reasons. Both of our decision procedures for unambiguisation and determinisation are in PSPACE, but our algorithm is not effective, i.e. in the case there is an equivalent deterministic/unambiguous automaton our algorithm does not construct it. An equivalent deterministic automaton can be constructed using our proof techniques, but

¹ The conjecture (now theorem) is stated more generally for any field.

the deterministic automaton would be of nonelementary size. We leave open whether the equivalent deterministic automaton need be this large, or whether the equivalent unambiguous automaton can be constructed using our proof techniques.

Related work

The seminal result for weighted automata over the rational field is due to Schützenberger [28], who proved that equivalence is decidable. The problem is known to be even in NC^2 [29, 18] (thus in particular in polynomial time). Other problems like containment or emptiness are undecidable already for probabilistic automata [24]. Recently these undecidable problems gained attention for weighted automata with restricted ambiguity. The goal is to determine the border of decidability between the classes: finitely ambiguous, polynomially-ambiguous, full class of weighted automata [14, 12, 4, 7, 10]. The determinisation problem is known to be a particular variant of register minimisation, see e.g. [11]. Recently, following the work of Bell and Smertnig [2], it was proved that register minimisation is also decidable over the rational field [5].

2 Preliminaries

2.1 Weighted automata

A weighted automaton \mathcal{A} is a tuple (Q, Σ, M, I, F) , where Q is a finite set of states, Σ is a finite alphabet, $M : \Sigma \rightarrow \mathbb{Q}^{Q \times Q}$ is a transition weighting function, and $I, F \subseteq \mathbb{Q}^Q$ are the initial and the final vectors, respectively.

For every word $w = a_1 \dots a_n$ we define the matrix $M(w) = M(a_1)M(a_2) \dots M(a_n)$. We denote the empty word by ϵ , and $M(\epsilon)$ is the identity matrix. For every word $w \in \Sigma^*$, the automaton outputs $\mathcal{A}(w) = I^T M(w) F$. We say two automata \mathcal{A} and \mathcal{B} over Σ are equivalent if $\mathcal{A}(w) = \mathcal{B}(w)$ for every $w \in \Sigma^*$.

We can also interpret a weighted automaton as a finite automaton with weighted edges: when $x \neq 0$ we denote $M(a)_{p,q} = x$ by a transition $p \xrightarrow{a:x} q$. We say a state q is initial if $I(q) \neq 0$ and final if $F(q) \neq 0$.

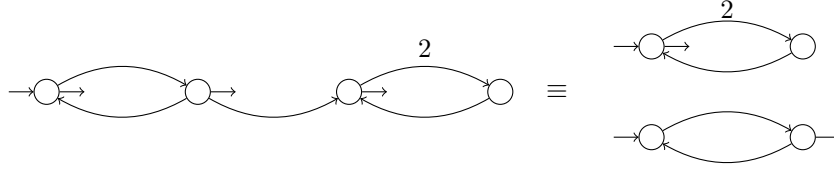
The size of the automaton \mathcal{A} , denoted $|\mathcal{A}|$, is the number of states of the automaton. The norm of \mathcal{A} , denoted $\|\mathcal{A}\|$, is the largest absolute value of numerators and denominators of numbers occurring in M, I and F . Observe the automaton can be represented using $O(|\mathcal{A}|^2 \log(\|\mathcal{A}\|))$ bits.

► **Definition 2.** We say a matrix M is p -triangular if there exists a permutation matrix P such that PMP^{-1} is upper triangular.

► **Remark 3.** The diagonal entries of an upper-triangular matrix are exactly the eigenvalues of the matrix. Since permutations do not change which entries are on the diagonal (only their order), when we refer to diagonal entries of p -triangular matrix, we equivalently refer to the eigenvalues of the matrices.

2.2 Paths, runs, counting runs, and the monoid of structures

A path of $w = a_1 \dots a_n$ in \mathcal{A} is a sequence of states q_1, \dots, q_{n+1} such that for $1 \leq i \leq n$ we have $q_i \xrightarrow{a_i:x_i} q_{i+1}$ (recall, this notation entails that $x_i \neq 0$). The value of a path is the product of the x_i . A path is a cycle if $q_1 = q_{n+1}$. A path is a run if q_1 is initial and q_{n+1} is final, and the value of the run is the product of $I(q_1), F(q_{n+1})$ and the value of the path.



■ **Figure 1** Example of a unary weighted automaton, where the input label of all edge is the letter a (omitted on the picture). Unlabelled edges are assumed to have weight 1. Observe, if n is even then $\mathcal{A}(a^n) = 1 + \sum_{i=0}^{n/2-1} 2^i = 2^{n/2}$ and if n is odd then $\mathcal{A}(a^n) = 1$. The automaton on the left is polynomially-ambiguous, and unambiguisable as depicted on the right, but the function is not determinisable.

A state q is reachable if there exists an initial state p and a word w such that there is a path from p to q . A state q is coreachable if there exists a final state r and a word w such that there is a path from q to r . Henceforth, we assume every state is reachable and coreachable (by trimming the automaton where necessary).

We denote by $\#runs_{\mathcal{A}}(w)$ the number of distinct runs on word w in \mathcal{A} .

► **Remark 4.** The existence of a run from q to p on w does not necessarily entail that $\mathcal{A}(w) \neq 0$. The value of all the runs on w may sum to zero, as \mathcal{A} is not necessarily non-negative.

We can also define the number of runs by treating the underlying finite automaton as a weighted automaton with weight 1 on every non-zero transition. Given $x \in \mathbb{Q}$, let $\bar{x} = 1$ if $x \neq 0$ and $\bar{x} = 0$ if $x = 0$. We extend the notion point wise to the transitions, initial and final states: $\overline{M}(a)_{p,q} = \overline{M(a)_{p,q}}$, $\overline{I}_q = \overline{I_q}$ and $\overline{F}_q = \overline{F_q}$. Then we have that $\#runs_{\mathcal{A}}(w) = \overline{I}^T \overline{M}(a_1) \overline{M}(a_2) \dots \overline{M}(a_n) \overline{F}$.

For a given matrix M , we say that \overline{M} is the *structure* of the matrix. The set of structures equipped with the matrix multiplication \otimes taken in the Boolean semiring (i.e. $1 + 1 = 1$) forms a monoid, often called the monoid of *Boolean matrices* in the literature. We say that a matrix M has *idempotent structure* if its structure is an idempotent element of this monoid: $\overline{M} = \overline{M} \otimes \overline{M}$.

2.3 Determinism, ambiguity and decision problems

► **Definition 5.** We say that an automaton \mathcal{A} is:

- deterministic if it has at most one non-zero entry in I , and $M(a)$ has at most one non-zero entry on every row for every $a \in \Sigma$,
- unambiguous if $\#runs_{\mathcal{A}}(w) \leq 1$ for every word $w \in \Sigma^*$,
- finitely ambiguous if there exists k such that $\#runs_{\mathcal{A}}(w) \leq k$ for every word $w \in \Sigma^*$,
- polynomially-ambiguous if there exists a polynomial p such that $\#runs_{\mathcal{A}}(w) \leq p(|w|)$ for every word $w \in \Sigma^*$, and
- exponentially ambiguous otherwise, in particular, $\#runs_{\mathcal{A}}(w) \leq |Q|^{|w|+1}$ for every w .

These characterisations lead to the following natural problems:

- The *determinisation problem* asks, given a weighted automaton \mathcal{A} , if there is an equivalent deterministic weighted automaton. If the answer is positive, we say \mathcal{A} is *determinisable*.
- The *unambiguisation problem* asks, given a weighted automaton \mathcal{A} , if there is an equivalent unambiguous weighted automaton. If the answer is positive, we say \mathcal{A} is *unambiguisable*.

An example polynomially-ambiguous weighted automaton that is unambiguisable but not determinisable is depicted in Figure 1.

2.4 Closure properties

We recall a standard result: weighted automata are closed under negation, difference and product.

- **Lemma 6.** *Let \mathcal{A}_1 and \mathcal{A}_2 be weighted automata over Σ that have size m_1 and m_2 .*
- *The function $-\mathcal{A} : u \mapsto -\mathcal{A}(u)$ is recognised by an automaton of size $|\mathcal{A}|$ and norm $\|\mathcal{A}\|$;*
 - *The function $\mathcal{A}_1 - \mathcal{A}_2 : u \mapsto \mathcal{A}_1(u) - \mathcal{A}_2(u)$ is recognised by an automaton of size $|\mathcal{A}_1| + |\mathcal{A}_2|$ and norm $\max\{\|\mathcal{A}_1\|, \|\mathcal{A}_2\|\}$;*
 - *The function $\mathcal{A}_1 \cdot \mathcal{A}_2 : u \mapsto \mathcal{A}_1(u) \cdot \mathcal{A}_2(u)$ is recognised by an automaton of size $|\mathcal{A}_1| \cdot |\mathcal{A}_2|$ and norm $\|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|$.*

Furthermore, if we assume that we can compute in space $O(\log(\|\mathcal{A}_1\|))$ (respectively $O(\log(\|\mathcal{A}_2\|))$) the weight of a transition in \mathcal{A}_1 (respectively \mathcal{A}_2) given by a letter $a \in \Sigma$ and two states of \mathcal{A}_1 (respectively \mathcal{A}_2), then we can compute in space $O(\log(\|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|))$ the weight of a transition in $\mathcal{A}_1 - \mathcal{A}_2$ or $\mathcal{A}_1 \cdot \mathcal{A}_2$ given by a letter $a \in \Sigma$ and two states of the corresponding automaton.

2.5 Assumptions

In this paper, without loss of generality, we only consider weighted automata that satisfy the following two assumptions:

Non-negative transitions We assume that the weight of every path is non-negative, although the run, when combined with I, F , may be negative. Formally, we assume that $M(a)_{p,q} \geq 0$ for all $a \in \Sigma, p, q \in Q$. Should the condition fail, there is a polynomial time algorithm to produce an equivalent weighted automaton whose matrix entries are non-negative. This can be done so that either the initial vector or the final vector is non-negative, so only one of these may have negative values. The full construction is available in Appendix B.1.

► **Remark 7.** A crucial consequence of this assumption is that for every word $u \in \Sigma^*$, each entry $(M(u))_{ij}$ of the matrix $M(u)$ is non-zero if and only if there exists (at least) one path labelled by u between i and j . Note that this is *not* necessarily the case if negative weights are allowed, since then distinct runs can cancel each other. Stated more formally, this assumption implies that the function mapping a word $u \in \Sigma^*$ to the structure $\overline{M}(u)$ of the corresponding matrix is a monoid homomorphism between the free monoid Σ^* and the finite monoid of Boolean matrices: $\overline{M}(uv) = \overline{M}(u) \cdot \overline{M}(v)$ for every $u, v \in \Sigma^*$.

Integer values We assume that the weights of the automaton are integer values, rather than rational values. That is, $M : \Sigma \rightarrow \mathbb{Z}^{Q \times Q}$, $I, F \in \mathbb{Z}^Q$.

In case the condition does not hold, an integer weighted automaton \mathcal{A}' can be constructed such that \mathcal{A} is unambiguisable (resp. determinisable) if and only if \mathcal{A}' is unambiguisable (resp. determinisable). Let X be the set of denominators of transition weights, initial weights and final weights and let $x = \text{lcm}(X)$. Note that $|x| \leq \|\mathcal{A}\|^{|X|}$, thus $\log(x) \leq |X| \log(\|\mathcal{A}\|)$, and observe that $|X|, \log(\|\mathcal{A}\|)$ are both polynomial in the size of the representation of \mathcal{A} .

We construct \mathcal{A}' from \mathcal{A} with M, I, F replaced by M', I', F' , where $I'(q) = xI(q), F'(q) = xF(q)$, and $M'(a)_{p,q} = xM(a)_{p,q}$ for all $a \in \Sigma, p, q \in Q$. Now $\mathcal{A}'(w) = x^{|w|+2}\mathcal{A}(w)$.

3 Overview of our main result

We start with an overview that presents the ideas behind our decision procedure, the technical details are presented in the following sections. Our algorithms deciding whether a weighted

automaton $\mathcal{A} = (Q, \Sigma, M, I, F)$ is unambiguisable, respectively determinisable, rely on the study of the behaviours of \mathcal{A} over families of words of the form $(uv^n w)_{n \in \mathbb{N}}$, with $u, v, w \in \Sigma^*$. Since $\mathcal{A}(uv^n w)$ is defined as $I \cdot M(u) \cdot M(v)^n \cdot M(w) \cdot F$, understanding these behaviours reduces to understanding powers of matrices.

► **Lemma 8.** *Let M be an $m \times m$ p -triangular matrix with the set of diagonal entries $\{d_1, d_2, \dots, d_k\} \subseteq \mathbb{N}$, and let $\vec{x}, \vec{y} \in \mathbb{Q}^m$. There exist k polynomials p_1, p_2, \dots, p_k such that*

$$\vec{x}^T \cdot M^n \cdot \vec{y} = \sum_{i=1}^k d_i^n \cdot p_i(n) \text{ for all } n \geq m. \quad (1)$$

Moreover, if the matrix M is invertible; p_1, p_2, \dots, p_k are all constant polynomials; and at most one p_j is not constantly 0, then

$$\vec{x}^T \cdot M^n \cdot \vec{y} = d_j^n \cdot \vec{x}^T \cdot \vec{y} \text{ for all } n \geq 0. \quad (2)$$

It is standard that a sequence $\vec{x}^T \cdot M^n \cdot \vec{y}$ forms a linear recurrence sequence which admits a closed form like the one given in Equation (1). Thus most of Lemma 8 can be derived from standard results on linear recursive sequences (see e.g. [15, Section 2]). We prove it in Appendix C.1 to get the precise form required in Equation (2). It is nontrivial as it holds for $n \geq 0$ (not just n big enough). This will be crucial later in the proofs. Overall, Lemma 8 has two purposes:

Equation (1) expresses the behaviours of \mathcal{A} over families of the form $(uv^n w)_{n \geq |\mathcal{A}|}$. This allows us to define the notion of *pumpability* (resp. blind pumpability) by forbidding *bad* behaviours (the ones that match no unambiguous automaton). We show that this is a necessary criterion for unambiguisable (resp. determinisable) (see Proposition 10).

Equation (2) allows us to establish pumpability (resp. blind pumpability) is sufficient to ensure unambiguisable (resp. determinisable) for polynomially-ambiguous WA (see Proposition 11).

One of the reasons we restrict ourselves to p -triangular matrices in Lemma 8 (and further definitions) is to avoid dealing with complex numbers. For a general matrix M an expression similar to Equation (1) also holds, with the d_i being *eigenvalues* of M (which happen to be the diagonal entries if M is p -triangular). Eigenvalues of a matrix over \mathbb{N} do not have to be rational numbers, which make it more difficult to distinguish good behaviours from bad.

Excluding bad behaviours As a direct consequence of Equation (1), we get that for every $u, v, w \in \Sigma^*$ such that $M(v)$ is a p -triangular matrix,

$$\mathcal{A}(uv^n w) = \sum_{i=1}^k d_i^n \cdot p_i(n) \text{ for every } n \geq |\mathcal{A}|,$$

where $\{d_1, d_2, \dots, d_k\}$ is the set of diagonal entries of $M(v)$, and p_1, p_2, \dots, p_k are polynomials. We show that if \mathcal{A} is unambiguisable, then this expression needs to collapse to some simple periodic expression for every choice of u, v, w . We formalise this through the notions of *pumpable automata* and *blindly pumpable automata* (in the latter, the pumping does not depend on the suffix, which, as we will show, is required for determinisability).

► **Definition 9.** *A weighted automaton \mathcal{A} is pumpable if for all $u, v, w \in \Sigma^*$ such that $M(v)$ is a p -triangular matrix there is an entry d of the diagonal of $M(v)$ satisfying*

$$\mathcal{A}(uv^{|\mathcal{A}|+n} w) = d^n \cdot \mathcal{A}(uv^{|\mathcal{A}|} w) \text{ for every } n \in \mathbb{N}.$$

Moreover, we say that \mathcal{A} is *blindly pumpable* if the entry d does not depend on the suffix w , that is, for all $u, v \in \Sigma^*$ such that $M(v)$ is a p -triangular matrix there is an entry d of the diagonal of $M(v)$ satisfying

$$\mathcal{A}(uv^{|\mathcal{A}|+n}w) = d^n \cdot \mathcal{A}(uv^{|\mathcal{A}|}w) \text{ for every } w \in \Sigma^* \text{ and } n \in \mathbb{N}.$$

The interest of this definition is reflected by Proposition 10, proved in Section 4.

► **Proposition 10.** *Every unambiguisable weighted automaton is pumpable, and every determinisable weighted automaton is blindly pumpable.*

Taking advantage of good behaviours Unfortunately, the converse statement of Proposition 10 does not hold in general, because of the following limitations:

1. Pumpability refers to p -triangular matrices, which do not appear in some automata;
2. Pumpability only guarantees a periodic behaviour over the words where some factor v is pumped several times: How do we ensure that this nice behaviour extends to words where such a repetition never happens, for instance square-free words?

To overcome these limitations, we restrict ourselves to *polynomially-ambiguous* automata. In this setting matrices with an idempotent structure are p -triangular matrices up to a rearrangement of the states (Lemma 21), and the presence of such matrices is guaranteed by Ramsey's theorem (Lemma 20). On top of this, we remark that Equation (2) in Lemma 8 would counteract the second limitation, but the periodic behaviour starts from the first appearance only for invertible matrices. The restriction to polynomially-ambiguous WA further allows us to show that sufficiently long words contain invertible-like idempotent entries in which Equation (2) can be applied.

Thus, restricting ourselves to polynomially-ambiguous automata allows us to get rid of the limitations preventing the converse of Proposition 10. We obtain the following result, proved in Section 5:

► **Proposition 11.** *Let \mathcal{A} be a polynomially-ambiguous automaton. If \mathcal{A} is pumpable then it is unambiguisable, and if \mathcal{A} is blindly pumpable then it is determinisable.*

Deciding Pumpability With Propositions 10 and 11, we have identified a class of weighted automata for which pumpability, respectively blind pumpability, characterises unambiguisability, respectively determinisability. We finally show in Section 6 that we can decide the two former notions in polynomial space:

► **Proposition 12.** *We can decide in polynomial space whether a given weighted automaton is pumpable, respectively blindly pumpable.*

Together, Propositions 10–12 entail our main theorem.

► **Theorem 1.** *Unambiguisability and determinisability are decidable in polynomial space for polynomially-ambiguous weighted automata.*

4 Pumpable automata

The goal of this section is to prove Proposition 10:

► **Proposition 10.** *Every unambiguisable weighted automaton is pumpable, and every determinisable weighted automaton is blindly pumpable.*

We begin by establishing the possible behaviours of unambiguous automata, respectively deterministic automata, over families of words of the form $(uv^n w)_{n \in \mathbb{N}}$ (Lemma 13). Then, to prove Proposition 10, we show that, given a weighted automaton \mathcal{A} , for each triple $u, v, w \in \Sigma^*$, if we take the possible values of $(\mathcal{A}(uv^n w))_{n \in \mathbb{N}}$ (which are described by Equation (1) of Lemma 8) and remove all the behaviours that do not fit an unambiguous (resp. deterministic) weighted automaton (which are described by Lemma 13), then we are left with exactly the behaviours fitting a pumpable (respectively blindly pumpable) automaton.

► **Lemma 13.** *Let \mathcal{A} be a weighted automaton.*

1. *If \mathcal{A} is unambiguous, then for all $u, v, w \in \Sigma^*$ there exist $m, d \in \mathbb{N}$ and $\lambda > 0$ such that*

$$\mathcal{A}(uv^{m+\lambda n}w) = d^n \cdot \mathcal{A}(uv^m w) \text{ for all } n \in \mathbb{N}.$$

2. *If \mathcal{A} is deterministic, then for all $u, v \in \Sigma^*$ there exist $m, d \in \mathbb{N}$ and $\lambda > 0$ such that*

$$\mathcal{A}(uv^{m+\lambda n}w) = d^n \cdot \mathcal{A}(uv^m w) \text{ for all } w \in \Sigma^* \text{ and } n \in \mathbb{N}.$$

Proof. These results follow from a standard pumping argument: in every automaton, long enough runs eventually visit a cycle. Since unambiguous weighted automata have at most one run over each input word, pumping such a cycle amounts to multiplying the weight of the cycle by a constant. On top of this, the second item follows from the fact that in a deterministic automaton the pumped constant cannot depend on the suffix that is yet to be read. Let us now prove these results formally.

1. Let \mathcal{A} be an unambiguous automaton, and let $u, v, w \in \Sigma^*$. First, remark that if for every $m \geq |\mathcal{A}|$ the automaton \mathcal{A} has no run over $uv^m w$, then $\mathcal{A}(uv^m w) = 0$, and the statement is easily satisfied by picking $m = |\mathcal{A}|$, $d = 0$ and $\lambda = 1$. Now let us suppose that there exists $m \geq |\mathcal{A}|$ such that \mathcal{A} has a run ρ over $uv^m w$. Since $|\mathcal{A}|$ denotes the number of states of \mathcal{A} , ρ can be factorised into three subruns such that the middle part is a cycle which processes a non-empty power v^λ of the word v . Therefore, if we denote by d the weight of this middle part, we get that for all $n \in \mathbb{N}$ the automaton \mathcal{A} has a run over $uv^{m+\lambda n}w$ which has weight $d^n \cdot \mathcal{A}(uv^m w)$. However, since \mathcal{A} is unambiguous by supposition, this is the only run over this input word, hence $\mathcal{A}(uv^{m+\lambda n}w) = d^n \cdot \mathcal{A}(uv^m w)$, which concludes the proof.

2. Let \mathcal{A} be a deterministic automaton, and let $u, v \in \Sigma^*$. The proof is nearly identical to the one for the unambiguous setting: If for every $m \geq |\mathcal{A}|$ the automaton \mathcal{A} has no path over $uv^m w$ starting in an initial state, then $\mathcal{A}(uv^m w) = 0$, and we are done. If there exists $m \geq |\mathcal{A}|$ such that \mathcal{A} has a path over $uv^m w$ starting from the initial state, then this path will visit a cycle while reading a non-empty power v^λ of v , hence for every $n \in \mathbb{N}$ there exists a path in \mathcal{A} over $uv^{m+\lambda n}w$ that starts in an initial state and has weight $d^n \cdot \mathcal{A}(uv^m w)$. Since \mathcal{A} is deterministic, there cannot be any other such path, hence for every $w \in \Sigma^*$ we get that $\mathcal{A}(uv^{m+\lambda n}w) = d^n \cdot \mathcal{A}(uv^m w)$, as required. ◀

Proof of Proposition 10. Fix a weighted automaton $\mathcal{A} = (Q, \Sigma, M, I, F)$.

Unambiguability implies pumpability Let us suppose that \mathcal{A} is unambiguability, and let \mathcal{U} be an unambiguous automaton equivalent to \mathcal{A} . Let $u, v, w \in \Sigma^*$ such that $M(v)$ is a p-triangular matrix. We compare the behaviour of \mathcal{A} and \mathcal{U} over the family of words $(uv^n w)_{n \in \mathbb{N}}$. The behaviour of \mathcal{A} is described by Equation (1) of Lemma 8:

$$\mathcal{A}(uv^n w) = \sum_{i=1}^k d_i^n \cdot p_i(n) \text{ for every } n \geq |\mathcal{A}|, \quad (3)$$

where $\{d_1, d_2, \dots, d_n\}$ is the set of diagonal entries of $M(v)$ and p_1, p_2, \dots, p_n are polynomials. The behaviour of \mathcal{U} is described by Lemma 13: There exist $d, m \in \mathbb{N}$ and $\lambda > 0$ such that

$$\mathcal{U}(uv^{m+\lambda n}w) = d^n \cdot \mathcal{U}(uv^m w) \text{ for all } n \in \mathbb{N}.$$

Since \mathcal{A} is equivalent to \mathcal{U} , these two behaviours need to match. Intuitively, this implies that Equation (3) collapses to a single term of the form $\mathcal{A}(uv^n w) = \delta^n \cdot C$ (so that it mirrors the behaviour of \mathcal{U}), which proves the desired result as this fits the definition of a pumpability. We now present the formal details proving this intuition. First, remark that

$$\left(\sum_{i=1}^k d_i^{m+\lambda n} \cdot p_i(m + \lambda n) \right) - d^n \cdot \mathcal{U}(uv^m w) = \mathcal{A}(uv^{m+\lambda n} w) - \mathcal{U}(uv^{m+\lambda n} w) = 0 \text{ for all } n \geq |\mathcal{A}|.$$

Let us rewrite the left-hand side as $\sum_{i=1}^k (d_i^\lambda)^n \cdot p'_i(n)$, where

$$p'_i(n) = \begin{cases} d_i^m p_i(m + \lambda n) & \text{if } d_i^\lambda \neq d; \\ d_i^m p_i(m + \lambda n) - \mathcal{U}(uv^m w) & \text{if } d_i^\lambda = d. \end{cases}$$

Since $\sum_{i=1}^k (d_i^\lambda)^n \cdot p'_i(n) = 0$ for all $n \in \mathbb{N}$ and the d_i are distinct positive integers, every p'_i is constantly 0: Otherwise, $\sum_{i=1}^k (d_i^\lambda)^n \cdot p'_i(n)$ would behave asymptotically like $(d_j^\lambda)^n \cdot p'_j(n)$ (contradicting the fact that it is 0), where d_j is the largest d_i such that the corresponding p'_i is not constantly 0. As a consequence, for every $1 \leq i \leq k$:

- If $d_i^\lambda \neq d$, then the fact that $p'_i(n) = d_i^m p_i(m + \lambda n)$ is constantly 0 implies that either $d_i = 0$ or $p_i(n)$ is constantly 0;
- If $d_i^\lambda = d$, then the fact that $p'_i(n) = d_i^m p_i(m + \lambda n) - \mathcal{U}(uv^m w)$ is constantly 0 implies that either $d_i = 0$ or $p_i(n)$ is constantly $\frac{\mathcal{U}(uv^m w)}{d_i^m}$.

We can update Equation (3) accordingly: If there exist d_j such that $d = d_j^\lambda$, then $\mathcal{A}(uv^n w) = d_j^n \cdot \frac{\mathcal{U}(uv^m w)}{d_j^m}$ for every $n \geq |\mathcal{A}|$. Otherwise, $\mathcal{A}(uv^n w) = 0$ for every $n \geq |\mathcal{A}|$. Remark that both cases fit the requirement of the definition of pumpability: $\mathcal{A}(uv^{|\mathcal{A}|+n} w) = d_j^n \cdot \mathcal{A}(uv^{|\mathcal{A}|} w)$. Since this holds for every triple $u, v, w \in \Sigma^*$ required in Definition 9 we deduce that \mathcal{A} is pumpable.

Determinisability implies blind pumpability If \mathcal{A} is equivalent to a deterministic automaton \mathcal{D} , in particular \mathcal{A} is unambiguisable, thus, as we just proved, it is pumpable. Towards building a contradiction, suppose that \mathcal{A} is not blindly pumpable: there exists $u, v, w_1, w_2 \in \Sigma^*$ such that $\mathcal{A}(uv^{|\mathcal{A}|+n} w_1) = d_1^n \cdot \mathcal{A}(uv^{|\mathcal{A}|} w_1)$ and $\mathcal{A}(uv^{|\mathcal{A}|+n} w_2) = d_2^n \cdot \mathcal{A}(uv^{|\mathcal{A}|} w_2)$ for all $n \in \mathbb{N}$, yet $d_1 \neq d_2$ and $\mathcal{A}(uv^{|\mathcal{A}|} w_1) \neq 0 \neq \mathcal{A}(uv^{|\mathcal{A}|} w_2)$. We compare these expressions with the behaviours of \mathcal{D} , described by Lemma 13: There exist $d, m \in \mathbb{N}$ and $\lambda > 0$ such that $\mathcal{D}(uv^{m+\lambda n} w_1) = d^n \cdot \mathcal{D}(uv^m w_1)$ and $\mathcal{D}(uv^{m+\lambda n} w_2) = d^n \cdot \mathcal{D}(uv^m w_2)$ for all $n \in \mathbb{N}$. Since d_1 and d_2 are distinct integers, d cannot match both d_1^λ and d_2^λ , hence \mathcal{A} and \mathcal{D} disagree on the value of either $uv^{m+\lambda n} w_1$ or $uv^{m+\lambda n} w_2$ for n sufficiently large. This contradicts the fact that the two automata are equivalent. ◀

5 Depumpable automata

This section is devoted to the proof of Proposition 11, which we recall:

► **Proposition 11.** *Let \mathcal{A} be a polynomially-ambiguous automaton. If \mathcal{A} is pumpable then it is unambiguisable, and if \mathcal{A} is blindly pumpable then it is determinisable.*

Our proof relies on the following lemma, which shows that, once we restrict ourselves to polynomially-ambiguous automata, pumpable automata are also *depumpable*:

► **Lemma 14** (Depumping lemma). *Let \mathcal{A} be a polynomially-ambiguous automaton. There exists a constant $R_{\mathcal{A}}$ such every word $u \in \Sigma^*$ satisfying $|u| = R_{\mathcal{A}}$ can be decomposed into three parts $u = u_1 u_2 u_3$ such that $u_2 \neq \epsilon$ and*

1. *If \mathcal{A} is pumpable, for each $v \in \Sigma^*$ there exist an entry d of the matrix $M(u_2)$ such that $\mathcal{A}(uv) = d \cdot \mathcal{A}(u_1 u_3 v)$;*
2. *If \mathcal{A} is blindly pumpable, there exists an entry d of the matrix $M(u_2)$ such that for all $v \in \Sigma^*$ we have $\mathcal{A}(uv) = d \cdot \mathcal{A}(u_1 u_3 v)$.*

Remark that the order of the quantifiers is different in the two items: In Item 2 the entry d does not depend on the suffix v . The proof of Lemma 14 is presented in Subsection 5.2.

Proof of Proposition 11. Let \mathcal{A} be a polynomially-ambiguous automaton. We prove both parts of Proposition 11 independently. Remark that we only use the assumption that \mathcal{A} is polynomially-ambiguous and pumpable (respectively blindly pumpable) to enable the use of the depumping lemma: In other words, extending the depumping lemma to a wider class of automata would result in extending Proposition 11 in the same manner.

Pumpability implies unambiguability Let us suppose that \mathcal{A} is pumpable. We prove that it is unambiguability relying on the following characterisation by Bell and Smertnig.

► **Theorem 15** ([2]). *A weighted automaton over integers \mathcal{A} is unambiguability if and only if the set of prime divisors of the set $\{\mathcal{A}(w) \mid w \in \Sigma^*\}$ is finite.*

We show that the set of prime divisors of the weights of \mathcal{A} is finite by building an upper bound according to the following intuition: For each $u \in \Sigma^*$, repeated application of the depumping lemma allows us to express $\mathcal{A}(u)$ as a product of the form $\mathcal{A}(u') \cdot \prod_{i=1}^n d_i$, where both $\mathcal{A}(u')$ and the d_i are bounded. This gives us a bound for the prime divisors of $\mathcal{A}(u)$.

Let us formalise this idea. Let $N \in \mathbb{N}$ be an integer such that for every $u \in \Sigma^*$ shorter than $R_{\mathcal{A}}$ all the entries of $M(u)$ are smaller than N and $\mathcal{A}(u) < N$. We prove by induction on the length that for every word $u \in \Sigma^*$, the prime factors of $\mathcal{A}(u)$ are all smaller than N .

The base case of the induction is immediate: If $|u| < R_{\mathcal{A}}$, then by definition of N we get that $\mathcal{A}(u)$ is smaller than N , thus its prime factors are also smaller than N .

Now let us suppose that $|u| \geq R_{\mathcal{A}}$ and that for every word u' shorter than u the prime factors of $\mathcal{A}(u')$ are smaller than N . By applying Lemma 14 to the prefix of size $R_{\mathcal{A}}$ of u we obtain a decomposition $u = u_1 u_2 u_3$ satisfying $0 < |u_2| \leq R_{\mathcal{A}}$ and $\mathcal{A}(u) = d \cdot \mathcal{A}(u_1 u_3)$ for some entry d of $M(u_2)$. We conclude by remarking that $d < N$ since $|u_2| < R_{\mathcal{A}}$, and the prime factors of $\mathcal{A}(u_1 u_3)$ are smaller than N by the induction hypothesis.

Blind pumpability implies determinisability Let us suppose that \mathcal{A} is blindly pumpable. We use the depumping Lemma to build a deterministic automaton \mathcal{D} equivalent to \mathcal{A} .

We begin with an informal description of the behaviour of \mathcal{D} . The states of \mathcal{D} are the words $u \in \Sigma^*$ satisfying $|u| \leq R_{\mathcal{A}}$. The automaton \mathcal{D} starts by keeping track in its state of the input word read so far, and preserves a weight equal to 1. Whenever \mathcal{D} reaches a state corresponding to a word u of length $R_{\mathcal{A}}$, it “depumps” it: According to Lemma 14 there exists a decomposition $u = u_1 u_2 u_3$ and an entry d of $M(u)$ such that for every suffix $w \in \Sigma^*$ $\mathcal{A}(uw) = d \cdot \mathcal{A}(u_1 u_3 w)$. Therefore, \mathcal{D} can immediately produce the weight d corresponding to the infix u_2 , and transition towards the state $u_1 u_3$. Finally, when \mathcal{D} finishes reading its

input, it produces the weight $\mathcal{A}(v)$ corresponding to its current state v . In order to prove that such an automaton is indeed equivalent to \mathcal{A} , we now formalise this construction.

We define the deterministic automaton $\mathcal{D} = (Q', \Sigma, M', I', F')$ as follows. First, we introduce the depumping function $\text{cut} : \Sigma^{R_{\mathcal{A}}} \rightarrow \Sigma^* \times \mathbb{Z}$ that *depumps deterministically* the words of size $R_{\mathcal{A}}$: it picks, for each $u \in \Sigma^*$ such that $|u| = R_{\mathcal{A}}$, a pair (u', d) satisfying

- $|u'| < R_{\mathcal{A}}$;
- d is an entry of $M(u')$;
- $\mathcal{A}(uv) = d \cdot \mathcal{A}(u'v)$ for every $v \in \Sigma^*$.

Note that this function exists: Lemma 14 guarantees the existence of at least one such pair (u', d) for every $u \in \Sigma^{R_{\mathcal{A}}}$. To ensure that \mathcal{D} is deterministic cut can pick any fixed pair for every $u \in \Sigma^{R_{\mathcal{A}}}$, for instance, the minimal pair according to the lexicographical order. We now define the components of \mathcal{D} :

- $Q' = \{q_u \in \Sigma^* \mid |u| < R_{\mathcal{A}}\}$;
- For every $u \in Q'$ and $a \in \Sigma$,
 - If $|ua| < R_{\mathcal{A}}$ then $q_u \xrightarrow{a:1} q_{ua}$;
 - If $|ua| = R_{\mathcal{A}}$ then $q_u \xrightarrow{a:d} q_{u'}$ where $\text{cut}(ua) = (u', d)$.
- $I'(q_{\epsilon}) = 1$ and $I'(q_u) = 0$ for every $u \neq \epsilon$;
- $F'(q_u) = \mathcal{A}(u)$.

The automaton \mathcal{D} is deterministic: There is exactly one initial state, and for every state q_u and letter a there is exactly one outgoing transition from u labelled by a . All that remains to show is that $\mathcal{D}(u) = \mathcal{A}(u)$ for every $u \in \Sigma^*$. To this end, we prove the following claim:

▷ **Claim 16.** The weight x of the (single) path $q_{\epsilon}, q_{u_1}, q_{u_2}, \dots, q_{u_n}$ of \mathcal{D} over u starting from the initial state q_{ϵ} satisfies $\mathcal{A}(uw) = x \cdot \mathcal{A}(u_n w)$ for every $w \in \Sigma^*$.

Remark that, by definition of the final vector F , this immediately implies that $\mathcal{D}(u) = \mathcal{A}(u)$. We prove the claim by induction on the size of u . If $|u| \leq R_{\mathcal{A}}$ the result is immediate: the path of \mathcal{D} over u that starts from the initial state has weight 1 and ends in the state q_u (thus $u_n = u$). Now suppose that $|u| > R_{\mathcal{A}}$. Let us denote $u = va$ with $v \in \Sigma^*$ and $a \in \Sigma$. Remark that the path of \mathcal{D} over v starting from the initial state is obtained by removing the last transition $q_{u_{n-1}} \xrightarrow{a:d} q_{u_n}$ of the path over u . Therefore, for every $w \in \Sigma^*$, applying the induction hypothesis to v yields that $\mathcal{A}(vw) = \mathcal{A}(vaw) = \frac{x}{d} \cdot \mathcal{A}(u_{n-1}aw)$. We distinguish two possibilities, depending on the type of transition used while reading the last letter a of u :

- If $|u_{n-1}a| < R_{\mathcal{A}}$, then $u_n = u_{n-1}a$ and $d = 1$, hence $\mathcal{A}(u) = \frac{x}{d} \cdot \mathcal{A}(u_{n-1}aw) = x \cdot \mathcal{A}(u_n w)$;
- If $|u_{n-1}a| = R_{\mathcal{A}}$, then $\text{cut}(u_{n-1}a) = (u_n, d)$, hence $\mathcal{A}(u) = \frac{x}{d} \cdot \mathcal{A}(u_{n-1}aw) = x \cdot \mathcal{A}(u_n w)$.

The last equality follows from the definition of cut . ◀

► **Remark 17.** While our decision procedure is in PSPACE, the size of the automaton \mathcal{D} , should it be constructed, depends on $R_{\mathcal{A}}$. We will see in the next section this will be quite big (non-elementary). Our decision procedure does not need to construct \mathcal{D} .

5.1 Replacing pumpable idempotents with invertible matrices

To prove Lemma 14 we will need a technical lemma that allows us to find pumpable fragments and replace them with invertible matrices. Some ideas here are similar as in [6], in particular the idea behind Claim 19 is similar to [6, Lemma 8].

Let us consider the family of function $(\text{tower}_r)_{r \in \mathbb{N}}$ defined inductively as follows:

$$\begin{aligned} \text{tower}_0(x) &= x && \text{for all } x \in \mathbb{N}; \\ \text{tower}_{r+1}(x) &= x \cdot \text{tower}_r(x^x) && \text{for all } x \in \mathbb{N}. \end{aligned}$$

► **Lemma 18.** *Let r be the maximal rank of the matrices $\{M(a) \mid a \in \Sigma\}$. Then for all $\ell \geq 1$, every word $u^* \in \Sigma^*$ satisfying $|u| \geq \text{tower}_r(2\ell \cdot |\Sigma|)$ can be decomposed as $u = u_{\vdash} u_1 u_2 \dots u_{\ell} u_{\dashv}$ such that*

- *for every $1 \leq i \leq \ell$ the word u_i is nonempty;*
- *for every $1 \leq i \leq j \leq \ell$ there exists an invertible matrix M satisfying, for all $n \geq 0$:*

$$M(u_{\vdash} u_1 u_2 \dots u_{i-1} (u_i \dots u_j)^n u_{j+1} \dots u_{\ell} u_{\dashv}) = M(u_{\vdash} u_1 u_2 \dots u_{i-1}) \cdot M^n \cdot M(u_{j+1} \dots u_{\ell} u_{\dashv}). \quad (4)$$

The proof will rely on the following claim.

▷ **Claim 19.** If the rank of ABA is equal to the rank of A , there exists an invertible matrix C satisfying $A(BA)^n = AC^n$ for all $n \geq 0$.

Proof. Since the rank of A and ABA are equal we conclude that their images are also the same. Let $V \subseteq \mathbb{Q}^m$ be this image. Note that BA can be seen as a linear transformation on \mathbb{Q}^m , and BA restricted to V , i.e. $BA|_V$, is a bijection. Let $\vec{v}_1, \dots, \vec{v}_s$ be a basis of V and let $\vec{v}_{s+1}, \dots, \vec{v}_m$ be such that $\vec{v}_1, \dots, \vec{v}_m$ is a basis of \mathbb{Q}^m .

Recall that a linear transformation $\mathbb{Q}^m \rightarrow \mathbb{Q}^m$ is uniquely defined by fixing the images of a basis. Let $\vec{w}_1, \dots, \vec{w}_s$ be the images of $\vec{v}_1, \dots, \vec{v}_s$ when applying the linear transformation BA , and observe that these also span V . The matrix C is defined by the linear transformation that maps \vec{v}_i to \vec{w}_i for all $1 \leq i \leq s$ and \vec{v}_i to \vec{v}_i for $s+1 \leq i \leq m$.

It is clear that C is invertible as its image has m independent vectors. To prove that $A(BA)^n = AC^n$ we prove that both linear transformations are defined in the same way on $\vec{v}_1, \dots, \vec{v}_m$. Indeed, vectors \vec{v}_i for $s+1 \leq m$ are mapped to $\vec{0}$ by both transformations. The remaining vectors have the same image since $BA|_V = C|_V$ and $BA|_V$ is a bijection. ◀

Proof of Lemma 18. We prove the lemma by induction on r .

If $r = 0$ the proof is straightforward: by definition of r for every letter $a \in \Sigma$ the matrix $M(a)$ has rank 0, thus it is the null matrix. Then for every word $u = a_1 a_2 \dots a_n$ of size $n \geq \text{tower}_0(2\ell \cdot |\Sigma|) = 2\ell \cdot |\Sigma| > \ell \cdot |\Sigma|$, the left-hand side of Equation (4) is equal to the null matrix for every choice of i, j and n . Therefore, we satisfy the statement by setting $u_{\vdash} = \epsilon$, $u_i = a_i$ for every $1 \leq i \leq \ell \cdot |\Sigma|$ and $u_{\dashv} = a_{\ell \cdot |\Sigma|+1} \dots a_n$.

Now let us suppose that $r > 0$ and that the lemma holds for $r-1$. Let $x = 2\ell \cdot |\Sigma|$, and let $u \in \Sigma^*$ be a word satisfying $|u| > \text{tower}_r(x) = x \text{tower}_{r-1}(x^x)$. We consider the decomposition $u = v_1 v_2 \dots v_{\text{tower}_{r-1}(x^x)} v$ such that $|v_i| = x$ for every $1 \leq i \leq \text{tower}_{r-1}(x^x)$. We distinguish two cases:

- Suppose that there exists $1 \leq i \leq \text{tower}_{r-1}(x^x)$ such that the rank of $M(v_i)$ is r . Then for every infix v' of v_i the rank of $M(v')$ is also r . Moreover, since $|v_i| = x > \ell \cdot |\Sigma|$, there is one letter $a \in \Sigma$ that occurs $\ell+1$ times in v_i . We get the statement of the lemma as a direct consequence of Claim 19.
- Suppose that for every $1 \leq i \leq \text{tower}_{r-1}(x^x)$ the matrix $M(v_i)$ has a rank smaller than r . In order to apply the induction hypothesis, we now consider the alphabet $\Gamma = \{v_i \mid 1 \leq i \leq \text{tower}_{r-1}(x^x)\}$. Then for every letter $a \in \Gamma$ we have that the rank of $M(a)$ is smaller than or equal to $r-1$. Moreover, since the length of each v_i is x , we get that $|\Gamma| \leq |\Sigma|^x$. Therefore, the word $w = v_1 v_2 \dots v_{\text{tower}_{r-1}(x^x)} \in \Gamma^*$ satisfies

$$|w| = \text{tower}_{r-1}(x^x) = \text{tower}_{r-1}((2\ell \cdot |\Sigma|)^x) > \text{tower}_{r-1}(2\ell \cdot |\Sigma|^x) \geq \text{tower}_{r-1}(2\ell \cdot |\Gamma|).$$

As a consequence, we can apply the induction hypothesis to obtain a decomposition of the word $w = v_1 v_2 \dots v_{\text{tower}_{r-1}(x^x)} \in \Gamma^*$, which can be transferred back to u . ◀

5.2 Proof of depumping lemma (Lemma 14)

We need two technical results. The first lemma, intuitively, shows that we can find idempotents in long enough words. It is a direct consequence of [17, Theorems 1 and 2].² The second lemma shows that idempotents for polynomially-ambiguous WA are p-triangular. It is proved in Appendix D.

► **Lemma 20.** *Given a weighted automaton \mathcal{A} : let $\ell = (3 \cdot 2^{4|\mathcal{A}|^2})^L$, where $L = \frac{|\mathcal{A}|^2 + |\mathcal{A}| + 2}{2}$. Let $u = u_1 \dots u_\ell \in \Sigma^+$, where $u_i \in \Sigma^+$ for all $1 \leq i \leq \ell$. There exist $1 \leq i \leq j \leq \ell$ such that $M(u_i \dots u_j)$ has idempotent structure.*

► **Lemma 21.** *Let \mathcal{A} be a polynomially-ambiguous weighted automaton. For every $u \in \Sigma^*$, if $M(u)$ has an idempotent structure then it is p-triangular.*

Proof of Lemma 14. We only prove Item 1, Item 2 can be proved nearly identically, up to the quantifier alternation.

Suppose \mathcal{A} is pumpable. We define ℓ as the constant in Lemma 20. We define $R_{\mathcal{A}} = \text{tower}_r(2\ell \cdot |\Sigma|)$ as the constant from Lemma 18. Let $u, v \in \Sigma^*$ such that $|u| = R_{\mathcal{A}}$.

We start by picking the decomposition $u = u_{\vdash} u_1 u_2 \dots u_\ell u_{\dashv}$ from Lemma 18. By Lemma 20 we can choose $1 \leq i \leq j \leq \ell$ such that $M(u_i \dots u_j)$ has an idempotent structure. By Lemma 21 we know that $M(u_i \dots u_j)$ is p-triangular. As \mathcal{A} is pumpable, there is an entry $d \in \mathbb{N}$ of the diagonal of $M(u_i \dots u_j)$ satisfying

$$\mathcal{A}(u_1 \dots u_{i-1} (u_i \dots u_j)^n u_{j+1} \dots u_\ell v) = d^{n-|\mathcal{A}|} \cdot \mathcal{A}(u_1 \dots u_{i-1} (u_i \dots u_j)^{|\mathcal{A}|} u_{j+1} \dots u_\ell v)$$

for all $n \geq |\mathcal{A}|$. Let us rewrite the left-hand side as a product of matrices: $I \cdot M(u_1 \dots u_{i-1}) \cdot M(u_i \dots u_j)^n \cdot M(u_{j+1} \dots u_\ell v) \cdot F$. Lemma 18 allows us to replace $M(u_i \dots u_j)$ with an invertible p-triangular matrix P . We get:

$$I \cdot M(u_1 \dots u_{i-1}) \cdot P^n \cdot M(u_{j+1} \dots u_\ell v) \cdot F = d^{n-|\mathcal{A}|} \cdot \mathcal{A}(u_1 \dots u_{i-1} (u_i \dots u_j)^{|\mathcal{A}|} u_{j+1} \dots u_\ell v)$$

for all $n \geq |\mathcal{A}|$. As the right-hand side is composed of a single power d^n multiplied by a constant polynomial and P is invertible and p-triangular, we can rewrite it according to Equation (2) of Lemma 8:

$$I \cdot M(u_1 \dots u_{i-1}) \cdot P^n \cdot M(u_{j+1} \dots u_\ell v) \cdot F = d^n \cdot I \cdot M(u_1 \dots u_{i-1}) \cdot M(u_{j+1} \dots u_\ell v) \cdot F$$

for all $n \in \mathbb{N}$. We revert P to $M(u_i \dots u_j)$, which yields the expression required by Lemma 14 once we set $n = 1$:

$$\mathcal{A}(u_1 \dots u_{i-1} (u_i \dots u_j)^n u_{j+1} \dots u_\ell v) = d^n \cdot \mathcal{A}(u_1 \dots u_{i-1} u_{j+1} \dots u_\ell v). \quad \blacktriangleleft$$

6 Deciding pumpability

This subsection is devoted to proving:

► **Proposition 12.** *We can decide in polynomial space whether a given weighted automaton is pumpable, respectively blindly pumpable.*

² The use of [17] is enabled thanks to our *non-negative transitions* assumption (see Remark 7).

6.1 Deciding pumpability

We start by showing pumpability is decidable, in fact, we will show that a seemingly weaker version of pumpability is decidable, however, we will observe that it is equivalent.

Let us recall (from Definition 9) that \mathcal{A} is pumpable if for all $u, v, w \in \Sigma^*$ such that $M(v)$ is p-triangular there is an entry d of the diagonal of $M(v)$ satisfying

$$\mathcal{A}(uv^{m+n}w) = d^n \cdot \mathcal{A}(uv^mw) \text{ for every } n \in \mathbb{N}. \quad (5)$$

However, weak pumpability requires only that Equation (5) applies only for $n = 1$:

► **Definition 22.** (*Weak Pumpability*) A weighted automaton \mathcal{A} of size m is weakly pumpable if for all $u, v, w \in \Sigma^*$ such that $M(v)$ is p-triangular there is an entry d of the diagonal of $M(v)$ satisfying

$$\mathcal{A}(uv^{m+1}w) = d \cdot \mathcal{A}(uv^mw). \quad (6)$$

It is clear that if \mathcal{A} is pumpable, then in particular the weaker property is satisfied. We prove the converse implication: if Equation (6) is satisfied by all triples u, v, w such that $M(v)$ is p-triangular, we show that each such triple also satisfies Equation (5). By applying Equation 6 to a family of the triples of the form $(uv^n, v, w)_{n \in \mathbb{N}}$, we get inductively that for every $n \in \mathbb{N}$, there exist m integers j_1, j_2, \dots, j_m summing up to n such that $\mathcal{A}(uv^{m+n}w) = \prod_{i=1}^m (M(v))_{ii}^{j_i} \cdot \mathcal{A}(uv^mw)$. However, Lemma 8 gives us another expression for these weights: $\mathcal{A}(uv^{m+n}w) = \sum_{i=1}^m (M(v))_{ii}^n \cdot p_i(n)$, where p_i are polynomials. The only way for these two expressions to match is that Equation 5 holds, which shows that \mathcal{A} is pumpable. We prove this formally (proof in Appendix E):

► **Lemma 23.** A weighted automaton \mathcal{A} is weakly pumpable if and only if \mathcal{A} is pumpable.

For every weighted automaton \mathcal{A} we show how to construct a weighted automaton $\mathcal{P}_{\mathcal{A}}$ that has size exponential with respect to \mathcal{A} , and maps every word to 0 if and only if \mathcal{A} is weakly pumpable. Then, deciding weak pumpability of \mathcal{A} amounts to deciding zeroness of $\mathcal{P}_{\mathcal{A}}$, which can be done in polynomial space (with respect to the size of \mathcal{A}). The next lemma presents constructions tailored to the study of pumpable automata to be used as building blocks to construct $\mathcal{P}_{\mathcal{A}}$: Given \mathcal{A} , we show how to construct automata that recognise the triples u, v, w such that $M(v)$ is p-triangular, and that compute the value mapped by \mathcal{A} to the words $(uv^n w)_{n \in \mathbb{N}}$ given only u, v, w .

► **Lemma 24.** Let $\mathcal{A} = (Q, \Sigma, M, I, F)$ be a weighted automaton of size $m = |Q|$ over Σ , and let $\$ \notin \Sigma$.

1. There exists an automaton \mathcal{T} of size $2^{m^2} + 3$ and norm 1 satisfying

$$\begin{aligned} \mathcal{T}(u\$v\$w) &= 1 && \text{for all } u, v, w \in \Sigma^* \text{ such that } M(v) \text{ is p-triangular;} \\ \mathcal{T}(u) &= 0 && \text{for all other } u \in (\Sigma \cup \{\$\})^*. \end{aligned}$$

2. For all $n \in \mathbb{N}$ there exist automata \mathcal{B}_n of size $m^{2n} + 2m$ and norm $\|\mathcal{A}\|^n$ satisfying

$$\mathcal{B}_n(u\$v\$w) = \mathcal{A}(uv^n w); \quad \text{for all } u, v, w \in \Sigma^*.$$

3. For all $1 \leq i \leq m$ there exists an automaton \mathcal{C}_i of size $m + 2$ and norm $\|\mathcal{A}\|$ satisfying

$$\mathcal{C}_i(u\$v\$w) = (M(v))_{ii} \quad \text{for all } u, v, w \in \Sigma^*.$$

► **Remark 25.** The values of \mathcal{B}_n and \mathcal{C}_i are unimportant and unspecified when the input does not take the form $u\$v\w for $u, v, w \in \Sigma$.

We are now ready to show how to decide pumpability in polynomial space. Let \mathcal{A} be a weighted automaton of size m , let $\$$ be a fresh symbol that is not in the alphabet of \mathcal{A} , and let \mathcal{T} , $(\mathcal{B}_n)_{n \in \mathbb{N}}$ and $(\mathcal{C}_i)_{1 \leq i \leq m}$ be the weighted automata constructed in Lemma 24. Using the result of 6 that the difference and product of functions recognised by weighted automata are themselves effectively expressible by weighted automata, let

$$\mathcal{P}_{\mathcal{A}} = \mathcal{T} \cdot \prod_{i=1}^m (\mathcal{B}_{m+1} - \mathcal{B}_m \mathcal{C}_i).$$

► **Lemma 26.** $\mathcal{P}_{\mathcal{A}}$ maps every word to 0 if and only if \mathcal{A} is weakly pumpable.

Proof. By definition, $\mathcal{P}_{\mathcal{A}}$ maps every word to 0 if and only if for every triple $u, v, w \in \Sigma^*$ such that $\mathcal{T}(u\$v\$w) > 0$ (i.e. $M(v)$ is p-triangular), there exists $1 \leq i \leq m$ such that $\mathcal{B}_{m+1}(u\$v\$w) = \mathcal{B}_m(u\$v\$w)\mathcal{C}_i(u\$v\$w)$, that is,

$$\text{there exists } 1 \leq i \leq m \text{ satisfying } \mathcal{A}(uv^{m+1}w) = (M(v))_{ii} \cdot \mathcal{A}(uw^m w). \quad \blacktriangleleft$$

It remains to confirm that we can check whether $\mathcal{P}_{\mathcal{A}}$ maps every word to 0 in space polynomial in m . First, remark that the size of $\mathcal{P}_{\mathcal{A}}$ is exponential in m :

$$|\mathcal{P}_{\mathcal{A}}| = |\mathcal{T}| \cdot \prod_{i=1}^m (|\mathcal{B}_{m+1}| + |\mathcal{B}_m| |\mathcal{C}_i|) = (2^{m^2} + 3) \cdot (m^{4m+1})^m \leq m^{6m^2} \quad (\text{assuming } m \geq 2).$$

and the norm is also exponential in $|\mathcal{A}|$, this entails that the weight of any edge can be encoded in polynomial space:

$$\|\mathcal{P}_{\mathcal{A}}\| = \|\mathcal{T}\| \cdot \prod_{i=1}^m \max\{\|\mathcal{B}_{m+1}\|, \|\mathcal{B}_m\| \|\mathcal{C}_i\|\} = 1 \cdot (\|\mathcal{A}\|^{m+1})^m = \|\mathcal{A}\|^{m^2+m}. \quad (7)$$

We show that zeroness of the exponential size automaton $\mathcal{P}_{\mathcal{A}}$ can be decided in PSPACE. It is known that deciding equivalence of \mathbb{Q} -weighted automata is in NC2 [29, 18]. Recall, NC is the class of problems decidable using a circuit of polynomial size and polylogarithmic depth, with branching width at most two. Such problems can be solved sequentially in polylogarithmic space [27].

In particular, this means the zeroness problem can be decided in polylogarithmic space in the size of the automaton. Thus applying the polylogarithmic space zeroness algorithm to the automaton $\mathcal{P}_{\mathcal{A}}$ requires only polylogarithmic space with respect to the exponential size automaton $\mathcal{P}_{\mathcal{A}}$, equivalently polynomial space with respect to \mathcal{A} . To conclude we show that any transition weight in $\mathcal{P}_{\mathcal{A}}$ can be computed in polynomial space, so that the equivalence procedure has access to any edge weight of $\mathcal{P}_{\mathcal{A}}$, and thus any bit of the representation of $\mathcal{P}_{\mathcal{A}}$, in PSPACE.

► **Lemma 27.** Given two states q, q' of $\mathcal{P}_{\mathcal{A}}$ and $a \in \Sigma \cup \{\$\}$, the transition weight $M_{\mathcal{P}_{\mathcal{A}}}(a)_{q,q'}$ can be computed in polynomial space.

Proof. First observe the same is true for $\mathcal{T}, \mathcal{B}_{m+1}, \mathcal{B}_m$ and \mathcal{C}_i for all $1 \leq i \leq m$. The automaton $\mathcal{P}_{\mathcal{A}}$ is built applying Lemma 6 polynomially many times. By induction, at each step the norm is at most exponential (in total bounded by Equation (7)) and the weight can be computed recursively in polynomial space using the second part of Lemma 6. \blacktriangleleft

6.2 Deciding blind pumpability

We show that we can also decide blind pumpability in PSPACE, again by reducing to the zeroness problem for weighted automata. Let us compare pumpability and blind pumpability:

- A pumpable automaton requires, for any (u, v, w) triple with $M(v)$ p-triangular, the existence of d such that $\mathcal{A}(uv^{m+n}w) = d^n \cdot \mathcal{A}(uv^m w)$ for all $n \in \mathbb{N}$.
- A blindly pumpable automaton requires, for any (u, v) pair with $M(v)$ p-triangular, the existence of d such that $\mathcal{A}(uv^{m+n}w) = d^n \cdot \mathcal{A}(uv^m w)$ for all $w \in \Sigma^*$ and $n \in \mathbb{N}$.

That is, in order to be blindly pumpable an automaton needs to be pumpable, and on top of that for any (u, v) pair the choice of d has to be the same for all w . We show how to decide this property, assuming it is already known that \mathcal{A} is pumpable using the previous section. We start with the assumption that it is already known that \mathcal{A} is pumpable, by first applying the algorithm of the previous section. We will then encode the requirement that the choice of d be the same for any two suffixes into a zeroness problem (as in the previous section). In order to do this we generalise the automata T and B_i of Lemma 24 to read two suffixes:

► **Lemma 28.** *There exists an automaton \mathcal{T}' such that $\mathcal{T}'(u\$v\$w\$w') = 1$ if $M(v)$ is p-triangular, and \mathcal{T}' is zero on all other inputs. There exist automata $\mathcal{B}_{1,n}, \mathcal{B}_{2,n}$ such that $\mathcal{B}_{1,n}(u\$v\$w\$w') = \mathcal{B}_n(u\$v\$w)$ and $\mathcal{B}_{2,n}(u\$v\$w\$w') = \mathcal{B}_n(u\$v\$w')$.*

Proof. These automata are easily obtained by modifying the constructions in Lemma 24. ◀

► **Lemma 29.** *Let m be the size of \mathcal{A} and let $\mathcal{Q}_{\mathcal{A}} = \mathcal{T}' \cdot (\mathcal{B}_{1,m+1} \cdot \mathcal{B}_{2,m} - \mathcal{B}_{1,m} \cdot \mathcal{B}_{2,m+1})$. Then \mathcal{A} is blindly pumpable if and only if \mathcal{A} is pumpable and $\mathcal{Q}_{\mathcal{A}}$ maps every word to zero.*

Proof. We verify for every u, v and every w, w' that the same d is used to assert that $\mathcal{A}(uv^{m+n}w) = d^n \cdot \mathcal{A}(uv^m w)$ and $\mathcal{A}(uv^{m+n}w') = d^n \cdot \mathcal{A}(uv^m w')$. Since \mathcal{A} is pumpable, the same d is used for every n , so it is sufficient to verify the property only at $n = 1$. In particular, whenever $\mathcal{A}(uvw) \neq 0$, we require that

$$\frac{\mathcal{A}(uv^{m+1}w)}{\mathcal{A}(uv^m w)} = \frac{\mathcal{A}(uv^{m+1}w')}{\mathcal{A}(uv^m w')} \text{ for all } w \in \Sigma^*. \quad (8)$$

Note that, if $\mathcal{A}(uv^m w) = 0$, then by pumpability $\mathcal{A}(uv^{m+n}w) = 0$. Then $\mathcal{A}(uv^{m+n}w) = d^n \mathcal{A}(uv^m w) = 0$ for any d , thus in particular the same choice of d can be made as $\mathcal{A}(uv^{m+n}w') = d^n \mathcal{A}(uv^m w') = 0$.

Observe that $\mathcal{Q}_{\mathcal{A}}$ is zero when either:

- \mathcal{T}' is zero, that is u, v, w, w' do not need to satisfy Equation (8), or,
- $\mathcal{A}(uv^{m+k}w) = 0$, or $\mathcal{A}(uv^{m+k}w') = 0$ for $k \in \{0, 1\}$, or
- for every u, v, w, w' Equation (8) holds. ◀

Like $\mathcal{P}_{\mathcal{A}}$, the size and norm of $\mathcal{Q}_{\mathcal{A}}$ is exponential, and we can test zeroness of $\mathcal{Q}_{\mathcal{A}}$ in PSPACE.

7 Conclusion

As mentioned in the introduction our PSPACE upper bounds are not constructive. If one would like to construct an equivalent deterministic automaton its size would be bounded by a tower of exponents (see Remark 17, and recall that the constant $R_{\mathcal{A}}$ is obtained from Lemma 18). We cannot extract the unambiguous automaton from our techniques³. Recall

³ Note that it can be constructed, just not necessarily from our techniques: since from the decision procedure one can be sure of its existence, we can enumerate unambiguous weighted automaton and test each for equivalence.

that for unambiguisation in the proof of Proposition 11 we rely on Bell and Smertnig’s result: Theorem 15. One could imagine a direct construction as we provide for the deterministic automaton. The issue is that for unambiguous automata one would need to keep track of all nonzero runs. Given an unambiguous automaton it is known that the number of such runs is bounded [30], but the size of the unambiguous automaton could be arbitrary big. A future work question is whether one could solve the determinisation and unambiguisation problems constructively in elementary time and space.

We do not provide any lower bound and we are unaware of such a result, even for the general class of weighted automata over rationals. We write a simple observation why obtaining lower bounds seems to be difficult: Suppose one wants to encode a problem, e.g. satisfiability of a SAT formula. One would like to define a weighted automaton \mathcal{A} that behaves like a deterministic (unambiguous) weighted automaton \mathcal{D} except if the formula is satisfied, which unlocks some nondeterministic (ambiguous) behaviour. The issue is that all natural encodings can be verified with an equivalence query to the deterministic (unambiguous) automaton \mathcal{D} , which over the rationals is in NC^2 [29].

References

- 1 Corentin Barloy, Nathanaël Fijalkow, Nathan Lhote, and Filip Mazowiecki. A robust class of linear recurrence sequences. *Inf. Comput.*, 289(Part):104964, 2022. doi:10.1016/j.ic.2022.104964.
- 2 Jason Bell and Daniel Smertnig. Noncommutative rational pólya series. *Selecta Mathematica*, 27(3):1–34, 2021.
- 3 Jason P. Bell and Daniel Smertnig. Computing the linear hull: Deciding deterministic? and unambiguous? for weighted automata over fields. In *LICS*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175691.
- 4 Paul C. Bell. Polynomially ambiguous probabilistic automata on restricted languages. *J. Comput. Syst. Sci.*, 127:53–65, 2022. doi:10.1016/j.jcss.2022.02.002.
- 5 Yahia Idriss Benalioua, Nathan Lhote, and Pierre-Alain Reynier. Register minimization of cost register automata over a field. *CoRR*, abs/2307.13505, 2023. arXiv:2307.13505, doi:10.48550/arXiv.2307.13505.
- 6 Georgina Bumpus, Christoph Haase, Stefan Kiefer, Paul-Ioan Stoienescu, and Jonathan Tanner. On the size of finite rational matrix semigroups. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 115:1–115:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.115.
- 7 Dmitry Chistikov, Stefan Kiefer, Andrzej S. Murawski, and David Purser. The big-o problem. *Log. Methods Comput. Sci.*, 18(1), 2022. doi:10.46298/lmcs-18(1:40)2022.
- 8 Thomas Colcombet. Unambiguity in automata theory. In Jeffrey O. Shallit and Alexander Okhotin, editors, *Descriptive Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings*, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015. doi:10.1007/978-3-319-19225-3_1.
- 9 Wojciech Czerwinski and Piotr Hofman. Language inclusion for boundedly-ambiguous vector addition systems is decidable. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 16:1–16:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.16.
- 10 Wojciech Czerwinski, Engel Lefauchaux, Filip Mazowiecki, David Purser, and Markus A. Whiteland. The boundedness and zero isolation problems for weighted automata over non-negative rationals. In Christel Baier and Dana Fisman, editors, *LICS ’22: 37th Annual*

- ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 15:1–15:13. ACM, 2022. doi:10.1145/3531130.3533336.
- 11 Laure Daviaud. Register complexity and determinisation of max-plus automata. *ACM SIGLOG News*, 7(2):4–14, 2020. doi:10.1145/3397619.3397621.
 - 12 Laure Daviaud, Marcin Jurdzinski, Ranko Lazic, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. When are emptiness and containment decidable for probabilistic automata? *J. Comput. Syst. Sci.*, 119:78–96, 2021. doi:10.1016/j.jcss.2021.01.006.
 - 13 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.
 - 14 Nathanaël Fijalkow, Cristian Riveros, and James Worrell. Probabilistic automata of bounded ambiguity. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CONCUR.2017.19.
 - 15 Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem’s problem—on the border between decidability and undecidability. Technical report, Citeseer, 2005.
 - 16 Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. Polynomial invariants for affine programs. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 530–539. ACM, 2018. doi:10.1145/3209108.3209142.
 - 17 Ismaël Jecker. A ramsey theorem for finite monoids. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 44:1–44:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.44.
 - 18 Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *Log. Methods Comput. Sci.*, 9(1), 2013. doi:10.2168/LMCS-9(1:8)2013.
 - 19 Daniel Kirsten and Ina Mäurer. On the determinization of weighted automata. *J. Autom. Lang. Comb.*, 10(2/3):287–312, 2005. doi:10.25596/jalc-2005-287.
 - 20 Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327(3):349–373, 2004. doi:10.1016/j.tcs.2004.02.049.
 - 21 Peter Kostolányi. Determinisability of unary weighted automata over the rational numbers. *Theor. Comput. Sci.*, 898:110–131, 2022. doi:10.1016/j.tcs.2021.11.002.
 - 22 Sylvain Lombardy and Jacques Sakarovitch. Sequential? *Theor. Comput. Sci.*, 356(1-2):224–244, 2006. doi:10.1016/j.tcs.2006.01.028.
 - 23 Mehryar Mohri. Finite-state transducers in language and speech processing. *Comput. Linguistics*, 23(2):269–311, 1997.
 - 24 Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.
 - 25 Mikhail A. Raskin. A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 138:1–138:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.138.
 - 26 Christophe Reutenauer. On polya series in noncommuting variables. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arithmetic, and Categorical Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 391–396. Akademie-Verlag, Berlin, 1979.
 - 27 Walter L. Ruzzo. On uniform circuit complexity. *J. Comput. Syst. Sci.*, 22(3):365–383, 1981. doi:10.1016/0022-0000(81)90038-6.

- 28 Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270, 1961. doi:10.1016/S0019-9958(61)80020-X.
- 29 Wen-Guey Tzeng. On path equivalence of nondeterministic finite automata. *Inf. Process. Lett.*, 58(1):43–46, 1996. doi:10.1016/0020-0190(96)00039-7.
- 30 Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991. doi:10.1016/0304-3975(91)90381-B.

A Proofs for Section 1 (Introduction)

In this section we describe the twin property for unambiguous weighted automata over the rational field. This is unnecessary for our proofs, but it gives an intuition why the unambiguisation and determinisation problems are similar.

Fix an unambiguous weighted automaton over rationals $\mathcal{A} = (Q, \Sigma, M, I, F)$. We say that it is trimmed if all states are reachable and coreachable (see Section 2). Note that for unambiguous weighted automata the existence of a run on word w is equivalent to $\mathcal{A}(w) \neq 0$. This is because for unambiguous weighted automata the sum operator is not used, and thus Remark 4 does not apply here. We write $q \xrightarrow{u} p$ if $M(u)_{q,p} \neq 0$. We write $I \xrightarrow{u} p$ if there exists $r \in Q$ such that $I(r) \neq 0$ and $M(u)_{r,p} \neq 0$. In words, there is a nonzero run from an initial state to p over the word u . Similarly, we write $p \xrightarrow{u} F$ if there exists $r \in Q$ such that $F(r) \neq 0$ and $M(u)_{p,r} \neq 0$. Note that if \mathcal{A} is unambiguous then r is unique both in runs from I and runs to F .

We say that \mathcal{A} has the twin property if for every $u, v \in \Sigma^*$ and $p, q \in Q$ such that $I \xrightarrow{u} p \xrightarrow{v} p$ and $I \xrightarrow{u} q \xrightarrow{v} q$ we have $|M(v)_{p,p}| = |M(v)_{q,q}|$.⁴ A standard argument shows that the twin property can be detected in polynomial time by the reduction to the equality test for weighted automata (which is in polynomial time [29, 18].)

► **Lemma 30.** *The twin property of an unambiguous automaton \mathcal{A} can be decided in polynomial time.*

Proof. We defined two automata $\mathcal{A}_1, \mathcal{A}_2$ operating on pairs of runs on \mathcal{A} . Intuitively, both $\mathcal{A}_1, \mathcal{A}_2$ will track and verify the pair of runs in which $I \xrightarrow{u} p \xrightarrow{v} p$ and $I \xrightarrow{u} q \xrightarrow{v} q$. \mathcal{A}_1 will output the value of $M(v)_{p,p}$ and \mathcal{A}_2 will output the value of $M(v)_{q,q}$, which we require to be equal. We give further details:

Let $T \subseteq Q \times \Sigma \times Q \times \mathbb{Z}$ be transitions of \mathcal{A} , and so $w \in (T \otimes T)^*$ describes a pairs of runs in \mathcal{A} . We call a pair of runs w valid if at each step both transitions are non-zero and take the same symbol in Σ , and the starting state of the transition complies with the previous. We define $\mathcal{A}_1, \mathcal{A}_2$ to have non-zero value on words of the form $w_1 \$ w_2$, where w_1 and w_2 are both valid pairs of runs on words u and v . We define $\mathcal{A}_1(w_1 \$ w_2) = |M_{p,p}(v)|$ where the first run in w_1 ends in state p and \mathcal{A}_2 takes values $|M_{q,q}(v)|$ where second run on u end in q .

It follows that, \mathcal{A} satisfies the twin property if and only if $\mathcal{A}_1 = \mathcal{A}_2$. Recall, deciding if $\mathcal{A}_1 = \mathcal{A}_2$ can be decided in polynomial time [29, 18].

It remains to verify that \mathcal{A}_1 and \mathcal{A}_2 can be constructed in polynomial time. We describe \mathcal{A}_1 (as \mathcal{A}_2 is symmetric). Take $|\mathcal{A}| + 1$ copies of \mathcal{A} : the first copy of \mathcal{A} deterministically tracks the first run of w_1 and every transition has weight 1. From state p , the $\$$ takes the automaton to a copy of \mathcal{A} in which the initial and accepting state is p and the weight of every transition has the absolute value of its usual weight (note that, p was final state of the first run in w_1). Note that it is safe to take the absolute value of the weight since \mathcal{A} is unambiguous. ◀

► **Proposition 31.** *A trimmed unambiguous weighted automaton \mathcal{A} over the rational field is determinisable if and only if it has the twin property.*

Proof. (\implies) For a contradiction, suppose that the twin property does not hold and let $u, v \in \Sigma^*$ and $p, q \in Q$ be such that $I \xrightarrow{u} p \xrightarrow{v} p$, $I \xrightarrow{u} q \xrightarrow{v} q$ and $|M(v)_{p,p}| \neq |M(v)_{q,q}|$.

⁴ Compared to standard twin property, e.g. used in [23, 19] for the tropical semiring, we needed to change the weights of cycles to their absolute values.

We denote x_p and x_q to be the absolute values of unique runs in $I \xrightarrow{u} p \xrightarrow{v} p$, $I \xrightarrow{u} q \xrightarrow{v} q$, and we write $y_p = |M(v)_{p,p}|$, $y_q = |M(v)_{q,q}|$. Since \mathcal{A} is trim there exist words w_p and w_q such that $p \xrightarrow{w_p} F$ and $q \xrightarrow{w_q} F$. Let z_p, z_q be the absolute values of the unique runs. Then $\mathcal{A}(uv^n w_p) = x_p z_p \cdot y_p^n$ and $\mathcal{A}(uv^n w_q) = x_q z_q \cdot y_q^n$. Recall that $|y_p| \neq |y_q|$ and thus $\lim_{n \rightarrow +\infty} \frac{|\mathcal{A}(uv^n w_p)|}{|\mathcal{A}(uv^n w_q)|}$ is either 0 or $+\infty$.

Suppose there is an equivalent deterministic automaton \mathcal{A}' , i.e. $\mathcal{A}' = \mathcal{A}$. For every n there is a unique run witnessing $I \xrightarrow{uv^n}$, let a_n be its absolute value. The words w_p, w_q and the size of \mathcal{A}' are fixed. Thus the values $|\mathcal{A}(uv^n w_p)|$ and $|\mathcal{A}(uv^n w_q)|$ are both equal to a_n multiplied by some constants. This is a contradiction.

(\Leftarrow) Suppose the twin property holds. The construction of the deterministic automaton is very similar to the second part of the proof of Proposition 11. Namely, the deterministic automaton remembers unprocessed parts of the input word. Once the word becomes big enough the automaton depumps a value. We only sketch the construction as it is standard.

The size of the word stored in the automaton is bounded by R such that for every word w of length at least R there exists an infix v such that $M(v)$ has idempotent structure. Such an R exists by standard Ramsey arguments, see e.g. [17]. For trim unambiguous weighted automata this means that $M(v)$ is diagonal. Let $w = uvv'$. The twin property guarantees that all nonzero values on the diagonal reachable by u must have the same absolute value. Thus every time we find such an idempotent v we can deterministically depump this absolute value. Note that to correctly evaluate the automaton we also need extra states that allow us to keep the sign of the runs. This information can be stored in finite space, as unambiguous automata have a bounded number of runs [30] and we only need to remember the parities of negative signs. \blacktriangleleft

B Proofs for Section 2 (Preliminaries)

► **Lemma 6.** *Let \mathcal{A}_1 and \mathcal{A}_2 be weighted automata over Σ that have size m_1 and m_2 .*

- *The function $-\mathcal{A} : u \mapsto -\mathcal{A}(u)$ is recognised by an automaton of size $|\mathcal{A}|$ and norm $\|\mathcal{A}\|$;*
- *The function $\mathcal{A}_1 - \mathcal{A}_2 : u \mapsto \mathcal{A}_1(u) - \mathcal{A}_2(u)$ is recognised by an automaton of size $|\mathcal{A}_1| + |\mathcal{A}_2|$ and norm $\max\{\|\mathcal{A}_1\|, \|\mathcal{A}_2\|\}$;*
- *The function $\mathcal{A}_1 \cdot \mathcal{A}_2 : u \mapsto \mathcal{A}_1(u) \cdot \mathcal{A}_2(u)$ is recognised by an automaton of size $|\mathcal{A}_1| \cdot |\mathcal{A}_2|$ and norm $\|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|$.*

Furthermore, if we assume that we can compute in space $O(\log(\|\mathcal{A}_1\|))$ (respectively $O(\log(\|\mathcal{A}_2\|))$) the weight of a transition in \mathcal{A}_1 (respectively \mathcal{A}_2) given by a letter $a \in \Sigma$ and two states of \mathcal{A}_1 (respectively \mathcal{A}_2), then we can compute in space $O(\log(\|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|))$ the weight of a transition in $\mathcal{A}_1 - \mathcal{A}_2$ or $\mathcal{A}_1 \cdot \mathcal{A}_2$ given by a letter $a \in \Sigma$ and two states of the corresponding automaton.

Proof. Negation is obtained by taking $-I$ as the initial vector, disjoint union of two automata encodes summation, which together give the difference.

The following construction gives the product: given (Q, Σ, M, I, F) , construct $(Q \times Q, \Sigma, M', I', F')$ with M' given by transitions $(q, q') \xrightarrow{a:x \cdot y} (p, p')$ whenever $q \xrightarrow{a:x} p$ in \mathcal{A}_1 and $q' \xrightarrow{a:y} p'$ in \mathcal{A}_2 . For all, $q, q' \in Q$ let $I'(q, q') = I(q)I(q')$ and $F'(q, q') = F(q)F(q')$.

Given $q, q' \in \mathcal{A}_1 - \mathcal{A}_2$ the transition probability of $M_{\mathcal{A}_1 - \mathcal{A}_2}(a)_{q,q'}$ can easily be computed by looking up $M_{\mathcal{A}_1}(a)_{q,q'}$ or $M_{\mathcal{A}_2}(a)_{q,q'}$ assuming q, q' were from the same automaton, or returning 0 if not.

Given $(q, q'), (p, p') \in \mathcal{A}_1 \cdot \mathcal{A}_2$ the transition probability of $M_{\mathcal{A}_1 \cdot \mathcal{A}_2}(a)_{q,q'}$ can easily be computed by taking the product of $M_{\mathcal{A}_1}(a)_{q,q'}$ and $M_{\mathcal{A}_2}(a)_{p,p'}$. \blacktriangleleft

B.1 Constructing an equivalent automaton with non-negative transitions

Given $\mathcal{A} = (Q, \Sigma, M, I, F)$, we construct equivalent $\mathcal{A}' = (Q', \Sigma, M', I', F')$. Let $Q' = \{q_- \mid q \in Q\} \cup \{q_+ \mid q \in Q\}$. Given a transition $q \xrightarrow{a:x} p$ in \mathcal{A} , we add the following transitions to \mathcal{A}' :

- $q_+ \xrightarrow{a:x} q_+$ and $q_- \xrightarrow{a:x} q_-$ if $x \geq 0$,
- $q_+ \xrightarrow{a:-x} q_-$ and $q_- \xrightarrow{a:-x} q_+$ if $x < 0$.

For every $q \in Q$, we let:

- $I(q_+) = I(q)$ if $I(q) \geq 0$ and 0 otherwise,
- $I(q_-) = -I(q)$ if $I(q) < 0$ and 0 otherwise,
- $F(q_+) = F(q)$,
- $F(q_-) = -F(q)$.

▷ **Claim 32.** $\mathcal{A}'(w) = \mathcal{A}(w)$ for all $w \in \Sigma^*$.

Proof. Consider a run on w from q to p in \mathcal{A} . We consider the equivalent run in \mathcal{A}' , which goes from either q_+ or q_- to p_+ or p_- . It is clear that the absolute value is maintained by the translation, we verify the correct sign is preserved. The sign of I', M' are positive, thus the sign of the run in \mathcal{A}' depends only on F' .

First suppose the number of negative transitions taken in the run is even. In this case the sign should be that of $I(q)F(q)$. If $I(q) \geq 0$, then the path goes from q_+ to p_+ , which has the same sign as $F(p)$. If $I(q) < 0$ then the path goes from q_- to p_- , in which case the sign of both I and F are swapped, thus the sign of $I(q)F(q)$ is the same as $I(q_-)F(p_-)$.

Secondly suppose the number of negative transitions taken in the run is odd, in which case the sign should be swapped from that of $I(q)F(p)$. If $I(q) \geq 0$, the sign should be opposite to $F(p)$, indeed the path goes from q_+ to p_- , for which the sign $F'(p_-)$ is swapped from that of $F(p)$. If $I(q) < 0$, the sign should be the same as $F(p)$, indeed the path goes from q_- to p_+ , for which the sign of $F'(p_+)$ is the same as $F(p)$. ◀

► **Remark 33.** As similar construction can assume that negative values appear only in the initial vector, rather than the final vector.

C Proofs for Section 3 (Overview of our main result)

C.1 Proof of Lemma 8

► **Lemma 8.** *Let M be an $m \times m$ p -triangular matrix with the set of diagonal entries $\{d_1, d_2, \dots, d_k\} \subseteq \mathbb{N}$, and let $\vec{x}, \vec{y} \in \mathbb{Q}^m$. There exist k polynomials p_1, p_2, \dots, p_k such that*

$$\vec{x}^T \cdot M^n \cdot \vec{y} = \sum_{i=1}^k d_i^n \cdot p_i(n) \text{ for all } n \geq m. \quad (1)$$

Moreover, if the matrix M is invertible; p_1, p_2, \dots, p_k are all constant polynomials; and at most one p_j is not constantly 0, then

$$\vec{x}^T \cdot M^n \cdot \vec{y} = d_j^n \cdot \vec{x}^T \cdot \vec{y} \text{ for all } n \geq 0. \quad (2)$$

The proof relies on two lemmas. Lemma 34 shows that every p -triangular matrix can be transformed (via an appropriate change of basis) into a diagonal block matrix where each block is an upper triangular matrix whose diagonal entries share the same value. It is a standard construction in linear algebra: for instance, such a change of basis is used to transform a matrix into its Jordan normal form.

► **Lemma 34.** *Let M be a p -triangular matrix with diagonal entries $\{d_1, d_2, \dots, d_k\} \subseteq \mathbb{N}$. There exists an invertible matrix P such that $P^{-1}MP = B_1 \oplus B_2 \oplus \dots \oplus B_k$ is a block diagonal matrix where each B_i is an upper triangular matrix whose diagonal entries all equal d_i .*

We use this result to reduce the study of the powers of a matrix to the study of the powers of its blocks. We move to Lemma 35 that shows that these blocks are easy to handle, using the fact that the diagonal entries of an individual block share the same value.

► **Lemma 35.** *Let B be an $m \times m$ upper triangular matrix whose diagonal entries all have the same value $d \in \mathbb{N}$, and let $\vec{x}, \vec{y} \in \mathbb{Q}^m$. There exists a polynomial p such that*

$$\vec{x}^T \cdot B^n \cdot \vec{y} = d^n \cdot p(n) \text{ for all } n \geq m.$$

Moreover, if $d > 0$ and p is a constant polynomial then

$$\vec{x}^T \cdot B^n \cdot \vec{y} = d^n \cdot \vec{x}^T \cdot \vec{y} \text{ for all } n \geq 0.$$

Proof of Lemma 35. Let B be an upper triangular $m \times m$ matrix whose diagonal entries all have the same value $d \in \mathbb{N}$. Remark that in the specific case where $d = 0$ the matrix B is nilpotent ($B^m = 0$), which immediately implies the desired statement: for all vectors $\vec{x}, \vec{y} \in \mathbb{Q}^m$ we have $\vec{x}^T \cdot B^n \cdot \vec{y} = 0$ for all $n \geq m$, hence every polynomial p fits. We now suppose that $d > 0$. We decompose B as the sum of the matrix $d \cdot \text{Id}$ obtained by multiplying the identity by d , and the remainder $B_0 = B - d \cdot \text{Id}$. This decomposition has the following properties:

- The two components commute multiplicatively: $(d \cdot \text{Id}) \cdot B_0 = dB_0 = B_0 \cdot (d \cdot \text{Id})$;
- B_0 is a strictly upper triangular matrix, and it is nilpotent: $B_0^m = 0$.

This allows us to give a nice expression for the value of the powers of B :

$$B^n = \sum_{i=0}^n \binom{n}{i} (d \cdot \text{Id})^{n-i} \cdot B_0^i = \sum_{i=0}^{m'} \binom{n}{i} d^{n-i} B_0^i = d^n \cdot \sum_{i=0}^{m'} \binom{n}{i} \cdot \frac{B_0^i}{d^i} \text{ with } m' = \min(n, m).$$

Therefore, for every pair of vectors $\vec{x}, \vec{y} \in \mathbb{Q}^m$ we get

$$\vec{x}^T \cdot B^n \cdot \vec{y} = d^n \cdot \underbrace{\sum_{i=0}^{m'} \binom{n}{i} \cdot \frac{\vec{x}^T \cdot B_0^i \cdot \vec{y}}{d^i}}_{p(n)} \text{ with } m' = \min(n, m). \quad (9)$$

As indicated by the underbrace, we let p denote the sum in Equation (9) in the case where $m' = m$. To conclude the proof, it remains to show that the expression p is a polynomial in n , and that if p is of degree 0 then $\vec{x}^T \cdot B^n \cdot \vec{y} = d^n \cdot \vec{x}^T \cdot \vec{y}$ for all $n \geq 0$. Let us focus on the summands composing p : For every $0 \leq i \leq m$ we have

$$\binom{n}{i} \cdot \frac{\vec{x}^T \cdot B_0^i \cdot \vec{y}}{d^i} = \frac{\vec{x}^T \cdot B_0^i \cdot \vec{y}}{\underbrace{d^i \cdot i(i-1)(i-2) \cdot \dots \cdot 2 \cdot 1}_{\in \mathbb{Q}}} \cdot n(n-1)(n-2) \dots (n-i+1).$$

This expression is a polynomial of degree i if $\vec{x}^T \cdot B_0^i \cdot \vec{y} \neq 0$, and it is constantly zero if $\vec{x}^T \cdot B_0^i \cdot \vec{y} = 0$. Since $p(n)$ is equal to the sum of all these expressions, we get that $p(n)$ is a polynomial whose degree is equal to the largest i satisfying $\vec{x}^T \cdot B_0^i \cdot \vec{y} \neq 0$ (or $p(n)$ is

constantly zero if no such i exists). In particular, if p is a constant polynomial we get that $\vec{x}^T \cdot B_0^i \cdot \vec{y} = 0$ for every $1 \leq i \leq m$, thus Equation (9) yields the desired expression:

$$\vec{x}^T \cdot B^n \cdot \vec{y} = d^n \cdot \binom{n}{0} \cdot \vec{x}^T \cdot B_0^0 \cdot \vec{y} = d^n \cdot \vec{x}^T \cdot \vec{y} \text{ for all } n \geq 0. \quad \blacktriangleleft$$

Proof of Lemma 8. Let M be an $m \times m$ upper triangular matrix with diagonal entries $\{d_1, d_2, \dots, d_m\} \subseteq \mathbb{N}$. We begin by transforming M into a block diagonal matrix according to Lemma 34: $P^{-1}MP = B_1 \oplus B_2 \oplus \dots \oplus B_k$. Since $M^n = P \cdot (B_1^n \oplus B_2^n \oplus \dots \oplus B_k^n) \cdot P^{-1}$, we get

$$\vec{x}^T \cdot M^n \cdot \vec{y} \stackrel{10.1}{=} \sum_{i=1}^k \vec{x}_i^T \cdot B_i^n \cdot \vec{y}_i \stackrel{10.2}{=} \sum_{i=1}^k d_i^n \cdot p_i(n) \text{ for all } n \geq m, \text{ where:} \quad (10)$$

10.1 The vectors $\vec{x}_1^T, \vec{x}_2^T, \dots, \vec{x}_k^T$, respectively $\vec{y}_1, \vec{y}_2, \dots, \vec{y}_k$, are obtained by cutting $\vec{x}^T \cdot P$, respectively $P^{-1} \cdot \vec{y}$, into parts whose sizes fit the blocks B_1, B_2, \dots, B_k ;

10.2 The polynomials p_1, p_2, \dots, p_k are obtained by applying Lemma 35 to each block.

This concludes the proof of Equation (1) of Lemma 8.

In order to prove the second part, we now suppose that the matrix M is invertible. Since M is an upper triangular matrix, this implies that none of its diagonal entries d_i equals zero. Therefore, if on top of that we suppose that p_1, p_2, \dots, p_k are all constant polynomials, and that only p_j is not constantly 0, we can apply the second part of Lemma 35 to get:

11.1 $\vec{x}_j^T \cdot B_j^n \cdot \vec{y}_j = d_j^n \vec{x}_j^T \cdot \vec{y}_j$ for all $n \in \mathbb{N}$;

11.2 For every $i \neq j$, $\vec{x}_i^T \cdot B_i^n \cdot \vec{y}_i = 0$ for all $n \in \mathbb{N}$ (thus in particular $\vec{x}_i^T \cdot \vec{y}_i = 0$).

We use these properties to refine Equation 10 into Equation (2) of Lemma 8:

$$\vec{x}^T \cdot M^n \cdot \vec{y} = \sum_{i=1}^k \vec{x}_i^T \cdot B_i^n \cdot \vec{y}_i \stackrel{11.1-11.2}{=} d_j^n \cdot \vec{x}_j^T \cdot \vec{y}_j \stackrel{11.2}{=} \sum_{i=1}^k d_j^n \cdot \vec{x}_i^T \cdot \vec{y}_i = d_j^n \cdot \vec{x}^T \cdot \vec{y} \text{ for all } n \in \mathbb{N}. \quad (11)$$

D Proofs for Section 5 (Depumpable automata)

► **Lemma 21.** *Let \mathcal{A} be a polynomially-ambiguous weighted automaton. For every $u \in \Sigma^*$, if $M(u)$ has an idempotent structure then it is p -triangular.*

Proof. Let \mathcal{A} be a polynomially-ambiguous automaton, let m denote the size of \mathcal{A} and let $u \in \Sigma^*$ such that $M(u)$ has an idempotent structure. We construct a permutation $\sigma : \{1, 2, \dots, m\} \rightarrow \{1, 2, \dots, m\}$ such that $P_\sigma^{-1} \cdot M(u) \cdot P_\sigma$ is an upper triangular matrix, where P_σ is defined as

$$(P_\sigma)_{ij} = \begin{cases} 1 & \text{if } j = \sigma(i); \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

The idea behind the construction of σ is the following: we show that, since $M(u)$ has an idempotent structure and \mathcal{A} is polynomially-ambiguous, the non-zero entries of $M(u)$ define a partial order on the set of indices $\{1, 2, \dots, m\}$. We then show that any permutation σ which sorts these indices from largest to smallest fits the desired requirements.

Formally, let us consider the binary relation \leq_u over $\{1, 2, \dots, m\}^2$ defined as

$$i \leq_u j \text{ if } i = j \text{ or } (M(u))_{ij} > 0. \quad (13)$$

We show that this relation is a partial order:

- Reflexivity: this follows immediately from the definition;
- Transitivity: If $i \leq_u j$ and $j \leq_u k$ with $i \neq j \neq k$, then we get $(M(uu))_{ik} > (M(u))_{ij} \cdot (M(u))_{jk} > 0$ as the entries of M are non-negative. Since $M(u)$ has an idempotent structure, this implies that $(M(u))_{ik} > 0$, hence we have $i \leq_u k$;
- Antisymmetry: suppose, towards building a contradiction, that there exist $i \neq j$ satisfying $i \leq_u j$ and $j \leq_u i$. This implies that $(M(uu))_{ii} > (M(u))_{ij} \cdot (M(u))_{ji} > 0$ hence, as M has an idempotent structure, $(M(u))_{ii} > 0$. Therefore, the automaton \mathcal{A} has two distinct cycles on i labelled by uv : one that loops twice on i (witnessed by $(M(u))_{ii} > 0$), and one that goes to j while reading the first copy of v ($(M(u))_{ij} > 0$), and back to i while reading the second one ($(M(u))_{ji} > 0$). This contradicts the fact that \mathcal{A} is polynomially-ambiguous: Weber and Seidl's [30] showed that an automaton is not polynomially-ambiguous if and only if there exists a state q and a word w such that there are at least two different cycles on q labelled by w .

Let σ be a permutation sorting $\{1, 2, \dots, n\}$ from largest to smallest according to \leq_u :

$$i \leq_u j \text{ implies } \sigma(i) \geq \sigma(j) \text{ for every } 1 \leq i, j \leq m. \quad (14)$$

Finally, let P_σ be the corresponding permutation matrix (as described by Equation (12)), and let $M' = P_\sigma^{-1} \cdot M(u) \cdot P_\sigma$. We conclude by showing that M' is an upper triangular matrix: For every $1 \leq i < j \leq m$ we have that $\sigma^{-1}(i) \not\leq_u \sigma^{-1}(j)$ (by the contrapositive of Equation (14)), therefore $(M(u))_{\sigma^{-1}(i)\sigma^{-1}(j)} = 0$ by Equation (13), and in turn:

$$(M')_{ij} = (P_\sigma^{-1} \cdot M(u) \cdot P_\sigma)_{ij} = (M(u))_{\sigma^{-1}(i)\sigma^{-1}(j)} = 0. \quad \blacktriangleleft$$

E Proofs for Section 6 (Deciding pumpability)

► **Lemma 23.** *A weighted automaton \mathcal{A} is weakly pumpable if and only if \mathcal{A} is pumpable.*

Proof. It is immediate that a pumpable automaton is weakly pumpable. We suppose that \mathcal{A} is a weakly pumpable automaton and show \mathcal{A} is pumpable. Consider any triple u, v, w , with $M(v)$ p-triangular, then for every $n \in \mathbb{N}$ we have $\mathcal{A}(uv^{m+n+1}w) = d \cdot \mathcal{A}(uv^{m+n}w)$ for some $d \in \{M(v)_{11}, \dots, M(v)_{mm}\}$. To show that \mathcal{A} is pumpable, we show that the same d is chosen for every $n \in \mathbb{N}$.

Consider the unary weighted automaton \mathcal{U} such that $\mathcal{U}(a^n) = \mathcal{A}(uv^n w)$ for every $n \in \mathbb{N}$. \mathcal{U} is constructed with alphabet $\{a\}$, over the same states as \mathcal{A} with initial vector $IM(u)$, final vector $M(w)F$ and $M_{\mathcal{U}}(a) = M(v)$.

Note that, by weak pumpability of \mathcal{A} , the set of prime divisors of $\{\mathcal{A}(uv^n w) \mid n \in \mathbb{N}\}$ is finite. Thus \mathcal{U} is unambiguisable by the characterisation of Theorem 15.

Therefore by Proposition 10, we have that \mathcal{U} is pumpable. In particular, consider the triple (ϵ, a, ϵ) : we have that $M(a)$ is p-triangular, so there is some d in the diagonal of $M(a)$ such that

$$\mathcal{U}(a^{m+n}) = d^n \mathcal{U}(a^m) \text{ for all } n \in \mathbb{N}$$

and by definition of \mathcal{U} , this entails that

$$\mathcal{A}(uv^{m+n}w) = d^n \mathcal{A}(uv^m w) \text{ for all } n \in \mathbb{N}.$$

Repeating for any u, v, w we have \mathcal{A} is pumpable. ◀

► **Lemma 24.** *Let $\mathcal{A} = (Q, \Sigma, M, I, F)$ be a weighted automaton of size $m = |Q|$ over Σ , and let $\$ \notin \Sigma$.*

1. There exists an automaton \mathcal{T} of size $2^{m^2} + 3$ and norm 1 satisfying

$$\begin{aligned} \mathcal{T}(u\$v\$w) &= 1 && \text{for all } u, v, w \in \Sigma^* \text{ such that } M(v) \text{ is p-triangular;} \\ \mathcal{T}(u) &= 0 && \text{for all other } u \in (\Sigma \cup \{\$\})^*. \end{aligned}$$

2. For all $n \in \mathbb{N}$ there exist automata \mathcal{B}_n of size $m^{2n} + 2m$ and norm $\|\mathcal{A}\|^n$ satisfying

$$\mathcal{B}_n(u\$v\$w) = \mathcal{A}(uv^n w); \quad \text{for all } u, v, w \in \Sigma^*.$$

3. For all $1 \leq i \leq m$ there exists an automaton \mathcal{C}_i of size $m + 2$ and norm $\|\mathcal{A}\|$ satisfying

$$\mathcal{C}_i(u\$v\$w) = (M(v))_{ii} \quad \text{for all } u, v, w \in \Sigma^*.$$

Proof of Lemma 24.

1. Let \mathcal{T} be the automaton with states $\{q_0, q_a\} \cup \{0, 1\}^{Q \times Q}$, that is, other than q_0, q_r and q_a , every state is a zero-one matrix indexed by Q . Recall the notation \overline{M} , where $\overline{M}_{i,j} = 1$ if $M_{i,j} \neq 0$ and 0 otherwise. Let the initial vector $I_{\mathcal{T}}$ be the zero vector, except that $I_{\mathcal{T}}(q_0) = 1$ and let the final vector be $F_{\mathcal{T}}$ be the zero vector, except that $F_{\mathcal{T}}(q_a) = 1$. Let \mathcal{T} have the following transitions:

- $q_0 \xrightarrow{a:1} q_0$, and $q_a \xrightarrow{a:1} q_a$ for each $a \in \Sigma$;
- $q_0 \xrightarrow{\$: 1} \text{Id}$, where Id is the identity matrix over Q ;
- For every $M \in \{0, 1\}^{Q \times Q}$, $a \in \Sigma$, let $N = \overline{M}M(a)$, we have $M \xrightarrow{a:1} \overline{N}$;
- If M is p-triangular we have $M \xrightarrow{\$: 1} q_a$;

Note that the automaton is deterministic such that $q_0 \xrightarrow{u\$v\$w:1} q_a$ if $M(v)$ is p-triangular, in which case the weight of the run is 1 as $F(q_a) = 1$, and otherwise no run reaches the final state. Observe $|\mathcal{T}| = 2^{|Q| \times |Q|} + 2$ and $\|\mathcal{T}\| = 1$.

2. \mathcal{B}_n will operate in three components separated by reading $\$$. The first and third components, reading u and w of $u\$v\w respectively are direct copies of \mathcal{A} . The second component, reading v but behaving like v^n , being the most interesting. For every state $q \in Q$, we let the copy q^1 of q represents the state in the first component and a copy q^3 represents the state in the third component. The transitions in the first and third components are as in \mathcal{A} .

The second component simulates n runs simultaneously using the state space $Q^{n-1} \times Q^n$. A state $(g_1, \dots, g_{n-1}, q_1, \dots, q_n)$, with $g_i, q_i \in Q$, denotes that the state of the i th run is q_i , and the guess that the i th run will terminate in state g_i at the end of v , and thus the $i + 1$ st run will start in g_i . Only runs with the correct guess will proceed to the third component.

Between the first and middle component we have:

$$q^1 \xrightarrow{\$: 1} (g_1, \dots, g_{n-1}, q, g_1, \dots, g_{n-1}) \text{ for all } g_1 \dots g_{n-1} \in Q \text{ and } q \in Q.$$

Within the middle component we have:

$$(g_1, \dots, g_{n-1}, q_1, \dots, q_n) \xrightarrow{a:x_1 \dots x_n} (g_1, \dots, g_{n-1}, q'_1, \dots, q'_n) \text{ for all } q_i \xrightarrow{a:x_i} q'_i \in \mathcal{A}.$$

From the middle component to the third we have:

$$(g_1, \dots, g_{n-1}, q_1, \dots, q_n) \xrightarrow{\$: 1} q_n \text{ if } g_i = q_i \text{ for all } 1 \leq i \leq n-1.$$

Let the initial and final vectors, $I_{\mathcal{B}_n}$ and $F_{\mathcal{B}_n}$, be the zero vectors, except $I_{\mathcal{B}_n}(q^1) = I(q)$ for every $q \in Q$ and $F_{\mathcal{B}_n}(q^3) = F(q)$ for every $q \in Q$.

Consider a word $uv^n w$, where $u = u_1 \dots u_{|u|}$, $v = v_1 \dots v_{|v|}$ and $w = w_1 \dots w_{|w|}$. We observe, there is a bijection between runs in \mathcal{A} on $uv^n w$ and runs in \mathcal{B} on $u\$v\w .

The bijection is as follows: Let the following be a run in \mathcal{A} on $uv^n w$:

- On u , the subrun $q_{u,1} \rightarrow \dots \rightarrow q_{u,|u|} \rightarrow q_{u,|u|+1}$ and having weights $I(q_{u,1}), x_{u,1}, \dots, x_{u,|u|}$,
- On v^n the subrun $q_{v,1,1} \rightarrow \dots q_{v,1,|v|+1} = q_{v,2,1} \rightarrow \dots \rightarrow q_{v,n,|v|+1}$, where $q_{u,|u|} = q_{v,1,1}$, and having weights $x_{v,1,1}, \dots, x_{v,1,|v|}, x_{v,2,1}, \dots, x_{v,n,|v|}$,
- On w the subrun $q_{w,1} \rightarrow \dots \rightarrow q_{w,|w|+1}$, where $q_{v,n,|v|+1} = q_{w,1}$. and having weights $x_{w,1}, \dots, x_{w,|w|+1}, F(q_{w,|w|+1})$.

This run is in bijection with the following run on $u\$v\w in \mathcal{B}_n .

- On u the subrun $q_{u,1}^1 \rightarrow \dots \rightarrow q_{u,|u|}^1 \rightarrow q_{u,|u|+1}^1$ has weights $I(q_{u,1}^1), x_{u,1}, \dots, x_{u,|u|}$.
- Which moves, with weight 1 on $\$$ to $(q_{v,2,1}, \dots, q_{v,n,1}, q_{v,1,1}, \dots, q_{v,n,1})$
- On v the subrun: $(q_{v,2,1}, \dots, q_{v,n,1}, q_{v,1,1}, \dots, q_{v,n,1}) \rightarrow (q_{v,2,1}, \dots, q_{v,n,1}, q_{v,1,2}, \dots, q_{v,n,2}) \rightarrow \dots \rightarrow (q_{v,2,1}, \dots, q_{v,n,1}, q_{v,1,|v|+1}, \dots, q_{v,n,|v|+1})$ has weights $(x_{v,1,1} \dots x_{v,n,1}), \dots, (x_{v,1,|v|} \dots x_{v,n,|v|})$, and
- Which moves, with weight 1 on $\$$ to $q_{w,1}^3 = q_{v,n,|v|+1}$, and
- On w the subrun $q_{w,1}^3 \rightarrow \dots \rightarrow q_{w,|w|}^4 \rightarrow q_{w,|w|+1}^1$, with weights $x_{w,1}, \dots, x_{w,|w|}, F(q_{w,|w|+1}^3)$.

Finally, we observe the weights have the same value:

$$I(q_{u,1}) \cdot x_{u,1} \dots x_{u,|u|} \cdot x_{v,1,1} \dots x_{v,1,|v|} \cdot x_{v,2,1} \dots x_{v,n,|v|} \cdot x_{w,1} \dots x_{w,|w|+1} \cdot F(q_{w,|w|+1}) = \\ I(q_{u,1}^1) \cdot x_{u,1} \dots x_{u,|u|} \cdot 1 \cdot (x_{v,1,1} \dots x_{v,n,1}) \cdot \dots \cdot (x_{v,1,|v|} \dots x_{v,n,|v|}) \cdot 1 \cdot x_{w,1} \dots x_{w,|w|+1} \cdot F(q_{w,|w|+1}^3).$$

3. \mathcal{C}_i is a copy of \mathcal{A} with two additional states q_0, q_f , in which $I_{\mathcal{C}_i}(q_0) = F_{\mathcal{C}_i}(q_f) = 1$ (and all other entries of $I_{\mathcal{C}_i}, F_{\mathcal{C}_i}$ are zero), and

$$q_0 \xrightarrow{\$: 1} i, \quad i \xrightarrow{\$: 1} q_f, \text{ and}$$

$$q_0 \xrightarrow{a : 1} q_0, \quad q_f \xrightarrow{a : 1} q_f \text{ for all } a \in \Sigma.$$

◀