Determinisation and Unambiguisation of Polynomially-Ambiguous Rational Weighted Automata

Ismaël Jecker[*†]
University of Franche-Comté
Besançon, France

Filip Mazowiecki[*‡]
f.mazowiecki@mimuw.edu.pl
University of Warsaw
Warsaw, Poland

David Purser[*]
D.Purser@liverpool.ac.uk
Department of Computer Science
University of Liverpool
Liverpool, UK

ABSTRACT

We study the determinisation and unambiguisation problems of weighted automata over the field of rationals: 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.

CCS CONCEPTS

• **Theory of computation → Formal languages and automata theory**; **Models of computation**.

KEYWORDS

Weighted automata, determinisation, unambiguisation

Full proofs can be found in the arXiv version [18].

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* [26], is when the words are assigned their probability of being accepted (this is a special case of the field of rationals).

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 (see the survey [24]). 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 field of rationals, 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 [22, 24].

Classes with bounded ambiguity can be defined for any automata model (not only weighted automata). The simplest and most studied 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 a mathematically 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 [21, 25]. In this case automata that can be determinised are characterised by forbidding a simple pattern in the automaton, called the *twins property*, which can be detected in polynomial time. Due to these results, papers often focus on the *unambiguisation problem*, i.e. whether there exists an equivalent unambiguous automaton. The mentioned work deals with the tropical semiring and its variants. A similar characterisation also works over the field of rationals, which we describe in **??**. 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 [33]. The decidability results of determinisation over the tropical semiring are for precisely these two classes [22, 24]. In general classes of automata with bounded ambiguity are well-known restrictions studied also in other areas [1, 9, 27].

We focus on the field of rationals, 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 [28]. 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

[*]Supported by the ERC grant INFSYS, agreement no. 950398.
[†]Supported by the EIPHI Graduate School, contract ANR-17-EURE-0002
[‡]Supported by Polish National Science Centre SONATA BIS-12 grant number 2022/46/E/ST6/00230.

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

fact that it is computable essentially follows from computability of the Zariski closure (not linear) [16]. In a recent paper Bell and Smertnig [3] focus on directly computing 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 field of rationals. Our result is the following

Theorem 1.1. *Unambiguisability and determinisability 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 recurrence sequences (see [1] or [23]). 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. 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 the deterministic automaton would be of nonelementary size. We leave open whether the equivalent deterministic automaton needs to be this large, and whether the equivalent unambiguous automaton can be constructed using our proof techniques.

Related work. For the max-plus semiring decidability of the unambiguisation and determinisation problems remain open. However, Kirsten and Lobardy [20] show that both are decidable for polynomially-ambiguous automata. The seminal result for weighted automata over the field of rationals is due to Schützenberger [30], who proved that equivalence is decidable. The problem is known to be even in NC^2 [19, 32] (thus in particular in polynomial time). Other problems like containment or emptiness are undecidable already for probabilistic automata [26]. 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, and the full class of weighted automata [4, 7, 10, 12, 14]. 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 field of rationals [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, $M : \Sigma \to \mathbb{Q}^{Q\times Q}$ 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 $M(\varepsilon)$ 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 2.1. We say a matrix M is *p-triangular* if there exists a permutation matrix P such that PMP^{-1} is upper triangular.

Remark 2.2. 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 $\mathcal{A}$ over $w = a_1 \dots a_n$ 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. A state q is reachable if there exists an initial state p such that there is a path from p to q. A state q is coreachable if there exists a finial state r 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 of $\mathcal{A}$ over the word w.

Remark 2.3. The existence of a run from p to q 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.

Given $x \in \mathbb{Q}$, let $\overline{x} = 1$ if $x \neq 0$ and $\overline{x} = 0$ 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 $\overline{F}_q = \overline{F_q}$. Moreover, for every word $w = a_1 \dots a_n$ we let $\overline{M}(a_1 a_2 \dots a_n) = \overline{M}(a_1) \otimes \overline{M}(a_2) \otimes \dots \otimes \overline{M}(a_n)$, where $\otimes$ is matrix multiplication over the Boolean semiring (i.e., $1 + 1 = 1$). 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 $\otimes$ 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: $\overline{M} = \overline{M} \otimes \overline{M}$.

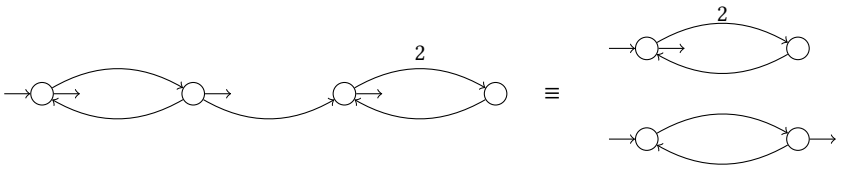

Figure 1: Example of a unary weighted automaton. The input label of all edge is the letter a (omitted on the picture), and unlabelled edges are assumed to have weight 1. If n is even then $\mathcal{A}(a^n) = 1 + \sum_{i=0}^{n/2-1} 2^i = 2^{n/2}$ and if n is odd then $\mathcal{A}(a^n) = 1$. The automaton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

2.3 Determinisim, ambiguity and decision problems

Definition 2.4. 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 $\mathfrak{p}$ such that $\#runs_{\mathcal{A}}(w) \leq \mathfrak{p}(|w|)$ for every word $w \in \Sigma^*$, and
- *exponentially ambiguous* otherwise, in particular, $\#runs_{\mathcal{A}}(w) \leq |Q|^{|w|+1}$ for every $w \in \Sigma^*$.

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 2.5. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be weighted automata over Σ.*

- *The function $-\mathcal{A}_1 : u \mapsto -\mathcal{A}_1(u)$ is recognised by an automaton of size $|\mathcal{A}_1|$ and norm $||\mathcal{A}_1||$;*
- *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 the weight of a given transition in $\mathcal{A}_1$ (respectively $\mathcal{A}_2$) in space $O(\log(||\mathcal{A}_1||))$ (respectively $O(\log(||\mathcal{A}_2||))$) then we can compute the weight of a given transition in $\mathcal{A}_1 - \mathcal{A}_2$ or $\mathcal{A}_1 \cdot \mathcal{A}_2$ in space $O(\log(||\mathcal{A}_1|| \cdot ||\mathcal{A}_2||))$.

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. Below is the formal construction in which the negative values only appear in the final vector, a similar construction allows them to appear only in the initial vector.

Given $\mathcal{A} = (Q, \Sigma, M, I, F)$, we construct an equivalent weighted automaton $\mathcal{A}' = (Q', \Sigma, M', I', F')$. 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 2.6. *$\mathcal{A}'(w) = \mathcal{A}(w)$ 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 2.7. 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: $\overline{M}(uv) = \overline{M}(u) \otimes \overline{M}(v)$ 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, $M : \Sigma \to \mathbb{Z}^{Q \times Q}$, $I, F \in \mathbb{Z}^Q$.

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). To this end, we rely on the following construction: Given an automaton $\mathcal{B} = (Q, \Sigma, M, I, F)$ and $t \in \mathbb{Q}$, let $\mathcal{B}_t = (Q, \Sigma, M', I', F)$ be the automaton constructed from $\mathcal{B}$ by multiplying all the weights by t, that is, $I'(q) = tI(q)$, $F'(q) = tF(q)$, and $M'(a)_{p,q} = tM(a)_{p,q}$ for all $a \in \Sigma$ and $p, q \in Q$.

We let $\mathcal{A}' = \mathcal{A}_x$, where $x \in \mathbb{Q}$ is defined as follows: Let X be the set of denominators of transition weights, initial weights and final weights of $\mathcal{A}$, and let $x = \mathrm{lcm}(X)$. Note that $|x| \leq ||\mathcal{A}||^{|X|}$, thus $\log(x) \leq |X| \log(||\mathcal{A}||)$, and observe that $|X|$ and $\log(||\mathcal{A}||)$ are both polynomial in the size of the representation of $\mathcal{A}$. As required, all the weights of $\mathcal{A}'$ are in $\mathbb{Z}$. Moreover, for every unambiguous automaton $\mathcal{B}$ equivalent to $\mathcal{A}$ the automaton $\mathcal{B}_x$ is unambiguous and equivalent to $\mathcal{A}'$, and for every unambiguous automaton $\mathcal{B}'$ equivalent to $\mathcal{A}'$ the automaton $\mathcal{B}'_{1/x}$ is unambiguous and equivalent to $\mathcal{A}$. The same holds when unambiguous is replaced by deterministic.

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 3.1. *Let M be an $m \times m$ matrix with the set of eigenvalues $\{d_1, d_2, \ldots, d_k\} \subseteq \mathbb{N}$, and let $\vec{x}, \vec{y} \in \mathbb{Q}^m$. There exist k polynomials $p_1, p_2, \ldots, p_k$ such that*

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

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

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

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 Equation (1) can be derived from standard results on linear recurrence sequences (see e.g. [15, Section 2]). We complete the proof in Section 3.1 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 in the later proofs. Overall, Lemma 3.1 has two purposes:

(1) 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 unambiguisability (resp. determinisability) (see Proposition 3.3).
(2) Equation (2) allows us to establish that pumpability (resp. blind pumpability) is sufficient to ensure unambiguisability (resp. determinisability) for polynomially-ambiguous WA (see Proposition 3.4).

Remark that the eigenvalues of a matrix over $\mathbb{N}$ do not have to be rational numbers, and might be complex numbers, which make it more difficult to distinguish good behaviours from bad. In further definitions we often restrict ourselves to p-triangular matrices to avoid dealing with complex numbers: As stated in Remark 2.2 the eigenvalues of such a matrix are its diagonal entries.

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 $\{d_1, d_2, \ldots, d_n\}$ is the set of diagonal entries of $M(v)$, and $p_1, p_2, \ldots, p_n$ 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 3.2. 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 3.3, proved in Section 4.

Proposition 3.3. *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 3.3 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 (Lemma 5.8), and the presence of such matrices is guaranteed by Ramsey's theorem (Lemma 5.7). On top of this, we remark that Equation (2) in Lemma 3.1 would counteract the second limitation, but the prescribed behaviour starts only from $m = 0$ only applies to 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 3.3. We obtain the following result, proved in Section 5:

Proposition 3.4. *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 3.3 and 3.4, 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 3.5. *We can decide in polynomial space whether a given weighted automaton is pumpable, respectively blindly pumpable.*

Together, Propositions 3.3 to 3.5 entail our main theorem.

Theorem 1.1. *Unambiguisability and determinisability are decidable in polynomial space for polynomially-ambiguous weighted automata.*

We now prove Lemma 3.1 in the next subsection and prove Propositions 3.3 to 3.5 in the next sections.

3.1 Proof of Lemma 3.1

The proof of Lemma 3.1 relies on two lemmas. Lemma 3.6 shows that every 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 [31, Appendix B].

Lemma 3.6. *Let M be a matrix with eigenvalues $\{d_1, d_2, \dots, d_k\} \subseteq \mathbb{N}$. There exists an invertible matrix P such that $P^{-1}MP = B_1 \oplus B_2 \oplus \cdots \oplus B_k$ 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 3.7 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 3.7. *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 3.7. 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 $d \cdot \mathsf{Id}$ obtained by multiplying the identity by d, and the remainder $B_0 = B - d \cdot \mathsf{Id}$. This decomposition has the following properties:

- The two components commute multiplicatively: $(d \cdot \mathsf{Id}) \cdot B_0 = dB_0 = B_0 \cdot (d \cdot \mathsf{Id})$;
- 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:

$$B^n = \sum_{i=0}^{n} \binom{n}{i} (d \cdot \mathsf{Id})^{n-i} \cdot B_0^i = \sum_{i=0}^{m'} \binom{n}{i} d^{n-i} B_0^i = d^n \cdot \sum_{i=0}^{m'} \binom{n}{i} \cdot \frac{B_0^i}{d^i}$$

with $m' = \min(n, m)$.

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

$$\vec{x}^T \cdot B^n \cdot \vec{y} = d^n \cdot \underbrace{\sum_{i=0}^{m'} \binom{n}{i} \cdot \frac{\vec{x}^T \cdot B_0^i \cdot \vec{y}}{d^i}}_{p(n)} \text{ with } m' = \min(n, m). \quad (3)$$

As indicated by the underbrace, we let p denote the sum in Equation (3) 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

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

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 (3) yields the desired expression:

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

Proof of Lemma 3.1. Let M be an $m \times m$ matrix with eigenvalues $\{d_1, d_2, \ldots, d_n\} \subseteq \mathbb{N}$. We begin by transforming M into a block diagonal matrix according to Lemma 3.6: $P^{-1}MP = B_1 \oplus B_2 \oplus \cdots \oplus B_k$. Since $M^n = P \cdot (B_1^n \oplus B_2^n \oplus \cdots \oplus B_k^n) \cdot P^{-1}$, we get

$$\vec{x}^T \cdot M^n \cdot \vec{y} \underset{4.1}{=} \sum_{i=1}^{k} \vec{x}_i^T \cdot B_i^n \cdot \vec{y}_i \underset{4.2}{=} \sum_{i=1}^{k} d_i^n \cdot p_i(n) \text{ for all } n \geq m, \text{ where:} \tag{4}$$

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

4.2 The polynomials $p_1, p_2, \ldots, p_n$ are obtained by applying Lemma 3.7 to each block.

This concludes the proof of Equation (1) of Lemma 3.1.

In order to prove the second part, we now suppose that the matrix M is invertible. This implies that none of its eigenvalues equals zero. Therefore, if on top of that we suppose that $p_1, p_2, \ldots, p_k$ are all constant polynomials, and that only p_j is not constantly 0, we can apply the second part of Lemma 3.7 to get:

5.1 $\vec{x}_j^T \cdot B_j^n \cdot \vec{y}_j = d_j^n \vec{x}_j^T \cdot \vec{y}_j$ for all $n \in \mathbb{N}$;

5.2 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 (4) into Equation (2) of Lemma 3.1:

$$\begin{aligned} \vec{x}^T \cdot M^n \cdot \vec{y} = \sum_{i=1}^{k} \vec{x}_i^T \cdot B_i^n \cdot \vec{y}_i &\underset{5.1-5.2}{=} d_j^n \cdot \vec{x}_j^T \cdot \vec{y}_j \\ &\underset{5.2}{=} \sum_{i=1}^{k} d_j^n \cdot \vec{x}_i^T \cdot \vec{y}_i \\ &= d_j^n \cdot \vec{x}^T \cdot \vec{y} \text{ for all } n \in \mathbb{N}. \end{aligned} \tag{5}$$

□

4 PUMPABLE AUTOMATA

The goal of this section is to prove Proposition 3.3:

Proposition 3.3. *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 4.1). Then, to prove Proposition 3.3, 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 3.1) and remove all the behaviours that do not fit an unambiguous (resp. deterministic) weighted automaton (which are described by Lemma 4.1), then we are left with exactly the behaviours fitting a pumpable (respectively blindly pumpable) automaton.

Lemma 4.1. *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 run by the weight of the cycle. 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 starting in an initial state, then $\mathcal{A}(uv^m w) = 0$ for every $w \in \Sigma^*$, and we are done. If there exists $m \geq |\mathcal{A}|$ such that $\mathcal{A}$ has a path over uv^m starting from the initial state with some weight x, then this path will visit a cycle while reading a non-empty power v^λ of v. Therefore, for every $n \in \mathbb{N}$ there exists a path in $\mathcal{A}$ over $uv^{m+\lambda n}$ that starts in an initial state and has weight $d^n \cdot x$. 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 3.3.
Fix a weighted automaton $\mathcal{A} = (Q, \Sigma, M, I, F)$.

Unambiguisability implies pumpability. Let us suppose that $\mathcal{A}$ is unambiguisable, 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 3.1:

$$\mathcal{A}(uv^n w) = \sum_{i=1}^{k} d_i^n \cdot p_i(n) \text{ for every } n \geq |\mathcal{A}|, \tag{6}$$

where $\{d_1, d_2, \ldots, d_n\}$ is the set of diagonal entries of $M(v)$ and $p_1, p_2, \ldots, p_n$ are polynomials. The behaviour of $\mathcal{U}$ is described by Lemma 4.1: 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 (6) 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 $d_i^\lambda \neq d$ 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 $d_i^\lambda = d$ 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 $d_i = 0$ (and $\mathcal{U}(uv^m w) = 0$), or $p_i(n)$ is constantly $\frac{\mathcal{U}(uv^m w)}{d_i^m}$.

We update Equation (6) 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: $\mathcal{A}(uv^{|\mathcal{A}|+n'} w) = d_j^{n'} \cdot \mathcal{A}(uv^{|\mathcal{A}|} w)$ for every $n' \geq 0$. Since this holds for every triple $u, v, w \in \Sigma^*$ required in Definition 3.2 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 4.1: 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 3.4, which we recall:

Proposition 3.4. *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 5.1 (Depumping lemma).
Let $\mathcal{A}$ be a polynomially-ambiguous automaton. There exists a constant $R_\mathcal{A}$ such that every word $u \in \Sigma^*$ satisfying $|u| = R_\mathcal{A}$ can be decomposed into three parts $u = u_1u_2u_3$ such that $u_2 \neq \varepsilon$ 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 $\mathcal{A}(uv) = d \cdot \mathcal{A}(u_1u_3v)$;*
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 $\mathcal{A}(uv) = d \cdot \mathcal{A}(u_1u_3v)$.*

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 5.1 is presented in Subsection 5.2.

Remark. The following proof 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 3.4 in the same manner.

Proof of Proposition 3.4. Let $\mathcal{A}$ be a polynomially-ambiguous automaton. We prove both parts of Proposition 3.4 independently.

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

Theorem 5.2 ([2]). *A weighted automaton over integers $\mathcal{A}$ is unambiguisable 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 image of $\mathcal{A}$ (the set of output values of $\mathcal{A}$ over all words) 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 $u \in \Sigma^*$ of length at most $R_\mathcal{A}$ 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 $|u| \leq R_\mathcal{A}$, then by definition of N we get that $A(u)$ is smaller than N, thus its prime factors are also smaller than N.

Now let us suppose that $|u| > R_\mathcal{A}$ and that for every word u' shorter than u the prime factors of $\mathcal{A}(u')$ are smaller than N. By applying Lemma 5.1 to the prefix of size $R_\mathcal{A}$ of u we obtain a decomposition $u = u_1u_2u_3v$ satisfying $0 < |u_2| \leq R_\mathcal{A}$ and $\mathcal{A}(u) = d \cdot \mathcal{A}(u_1u_3v)$ for some entry d of $M(u_2)$. We conclude by remarking that $d < N$ since $|u_2| \leq R_\mathcal{A}$, and the prime factors of $\mathcal{A}(u_1u_3v)$ 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 5.1 there exists a decomposition $u = u_1u_2u_3$ and an entry d of $M(u)$ such that $\mathcal{A}(uw) = d \cdot \mathcal{A}(u_1u_3w)$ for every suffix $w \in \Sigma^*$. Therefore, $\mathcal{D}$ can immediately produce the weight d corresponding to the infix u_2, and transition towards the state u_1u_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 $\text{cut} : \Sigma^{R_{\mathcal{A}}} \to \Sigma^* \times \mathbb{Z}$ 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

- $u = u_1u_2u_3$ for some $u_1, u_2, u_3 \in \Sigma^*$ and $u' = u_1u_3$;
- $|u'| < R_{\mathcal{A}}$;
- d is an entry of $M(u_2)$;
- $\mathcal{A}(uv) = d \cdot \mathcal{A}(u_1u_3v)$ for every $v \in \Sigma^*$.

Note that this function exists: Lemma 5.1 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 $\mathcal{D}$:

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

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 5.3. *The weight x of the (single) path $q_\varepsilon, q_{u_1}, q_{u_2}, \ldots, q_{u_n}$ of $\mathcal{D}$ over u starting from the initial state q_ε satisfies $\mathcal{A}(uw) = x \cdot \mathcal{A}(u_nw)$ 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 $|u| < R_{\mathcal{A}}$ 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 $|u| \geq R_{\mathcal{A}}$. 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 $\mathcal{A}(uw) = \mathcal{A}(vaw) = \frac{x}{d} \cdot \mathcal{A}(u_{n-1}aw)$. 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
$$\mathcal{A}(u) = \frac{x}{d} \cdot \mathcal{A}(u_{n-1}aw) = x \cdot \mathcal{A}(u_nw);$$
- If $|u_{n-1}a| = R_{\mathcal{A}}$, then $\text{cut}(u_{n-1}a) = (u_n, d)$, hence
$$\mathcal{A}(u) = \frac{x}{d} \cdot \mathcal{A}(u_{n-1}aw) = x \cdot \mathcal{A}(u_nw).$$

The last equality follows from the definition of cut. □

Remark 5.4. 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 5.1 we will need a technical lemma that allows us to find pumpable fragments and replace them with invertible matrices. Some ideas here are similar to those in [6], in particular the idea behind Claim 5.6 is similar to [6, Lemma 8].

Let us consider the family of functions $(\text{tower}_r)_{r \in \mathbb{N}}$ 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 5.5. *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 $u = u_\vdash u_1u_2 \ldots u_\ell u_\dashv$ 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 $n \geq 0$:*

$$\begin{aligned} &M(u_\vdash u_1u_2 \ldots u_{i-1}(u_i \ldots u_j)^n u_{j+1} \ldots u_\ell u_\dashv)\\ &\qquad = M(u_\vdash u_1u_2 \ldots u_{i-1}) \cdot M^n \cdot M(u_{j+1} \ldots u_\ell u_\dashv). \end{aligned} \quad (7)$$

The proof will rely on the following claim.

Claim 5.6. *If the rank of ABA is equal to the rank of A, there exists an invertible matrix C satisfying $(AB)^nA = C^nA$ for all $n \geq 0$.*

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

Recall that a linear transformation $\mathbb{Q}^m \to \mathbb{Q}^m$ is uniquely defined by fixing the images of a basis. Let $\vec{w}_1, \ldots, \vec{w}_s$ be the images of $\vec{v}_1, \ldots, \vec{v}_s$ when applying the linear transformation AB, 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. Also by definition $(AB)A = CA$. The general property for all powers is proved by induction:

$$(AB)^nA = (AB)^{n-1}ABA = C^{n-1}ABA = C^nA.$$ □

Proof of Lemma 5.5. We prove the lemma by induction on r, the maximal rank of the matrices $\{M(a) \mid a \in \Sigma\}$.

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 $u = a_1a_2 \ldots a_n$ of size $n \geq \text{tower}_0(2\ell \cdot |\Sigma|) = 2\ell \cdot |\Sigma| > \ell \cdot |\Sigma|$, the left-hand side of Equation (7) is equal to the null matrix for every choice of i, j and n. Therefore, we satisfy the statement by setting $u_\vdash = \varepsilon$, $u_i = a_i$ for every $1 \leq i \leq \ell \cdot |\Sigma|$ and

$u_\dashv = a_{\ell\cdot|\Sigma|+1} \dots a_n$. Note that the lemma holds for any choice of an invertible matrix M.

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 $1 \le i \le \text{tower}_{r-1}(x^x)$. We distinguish two cases:

- Suppose that there exists $1 \le i \le \text{tower}_{r-1}(x^x)$ such that the rank of $M(v_i)$ is r. Since $|v_i| = x > \ell \cdot |\Sigma|$, there is one letter $a \in \Sigma$ that occurs $\ell+1$ times in v_i:

  $v_i = w_0 a w_1 a w_2 \dots a w_\ell a w_{\ell+1}$ for some $w_0, w_1, \dots, w_{\ell+1} \in \Sigma^*$.

  Moreover, since the rank of both $M(v_i)$ and $M(a)$ is r, for every $1 \le i \le j \le \ell$ the rank of $M(a)M(w_i \dots a w_j)M(a)$ is also r. As a consequence, Claim 5.6 guarantees the existence of an invertible matrix C satifying for all $n \ge 0$:

  $$M((a w_i \dots a w_j)^n a) = M(a w_i \dots a w_j)^n \cdot M(a) = C^n \cdot M(a).$$

  Therefore, the decomposition $u = u_\vdash u_1 u_2 \dots u_\ell u_\dashv$ where $u_i = a w_i$ for every $1 \le i \le \ell$ satisfies the statement of the lemma.
- Suppose that for every $1 \le i \le \text{tower}_{r-1}(x^x)$ the matrix $M(v_i)$ has a rank smaller than r. In order to apply the induction hypothesis, we now consider the alphabet $\Gamma = \{v_i \mid 1 \le i \le \text{tower}_{r-1}(x^x)\}$. 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 $|\Gamma| \le |\Sigma|^x$. Therefore, the word $w = v_1 v_2 \dots v_{\text{tower}_{r-1}(x^x)} \in \Gamma^*$ satisfies

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

  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 5.1)

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 **??**.

Lemma 5.7. *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 $1 \le i \le \ell$. There exist $1 \le i \le j \le \ell$ such that $M(u_i \dots u_j)$ has idempotent structure.*

Lemma 5.8. *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 $\sigma : \{1, 2, \dots, m\} \to \{1, 2, \dots, m\}$ such that $P_\sigma^{-1} \cdot M(u) \cdot P_\sigma$ is an upper triangular matrix, where P_σ is defined as

$$(P_\sigma)_{ij} = \begin{cases} 1 & \text{if } j = \sigma(i); \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

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 $\le_u$ over $\{1, 2, \dots, m\}^2$ defined as

$$i \le_u j \text{ if } i = j \text{ or } (M(u))_{ij} > 0. \tag{9}$$

We show that this relation is a partial order:

- Reflexivity: this follows immediately from the definition;
- Transitivity: If $i \le_u j$ and $j \le_u k$ with $i \ne j \ne k$, 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 $i \le_u k$;
- Antisymmetry: suppose, towards building a contradiction, that there exist $i \ne j$ satisfying $i \le_u j$ and $j \le_u i$. 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 uu: one that loops twice on i (witnessed by $(M(u))_{ii} > 0$), and one that goes to j while reading the first copy of u ($(M(u))_{ij} > 0$), 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 [33] showed that an automaton is not polynomially-ambiguous if and only if there exists a state q and a word w such that the 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 $\le_u$:

$$i \le_u j \text{ implies } \sigma(i) \ge \sigma(j) \text{ for every } 1 \le i, j \le m. \tag{10}$$

Finally, let P_σ be the corresponding permutation matrix (as described by Equation (8)), 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 $1 \le i < j \le m$ we have that $\sigma^{-1}(i) \not\le_u \sigma^{-1}(j)$ (by the contrapositive of Equation (10)), therefore $(M(u))_{\sigma^{-1}(i)\sigma^{-1}(j)} = 0$ by Equation (9), and in turn:

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

Proof of Lemma 5.1. 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 5.7. We define $R_\mathcal{A} = \text{tower}_r(2\ell \cdot |\Sigma|)$ as the constant from Lemma 5.5. 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 5.5. By Lemma 5.7 we can choose $1 \le i \le j \le \ell$ such that $M(u_i \dots u_j)$ has an idempotent structure. By Lemma 5.8 we know that $M(u_i \dots u_j)$ is p-triangular. As $\mathcal{A}$ is pumpable, there is

[2] The use of [17] is enabled thanks to our *non-negative transitions* assumption (see Remark 2.7).

an entry $d \in \mathbb{N}$ of the diagonal of $M(u_i \dots u_j)$ satisfying

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

for all $n \geq |\mathcal{A}|$. Let us rewrite the left-hand side as a product of matrices: $I \cdot M(u_\vdash u_1 \dots u_{i-1}) \cdot M(u_i \dots u_j)^n \cdot M(u_{j+1} \dots u_\ell u_\dashv v) \cdot F$. Lemma 5.5 allows us to replace $M(u_i \dots u_j)$ with an invertible matrix P. We get:

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

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, we can rewrite it according to Equation (2) of Lemma 3.1:

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

for all $n \in \mathbb{N}$. We revert P to $M(u_i \dots u_j)$, which yields the expression required by Lemma 5.1 once we set $n = 1$:

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

6 DECISION PROCEDURES

This section is devoted to proving:

Proposition 3.5. *We can decide in polynomial space whether a given weighted automaton is pumpable, respectively blindly pumpable.*

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 3.2) 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

$$\mathcal{A}(uv^{|\mathcal{A}|+n}w) = d^n \cdot \mathcal{A}(uv^{|\mathcal{A}|}w) \text{ for every } n \in \mathbb{N}. \tag{11}$$

However, weak pumpability only requires that Equation (11) applies for $n = 1$:

Definition 6.1. (Weak Pumpability) A weighted automaton $\mathcal{A}$ 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

$$\mathcal{A}(uv^{|\mathcal{A}|+1}w) = d \cdot \mathcal{A}(uv^{|\mathcal{A}|}w). \tag{12}$$

We show that pumpability and weak pumpability are equivalent:

Lemma 6.2. *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 now suppose that $\mathcal{A}$ is a weakly pumpable automaton and show $\mathcal{A}$ is also pumpable. Let m denote the size of $\mathcal{A}$, and consider any triple u, v, w with $M(v)$ p-triangular. Then for all $n > 0$ considering the triple uv^{n-1}, v, w yields the existence of a diagonal entry d_n of $M(v)$ satisfying

$$\mathcal{A}(uv^{m+n+1}w) = d_n \cdot \mathcal{A}(uv^{m+n}w). \tag{13}$$

To prove that $\mathcal{A}$ is pumpable, we show that all the d_n are identical.

Consider the unary weighted automaton $\mathcal{U}$ satisfying $\mathcal{U}(a^n) = \mathcal{A}(uv^n w)$ for every $n \in \mathbb{N}$. The automaton $\mathcal{U}$ is constructed with alphabet $\{a\}$, over the same state space as $\mathcal{A}$ with initial vector $IM(u)$, final vector $M(w)F$ and $M_{\mathcal{U}}(a) = M(v)$. Equation (13) yields a sequence $(d_i)_{i \in \mathbb{N}}$ of diagonal entries of $M(v)$ such that

$$\mathcal{U}(a^n) = d_1 d_2 d_3 \dots d_n \mathcal{A}(uv^m w) \text{ for all } n \in \mathbb{N}.$$

Since $M(v)$ has at most m distinct diagonal entries, this implies that the set of prime divisors of $\{\mathcal{U}(a^n) \mid n \in \mathbb{N}\}$ is finite. Therefore $\mathcal{U}$ is unambiguisable by the characterisation of Theorem 5.2, and Proposition 3.3 yields that $\mathcal{U}$ is pumpable. In particular, consider the triple $(\varepsilon, a, \varepsilon)$: we have that $M(a)$ is p-triangular, so there is some d in the diagonal of $M(a)$ such that

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

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. $\square$

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

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

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

(2) For all $n \in \mathbb{N}$ there exist an automaton $\mathcal{B}_n$ 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 C_i of size $m + 2$ and norm $||\mathcal{A}||$ satisfying

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

Remark 6.4. The values of $\mathcal{B}_n$ and C_i are unimportant and unspecified when the input does not take the form $u\$v\$w$ for $u, v, w \in \Sigma$.

Proof Sketch.

(1) The main states of $\mathcal{T}$ are the finite set of path structures $\{\overline{M}(v) \mid v \in \Sigma^*\}$ (finitely many 0-1 matrices). On $u\$v\$w$, the current path structure is updated while reading v, while u and w have no effect (the automaton loops in a holding state before the first \$ and different holding states after the second \$). The automaton outputs with weight 1 if it was in a state representing a p-triangular matrix at the point of reading the second \$, and 0 otherwise.
(2) On $u\$v\$w$, the automaton $\mathcal{B}_n$ behaves like $\mathcal{A}$ on u and w. While reading v, $\mathcal{B}_n$ simultaneously guesses n runs by taking the n-fold product of the automaton, with transition weights that are the product of the weights in each of the n copies. The automaton must check that the ith guessed run is a valid continuation of the $(i-1)$th run (for $2 \leq i \leq n$); the state remembers the starting state for each copy for the duration of the run and then verifies that it matches the final state of the previous copy at the point of reading the second \$.
(3) On $u\$v\$w$, C_i ignores u and w and behaves like $\mathcal{A}$ while reading v. So that only runs from i to i are considered in $\mathcal{A}$, the first \$ leads to state i and the second \$ can only leave state i. □

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 $(C_i)_{1\leq i\leq m}$ be the weighted automata constructed in Lemma 6.3. Using the result of Lemma 2.5 that the difference and product of functions recognised by weighted automata are themselves effectively expressible by weighted automata, let

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

Lemma 6.5. *$\mathcal{P}_{\mathcal{A}}$ 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 $1 \leq i \leq m$ satisfying

$$\mathcal{B}_{m+1}(u\$v\$w) = \mathcal{B}_m(u\$v\$w)C_i(u\$v\$w),$$

that is, there exists $1 \leq i \leq m$ satisfying

$$\mathcal{A}(uv^{m+1}w) = (M(v))_{ii} \cdot \mathcal{A}(uv^m w).$$ □

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:

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

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

$$||\mathcal{P}_{\mathcal{A}}|| = ||\mathcal{T}|| \cdot \prod_{i=1}^{m} \max\{||\mathcal{B}_{m+1}||, ||\mathcal{B}_m|||C_i||\}$$
$$= 1 \cdot (||\mathcal{A}||^{m+1})^m = ||\mathcal{A}||^{m^2+m}. \tag{14}$$

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 NC^2 [19, 32]. 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 [29].

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 $\mathcal{P}_A$ 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 6.6. *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 C_i for all $1 \leq i \leq m$. The automaton $\mathcal{P}_{\mathcal{A}}$ is built applying Lemma 2.5 polynomially many times. By induction, at each step the norm is at most exponential (in total bounded by Equation (14)) and the weight can be computed recursively in polynomial space using the second part of Lemma 2.5. □

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 start with the assumption that it is already known that $\mathcal{A}$ is pumpable, by first applying the algorithm of the previous subsection. 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 subsection). In order to do this we generalise the automata $\mathcal{T}$ and $\mathcal{B}_i$ of Lemma 6.3 to read two suffixes:

Lemma 6.7. *Let $\mathcal{A} = (Q, \Sigma, M, I, F)$ be a weighted automaton of size $m = |\mathcal{A}|$ over Σ, and let $\$ \notin \Sigma$.*

(1) *There exists an automaton $\mathcal{T}'$ of size $2^{m^2}+3$ and norm 1 satisfying*

$$\mathcal{T}'(u\$v\$w\$w') = 1 \quad \text{for all } u,v,w,w' \in \Sigma^* \text{ such that } M(v) \text{ is p-triangular;}$$
$$\mathcal{T}'(u) = 0 \quad \text{for all other } u \in (\Sigma \cup \{\$\})^*.$$

(2) *For all $n \in \mathbb{N}$ there exist two automata $\mathcal{B}_{1,n}$, $\mathcal{B}_{2,n}$ of size $m^{2n}+2m+1$ and norm $||\mathcal{A}||^n$ satisfying*

$$\mathcal{B}_{1,n}(u\$v\$w\$w') = \mathcal{B}_n(u\$v\$w) \quad \text{for all } u,v,w,w' \in \Sigma^*;$$
$$\mathcal{B}_{2,n}(u\$v\$w\$w') = \mathcal{B}_n(u\$v\$w') \quad \text{for all } u,v,w,w' \in \Sigma^*.$$

Proof. These automata are easily obtained by modifying the constructions in Lemma 6.3. □

We combine these automata to construct an automaton that can be used to check blind pumbability. Given an automaton $\mathcal{A}$ of size m, let

$$\mathcal{Q}_{\mathcal{A}} = \mathcal{T}' \cdot (\mathcal{B}_{1,m+1} \cdot \mathcal{B}_{2,m} - \mathcal{B}_{1,m} \cdot \mathcal{B}_{2,m+1}).$$

Lemma 6.8. *Let $\mathcal{A}$ be a pumpable automaton. Then $\mathcal{A}$ is blindly pumpable if and only if $\mathcal{Q}_{\mathcal{A}}$ maps every word to zero.*

Proof. Suppose that $\mathcal{A}$ is pumpable. Then for every $u,v,w,w' \in \Sigma^*$, if $M(v)$ is p-triangular there exist two entries d_1 and d_2 of the diagonal of $M(v)$ satisfying

$$\mathcal{A}(uv^{m+n}w) = d_1^n \cdot \mathcal{A}(uv^m w) \text{ for all } n \geq 0; \tag{15}$$
$$\mathcal{A}(uv^{m+n}w') = d_2^n \cdot \mathcal{A}(uv^m w') \text{ for all } n \geq 0. \tag{16}$$

A direct consequence of these equations is that $\mathcal{A}(uv^m w) = 0$ if and only if $\mathcal{A}(uv^{m+n}w) = 0$ for all $n \geq 0$; and $\mathcal{A}(uv^m w') = 0$ if and only if $\mathcal{A}(uv^{m+n}w') = 0$ for all $n \geq 0$.

To decide whether $\mathcal{A}$ is blindly pumpable, we need to check whether there exists a single entry $d = d_1 = d_2$ that satisfies both Equation (15) and Equation (16). Remark that it is sufficient to verify the property at $n = 1$ as the choice of d is the same for all $n \geq 1$. In other words we need to check that either $\mathcal{A}(uv^m w) = 0$ or $\mathcal{A}(uvw') = 0$ (in these cases every d work), or

$$\frac{\mathcal{A}(uv^{m+1}w)}{\mathcal{A}(uv^m w)} = \frac{\mathcal{A}(uv^{m+1}w')}{\mathcal{A}(uv^m w')}. \tag{17}$$

To conclude the proof, observe that for every $u,v,w,w' \in \Sigma^*$ the automaton $\mathcal{Q}_{\mathcal{A}}$ maps $u\$v\$w\$w'$ to 0 if and only if either:

- $\mathcal{T}'$ is zero, that is, $M(v)$ is not p-triangular;
- $\mathcal{A}(uv^m w) = \mathcal{A}(uv^{m+1}w) = 0$;
- $\mathcal{A}(uv^m w') = \mathcal{A}(uv^{m+1}w') = 0$;
- Equation (17) 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 5.4, and recall that the constant $R_{\mathcal{A}}$ is obtained from Lemma 5.5). We cannot extract the unambiguous automaton from our techniques[3]. Recall that for unambiguisation in the proof of Proposition 3.4 we rely on Bell and Smertnig's result: Theorem 5.2. 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 [33], but the size of the unambiguous automaton could be arbitrarily big. We leave open whether one could solve the determinisation and unambiguisation problems *constructively* in elementary time and space. The result of Bell and Smertnig applies to any field, while we have focused on the field of rationals, we leave open whether our result can be extended to other fields.

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 [32].

REFERENCES

[1] Corentin Barloy, Nathanaël Fijalkow, Nathan Lhote, and Filip Mazowiecki. 2022. A robust class of linear recurrence sequences. *Inf. Comput.* 289, Part (2022), 104964. https://doi.org/10.1016/j.ic.2022.104964

[2] Jason Bell and Daniel Smertnig. 2021. Noncommutative rational Pólya series. *Selecta Mathematica* 27, 3 (2021), 1–34.

[3] Jason P. Bell and Daniel Smertnig. 2023. Computing the linear hull: Deciding Deterministic? and Unambiguous? for weighted automata over fields. In *LICS*. 1–13. https://doi.org/10.1109/LICS56636.2023.10175691

[4] Paul C. Bell. 2022. Polynomially ambiguous probabilistic automata on restricted languages. *J. Comput. Syst. Sci.* 127 (2022), 53–65. https://doi.org/10.1016/j.jcss.2022.02.002

[5] Yahia Idriss Benalioua, Nathan Lhote, and Pierre-Alain Reynier. 2023. Register Minimization of Cost Register Automata over a Field. *CoRR* abs/2307.13505 (2023). https://doi.org/10.48550/arXiv.2307.13505 arXiv:2307.13505

[6] Georgina Bumpus, Christoph Haase, Stefan Kiefer, Paul-Ioan Stoienescu, and Jonathan Tanner. 2020. On the Size of Finite Rational Matrix Semigroups. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference) (LIPIcs, Vol. 168)*, Artur Czumaj, Anuj Dawar, and Emanuela Merelli (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 115:1–115:13. https://doi.org/10.4230/LIPIcs.ICALP.2020.115

[7] Dmitry Chistikov, Stefan Kiefer, Andrzej S. Murawski, and David Purser. 2022. The Big-O Problem. *Log. Methods Comput. Sci.* 18, 1 (2022). https://doi.org/10.46298/lmcs-18(1:40)2022

[8] Thomas Colcombet. 2015. Unambiguity in Automata Theory. In *Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9118)*, Jeffrey O. Shallit and Alexander Okhotin (Eds.). Springer, 3–18. https://doi.org/10.1007/978-3-319-19225-3_1

[9] Wojciech Czerwinski and Piotr Hofman. 2022. Language Inclusion for Boundedly-Ambiguous Vector Addition Systems Is Decidable. In *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland (LIPIcs, Vol. 243)*, Bartek Klin, Slawomir Lasota, and Anca Muscholl (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 16:1–16:22. https://doi.org/10.4230/LIPIcs.CONCUR.2022.16

[10] Wojciech Czerwinski, Engel Lefaucheux, Filip Mazowiecki, David Purser, and Markus A. Whiteland. 2022. The boundedness and zero isolation problems for weighted automata over nonnegative rationals. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5,*

[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 automata and test each for equivalence.

2022, Christel Baier and Dana Fisman (Eds.). ACM, 15:1–15:13. https://doi.org/10.1145/3531130.3533336

[11] Laure Daviaud. 2020. Register complexity and determinisation of max-plus automata. *ACM SIGLOG News* 7, 2 (2020), 4–14. https://doi.org/10.1145/3397619.3397621

[12] Laure Daviaud, Marcin Jurdzinski, Ranko Lazic, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. 2021. When are emptiness and containment decidable for probabilistic automata? *J. Comput. Syst. Sci.* 119 (2021), 78–96. https://doi.org/10.1016/j.jcss.2021.01.006

[13] Manfred Droste, Werner Kuich, and Heiko Vogler. 2009. *Handbook of weighted automata.* Springer Science & Business Media.

[14] Nathanaël Fijalkow, Cristian Riveros, and James Worrell. 2017. Probabilistic Automata of Bounded Ambiguity. In *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany (LIPIcs, Vol. 85)*, Roland Meyer and Uwe Nestmann (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:14. https://doi.org/10.4230/LIPIcs.CONCUR.2017.19

[15] Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. 2005. *Skolem's problem–on the border between decidability and undecidability.* Technical Report. Citeseer.

[16] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. 2018. Polynomial Invariants for Affine Programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, Anuj Dawar and Erich Grädel (Eds.). ACM, 530–539. https://doi.org/10.1145/3209108.3209142

[17] Ismaël Jecker. 2021. A Ramsey Theorem for Finite Monoids. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference) (LIPIcs, Vol. 187)*, Markus Bläser and Benjamin Monmege (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 44:1–44:13. https://doi.org/10.4230/LIPIcs.STACS.2021.44

[18] Ismaël Jecker, Filip Mazowiecki, and David Purser. 2023. Determinisation and Unambiguisation of Polynomially-Ambiguous Rational Weighted Automata. arXiv:2310.02204 [cs.FL]

[19] Stefan Kiefer, Andrzej S. Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. 2013. On the Complexity of Equivalence and Minimisation for Q-weighted Automata. *Log. Methods Comput. Sci.* 9, 1 (2013). https://doi.org/10.2168/LMCS-9(1:8)2013

[20] Daniel Kirsten and Sylvain Lombardy. 2009. Deciding Unambiguity and Sequentiality of Polynomially Ambiguous Min-Plus Automata. In *26th International Symposium on Theoretical Aspects of Computer Science (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 3)*, Susanne Albers and Jean-Yves Marion (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 589–600. https://doi.org/10.4230/LIPIcs.STACS.2009.1850

[21] Daniel Kirsten and Ina Mäurer. 2005. On the Determinization of Weighted Automata. *J. Autom. Lang. Comb.* 10, 2/3 (2005), 287–312. https://doi.org/10.25596/jalc-2005-287

[22] Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. 2004. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.* 327, 3 (2004), 349–373. https://doi.org/10.1016/j.tcs.2004.02.049

[23] Peter Kostolányi. 2022. Determinisability of unary weighted automata over the rational numbers. *Theor. Comput. Sci.* 898 (2022), 110–131. https://doi.org/10.1016/j.tcs.2021.11.002

[24] Sylvain Lombardy and Jacques Sakarovitch. 2006. Sequential? *Theor. Comput. Sci.* 356, 1-2 (2006), 224–244. https://doi.org/10.1016/j.tcs.2006.01.028

[25] Mehryar Mohri. 1997. Finite-State Transducers in Language and Speech Processing. *Comput. Linguistics* 23, 2 (1997), 269–311.

[26] Azaria Paz. 1971. *Introduction to probabilistic automata.* Academic Press.

[27] Mikhail A. Raskin. 2018. A Superpolynomial Lower Bound for the Size of Non-Deterministic Complement of an Unambiguous Automaton. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic (LIPIcs, Vol. 107)*, Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 138:1–138:11. https://doi.org/10.4230/LIPIcs.ICALP.2018.138

[28] Christophe Reutenauer. 1979. On Polya series in noncommuting variables. In *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, Lothar Budach (Ed.). Akademie-Verlag, Berlin, 391–396.

[29] Walter L. Ruzzo. 1981. On Uniform Circuit Complexity. *J. Comput. Syst. Sci.* 22, 3 (1981), 365–383. https://doi.org/10.1016/0022-0000(81)90038-6

[30] Marcel Paul Schützenberger. 1961. On the Definition of a Family of Automata. *Inf. Control.* 4, 2-3 (1961), 245–270. https://doi.org/10.1016/S0019-9958(61)80020-X

[31] Gilbert Strang. 2022. *Introduction to linear algebra.* SIAM.

[32] Wen-Guey Tzeng. 1996. On Path Equivalence of Nondeterministic Finite Automata. *Inf. Process. Lett.* 58, 1 (1996), 43–46. https://doi.org/10.1016/0020-0190(96)00039-7

[33] Andreas Weber and Helmut Seidl. 1991. On the Degree of Ambiguity of Finite Automata. *Theor. Comput. Sci.* 88, 2 (1991), 325–349. https://doi.org/10.1016/0304-3975(91)90381-B