Minimisation of Multiplicity Tree Automata

Stefan Kiefer, Ines Marusic, and James Worrell

University of Oxford, UK

Abstract. We consider the problem of minimising the number of states in a multiplicity tree automaton over the field of rational numbers. We give a minimisation algorithm that runs in polynomial time assuming unit-cost arithmetic. We also show that a polynomial bound in the standard Turing model would require a breakthrough in the complexity of polynomial identity testing by proving that the latter problem is logspace equivalent to the decision version of minimisation. The developed techniques also improve the state of the art in multiplicity word automata: we give an NC algorithm for minimising multiplicity word automata. Finally, we consider the minimal consistency problem: does there exist an automaton with n states that is consistent with a given finite sample of weight-labelled words or trees? We show that this decision problem is complete for the existential theory of the rationals, both for words and for trees of a fixed alphabet rank.

1 Introduction

Minimisation is a fundamental problem in automata theory that is closely related to both learning and equivalence testing. In this work we analyse the complexity of minimisation for multiplicity automata, i.e., weighted automata over a field. We take a comprehensive view, looking at multiplicity automata over both words and trees and considering both function and decision problems. We also look at the closely related problem of obtaining a minimal automaton consistent with a given finite set of observations. We characterise the complexity of these problems in terms of arithmetic and Boolean circuit classes. In particular, we give relationships to longstanding open problems in arithmetic complexity theory.

Multiplicity tree automata were first introduced by Berstel and Reutenauer [1] under the terminology of linear representations of a tree series. They generalise multiplicity word automata, introduced by Schützenberger [25], which can be viewed as multiplicity tree automata on unary trees. The minimisation problem for multiplicity word automata has long been known to be solvable in polynomial time [25].

In this work, we give a new procedure for computing minimal word automata and thereby place minimisation in NC improving also on a randomised NC procedure in [22]. (Recall that NL $\subseteq$ NC $\subseteq$ P, where NC comprises those languages having L-uniform Boolean circuits of polylogarithmic depth and polynomial size, or, equivalently, those problems solvable in polylogarithmic time on

A. Pitts (Ed.): FOSSACS 2015, LNCS 9034, pp. 297–311, 2015.
DOI: 10.1007/978-3-662-46678-0_19

parallel random-access machines with polynomially many processors.) By comparison, minimising deterministic word automata is NL-complete [12], while minimising non-deterministic word automata is PSPACE-complete [20].

Over trees, we give what is (to the best of our knowledge) the first complexity analysis of the problem of minimising multiplicity automata. We present an algorithm that minimises a given tree automaton $\mathcal{A}$ in time $O\left(|\mathcal{A}|^2 \cdot r\right)$ where r is the maximum alphabet rank, assuming unit-cost arithmetic. This procedure can be viewed as a concrete version of the construction of a syntactic algebra of a recognisable tree series in [4]. We thus place the problem within PSPACE in the conventional Turing model. We are moreover able to precisely characterise the complexity of the decision version of the minimisation problem as being logspace equivalent to the arithmetic circuit identity testing (ACIT) problem, commonly also called the polynomial identity testing problem. The latter problem is very well studied, with a variety of randomised polynomial-time algorithms, but, as yet, no deterministic polynomial-time procedure. In previous work we have reduced equivalence testing of multiplicity tree automata to ACIT [24]; the advance here is to reduce the more general problem of minimisation also to ACIT.

Finally, we consider the problem of computing a minimal multiplicity automaton consistent with a finite set of input-output behaviours. This is a natural learning problem whose complexity for non-deterministic finite automata was studied by Gold [17]. For multiplicity word automata over a field $\mathbb{F}$, we show that the decision version of this problem is logspace equivalent to the problem of deciding the truth of existential first-order sentences over the field $(\mathbb{F}, +, \cdot, 0, 1)$, a long-standing open problem in case $\mathbb{F} = \mathbb{Q}$. Furthermore we show that the same result holds for multiplicity tree automata of a fixed alphabet rank, but we leave open the complexity of the problem for general multiplicity tree automata.

The full version of this paper is available as [21].

Further Related Work. Based on a generalisation of the Myhill-Nerode theorem to trees, one obtains a procedure for minimising deterministic tree automata that runs in time quadratic in the size of the input automaton [7,11]. There have also been several works on minimising deterministic tree automata with weights in a semi-field (that is, a semi-ring with multiplicative inverses). In particular, Maletti [23] gives a polynomial-time algorithm in this setting, assuming unit cost for arithmetic in the semi-field. In the non-deterministic case, Carme et al. [10] define the subclass of *residual finite* non-deterministic tree automata. They show that this class expresses the class of regular tree languages and admits a polynomial-space minimisation procedure.

2 Preliminaries

Let $\mathbb{N}$ and $\mathbb{N}_0$ denote the set of all positive and non-negative integers, respectively. For every $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, 2, \ldots, n\}$.

Matrices and Vectors. Let $n \in \mathbb{N}$. We write I_n for the identity matrix of order n. For every $i \in [n]$, we write e_i for the i^{th} n-dimensional coordinate row

vector. For any matrix A, we write A_i for its i^{th} row, A^j for its j^{th} column, and $A_{i,j}$ for its $(i,j)^{\text{th}}$ entry. Given nonempty subsets I and J of the rows and columns of A, respectively, we write $A_{I,J}$ for the submatrix $(A_{i,j})_{i\in I, j\in J}$ of A.

Let A be an $m \times n$ matrix with entries in a field $\mathbb{F}$. The *row space* of A, written $RS(A)$, is the subspace of $\mathbb{F}^n$ spanned by the rows of A. The *column space* of A, written $CS(A)$, is the subspace of $\mathbb{F}^m$ spanned by the columns of A.

Given a set $S \subseteq \mathbb{F}^n$, we use $\langle S \rangle$ to denote the vector subspace of $\mathbb{F}^n$ that is spanned by S, where we often omit the braces when denoting S.

Kronecker Product. Let A be an $m_1 \times n_1$ matrix and B an $m_2 \times n_2$ matrix. The *Kronecker product* of A by B, written as $A \otimes B$, is an $m_1 m_2 \times n_1 n_2$ matrix where $(A \otimes B)_{(i_1-1)m_2+i_2,(j_1-1)n_2+j_2} = A_{i_1,j_1} \cdot B_{i_2,j_2}$ for every $i_1 \in [m_1]$, $i_2 \in [m_2]$, $j_1 \in [n_1]$, $j_2 \in [n_2]$.

The Kronecker product is bilinear, associative, and has the following *mixed-product property*: For any matrices A, B, C, D such that products $A \cdot C$ and $B \cdot D$ are defined, it holds that $(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$.

For every $k \in \mathbb{N}_0$ we define the *k-fold Kronecker power* of a matrix A, written as $A^{\otimes k}$, inductively by $A^{\otimes 0} = I_1$ and $A^{\otimes k} = A^{\otimes(k-1)} \otimes A$ for $k \geq 1$.

Multiplicity Word Automata. Let Σ be a finite alphabet and ε be the empty word. The set of all words over Σ is denoted by Σ^*, and the length of a word $w \in \Sigma^*$ is denoted by $|w|$. For any $n \in \mathbb{N}_0$ we write $\Sigma^n := \{w \in \Sigma^* : |w| = n\}$, $\Sigma^{\leq n} := \bigcup_{l=0}^{n} \Sigma^l$, and $\Sigma^{<n} := \Sigma^{\leq n} \setminus \Sigma^n$. Given two words $x, y \in \Sigma^*$, we denote by xy the concatenation of x and y. Given two sets $X, Y \subseteq \Sigma^*$, we define $XY := \{xy : x \in X, y \in Y\}$.

Let $\mathbb{F}$ be a field. A *word series* over Σ with coefficients in $\mathbb{F}$ is a mapping $f : \Sigma^* \to \mathbb{F}$. The *Hankel matrix* of f is the matrix $H : \Sigma^* \times \Sigma^* \to \mathbb{F}$ such that $H_{x,y} = f(xy)$ for all $x, y \in \Sigma^*$.

An $\mathbb{F}$-*multiplicity word automaton* ($\mathbb{F}$-*MWA*) is a 5-tuple $\mathcal{A} = (n, \Sigma, \mu, \alpha, \gamma)$ which consists of the *dimension* $n \in \mathbb{N}_0$ representing the number of states, a finite alphabet Σ, a function $\mu : \Sigma \to \mathbb{F}^{n\times n}$ assigning a *transition matrix* $\mu(\sigma)$ to each $\sigma \in \Sigma$, the *initial weight vector* $\alpha \in \mathbb{F}^{1\times n}$, and the *final weight vector* $\gamma \in \mathbb{F}^{n\times 1}$. We extend the function μ from Σ to Σ^* by $\mu(\varepsilon) := I_n$ and $\mu(\sigma_1 \dots \sigma_k) := \mu(\sigma_1) \cdot \ldots \cdot \mu(\sigma_k)$ for any $\sigma_1, \dots, \sigma_k \in \Sigma$. It is easy to see that $\mu(xy) = \mu(x) \cdot \mu(y)$ for any $x, y \in \Sigma^*$. Automaton $\mathcal{A}$ *recognises* the word series $\|\mathcal{A}\| : \Sigma^* \to \mathbb{F}$ where $\|\mathcal{A}\|(w) = \alpha \cdot \mu(w) \cdot \gamma$ for every $w \in \Sigma^*$.

Finite Trees. A *ranked alphabet* is a tuple (Σ, rk) where Σ is a nonempty finite set of symbols and $rk : \Sigma \to \mathbb{N}_0$ is a function. Ranked alphabet (Σ, rk) is often written Σ for short. For every $k \in \mathbb{N}_0$, we define the set of all *k-ary* symbols $\Sigma_k := rk^{-1}(\{k\})$. We say that Σ has *rank* r if $r = \max\{rk(\sigma) : \sigma \in \Sigma\}$.

The set of *Σ-trees* (*trees* for short), written T_Σ, is the smallest set T satisfying (i) $\Sigma_0 \subseteq T$, and (ii) if $\sigma \in \Sigma_k$, $t_1, \dots, t_k \in T$ then $\sigma(t_1, \dots, t_k) \in T$. The *height* of a tree t, $height(t)$, is defined by $height(t) = 0$ if $t \in \Sigma_0$, and $height(t) = 1 + \max_{i\in[k]} height(t_i)$ if $t = \sigma(t_1, \dots, t_k)$ for some $k \geq 1$. For any $n \in \mathbb{N}_0$ we write $T_\Sigma^n := \{t \in T_\Sigma : height(t) = n\}$, $T_\Sigma^{\leq n} := \bigcup_{l=0}^{n} T_\Sigma^l$, and $T_\Sigma^{<n} := T_\Sigma^{\leq n} \setminus T_\Sigma^n$.

Let $\Box$ be a nullary symbol not contained in Σ. The set C_Σ of *Σ-contexts* (*contexts* for short) is the set of $(\{\Box\} \cup \Sigma)$-trees in which $\Box$ occurs exactly once. Let $n \in \mathbb{N}_0$. We denote by C_Σ^n the set of all contexts $c \in C_\Sigma$ where the distance between the root and the $\Box$-labelled node of c is equal to n. Moreover, we write $C_\Sigma^{\leq n} := \bigcup_{l=0}^{n} C_\Sigma^l$ and $C_\Sigma^{<n} := C_\Sigma^{\leq n} \setminus C_\Sigma^n$. A *subtree* of $c \in C_\Sigma$ is a Σ-tree consisting of a node in c and all of its descendants. Given a set $S \subseteq T_\Sigma$, we denote by $C_{\Sigma,S}^n$ the set of all contexts $c \in C_\Sigma^n$ where every subtree of c is an element of S; we moreover write $C_{\Sigma,S}^{\leq n} := \bigcup_{l=0}^{n} C_{\Sigma,S}^l$ and $C_{\Sigma,S}^{<n} := C_{\Sigma,S}^{\leq n} \setminus C_{\Sigma,S}^n$.

Given $c \in C_\Sigma$ and $t \in T_\Sigma \dot\cup C_\Sigma$, we write $c[t]$ for the tree obtained by substituting t for $\Box$ in c. Let $\mathbb{F}$ be a field. A *tree series* over Σ with coefficients in $\mathbb{F}$ is a mapping $f : T_\Sigma \to \mathbb{F}$. The *Hankel matrix* of $f : T_\Sigma \to \mathbb{F}$ is the matrix $H : T_\Sigma \times C_\Sigma \to \mathbb{F}$ such that $H_{t,c} = f(c[t])$ for every $t \in T_\Sigma$ and $c \in C_\Sigma$.

Multiplicity Tree Automata. Let $\mathbb{F}$ be a field. An *$\mathbb{F}$-multiplicity tree automaton* ($\mathbb{F}$-*MTA*) is a 4-tuple $\mathcal{A} = (n, \Sigma, \mu, \gamma)$ which consists of the *dimension* $n \in \mathbb{N}_0$ representing the number of states, a ranked alphabet Σ, the *tree representation* $\mu = \{\mu(\sigma) : \sigma \in \Sigma\}$ where for every symbol $\sigma \in \Sigma$, $\mu(\sigma) \in \mathbb{F}^{n^{rk(\sigma)} \times n}$ represents the *transition matrix* associated to σ, and the *final weight vector* $\gamma \in \mathbb{F}^{n \times 1}$. We speak of an MTA if the field $\mathbb{F}$ is clear from the context or irrelevant. The *size* of $\mathcal{A}$, written as $|\mathcal{A}|$, is the total number of entries in all transition matrices and the final weight vector of $\mathcal{A}$, i.e., $|\mathcal{A}| := \sum_{\sigma \in \Sigma} n^{rk(\sigma)+1} + n$.

We extend the tree representation μ from Σ to T_Σ by $\mu(\sigma(t_1, \ldots, t_k)) := (\mu(t_1) \otimes \cdots \otimes \mu(t_k)) \cdot \mu(\sigma)$ for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$. Automaton $\mathcal{A}$ *recognises* the tree series $\|\mathcal{A}\| : T_\Sigma \to \mathbb{F}$ where $\|\mathcal{A}\|(t) = \mu(t) \cdot \gamma$ for every $t \in T_\Sigma$.

We further extend μ from T_Σ to C_Σ by treating $\Box$ as a unary symbol and defining $\mu(\Box) := I_n$. This allows to define $\mu(c) \in \mathbb{F}^{n \times n}$ for every $c = \sigma(t_1, \ldots, t_k) \in C_\Sigma$ inductively as $\mu(c) := (\mu(t_1) \otimes \cdots \otimes \mu(t_k)) \cdot \mu(\sigma)$. It is easy to see that for every $t \in T_\Sigma \dot\cup C_\Sigma$ and $c \in C_\Sigma$, $\mu(c[t]) = \mu(t) \cdot \mu(c)$.

MWAs can be seen as a special case of MTAs: An MWA $(n, \Sigma, \mu, \alpha, \gamma)$ "is" the MTA $(n, \Sigma \dot\cup \{\sigma_0\}, \mu, \gamma)$ where the symbols in Σ are unary, symbol σ_0 is nullary, and $\mu(\sigma_0) = \alpha$. That is, we view $(\Sigma \dot\cup \{\sigma_0\})$-trees as words over Σ by omitting the leaf symbol σ_0. Hence if a result holds for MTAs, it also holds for MWAs. Some concepts, such as contexts, would formally need adaptation, however we omit such adaptations as they are straightforward. Therefore, we freely view MWAs as MTAs whenever convenient.

Two MTAs $\mathcal{A}_1, \mathcal{A}_2$ are said to be *equivalent* if $\|\mathcal{A}_1\| = \|\mathcal{A}_2\|$. An MTA is said to be *minimal* if no equivalent automaton has strictly smaller dimension. The following result was first shown by Habrard and Oncina [18], although a closely related result was given by Bozapalidis and Louscou-Bozapalidou [6].

Theorem 1 ([6,18]). *Let Σ be a ranked alphabet, $\mathbb{F}$ be a field, and $f : T_\Sigma \to \mathbb{F}$. Let H be the Hankel matrix of f. Then, f is recognised by some MTA if and only if H has finite rank over $\mathbb{F}$. In case H has finite rank over $\mathbb{F}$, the dimension of a minimal MTA recognising f is rank(H) over $\mathbb{F}$.*

It follows from Theorem 1 that an $\mathbb{F}$-MTA $\mathcal{A}$ of dimension n is minimal if and only if the Hankel matrix of $\|\mathcal{A}\|$ has rank n over $\mathbb{F}$.

Remark 2. Theorem 1 specialised to word automata was proved by Carlyle and Paz [9] and Fliess [16]. Their proofs show that if $X, Y \subseteq \Sigma^*$ are such that $rank(H_{X,Y}) = rank(H)$, then f is uniquely determined by $H_{X,Y}$ and $H_{X\Sigma,Y}$.

The following closure properties for MTAs can be found in [1,3]; see also [21].

Proposition 3. *Let $\mathcal{A}_1 = (n_1, \Sigma, \mu_1, \gamma_1)$, $\mathcal{A}_2 = (n_2, \Sigma, \mu_2, \gamma_2)$ be two $\mathbb{F}$-MTAs. One can construct an $\mathbb{F}$-MTA $\mathcal{A}_1 - \mathcal{A}_2$, called the* difference *of $\mathcal{A}_1$ and $\mathcal{A}_2$, such that $\|\mathcal{A}_1 - \mathcal{A}_2\| = \|\mathcal{A}_1\| - \|\mathcal{A}_2\|$. Secondly, one can construct an $\mathbb{F}$-MTA $\mathcal{A}_1 \times \mathcal{A}_2 = (n_1 \cdot n_2, \Sigma, \mu, \gamma_1 \otimes \gamma_2)$, called the* product *of $\mathcal{A}_1$ by $\mathcal{A}_2$, such that $\mu(t) = \mu_1(t) \otimes \mu_2(t)$ for every $t \in T_\Sigma$, $\mu(c) = \mu_1(c) \otimes \mu_2(c)$ for every $c \in C_\Sigma$, and $\|\mathcal{A}_1 \times \mathcal{A}_2\| = \|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|$. When $\mathbb{F} = \mathbb{Q}$, both automata $\mathcal{A}_1 - \mathcal{A}_2$ and $\mathcal{A}_1 \times \mathcal{A}_2$ can be computed from $\mathcal{A}_1$ and $\mathcal{A}_2$ in logarithmic space.*

3 Fundamentals of Minimisation

In this section we prepare the ground for minimisation algorithms. Let us fix a field $\mathbb{F}$ for the rest of this section and assume that all automata are over $\mathbb{F}$. We also fix an MTA $\mathcal{A} = (n, \Sigma, \mu, \gamma)$ for the rest of the section. We will construct from $\mathcal{A}$ another MTA $\tilde{\mathcal{A}}$ which we show to be equivalent to $\mathcal{A}$ and minimal. A crucial ingredient for this construction are special vector spaces induced by $\mathcal{A}$, called the forward space and backward space.

3.1 Forward and Backward Space

The *forward space* $\mathcal{F}$ of $\mathcal{A}$ is the (row) vector space $\mathcal{F} := \langle \mu(t) : t \in T_\Sigma \rangle$ over $\mathbb{F}$. The *backward space* $\mathcal{B}$ of $\mathcal{A}$ is the (column) vector space $\mathcal{B} := \langle \mu(c) \cdot \gamma : c \in C_\Sigma \rangle$ over $\mathbb{F}$. The following Propositions 4 and 5, proved in [21], provide fundamental characterisations of $\mathcal{F}$ and $\mathcal{B}$, respectively.

Proposition 4. *The forward space $\mathcal{F}$ has the following properties:*

(a) The space $\mathcal{F}$ is the smallest vector space V over $\mathbb{F}$ such that for all $k \in \mathbb{N}_0$, $v_1, \ldots, v_k \in V$, and $\sigma \in \Sigma_k$ it holds that $(v_1 \otimes \cdots \otimes v_k) \cdot \mu(\sigma) \in V$.
(b) The set of row vectors $\{\mu(t) : t \in T_\Sigma^{\leq n}\}$ spans $\mathcal{F}$.

Proposition 5. *Let S be a set of Σ-trees such that $\{\mu(t) : t \in S\}$ spans $\mathcal{F}$. The backward space $\mathcal{B}$ has the following properties:*

(a) The space $\mathcal{B}$ is the smallest vector space V over $\mathbb{F}$ such that $\gamma \in V$, and for every $v \in V$ and $c \in C^1_{\Sigma,S}$ it holds that $\mu(c) \cdot v \in V$.
(b) The set of column vectors $\{\mu(c) \cdot \gamma : c \in C^{\leq n}_{\Sigma,S}\}$ spans $\mathcal{B}$.

3.2 A Minimal Automaton

Let F and B be matrices whose rows and columns span $\mathcal{F}$ and $\mathcal{B}$, respectively. That is, $RS(F) = \mathcal{F}$ and $CS(B) = \mathcal{B}$. We discuss later (Section 4.1) how to efficiently compute F and B. The following lemma states that $rank(F \cdot B)$ is the dimension of a minimal automaton equivalent to $\mathcal{A}$.

Lemma 6. *A minimal automaton equivalent to $\mathcal{A}$ has $m := rank(F \cdot B)$ states.*

Proof. Let H be the Hankel matrix of $\|\mathcal{A}\|$. Define the matrix $\overline{F} \in \mathbb{F}^{T_\Sigma \times [n]}$ where $\overline{F}_t = \mu(t)$ for every $t \in T_\Sigma$. Define the matrix $\overline{B} \in \mathbb{F}^{[n] \times C_\Sigma}$ where $\overline{B}^c = \mu(c) \cdot \gamma$ for every $c \in C_\Sigma$. For every $t \in T_\Sigma$ and $c \in C_\Sigma$ we have by the definitions that

$$H_{t,c} = \|\mathcal{A}\|(c[t]) = \mu(c[t]) \cdot \gamma = \mu(t) \cdot \mu(c) \cdot \gamma = \overline{F}_t \cdot \overline{B}^c \,,$$

hence $H = \overline{F} \cdot \overline{B}$. Note that

$$RS(\overline{F}) = \mathcal{F} = RS(F) \qquad \text{and} \qquad CS(\overline{B}) = \mathcal{B} = CS(B)\,. \tag{1}$$

We now have $m = rank(H) = rank(\overline{F} \cdot \overline{B}) = rank(F \cdot B)$, where the first equality is by Theorem 1, and the last equality is by (1) and a general linear-algebra argument, see [21]. ■

By definition, there exist m rows of $F \cdot B$ that span $RS(F \cdot B)$. The corresponding m rows of F form a matrix $\tilde{F} \in \mathbb{F}^{m \times n}$ with $RS(\tilde{F} \cdot B) = RS(F \cdot B)$. Define a multiplicity tree automaton $\tilde{\mathcal{A}} = (m, \Sigma, \tilde{\mu}, \tilde{\gamma})$ with $\tilde{\gamma} = \tilde{F} \cdot \gamma$ and

$$\tilde{\mu}(\sigma) \cdot \tilde{F} \cdot B = \tilde{F}^{\otimes k} \cdot \mu(\sigma) \cdot B \qquad \text{for every } \sigma \in \Sigma_k. \tag{2}$$

We show that $\tilde{\mathcal{A}}$ minimises $\mathcal{A}$:

Proposition 7. *The MTA $\tilde{\mathcal{A}}$ is well defined and is a minimal automaton equivalent to $\mathcal{A}$.*

We provide a proof in [21]. Due to the importance of Proposition 7, we sketch its proof in the rest of this subsection. We do this by proving Proposition 7 for multiplicity *word* automata. The main arguments are similar for the tree case.

Let $\mathcal{A} = (n, \Sigma, \mu, \alpha, \gamma)$ be an MWA. The forward and backward space can then be written as $\mathcal{F} = \langle \alpha \cdot \mu(w) : w \in \Sigma^* \rangle$ and $\mathcal{B} = \langle \mu(w) \cdot \gamma : w \in \Sigma^* \rangle$, respectively. The MWA $\tilde{\mathcal{A}}$ can be written as $\tilde{\mathcal{A}} = (m, \Sigma, \tilde{\mu}, \tilde{\alpha}, \tilde{\gamma})$ with $\tilde{\gamma} = \tilde{F} \cdot \gamma$,

$$\tilde{\alpha} \cdot \tilde{F} \cdot B = \alpha \cdot B \qquad \text{and} \tag{3}$$

$$\tilde{\mu}(\sigma) \cdot \tilde{F} \cdot B = \tilde{F} \cdot \mu(\sigma) \cdot B \qquad \text{for every } \sigma \in \Sigma. \tag{4}$$

First, we show that $\tilde{\mathcal{A}}$ is a well-defined automaton:

Lemma 8. *There exists a unique vector $\tilde{\alpha}$ satisfying Equation (3). For every symbol $\sigma \in \Sigma$, there exists a unique matrix $\tilde{\mu}(\sigma)$ satisfying Equation (4).*

Proof. Since the rows of $\tilde{F} \cdot B$ form a basis of $RS(F \cdot B)$, it suffices to prove that $\alpha \cdot B \in RS(F \cdot B)$ and $RS(\tilde{F} \cdot \mu(\sigma) \cdot B) \subseteq RS(F \cdot B)$ for every $\sigma \in \Sigma$. By a general linear-algebra argument (see [21]), it further suffices to prove that $\alpha \in RS(F)$ and $RS(\tilde{F} \cdot \mu(\sigma)) \subseteq RS(F)$ for every $\sigma \in \Sigma$.

We have $\alpha = \alpha \cdot \mu(\varepsilon) \in \mathcal{F} = RS(F)$. Let $i \in [m]$. Since $\tilde{F}_i \in RS(F) = \mathcal{F}$, it follows from Proposition 4 (a) that $(\tilde{F} \cdot \mu(\sigma))_i = \tilde{F}_i \cdot \mu(\sigma) \in \mathcal{F}$ for all $\sigma \in \Sigma$. ■

We now show that the automaton $\tilde{\mathcal{A}}$ minimises $\mathcal{A}$:

Lemma 9. *Automaton $\tilde{\mathcal{A}}$ is a minimal MWA equivalent to $\mathcal{A}$.*

Proof. First, we show that $\tilde{\alpha}\tilde{\mu}(w)\tilde{F}B = \alpha\mu(w)B$ for every $w \in \Sigma^*$. Our proof is by induction on the length of w. For the base case, we have $w = \varepsilon$ and by definition of $\tilde{\mathcal{A}}$ it holds that $\tilde{\alpha}\tilde{\mu}(\varepsilon)\tilde{F}B = \tilde{\alpha}\tilde{F}B = \alpha B = \alpha\mu(\varepsilon)B$.

For the induction step, let $l \in \mathbb{N}_0$ and assume that $\tilde{\alpha}\tilde{\mu}(w)\tilde{F}B = \alpha\mu(w)B$ holds for every $w \in \Sigma^l$. Take any $w \in \Sigma^l$ and $\sigma \in \Sigma$. For every $b \in \mathcal{B}$ we have by Proposition 5 (a) that $\mu(\sigma)b \in \mathcal{B}$, and thus by the induction hypothesis

$$\tilde{\alpha}\tilde{\mu}(w\sigma)\tilde{F}b = \tilde{\alpha}\tilde{\mu}(w)\tilde{\mu}(\sigma)\tilde{F}b \overset{\text{Eq. (4)}}{=} \tilde{\alpha}\tilde{\mu}(w)\tilde{F}\mu(\sigma)b = \alpha\mu(w)\mu(\sigma)b = \alpha\mu(w\sigma)b,$$

which completes the induction. Now for any $w \in \Sigma^*$, since $\gamma \in \mathcal{B}$ we have

$$\|\tilde{\mathcal{A}}\|(w) = \tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{\gamma} = \tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{F} \cdot \gamma = \alpha \cdot \mu(w) \cdot \gamma = \|\mathcal{A}\|(w).$$

Hence, automata $\tilde{\mathcal{A}}$ and $\mathcal{A}$ are equivalent. Minimality follows from Lemma 6. ■

By a result of Bozapalidis and Alexandrakis [5, Proposition 4], all equivalent minimal MTAs are equal up to a change of basis. Thus the MTA $\tilde{\mathcal{A}}$ is "canonical" in the sense that any minimal MTA equivalent to $\mathcal{A}$ can be obtained from $\tilde{\mathcal{A}}$ via a linear transformation: any m-dimensional MTA $\tilde{\mathcal{A}}' = (m, \Sigma, \tilde{\mu}', \tilde{\gamma}')$ is equivalent to $\mathcal{A}$ if and only if there exists an invertible matrix $U \in \mathbb{F}^{m \times m}$ such that $\tilde{\gamma}' = U \cdot \tilde{\gamma}$ and $\tilde{\mu}'(\sigma) = U^{\otimes rk(\sigma)} \cdot \tilde{\mu}(\sigma) \cdot U^{-1}$ for every $\sigma \in \Sigma$.

3.3 Spanning Sets for the Forward and Backward Spaces

The minimal automaton $\tilde{\mathcal{A}}$ from Section 3.2 is defined in terms of matrices F and B whose rows and columns span the forward space $\mathcal{F}$ and the backward space $\mathcal{B}$, respectively. In fact, the central algorithmic challenge for minimisation lies in the efficient computation of those matrices. In this section we prove a key proposition, Proposition 10 below, suggesting a way to compute F and B, which we exploit in Sections 4.2 and 5.

Propositions 4 and 5 and their proofs already suggest an efficient algorithm for iteratively computing bases of $\mathcal{F}$ and $\mathcal{B}$. We make this algorithm more explicit and analyse its unit-cost complexity in Section 4.1. The drawback of the resulting algorithm will be the use of "if-conditionals": the algorithm branches according to whether certain sets of vectors are linearly independent. Such conditionals are ill-suited for efficient *parallel* algorithms and also for many-one reductions. Thus it cannot be used for an NC-algorithm nor for a reduction to ACIT.

The following proposition exhibits polynomial-size sets of spanning vectors for $\mathcal{F}$ and $\mathcal{B}$, which, as we will see later, can be computed efficiently without branching. The proposition is based on the *product* automaton $\mathcal{A} \times \mathcal{A}$ defined

in Proposition 3. It defines a sequence $(f(l))_{l\in\mathbb{N}}$ of row vectors and a sequence $(b(l))_{l\in\mathbb{N}}$ of square matrices. Part (a) states that the vector $f(n)$ and the matrix $b(n)$ determine matrices F and B, whose rows and columns span $\mathcal{F}$ and $\mathcal{B}$, respectively. Part (b) gives a recursive characterization of the sequences $(f(l))_{l\in\mathbb{N}}$ and $(b(l))_{l\in\mathbb{N}}$. This allows for an efficient computation of $f(n)$ and $b(n)$.

Proposition 10. *Let Σ have rank r. Let MTA $\mathcal{A}\times\mathcal{A} = (n^2, \Sigma, \mu', \gamma^{\otimes 2})$ be the product of $\mathcal{A}$ by $\mathcal{A}$. For every $l \in \mathbb{N}$, define $f(l) := \sum_{t\in T_\Sigma^{\le l}} \mu'(t) \in \mathbb{F}^{1\times n^2}$ and $b(l) := \sum_{c\in C_{\Sigma,T_\Sigma^{\le n}}^{<l}} \mu'(c) \in \mathbb{F}^{n^2\times n^2}$.*

(a) Let $F \in \mathbb{F}^{n\times n}$ be the matrix with $F_{i,j} = f(n)\cdot(e_i\otimes e_j)^\top$. Let $B \in \mathbb{F}^{n\times n}$ be the matrix with $B_{i,j} = (e_i\otimes e_j)\cdot b(n)\cdot\gamma^{\otimes 2}$. Then, $RS(F) = \mathcal{F}$ and $CS(B) = \mathcal{B}$.
(b) We have $f(1) = \sum_{\sigma\in\Sigma_0}\mu'(\sigma)$, $b(1) = I_{n^2}$, and for all $l \in \mathbb{N}$:

$$f(l+1) = \sum_{k=0}^{r} f(l)^{\otimes k} \sum_{\sigma\in\Sigma_k} \mu'(\sigma)$$

$$b(l+1) = I_{n^2} + \sum_{k=1}^{r}\sum_{j=1}^{k} \left(f(n)^{\otimes(j-1)} \otimes b(l) \otimes f(n)^{\otimes(k-j)} \right) \sum_{\sigma\in\Sigma_k} \mu'(\sigma)$$

Proof (sketch). We provide a proof in [21]. Here we only prove the statement $RS(F) = \mathcal{F}$ from part (a). Let $\widehat{F} \in \mathbb{F}^{T_\Sigma^{\le n}\times[n]}$ be the matrix such that $\widehat{F}_t = \mu(t)$ for every $t \in T_\Sigma^{\le n}$. From Proposition 4 (b) it follows that $RS(\widehat{F}) = \mathcal{F}$. By a general linear-algebra argument (see [21]) we have $RS(\widehat{F}^\top\widehat{F}) = RS(\widehat{F})$ and hence $RS(\widehat{F}^\top\widehat{F}) = \mathcal{F}$. Thus in order to prove that $RS(F) = \mathcal{F}$, it suffices to show that $\widehat{F}^\top\widehat{F} = F$. Indeed, using the mixed-product property of the Kronecker product, we have for all $i, j \in [n]$:

$$(\widehat{F}^\top\widehat{F})_{i,j} = (\widehat{F}^\top)_i \cdot (\widehat{F})^j = \sum_{t\in T_\Sigma^{\le n}} \mu(t)_i \cdot \mu(t)_j = \sum_{t\in T_\Sigma^{\le n}} (\mu(t)\cdot e_i^\top)\otimes(\mu(t)\cdot e_j^\top)$$

$$= \left(\sum_{t\in T_\Sigma^{\le n}} (\mu(t)\otimes\mu(t))\right)(e_i\otimes e_j)^\top \overset{\text{Prop. 3}}{=} \left(\sum_{t\in T_\Sigma^{\le n}} \mu'(t)\right)(e_i\otimes e_j)^\top$$

$$= f(n)\cdot(e_i\otimes e_j)^\top. \qquad \blacksquare$$

Loosely speaking, Proposition 10 says that the sum over a small subset of the forward space of the product automaton encodes a spanning set of the whole forward space of the original automaton, and similarly for the backward space.

4 Minimisation Algorithms

In this section we devise algorithms for minimising a given multiplicity automaton: Section 4.1 considers general MTAs, while Section 4.2 considers MWAs. For the sake of a complexity analysis in standard models, we fix the field $\mathbb{F} = \mathbb{Q}$.

4.1 Minimisation of Multiplicity Tree Automata

In this section we describe an implementation of the algorithm implicit in Section 3.2, and analyse the number of operations. We denote by r the rank of Σ.

Step 1 "Forward". The first step is to compute a matrix F whose rows form a basis of $\mathcal{F}$. Seidl [26] outlines a saturation-based algorithm for that and proves that the algorithm takes polynomial time assuming unit-cost arithmetic. Based on Proposition 4 (a) we give in [21] an explicit version of Seidl's algorithm. This allows for the following lemma:

Lemma 11. *There is an algorithm that, given a $\mathbb{Q}$-MTA (n, Σ, μ, γ), computes a matrix F whose rows span the forward space $\mathcal{F}$. Each row of F equals $\mu(t)$ for some tree $t \in T_\Sigma^{\leq n}$. The algorithm executes $O\left(\sum_{k=0}^{r} |\Sigma_k| \cdot n^{2k+1}\right)$ operations.*

Step 2 "Backward". The next step suggested in Section 3.2 is to compute a matrix B whose columns form a basis of $\mathcal{B}$. Each row of the matrix F computed by the algorithm from Lemma 11 equals $\mu(t)$ for some tree $t \in T_\Sigma^{\leq n}$. Let S denote the set of those trees. By Proposition 5 (a) we have that $\mathcal{B}$ is the smallest vector space $V \subseteq \mathbb{Q}^n$ such that $\gamma \in V$ and $M \cdot v \in V$ for all $M \in \mathcal{M} := \{\mu(c) : c \in C^1_{\Sigma,S}\}$ and $v \in V$. Tzeng [27] shows, for an arbitrary column vector $\gamma \in \mathbb{Q}^n$ and an arbitrary finite set of matrices $\mathcal{M} \subseteq \mathbb{Q}^{n \times n}$, how to compute a basis of V in time $O(|\mathcal{M}| \cdot n^4)$. This can be improved to $O(|\mathcal{M}| \cdot n^3)$ (see, e.g., [14]). This leads to the following lemma (full proof in [21]):

Lemma 12. *Given the matrix F from Lemma 11, a matrix B whose columns span $\mathcal{B}$ can be computed with $O\left(\sum_{k=1}^{r} |\Sigma_k| \cdot (kn^{2k} + kn^{k+2})\right)$ operations.*

Step 3 "Solve". The final step suggested in Section 3.2 has two substeps. The first substep is to compute a matrix $\tilde{F} \in \mathbb{Q}^{m \times n}$, where $m = rank(F \cdot B)$ and $RS(\tilde{F} \cdot B) = RS(F \cdot B)$. Matrix $\tilde{F}$ can be computed from F by going through the rows of F one by one and including only those rows that are linearly independent of the previous rows when multiplied by B. This can be done in time $O(n^3)$, e.g., by transforming matrix $F \cdot B$ into a triangular form using Gaussian elimination.

The second substep is to compute the minimal MTA $\tilde{\mathcal{A}}$. The vector $\tilde{\gamma} = \tilde{F} \cdot \gamma$ is easy to compute. Solving Equation (2) for each $\tilde{\mu}(\sigma)$ can be done via Gaussian elimination in time $O(n^3)$, however, the bottleneck is the computation of $\tilde{F}^{\otimes k} \cdot \mu(\sigma)$ for every $\sigma \in \Sigma_k$, which takes $O\left(\sum_{k=0}^{r} |\Sigma_k| \cdot n^k \cdot n^k \cdot n\right) = O\left(\sum_{k=0}^{r} |\Sigma_k| \cdot n^{2k+1}\right)$ operations. Combining the results of this section, we get:

Theorem 13. *There is an algorithm that transforms a given $\mathbb{Q}$-MTA $\mathcal{A}$ into an equivalent minimal $\mathbb{Q}$-MTA. Assuming unit-cost arithmetic, the algorithm takes time $O\left(\sum_{k=0}^{r} |\Sigma_k| \cdot (n^{2k+1} + kn^{2k} + kn^{k+2})\right)$, which is $O\left(|\mathcal{A}|^2 \cdot r\right)$.*

4.2 Minimisation of Multiplicity Word Automata in NC

In this section we consider the problem of minimising a given $\mathbb{Q}$-MWA $\mathcal{A} = (n, \Sigma, \mu, \alpha, \gamma)$. We prove the following result:

Theorem 14. *There is an* NC *algorithm that transforms a given $\mathbb{Q}$-MWA into an equivalent minimal $\mathbb{Q}$-MWA. In particular, given a $\mathbb{Q}$-MWA and a number $d \in \mathbb{N}_0$, one can decide in* NC *whether there exists an equivalent $\mathbb{Q}$-MWA of dimension at most d.*

Theorem 14 improves on two results of [22]. First, [22, Theorem 4.2] states that deciding whether a $\mathbb{Q}$-MWA is minimal is in NC. Second, [22, Theorem 4.5] states the same thing as our Theorem 14, but with NC replaced with *randomised* NC.

Proof (of Theorem 14). The algorithm relies on Propositions 7 and 10. Let $\mathcal{A} = (n, \Sigma, \mu, \alpha, \gamma)$ be the given $\mathbb{Q}$-MWA. In the notation of Proposition 10, we have for all $l \in \mathbb{N}$ that $b(l+1) = I_{n^2} + b(l) \cdot \sum_{\sigma \in \Sigma} \mu'(\sigma)$. From here one can easily show, using an induction on l, that $b(n) = \sum_{k=0}^{n-1} \left(\sum_{\sigma \in \Sigma} \mu'(\sigma)\right)^k$. It follows for the matrix $B \in \mathbb{Q}^{n \times n}$ from Proposition 10 that for all $i, j \in [n]$:

$$B_{i,j} = (e_i \otimes e_j) \cdot \left(\sum_{k=0}^{n-1} \Big(\sum_{\sigma \in \Sigma} \mu'(\sigma) \Big)^k \right) \cdot \gamma^{\otimes 2}$$

Similarly, we have for the matrix $F \in \mathbb{Q}^{n \times n}$ from Proposition 10 and all $i, j \in [n]$:

$$F_{i,j} = \alpha^{\otimes 2} \cdot \left(\sum_{k=0}^{n-1} \Big(\sum_{\sigma \in \Sigma} \mu'(\sigma) \Big)^k \right) \cdot (e_i \otimes e_j)^\top.$$

The matrices F, B can be computed in NC since sums and matrix powers can be computed in NC [13]. Next we show how to compute in NC the matrix $\tilde{F}$, which is needed to compute the minimal $\mathbb{Q}$-MWA $\tilde{\mathcal{A}}$ from Section 3.2. Our NC algorithm includes the i^{th} row of F (i.e., F_i) in $\tilde{F}$ if and only if $rank(F_{[i],[n]} \cdot B) > rank(F_{[i-1],[n]} \cdot B)$. This can be done in NC since the rank of a matrix can be computed in NC [19]. It remains to compute $\tilde{\gamma} := \tilde{F}\gamma$ and solve Equations (3) and (4) for $\tilde{\alpha}$ and $\tilde{\mu}(\sigma)$, respectively. Both are easily done in NC. ■

5 Decision Problem

In this section we characterise the complexity of the following decision problem: Given a $\mathbb{Q}$-MTA and a number $d \in \mathbb{N}_0$, the *minimisation* problem asks whether there is an equivalent $\mathbb{Q}$-MTA of dimension at most d. We show, in Theorem 15 below, that this problem is interreducible with the ACIT problem.

The latter problem can be defined as follows. An *arithmetic circuit* is a finite directed acyclic vertex-labelled multigraph whose vertices, called *gates*, have indegree 0 or 2. Vertices of indegree 0, called *input gates*, are labelled with a non-negative integer or a variable from the set $\{x_i : i \in \mathbb{N}\}$. Vertices of indegree 2 are

labelled with one of the arithmetic operations $+$, $\times$, or $-$. One can associate, in a straightforward inductive way, each gate with the polynomial it computes. The *Arithmetic Circuit Identity Testing (*ACIT*)* problem asks, given an arithmetic circuit and a gate, whether the polynomial computed by the gate is equal to the zero polynomial. We show:

Theorem 15. *Minimisation is logspace interreducible with* ACIT*.*

We consider the lower and the upper bound separately.

Lower Bound. Given a $\mathbb{Q}$-MTA $\mathcal{A}$, the *zeroness* problem asks whether $\|\mathcal{A}\|(t) = 0$ for all trees t. Observe that $\|\mathcal{A}\|(t) = 0$ for all trees t if and only if there exists an equivalent automaton of dimension 0. Therefore, zeroness is a special case of minimisation. We prove:

Proposition 16. *There is a logspace reduction from* ACIT *to zeroness.*

This implies ACIT-hardness of minimisation.

Proof (of Proposition 16). It is shown in [24] that the *equivalence* problem for $\mathbb{Q}$-MTAs is logspace equivalent to ACIT. This problems asks, given two $\mathbb{Q}$-MTAs $\mathcal{A}_1$ and $\mathcal{A}_2$, whether $\|\mathcal{A}_1\|(t) = \|\mathcal{A}_2\|(t)$ holds for all trees t. By Proposition 3 one can reduce this problem to zeroness in logarithmic space. ■

Upper Bound. We prove:

Proposition 17. *There is a logspace reduction from minimisation to* ACIT*.*

Proof. Let $\mathcal{A} = (n, \Sigma, \mu, \gamma)$ be the $\mathbb{Q}$-MTA, and $d \in \mathbb{N}_0$ the given number. In our reduction to ACIT we allow input gates with rational labels as well as division gates. Rational numbers and division gates can be eliminated in a standard way by constructing separate gates for the numerators and denominators of the rational numbers computed by the original gates.

By Lemma 6, the dimension of a minimal automaton equivalent to $\mathcal{A}$ is $m := \mathit{rank}(F \cdot B)$ where F, B are matrices with $RS(F) = \mathcal{F}$ and $CS(B) = \mathcal{B}$. Thus we have $m \leq d$ if and only if $\mathit{rank}(F \cdot B) \leq d$. The recursive characterisation of F and B from Proposition 10 allows us to compute in logarithmic space an arithmetic circuit for $F \cdot B$. Thus, the result follows from Lemma 18 below. ■

The following lemma follows easily from the well-known NC procedure for computing matrix rank [15].

Lemma 18. *Let $M \in \mathbb{Q}^{m \times n}$. Let $d \in \mathbb{N}_0$. The problem of deciding whether $\mathit{rank}(M) \leq d$ is logspace reducible to* ACIT*.*

Proof. We have $\mathit{rank}(M) \leq d$ if and only if $\dim \mathit{ker}(M) \geq n - d$. As $\mathit{ker}(M) = \mathit{ker}(M^T M)$, this is equivalent to $\dim \mathit{ker}(M^T M) \geq n - d$. Now $M^T M$ is Hermitian, so $\dim \mathit{ker}(M^T M) \geq n - d$ if and only if the $n - d$ lowest-order coefficients of the characteristic polynomial of $M^T M$ are all zero [19]. But these coefficients are representable by arithmetic circuits with inputs from M (see [15]). ■

We emphasise that our reduction to ACIT is a many-one reduction, thanks to Proposition 10: our reduction computes only a single instance of ACIT; there are no if-conditionals.

6 Minimal Consistent Multiplicity Automaton

Fix a field $\mathbb{F}$ of characteristic 0. A natural computational problem is to compute an $\mathbb{F}$-MWA $\mathcal{A}$ of minimal dimension that is consistent with a given finite set of $\mathbb{F}$-weighted words $S = \{(w_1, r_1), \ldots, (w_m, r_m)\}$, where $w_i \in \Sigma^*$ and $r_i \in \mathbb{F}$ for every $i \in [m]$. Here *consistency* means that $\|\mathcal{A}\|(w_i) = r_i$ for every $i \in [m]$.

The above problem can be studied in the Blum-Shub-Smale model [2] of computation over a field $\mathbb{F}$. Since we wish to stay within the conventional Turing model, we consider instead a decision version of the problem, which we call *minimal consistency problem*, in which the output weights r_i are all rational numbers and we ask whether there exists an $\mathbb{F}$-MWA consistent with the set of input-output behaviours S that has dimension at most some non-negative integer bound n. We show that the minimal consistency problem is logspace equivalent to the problem of deciding the truth of first-order sentences over the field $(\mathbb{F}, +, \cdot, 0, 1)$. In case $\mathbb{F} = \mathbb{R}$ the latter problem is in PSPACE [8], whereas over $\mathbb{Q}$ decidability is open. This should be compared with the result that the problem of finding the smallest deterministic automaton consistent with a set of accepted or rejected strings is NP-complete [17].

The reduction of the minimal consistency problem to the decision problem for existential sentences is immediate. The idea is to represent an $\mathbb{F}$-MWA $\mathcal{A} = (n, \Sigma, \mu, \alpha, \gamma)$ "symbolically" by introducing separate variables for each entry of the initial weight vector α, final weight vector γ, and each transition matrix $\mu(\sigma)$, $\sigma \in \Sigma$. Then, consistency of automaton $\mathcal{A}$ with a given finite sample $S \subseteq \Sigma^* \times \mathbb{Q}$ can directly be written as an existential sentence.

Conversely, we reduce the decision problem for sentences of the form

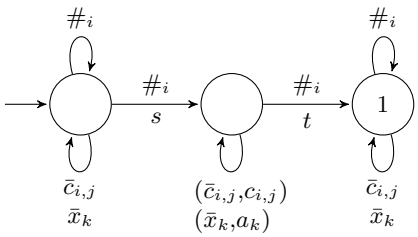
$$\exists x_1 \ldots \exists x_n \bigwedge_{i=1}^{m} f_i(x_1, \ldots, x_n) = 0\,, \tag{5}$$

where $f_i(x_1, \ldots, x_n) = \sum_{j=1}^{l_i} c_{i,j} x_1^{k_{i,j,1}} \cdots x_n^{k_{i,j,n}}$ is a polynomial with rational coefficients, to the minimal consistency problem. It suffices to consider conjunctions of positive atoms in the matrix of (5) since $f = 0 \vee g = 0$ is equivalent to $\exists x\,(x^2 - x = 0 \wedge xf = 0 \wedge (1-x)g = 0)$ and $f \neq 0$ is equivalent to $\exists x\,(fx = 1)$ for polynomials f and g.

Define an alphabet $\Sigma = \{s, t\} \cup \{\#_i, \bar{c}_{i,j}, \bar{x}_k : i \in [m], j \in [l_i], k \in [n]\}$, including symbols $\bar{c}_{i,j}$ and $\bar{x}_k$ for each coefficient $c_{i,j}$ and variable x_k respectively. Over alphabet Σ we consider the 3-dimensional $\mathbb{F}$-MWA $\mathcal{A}$, depicted in Figure 1 (b). The transitions in this automaton are annotated by label-weight pairs in $\Sigma \times \mathbb{F}$ or simply by labels from Σ, in which case the weight is assumed to be 1. Recall that the weights $c_{i,j}$ are coefficients of the polynomials f_i. For each $k \in [n]$, the weight a_k is a fixed but arbitrary element of $\mathbb{F}$.

	st	t	ε
ε	1	0	0
s	0	1	0
st	0	0	1
$\#_i$	1	1	0
$s\#_i$	0	0	1
$st\#_i$	0	0	1
$\bar{c}_{i,j}$	1	0	0
$s\bar{c}_{i,j}$	0	$c_{i,j}$	0
$st\bar{c}_{i,j}$	0	0	1
$\bar{x}_k$	1	0	0
$s\bar{x}_k$	0	a_k	0
$st\bar{x}_k$	0	0	1
t	0	0	0
stt	0	0	0
ss	0	0	0
sts	0	0	0

(a)



(b)

Fig. 1. The left figure (a) shows a Hankel-matrix fragment $\tilde{H}$, where $i \in [m]$, $j \in [l_i]$, $k \in [n]$. The right figure (b) shows a graph representation of the automaton $\mathcal{A}$.

Define $X, Y \subseteq \Sigma^*$ by $X = \{\varepsilon, s, st\}$ and $Y = \{st, t, \varepsilon\}$, and consider the fragment $\tilde{H} = H_{X \cup X\Sigma, Y}$, shown in Figure 1 (a), of the Hankel matrix H of $\mathcal{A}$. Since $rank(\tilde{H}) = 3 = rank(H)$, from Remark 2 it follows that any 3-dimensional $\mathbb{F}$-MWA $\mathcal{A}'$ that is consistent with $\tilde{H}$ is equivalent to $\mathcal{A}$.

Now for every $i \in [m]$, we encode polynomial f_i by the word

$$w_i := \#_i \bar{c}_{i,1} \bar{x}_1^{k_{i,1,1}} \dots \bar{x}_n^{k_{i,1,n}} \dots \#_i \bar{c}_{i,l_i} \bar{x}_1^{k_{i,l_i,1}} \dots \bar{x}_n^{k_{i,l_i,n}} \#_i$$

over alphabet Σ. Note that w_i comprises l_i 'blocks' of symbols, corresponding to the l_i monomials in f_i, with each block enclosed by two $\#_i$ symbols. From the definition of w_i it follows that $\|\mathcal{A}\|(w_i) = f_i(a_1, \dots, a_n)$.

Define the set $S \subseteq \Sigma^* \times \mathbb{Q}$ of weighted words as $S := S_1 \cup S_2$, where S_1 is the set of all pairs $(uv, \tilde{H}_{u,v})$ with $u \in X \cup X\Sigma$, $v \in Y$, and $uv \notin \{s\bar{x}_k t : k \in [n]\}$, and $S_2 := \{(w_i, 0) : i \in [m]\}$.

Any 3-dimensional $\mathbb{F}$-MWA $\mathcal{A}'$ consistent with S_1 is equivalent to an automaton of the form $\mathcal{A}$ for some $a_1, \dots, a_n \in \mathbb{F}$. If $\mathcal{A}'$ is moreover consistent with S_2, then $f_i(a_1, \dots, a_n) = 0$ for every $i \in [m]$. From this observation we have the following proposition (proof in [21]).

Proposition 19. *The sample S is consistent with a 3-dimensional $\mathbb{F}$-MWA if and only if the sentence (5) is true in $\mathbb{F}$.*

From Proposition 19 we derive the main result of this section:

Theorem 20. *The minimal consistency problem for $\mathbb{F}$-MWAs is logspace equivalent to the decision problem for existential first-order sentences over $\mathbb{F}$.*

Theorem 20 also holds for $\mathbb{F}$-MTAs of a fixed alphabet rank, because the minimal consistency problem can be reduced to the decision problem for existential first-order sentences over $\mathbb{F}$ in similar manner to the case for words. Here, fixing the alphabet rank keeps the reduction in polynomial time.

7 Conclusions and Future Work

We have looked at the problem of minimising a given multiplicity tree automaton from several angles. Specifically, we have analysed the complexity of *computing* a minimal automaton in the unit-cost model, of the minimisation *decision problem*, and of the *minimal consistency problem*. One of the key technical contributions of our work is Proposition 10, which, based on the product of a given automaton by itself, provides small spanning sets for forward space $\mathcal{F}$ and backward space $\mathcal{B}$. This technology also led us to an NC algorithm for minimising multiplicity *word* automata, thus improving the best previous algorithms (polynomial time and randomised NC).

It is an open question whether the complexity of the minimal consistency problem for $\mathbb{F}$-MTAs is higher if the alphabet rank is not fixed. We also plan to investigate *probabilistic* tree automata, a class that lies strictly between deterministic and multiplicity tree automata.

Acknowledgements. The authors would like to thank Michael Benedikt for stimulating discussions, and anonymous referees for their helpful suggestions. Kiefer is supported by a University Research Fellowship of the Royal Society. Marusic and Worrell gratefully acknowledge the support of the EPSRC.

References

1. Berstel, J., Reutenauer, C.: Recognizable formal power series on trees. Theoretical Computer Science 18(2), 115–148 (1982)
2. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and real computation. Springer (1997)
3. Borchardt, B.: A pumping lemma and decidability problems for recognizable tree series. Acta Cybern. 16(4), 509–544 (2004)
4. Bozapalidis, S.: Effective construction of the syntactic algebra of a recognizable series on trees. Acta Inf. 28(4), 351–363 (1991)
5. Bozapalidis, S., Alexandrakis, A.: Représentations matricielles des séries d'arbre reconnaissables. RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications 23(4), 449–459 (1989)
6. Bozapalidis, S., Louscou-Bozapalidou, O.: The rank of a formal tree power series. Theoretical Computer Science 27(1), 211–215 (1983)

7. Brainerd, W.S.: The minimalization of tree automata. Information and Control 13(5), 484–491 (1968)
8. Canny, J.: Some algebraic and geometric computations in PSPACE. In: Proceedings of STOC 1988, pp. 460–467. ACM (1988)
9. Carlyle, J.W., Paz, A.: Realizations by stochastic finite automata. Journal of Computer and System Sciences 5(1), 26–40 (1971)
10. Carme, J., Gilleron, R., Lemay, A., Terlutte, A., Tommasi, M.: Residual finite tree automata. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 171–182. Springer, Heidelberg (2003)
11. Carrasco, R.C., Daciuk, J., Forcada, M.L.: An implementation of deterministic tree automata minimization. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 122–129. Springer, Heidelberg (2007)
12. Cho, S., Huynh, D.T.: The parallel complexity of finite-state automata problems. Inf. Comput. 97(1), 1–22 (1992)
13. Cook, S.A.: A taxonomy of problems with fast parallel algorithms. Information and Control 64(1-3), 2–22 (1985)
14. Cortes, C., Mohri, M., Rastogi, A.: On the computation of some standard distances between probabilistic automata. In: Ibarra, O.H., Yen, H.-C. (eds.) CIAA 2006. LNCS, vol. 4094, pp. 137–149. Springer, Heidelberg (2006)
15. Csanky, L.: Fast parallel matrix inversion algorithms. SIAM J. Comput. 5(4), 618–623 (1976)
16. Fliess, M.: Matrices de Hankel. Journal de Mathématiques Pures et Appliquées 53, 197–222 (1974)
17. Gold, E.M.: Complexity of automaton identification from given data. Information and Control 37(3), 302–320 (1978)
18. Habrard, A., Oncina, J.: Learning multiplicity tree automata. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 268–280. Springer, Heidelberg (2006)
19. Ibarra, O.H., Moran, S., Rosier, L.E.: A note on the parallel complexity of computing the rank of order n matrices. Information Processing Letters 11(4/5), 162 (1980)
20. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. SIAM J. Comput. 22(6), 1117–1141 (1993)
21. Kiefer, S., Marusic, I., Worrell, J.: Minimisation of multiplicity tree automata. Technical report, arxiv.org (2014), `http://arxiv.org/abs/1410.535`
22. Kiefer, S., Murawski, A., Ouaknine, J., Wachter, B., Worrell, J.: On the complexity of equivalence and minimisation for $\mathbb{Q}$-weighted automata. Logical Methods in Computer Science 9(1) (2013)
23. Maletti, A.: Minimizing deterministic weighted tree automata. Inf. Comput. 207(11), 1284–1299 (2009)
24. Marusic, I., Worrell, J.: Complexity of equivalence and learning for multiplicity tree automata. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014, Part I. LNCS, vol. 8634, pp. 414–425. Springer, Heidelberg (2014)
25. Schützenberger, M.P.: On the definition of a family of automata. Information and Control 4(2-3), 245–270 (1961)
26. Seidl, H.: Deciding equivalence of finite tree automata. SIAM J. Comput. 19(3), 424–437 (1990)
27. Tzeng, W.-G.: A polynomial-time algorithm for the equivalence of probabilistic automata. SIAM J. Comput. 21(2), 216–227 (1992)