Logical Methods in Computer Science
Vol. 13(1:16)2017, pp. 1–25
www.lmcs-online.org

Submitted Dec. 2, 2015
Published Mar. 28, 2017

MINIMISATION OF MULTIPLICITY TREE AUTOMATA [*]

(No equivalence)

STEFAN KIEFER, INES MARUŠIĆ, AND JAMES WORRELL

University of Oxford, UK
e-mail address: {stefan.kiefer, ines.marusic, james.worrell}@cs.ox.ac.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 a given number of states that is consistent with a given finite sample of weight-labelled words or trees? We show that, over both words and trees, this decision problem is interreducible with the problem of deciding the truth of existential first-order sentences over the field of rationals—whose decidability is a longstanding open problem.

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. Minimisation of multiplicity and weighted automata has numerous applications including image compression [1] and reducing the space complexity of speech recognition tasks [29, 19].

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 [4] under the terminology of linear representations of a tree series. They generalise multiplicity word automata, introduced by Schützenberger [32], which can be viewed as multiplicity tree

Key words and phrases: weighted automata, tree automata, minimisation, arithmetic circuit identity testing, consistent automaton.

[*] This is a full and improved version of the FoSSaCS'15 paper with the same title. The current paper contains full proofs of all results reported there and complete definitions of all the minimisation algorithms.

LOGICAL METHODS IN COMPUTER SCIENCE

DOI:10.23638/LMCS-13(1:16)2017

automata on unary trees. The minimisation problem for multiplicity word automata has long been known to be solvable in polynomial time (in the Turing model) [32, 35].

In this work, we give a new procedure for computing minimal multiplicity word automata and thereby place minimisation in NC, improving also on a randomised NC procedure in [25]. (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 parallel random-access machines with polynomially many processors.) By comparison, it is known that minimising deterministic word automata is NL-complete [14], while minimising non-deterministic word automata is PSPACE-complete [24]. The latter result shows, in particular, that the bounds obtained in this paper over $\mathbb{Q}$ do not apply to weighted automata over an arbitrary semi-ring, because non-deterministic automata can be viewed as weighted automata over the Boolean semi-ring.

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 multiplicity tree automaton $\mathcal{A}$ in time $O\left(|\mathcal{A}|^2 \cdot r\right)$, where $|\mathcal{A}|$ is the size of $\mathcal{A}$ and 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 by Bozapalidis [6]. We thus place the problem within PSPACE in the conventional Turing model, since a polynomial-time decidable problem in the unit-cost model lies in PSPACE (see, e.g., [2]). We are moreover able to precisely characterise the complexity of the decision version of the minimisation problem, showing that it is logspace equivalent to the *arithmetic circuit identity testing (* ACIT*)* problem, commonly also called the *polynomial identity testing* problem. As far as we can tell, obtaining this complexity bound requires departing from the framework of Bozapalidis [6]. The ACIT problem is very well studied, with a variety of randomised polynomial-time algorithms [18, 33, 36], but, as yet, no deterministic polynomial-time procedure (see [3]). In previous work we have reduced equivalence testing of multiplicity tree automata to ACIT [28]; the advance here is to reduce the more general problem of minimisation also to ACIT.

More general problems are harder, so this is not so much of an advancement…?

Lastly, 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 deterministic finite automata was studied by Gold [21], who showed that the problem of exactly identifying the smallest deterministic finite automaton consistent with a set of accepted and rejected words is NP-hard. For multiplicity word automata over the field $\mathbb{Q}$, we show that the decision version of this problem, which we call the *minimal consistency problem*, is logspace equivalent to the problem of deciding the truth of existential first-order sentences over the structure $(\mathbb{Q}, +, \cdot, 0, 1)$, a longstanding open problem (see [31]). We observe that, by contrast, the minimal consistency problem for multiplicity word automata over the field $\mathbb{R}$ is in PSPACE, and likewise for multiplicity tree automata over $\mathbb{R}$ that have a fixed alphabet rank.

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 [9, 13]. There have also been several works on minimising deterministic tree automata with weights in a semi-field (*i.e.*, a semi-ring with multiplicative inverses). In particular, Maletti [27] 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.* [12] 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 nonnegative integers, respectively. For every $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, 2, \dots, n\}$ and 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. We write $\mathbf{0}_n$ for the n-dimensional zero 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.

Given a field $\mathbb{F}$ and 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.

2.1. **Row and Column Spaces.**

Let $\mathbb{F}$ be either the field of rationals $\mathbb{Q}$ or the field of reals $\mathbb{R}$. Let A be an $m \times n$ matrix with entries in $\mathbb{F}$. The *row space* of A, written as $RS(A)$, is the subspace of $\mathbb{F}^n$ spanned by the rows of A. The *column space* of A, written as $CS(A)$, is the subspace of $\mathbb{F}^m$ spanned by the columns of A. That is, $RS(A) = \langle v \cdot A : v \in \mathbb{F}^m \rangle$ and $CS(A) = \langle A \cdot v^\top : v \in \mathbb{F}^n \rangle$.

The following Lemmas 2.1-2.3 contain some basic results about row and column spaces that we will use in this paper.

Lemma 2.1. *Let A_1, A_2 be matrices such that $RS(A_1) \subseteq RS(A_2)$. For any matrix B such that $A_1 \cdot B$ (and thus also $A_2 \cdot B$) is defined, we have that*

$$RS(A_1 \cdot B) \subseteq RS(A_2 \cdot B).$$

Proof. Suppose $A_1 \in \mathbb{F}^{m_1 \times n}$ and $A_2 \in \mathbb{F}^{m_2 \times n}$. For every vector $v_1 \in \mathbb{F}^{m_1}$, it holds that $v_1 \cdot A_1 \in RS(A_1) \subseteq RS(A_2)$. Hence, there exists a vector $v_2 \in \mathbb{F}^{m_2}$ such that $v_1 \cdot A_1 = v_2 \cdot A_2$. Thus

$$\begin{aligned} RS(A_1 \cdot B) &= \langle v_1 \cdot A_1 \cdot B : v_1 \in \mathbb{F}^{m_1} \rangle \\ &\subseteq \langle v_2 \cdot A_2 \cdot B : v_2 \in \mathbb{F}^{m_2} \rangle = RS(A_2 \cdot B), \end{aligned}$$

which completes the proof. □

Lemma 2.2. *For any matrix $A \in \mathbb{F}^{m \times n}$, it holds that $RS(A^\top A) = RS(A)$.*

Proof. For any $x \in \mathbb{F}^n$ such that $(A^\top A)x^\top = \mathbf{0}_n^\top$ we have

$$(Ax^\top)^\top A x^\top \;=\; x A^\top A x^\top \;=\; x \mathbf{0}_n^\top \;=\; 0\,,$$

and hence $Ax^\top = \mathbf{0}_m^\top$. Conversely, for any $x \in \mathbb{F}^n$ with $Ax^\top = \mathbf{0}_m^\top$ we have $(A^\top A)x^\top = \mathbf{0}_n^\top$. Therefore, matrices A and $A^\top A$ have the same null space and hence the same row space. □

Lemma 2.3. *Let A_1, A_2, B_1, B_2 be matrices of dimension $n_1 \times m$, $n_2 \times m$, $m \times n_3$, $m \times n_4$, respectively. If $RS(A_1) = RS(A_2)$ and $CS(B_1) = CS(B_2)$, then*

$$rank(A_1 \cdot B_1) = rank(A_2 \cdot B_2).$$

Proof. By definition of rank as the dimension of row or column space, we have

$$\begin{aligned} rank(A_1 \cdot B_1) &= dim\ \langle x \cdot A_1 \cdot B_1 : x \in \mathbb{F}^{n_1} \rangle \\ &= dim\ \langle x \cdot A_2 \cdot B_1 : x \in \mathbb{F}^{n_2} \rangle && \text{(using } RS(A_1) = RS(A_2)\text{)} \\ &= dim\ \langle A_2 \cdot B_1 \cdot x^\top : x \in \mathbb{F}^{n_3} \rangle \\ &= dim\ \langle A_2 \cdot B_2 \cdot x^\top : x \in \mathbb{F}^{n_4} \rangle && \text{(using } CS(B_1) = CS(B_2)\text{)} \\ &= rank(A_2 \cdot B_2). \end{aligned}$$

This completes the proof. □

2.2. 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 $k \geq 1$.

Let $k \in \mathbb{N}$, and let $n_1, \ldots, n_k \in \mathbb{N}$. Suppose A is a matrix with $n_1 \cdot \ldots \cdot n_k$ rows. For every $(i_1, \ldots, i_k) \in [n_1] \times \cdots \times [n_k]$, we use $A_{(i_1,\ldots,i_k)}$ to denote the $(\sum_{l=1}^{k-1} (i_l - 1) \cdot (\prod_{p=l+1}^{k} n_p) + i_k)^{\text{th}}$ row of A. Let $A_1, \ldots, A_k$ be matrices such that for every $l \in [k]$, A_l has n_l rows. It can easily be shown using induction on k that for every $(i_1, \ldots, i_k) \in [n_1] \times \cdots \times [n_k]$,

$$(A_1 \otimes \cdots \otimes A_k)_{(i_1,\ldots,i_k)} = (A_1)_{i_1} \otimes \cdots \otimes (A_k)_{i_k}. \tag{2.1}$$

We write $\bigotimes_{l=1}^{k} A_l := A_1 \otimes \cdots \otimes A_k$.

For any $k \in \mathbb{N}_0$ and matrices $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ where product $A_l \cdot B_l$ is defined for every $l \in [k]$, we have

$$(A_1 \otimes \cdots \otimes A_k) \cdot (B_1 \otimes \cdots \otimes B_k) = (A_1 \cdot B_1) \otimes \cdots \otimes (A_k \cdot B_k). \tag{2.2}$$

This follows easily from the mixed-product property by induction on k.

2.3. 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 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 defining $\mu(\varepsilon) := I_n$, and $\mu(\sigma_1 \cdots \sigma_k) := \mu(\sigma_1) \cdot \ldots \cdot \mu(\sigma_k)$ for any $\sigma_1, \ldots, \sigma_k \in \Sigma$. 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 $w \in \Sigma^*$.

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

Let $\Box$ be a nullary symbol not contained in Σ. The set C_Σ of Σ-*contexts* (*contexts* for short) is the set of all $(\{\Box\} \cup \Sigma)$-trees in which $\Box$ occurs exactly once. Let $n \in \mathbb{N}_0$. We denote by C^n_Σ 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 $C^{\leq n}_\Sigma := \bigcup_{l=0}^n C^l_\Sigma$ and $C^{<n}_\Sigma := C^{\leq n}_\Sigma \setminus C^n_\Sigma$. 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 $C^n_{\Sigma,S}$ the set of all contexts $c \in C^n_\Sigma$ where every subtree of c is an element of S. Moreover, we write $C^{\leq n}_{\Sigma,S} := \bigcup_{l=0}^n C^l_{\Sigma,S}$ and $C^{<n}_{\Sigma,S} := C^{\leq n}_{\Sigma,S} \setminus C^n_{\Sigma,S}$.

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 $c \in C_\Sigma$.

2.5. **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 defining

$$\mu(\sigma(t_1, \ldots, t_k)) := (\mu(t_1) \otimes \cdots \otimes \mu(t_k)) \cdot \mu(\sigma)$$

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 [22], although a closely-related result was given by Bozapalidis and Louscou-Bozapalidou [8].

Theorem 2.4 ([8, 22])**.** *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 2.4 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 2.5. Theorem 2.4 specialised to word automata was proved by Carlyle and Paz [11] and Fliess [20]. 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 $H_{X\Sigma,Y}$.

In the remainder of this section, we prove some closure properties for MTAs. First, we give two definitions: the product and the difference of two $\mathbb{F}$-MTAs. Let $\mathcal{A}_1 = (n_1, \Sigma, \mu_1, \gamma_1)$ and $\mathcal{A}_2 = (n_2, \Sigma, \mu_2, \gamma_2)$ be two $\mathbb{F}$-multiplicity tree automata. The *difference* of $\mathcal{A}_1$ and $\mathcal{A}_2$, written as $\mathcal{A}_1 - \mathcal{A}_2$, is the $\mathbb{F}$-multiplicity tree automaton (n, Σ, μ, γ) where:

- $n = n_1 + n_2$;
- For every $\sigma \in \Sigma$ and any $i \in [(n_1 + n_2)^{rk(\sigma)}]$, $j \in [n_1 + n_2]$,
$$\mu(\sigma)_{i,j} = \begin{cases} \mu_1(\sigma)_{i,j} & \text{if } i \le n_1^{rk(\sigma)} \text{and } j \le n_1 \\ \mu_2(\sigma)_{i,j} & \text{if } i > (n_1+n_2)^{rk(\sigma)} - n_2^{rk(\sigma)} \text{and } j > n_1 \\ 0 & \text{otherwise;} \end{cases}$$
- $\gamma = \begin{bmatrix} \gamma_1 \\ -\gamma_2 \end{bmatrix}$.

The *product* of $\mathcal{A}_1$ by $\mathcal{A}_2$, written as $\mathcal{A}_1 \times \mathcal{A}_2$, is the $\mathbb{F}$-multiplicity tree automaton (n, Σ, μ, γ) where:

- $n = n_1 \cdot n_2$;
- For every $\sigma \in \Sigma_k$, $\mu(\sigma) = P_k \cdot (\mu_1(\sigma) \otimes \mu_2(\sigma))$ where P_k is a permutation matrix of order $(n_1 \cdot n_2)^k$ uniquely defined (see Remark 2.6 below) by
$$(u_1 \otimes \cdots \otimes u_k) \otimes (v_1 \otimes \cdots \otimes v_k) = ((u_1 \otimes v_1) \otimes \cdots \otimes (u_k \otimes v_k)) \cdot P_k \tag{2.3}$$
for all $u_1, \ldots, u_k \in \mathbb{F}^{1\times n_1}$ and $v_1, \ldots, v_k \in \mathbb{F}^{1\times n_2}$;
- $\gamma = \gamma_1 \otimes \gamma_2$.

Remark 2.6. In the following we argue that for every k, matrix P_k is well-defined by Equation (2.3). To do this, it suffices to show that P_k is well-defined on a set of basis vectors of $\mathbb{F}^{1\times n_1}$ and $\mathbb{F}^{1\times n_2}$ and then extend linearly. To that end, let $(e^1_i)_{i\in[n_1]}$ and $(e^2_j)_{j\in[n_2]}$ be bases of $\mathbb{F}^{1\times n_1}$ and $\mathbb{F}^{1\times n_2}$, respectively. Then
$$E_1 := \{(e^1_{i_1} \otimes \cdots \otimes e^1_{i_k}) \otimes (e^2_{j_1} \otimes \cdots \otimes e^2_{j_k}) : i_1, \ldots, i_k \in [n_1], j_1, \ldots, j_k \in [n_2]\}$$

and

$$E_2 := \{(e^1_{i_1} \otimes e^2_{j_1}) \otimes \cdots \otimes (e^1_{i_k} \otimes e^2_{j_k}) : i_1, \ldots, i_k \in [n_1], j_1, \ldots, j_k \in [n_2]\}$$

are two bases of the vector space $\mathbb{F}^{1\times n_1 n_2}$. Therefore, P_k is well-defined as an invertible matrix mapping basis E_1 to basis E_2.

We now turn to the closure properties for MTAs:

Proposition 2.7. *Let $\mathcal{A}_1 = (n_1, \Sigma, \mu_1, \gamma_1)$ and $\mathcal{A}_2 = (n_2, \Sigma, \mu_2, \gamma_2)$ be two $\mathbb{F}$-MTAs. For their difference $\mathcal{A}_1 - \mathcal{A}_2$, it holds that $\|\mathcal{A}_1 - \mathcal{A}_2\| = \|\mathcal{A}_1\| - \|\mathcal{A}_2\|$. For their product $\mathcal{A}_1 \times \mathcal{A}_2 = (n, \Sigma, \mu, \gamma)$, the following properties hold:*

(i) *for every $t \in T_\Sigma$, $\mu(t) = \mu_1(t) \otimes \mu_2(t)$;*
(ii) *for every $c \in C_\Sigma$, $\mu(c) = \mu_1(c) \otimes \mu_2(c)$;*
(iii) $\|\mathcal{A}_1 \times \mathcal{A}_2\| = \|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|$.

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.

Proof. The result for the difference automaton is shown in [4, Proposition 3.1]. Results (i) and (iii) for the product automaton are shown in [4, Proposition 5.1]; see also [5]. In the following we prove the remainder of the proposition.

We prove result (ii) using induction on the distance between the root and the $\square$-labelled node of c. The base case is $c = \square$. Here by definition we have that

$$\mu(c) = \mu(\square) = I_{n_1 \cdot n_2} = I_{n_1} \otimes I_{n_2} = \mu_1(\square) \otimes \mu_2(\square) = \mu_1(c) \otimes \mu_2(c).$$

For the induction step, let $h \in \mathbb{N}_0$ and assume that (ii) holds for every context $c \in C^h_\Sigma$. Take any $c \in C^{h+1}_\Sigma$. Without loss of generality we can assume that $c = \sigma(c_1, t_2, \ldots, t_k)$ for some $k \geq 1$, $\sigma \in \Sigma_k$, $c_1 \in C^h_\Sigma$, and $t_2, \ldots, t_k \in T_\Sigma$. By the induction hypothesis, result (i), Equation (2.3), and the mixed-product property of Kronecker product, we now have

$$\begin{aligned}
\mu(c) &= \left(\mu(c_1) \otimes \bigotimes_{j=2}^{k} \mu(t_j)\right) \cdot \mu(\sigma) \\
&= \left((\mu_1(c_1) \otimes \mu_2(c_1)) \otimes \bigotimes_{j=2}^{k} (\mu_1(t_j) \otimes \mu_2(t_j))\right) \cdot P_k \cdot (\mu_1(\sigma) \otimes \mu_2(\sigma)) \\
&= \left(\left(\mu_1(c_1) \otimes \bigotimes_{j=2}^{k} \mu_1(t_j)\right) \otimes \left(\mu_2(c_1) \otimes \bigotimes_{j=2}^{k} \mu_2(t_j)\right)\right) \cdot (\mu_1(\sigma) \otimes \mu_2(\sigma)) \\
&= \left(\left(\mu_1(c_1) \otimes \bigotimes_{j=2}^{k} \mu_1(t_j)\right) \cdot \mu_1(\sigma)\right) \otimes \left(\left(\mu_2(c_1) \otimes \bigotimes_{j=2}^{k} \mu_2(t_j)\right) \cdot \mu_2(\sigma)\right) \\
&= \mu_1(c) \otimes \mu_2(c).
\end{aligned}$$

This completes the proof of result (ii) by induction.

Now let $\mathbb{F} = \mathbb{Q}$. The $\mathbb{Q}$-MTA $\mathcal{A}_1 \times \mathcal{A}_2$ can be computed using a deterministic Turing machine which scans the transition matrices and the final weight vectors of $\mathcal{A}_1$ and $\mathcal{A}_2$, and then writes down the entries of the transition matrices and the final weight vector of their product $\mathcal{A}_1 \times \mathcal{A}_2$ onto the output tape. This computation requires maintaining only a constant number of pointers, which takes logarithmic space in the representation of automata $\mathcal{A}_1$ and $\mathcal{A}_2$. Hence, the Turing machine computing the automaton $\mathcal{A}_1 \times \mathcal{A}_2$ uses logarithmic space in the work tape. Analogously, the $\mathbb{Q}$-MTA $\mathcal{A}_1 - \mathcal{A}_2$ 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 the 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 3.1 and 3.2 provide fundamental characterisations of $\mathcal{F}$ and $\mathcal{B}$, respectively.

Proposition 3.1. *The forward space $\mathcal{F}$ has the following properties:*

(a) *The forward 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 $\{\mu(t) : t \in T_\Sigma^{<n}\}$ spans $\mathcal{F}$.*

Proof. We start by proving result (a). Here we first show that $\mathcal{F}$ has the closure property stated in (a). To this end, let us take any $k \in \mathbb{N}_0$, $v_1, \ldots, v_k \in \mathcal{F}$, and $\sigma \in \Sigma_k$. By definition of the forward space $\mathcal{F}$, for every $i \in [k]$ we can express vector $v_i \in \mathcal{F}$ as

$$v_i = \sum_{j_i=1}^{m_i} \alpha^i_{j_i} \mu(t^i_{j_i})$$

for some integer $m_i \in \mathbb{N}$, scalars $\alpha^i_1, \ldots, \alpha^i_{m_i} \in \mathbb{F}$, and trees $t^i_1, \ldots, t^i_{m_i} \in T_\Sigma$. From here, using bilinearity of Kronecker product we get that

$$\begin{aligned}(v_1 \otimes \cdots \otimes v_k) \cdot \mu(\sigma) &= \left(\left(\sum_{j_1=1}^{m_1} \alpha^1_{j_1} \mu(t^1_{j_1}) \right) \otimes \cdots \otimes \left(\sum_{j_k=1}^{m_k} \alpha^k_{j_k} \mu(t^k_{j_k}) \right) \right) \cdot \mu(\sigma) \\ &= \sum_{j_1=1}^{m_1} \cdots \sum_{j_k=1}^{m_k} \alpha^1_{j_1} \cdots \alpha^k_{j_k} \left(\mu(t^1_{j_1}) \otimes \cdots \otimes \mu(t^k_{j_k}) \right) \cdot \mu(\sigma) \\ &= \sum_{j_1=1}^{m_1} \cdots \sum_{j_k=1}^{m_k} \alpha^1_{j_1} \cdots \alpha^k_{j_k} \cdot \mu(\sigma(t^1_{j_1}, \ldots, t^k_{j_k})).\end{aligned}$$

Since $\mathcal{F}$ is a vector space, the above equation implies that $(v_1 \otimes \cdots \otimes v_k) \cdot \mu(\sigma) \in \mathcal{F}$.

Let V be any vector space 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 $(v_1 \otimes \cdots \otimes v_k) \cdot \mu(\sigma) \in V$. We claim that $\mathcal{F} \subseteq V$. To prove this, it suffices to show that $\mu(t) \in V$ for every $t \in T_\Sigma$. Here we give a proof by induction on *height*(t). The base case $t \in \Sigma_0$ is trivial. For the induction step, let $h \in \mathbb{N}_0$ and assume that $\mu(t) \in V$ for all $t \in T_\Sigma^{\leq h}$. Take any $t \in T_\Sigma^{h+1}$. Then, $t = \sigma(t_1, \ldots, t_k)$ for some $k \geq 1$, $\sigma \in \Sigma_k$, and

$t_1, \dots, t_k \in T_\Sigma^{\leq h}$. The induction hypothesis now implies that $\mu(t_1), \dots, \mu(t_k) \in V$. By the choice of V, we therefore have that $\mu(t) = (\mu(t_1) \otimes \cdots \otimes \mu(t_k)) \cdot \mu(\sigma) \in V$. This completes the proof by induction.

The proof of result (b) follows from [34, Main Lemma 4.1]. □

Proposition 3.2. *Let $S \subseteq T_\Sigma$ be a set of trees such that $\{\mu(t) : t \in S\}$ spans $\mathcal{F}$. Then, the following properties hold:*

(a) *The backward space $\mathcal{B}$ is the smallest vector space V over $\mathbb{F}$ such that:*
  (1) *$\gamma \in V$.*
  (2) *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 $\{\mu(c) \cdot \gamma : c \in C^{<n}_{\Sigma,S}\}$ spans $\mathcal{B}$.*

Proof. First, we prove result (a). We have that $\gamma = \mu(\Box) \cdot \gamma \in \mathcal{B}$, hence $\mathcal{B}$ satisfies property 1. To see that $\mathcal{B}$ satisfies property 2, let us take any $v \in \mathcal{B}$ and $c \in C^1_{\Sigma,S}$. By definition of $\mathcal{B}$, the vector v can be expressed as

$$v = \sum_{i=1}^{m} \alpha_i \cdot \mu(c_i) \cdot \gamma$$

for some integer $m \in \mathbb{N}$, scalars $\alpha_1, \dots, \alpha_m \in \mathbb{F}$, and contexts $c_1, \dots, c_m \in C_\Sigma$. Thus by bilinearity of matrix multiplication we have

$$\begin{aligned}\mu(c) \cdot v &= \mu(c) \cdot \left(\sum_{i=1}^{m} \alpha_i \cdot \mu(c_i) \cdot \gamma \right) \\ &= \sum_{i=1}^{m} \alpha_i \cdot (\mu(c) \cdot \mu(c_i) \cdot \gamma) = \sum_{i=1}^{m} \alpha_i \cdot \mu(c_i[c]) \cdot \gamma,\end{aligned}$$

which implies that $\mu(c) \cdot v \in \mathcal{B}$ since $\mathcal{B}$ is a vector space. Therefore, $\mathcal{B}$ satisfies properties 1 and 2.

Let now V be any vector space over $\mathbb{F}$ satisfying properties 1 and 2. In order to show that $\mathcal{B} \subseteq V$, it suffices to show that $\mu(c) \cdot \gamma \in V$ for every $c \in C_\Sigma$. We prove the latter result using induction on the distance between the root and the $\Box$-labelled node of c. For the induction basis, let the distance be 0, *i.e.*, $c = \Box$. Then we have $\mu(c) \cdot \gamma = \gamma \in V$ by property 1. For the induction step, let $h \in \mathbb{N}_0$ and assume that $\mu(c) \cdot \gamma \in V$ for all $c \in C_\Sigma^{\leq h}$. Take any $c \in C_\Sigma^{h+1}$. Let $c' \in C^1_\Sigma$ and $c'' \in C^h_\Sigma$ be such that $c = c''[c']$. Without loss of generality we can assume that $c' = \sigma(\Box, \tau_2, \dots, \tau_k)$ where $k \geq 1$, $\sigma \in \Sigma_k$, and $\tau_2, \dots, \tau_k \in T_\Sigma$. Since $\mathcal{F} = \langle \mu(t) : t \in S \rangle$, for every $i \in \{2, \dots, k\}$ there is an integer $m_i \in \mathbb{N}$, scalars $\alpha^i_1, \dots, \alpha^i_{m_i} \in \mathbb{F}$, and trees $t^i_1, \dots, t^i_{m_i} \in S$ such that

$$\mu(\tau_i) = \sum_{j_i=1}^{m_i} \alpha^i_{j_i} \mu(t^i_{j_i}).$$

From here, using bilinearity of Kronecker product, it follows that

$$\begin{aligned}\mu(c) \cdot \gamma &= \mu(c') \cdot \mu(c'') \cdot \gamma \\ &= (I_n \otimes \mu(\tau_2) \otimes \cdots \otimes \mu(\tau_k)) \mu(\sigma) \cdot \mu(c'') \cdot \gamma \\ &= \left(I_n \otimes \left(\sum_{j_2=1}^{m_2} \alpha^2_{j_2} \mu(t^2_{j_2}) \right) \otimes \cdots \otimes \left(\sum_{j_k=1}^{m_k} \alpha^k_{j_k} \mu(t^k_{j_k}) \right) \right) \mu(\sigma) \cdot \mu(c'') \cdot \gamma\end{aligned}$$

$$= \sum_{j_2=1}^{m_2} \cdots \sum_{j_k=1}^{m_k} \alpha^2_{j_2} \cdots \alpha^k_{j_k} \cdot \left(I_n \otimes \mu(t^2_{j_2}) \otimes \cdots \otimes \mu(t^k_{j_k}) \right) \mu(\sigma) \cdot \mu(c'') \cdot \gamma$$
$$= \sum_{j_2=1}^{m_2} \cdots \sum_{j_k=1}^{m_k} \alpha^2_{j_2} \cdots \alpha^k_{j_k} \cdot \mu(\sigma(\square, t^2_{j_2}, \ldots, t^k_{j_k})) \cdot \mu(c'') \cdot \gamma \, ,$$

where we note that $\sigma(\square, t^2_{j_2}, \ldots, t^k_{j_k}) \in C^1_{\Sigma,S}$ for every $j_2 \in [m_2], \ldots, j_k \in [m_k]$. Moreover, we have $\mu(c'') \cdot \gamma \in V$ by the induction hypothesis. Thus by property 2 we have $\mu(\sigma(\square, t^2_{j_2}, \ldots, t^k_{j_k})) \cdot \mu(c'') \cdot \gamma \in V$ for every $j_2 \in [m_2], \ldots, j_k \in [m_k]$. Since V is a vector space, we conclude that $\mu(c) \cdot \gamma \in V$. This completes the proof of result (a) by induction.

We denote by $C_{\Sigma,S}$ the set of all $c \in C_\Sigma$ where every subtree of c is an element of S. It follows easily from part (a) that $\langle \mu(c) \cdot \gamma : c \in C_{\Sigma,S} \rangle = \mathcal{B}$ since $\langle \mu(c) \cdot \gamma : c \in C_{\Sigma,S} \rangle$ satisfies properties 1 and 2. Thus in order to prove result (b), it suffices to show that the set $\{\mu(c) \cdot \gamma : c \in C^{<n}_{\Sigma,S}\}$ spans $\langle \mu(c) \cdot \gamma : c \in C_{\Sigma,S} \rangle$. We show this using an argument that was similarly given, *e.g.*, in [30]. If γ is the zero vector $\mathbf{0}_n^\top$, the statement is trivial. Let us now assume that $\gamma \neq \mathbf{0}_n^\top$. For every $i \in \mathbb{N}$, we define the vector space $\mathcal{B}^i := \langle \mu(c) \cdot \gamma : c \in C^{<i}_{\Sigma,S} \rangle$ over $\mathbb{F}$. Since $\mathcal{B}^i$ is a subspace of $\mathcal{B}^{i+1}$ for every $i \in \mathbb{N}$, we have

$$1 \;\leq\; \mathit{dim}(\mathcal{B}^1) \;\leq\; \mathit{dim}(\mathcal{B}^2) \;\leq\; \cdots \;\leq\; \mathit{dim}(\mathcal{B}^{n+1}) \;\leq\; n \, , \tag{3.1}$$

where the first inequality holds because $\gamma \neq \mathbf{0}_n^\top$, and the last inequality holds because $\mathcal{B}^i \subseteq \mathbb{F}^n$ for all $i \in \mathbb{N}$. Not all inequalities in the inequality chain (3.1) can be strict, so we must have $\mathcal{B}^{i_0} = \mathcal{B}^{i_0+1}$ for some $i_0 \in [n]$. We claim that $\mathcal{B}^i = \mathcal{B}^{i+1}$ for all $i \geq i_0$. We give a proof by induction on i. The base case $i = i_0$ holds by definition of i_0. For the induction step, let $i \geq i_0$ and assume that $\mathcal{B}^i = \mathcal{B}^{i+1}$. Note that, by definition, for all $j \in \mathbb{N}$ we have $\mathcal{B}^{j+1} = \langle \gamma, \mu(c) \cdot \mathcal{B}^j : c \in C^1_{\Sigma,S} \rangle$. Using this result for $j \in \{i, i+1\}$, we obtain:

$$\mathcal{B}^{i+1} \;=\; \langle \gamma, \mu(c) \cdot \mathcal{B}^i : c \in C^1_{\Sigma,S} \rangle \;=\; \langle \gamma, \mu(c) \cdot \mathcal{B}^{i+1} : c \in C^1_{\Sigma,S} \rangle \;=\; \mathcal{B}^{i+2}$$

where the middle equation holds by the induction hypothesis. This completes the proof by induction, and we thus conclude that $\mathcal{B}^i = \mathcal{B}^{i+1}$ for all $i \geq i_0$. Since $n \geq i_0$, it follows that $\mathcal{B}^n = \bigcup_{i \geq n} \mathcal{B}^i$. Since $(\mathcal{B}^i)_{i \in \mathbb{N}}$ is an increasing sequence of vector spaces, we have $\mathcal{B}^n = \bigcup_{i \in \mathbb{N}} \mathcal{B}^i = \langle \mu(c) \cdot \gamma : c \in C_{\Sigma,S} \rangle$ as required. □

3.2. **A Minimal Automaton.** Let F and B be matrices whose rows and columns, respectively, span $\mathcal{F}$ and $\mathcal{B}$. 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 $\mathit{rank}(F \cdot B)$ is the dimension of a minimal automaton equivalent to $\mathcal{A}$.

Lemma 3.3. *A minimal automaton equivalent to $\mathcal{A}$ has $m := \mathit{rank}(F \cdot B)$ 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

$$RS(\overline{F}) = \mathcal{F} = RS(F) \qquad \text{and} \qquad CS(\overline{B}) = \mathcal{B} = CS(B) \, . \tag{3.2}$$

We now have $m = \mathit{rank}(H) = \mathit{rank}(\overline{F} \cdot \overline{B}) = \mathit{rank}(F \cdot B)$, where the first equality holds by Theorem 2.4 and the last equality holds by (3.2) and Lemma 2.3. □

Since $m = \mathit{rank}(F \cdot B)$, 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

$$\tilde{\mu}(\sigma) \cdot \tilde{F} \cdot B = \tilde{F}^{\otimes k} \cdot \mu(\sigma) \cdot B \qquad \text{for every } \sigma \in \Sigma_k. \tag{3.3}$$

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

Proposition 3.4. *The MTA $\tilde{\mathcal{A}}$ is well-defined and is a minimal automaton equivalent to $\mathcal{A}$.*

Before giving a full proof of Proposition 3.4 later in this subsection, we now prove this result for multiplicity *word* automata, stated as Proposition 3.5 below, which will be used in Section 4.2. The main arguments are similar for the tree case, but slightly more involved.

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 $\tilde{\gamma} = \tilde{F} \cdot \gamma$,

$$\tilde{\alpha} \cdot \tilde{F} \cdot B = \alpha \cdot B, \qquad \text{and} \tag{3.4}$$

$$\tilde{\mu}(\sigma) \cdot \tilde{F} \cdot B = \tilde{F} \cdot \mu(\sigma) \cdot B \qquad \text{for every } \sigma \in \Sigma. \tag{3.5}$$

Proposition 3.5. *The MWA $\tilde{\mathcal{A}}$ is well-defined and is a minimal automaton equivalent to $\mathcal{A}$.*

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

Lemma 3.6. *There exists a unique vector $\tilde{\alpha}$ satisfying Equation (3.4). For every $\sigma \in \Sigma$, there exists a unique matrix $\tilde{\mu}(\sigma)$ satisfying Equation (3.5).*

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 Lemma 2.1, 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 3.1 (a) that $(\tilde{F} \cdot \mu(\sigma))_i = \tilde{F}_i \cdot \mu(\sigma) \in \mathcal{F}$ for all $\sigma \in \Sigma$. □

We complete the proof of Proposition 3.5 by showing that MWA $\tilde{\mathcal{A}}$ minimises $\mathcal{A}$:

Lemma 3.7. *The automaton $\tilde{\mathcal{A}}$ is a minimal MWA equivalent to $\mathcal{A}$.*

Proof. We claim that for every $w \in \Sigma^*$,

$$\tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{F} \cdot B = \alpha \cdot \mu(w) \cdot B. \tag{3.6}$$

Our proof is by induction on $|w|$. For the base case $w = \varepsilon$, we have

$$\tilde{\alpha} \cdot \tilde{\mu}(\varepsilon) \cdot \tilde{F} \cdot B = \tilde{\alpha} \cdot \tilde{F} \cdot B \overset{\text{Eq. (3.4)}}{=} \alpha \cdot B = \alpha \cdot \mu(\varepsilon) \cdot B.$$

For the induction step, let $l \in \mathbb{N}_0$ and assume that (3.6) holds for every $w \in \Sigma^l$. Take any $w \in \Sigma^l$ and $\sigma \in \Sigma$. For every $b \in \mathcal{B} = CS(B)$ we have by Proposition 3.2 (a) that $\mu(\sigma) \cdot b \in \mathcal{B}$, and thus by the induction hypothesis for $w \in \Sigma^l$ it follows

$$\begin{aligned} \tilde{\alpha} \cdot \tilde{\mu}(w\sigma) \cdot \tilde{F} \cdot b = \tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{\mu}(\sigma) \cdot \tilde{F} \cdot b &\overset{\text{Eq. (3.5)}}{=} \tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{F} \cdot \mu(\sigma) \cdot b \\ &= \alpha \cdot \mu(w) \cdot \mu(\sigma) \cdot b = \alpha \cdot \mu(w\sigma) \cdot b \end{aligned}$$

which completes the proof by induction.

Now for any $w \in \Sigma^*$, since $\gamma \in \mathcal{B}$ we have

$$\|\tilde{\mathcal{A}}\|(w) = \tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{\gamma} = \tilde{\alpha} \cdot \tilde{\mu}(w) \cdot \tilde{F} \cdot \gamma \overset{\text{Eq. (3.6)}}{=} \alpha \cdot \mu(w) \cdot \gamma = \|\mathcal{A}\|(w).$$

Hence, MWAs $\tilde{\mathcal{A}}$ and $\mathcal{A}$ are equivalent. Minimality of $\tilde{\mathcal{A}}$ follows from Lemma 3.3. □

We are now ready to prove Proposition 3.4 in its full generality. The proof is split in two lemmas, Lemmas 3.8 and 3.9, which together imply Proposition 3.4. First, we show that $\tilde{\mathcal{A}}$ is a well-defined multiplicity tree automaton:

Lemma 3.8. *For every $\sigma \in \Sigma_k$, there exists a unique matrix $\tilde{\mu}(\sigma)$ satisfying Equation (3.3).*

Proof. Since the rows of $\tilde{F} \cdot B$ form a basis of $RS(F \cdot B)$, it suffices to prove that

$$RS(\tilde{F}^{\otimes k} \cdot \mu(\sigma) \cdot B) \subseteq RS(F \cdot B).$$

By Lemma 2.1, to do this it suffices to prove that $RS(\tilde{F}^{\otimes k} \cdot \mu(\sigma)) \subseteq RS(F)$. Let us therefore take an arbitrary row $(\tilde{F}^{\otimes k} \cdot \mu(\sigma))_{(i_1,\ldots,i_k)}$ of $\tilde{F}^{\otimes k} \cdot \mu(\sigma)$, where $(i_1, \ldots, i_k) \in [m]^k$. We have

$$(\tilde{F}^{\otimes k} \cdot \mu(\sigma))_{(i_1,\ldots,i_k)} = (\tilde{F}^{\otimes k})_{(i_1,\ldots,i_k)} \cdot \mu(\sigma) \overset{\text{Eq. (2.1)}}{=} (\tilde{F}_{i_1} \otimes \cdots \otimes \tilde{F}_{i_k}) \cdot \mu(\sigma).$$

Since $\tilde{F}_{i_1}, \ldots, \tilde{F}_{i_k} \in RS(\tilde{F}) \subseteq RS(F) = \mathcal{F}$, we have that $(\tilde{F}_{i_1} \otimes \cdots \otimes \tilde{F}_{i_k}) \cdot \mu(\sigma) \in \mathcal{F}$ by Proposition 3.1 (a). Therefore, $(\tilde{F}^{\otimes k} \cdot \mu(\sigma))_{(i_1,\ldots,i_k)} \in \mathcal{F} = RS(F)$. □

Next, we show that MTA $\tilde{\mathcal{A}}$ minimises $\mathcal{A}$:

Lemma 3.9. *The automaton $\tilde{\mathcal{A}}$ is a minimal MTA equivalent to $\mathcal{A}$.*

Proof. First we show that for every $t \in T_\Sigma$,

$$\tilde{\mu}(t) \cdot \tilde{F} \cdot B = \mu(t) \cdot B. \tag{3.7}$$

Our proof is by induction on *height*(t). The base case $t = \sigma \in \Sigma_0$ follows immediately from Equation (3.3). For the induction step, let $h \in \mathbb{N}_0$ and assume that (3.7) holds for every $t \in T_\Sigma^{\leq h}$. Take any tree $t \in T_\Sigma^{h+1}$. Then $t = \sigma(t_1, \ldots, t_k)$ for some $k \geq 1$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma^{\leq h}$. Using bilinearity of Kronecker product we get that

$$\begin{aligned}
\tilde{\mu}(t) \cdot \tilde{F} \cdot B &= (\tilde{\mu}(t_1) \otimes \cdots \otimes \tilde{\mu}(t_k)) \cdot \tilde{\mu}(\sigma) \cdot \tilde{F} \cdot B \\
&= (\tilde{\mu}(t_1) \otimes \cdots \otimes \tilde{\mu}(t_k)) \cdot \tilde{F}^{\otimes k} \cdot \mu(\sigma) \cdot B && \text{by Eq. (3.3)} \\
&= ((\tilde{\mu}(t_1)\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B && \text{by Eq. (2.2)} \\
&= (\tilde{\mu}(t_1)\tilde{F}) \cdot (I_n \otimes (\tilde{\mu}(t_2)\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B.
\end{aligned}$$

Since $RS(\tilde{F}) \subseteq \mathcal{F}$, for every $i \in \{2, \ldots, k\}$ it holds that $\tilde{\mu}(t_i)\tilde{F} \in \mathcal{F}$. Since $I_n = \mu(\square) \in \mathcal{F}$, we now have that $(I_n \otimes (\tilde{\mu}(t_2)\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B \in \mathcal{B}$ by Proposition 3.2 (a). Thus by the induction hypothesis for $t_1 \in T_\Sigma^{\leq h}$, we have

$$\begin{aligned}
\tilde{\mu}(t) \cdot \tilde{F} \cdot B &= \mu(t_1) \cdot (I_n \otimes (\tilde{\mu}(t_2)\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B \\
&= (\mu(t_1) \otimes (\tilde{\mu}(t_2)\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B.
\end{aligned}$$

From here we argue inductively as follows: Assume that for some $l \in [k-1]$,

$$\tilde{\mu}(t) \cdot \tilde{F} \cdot B = (\mu(t_1) \otimes \cdots \otimes \mu(t_l) \otimes (\tilde{\mu}(t_{l+1})\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B.$$

Then by bilinearity of Kronecker product, we get that $\tilde{\mu}(t) \cdot \tilde{F} \cdot B$ is equal to

$$(\tilde{\mu}(t_{l+1})\tilde{F}) \cdot (\mu(t_1) \otimes \cdots \otimes \mu(t_l) \otimes I_n \otimes (\tilde{\mu}(t_{l+2})\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B.$$

Here $(\mu(t_1) \otimes \cdots \otimes \mu(t_l) \otimes I_n \otimes (\tilde{\mu}(t_{l+2})\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B \in \mathcal{B}$ by the same reasoning as above. The induction hypothesis for $t_{l+1} \in T_\Sigma^{\leq h}$ now implies

$$\begin{aligned}
&\tilde{\mu}(t) \cdot \tilde{F} \cdot B \\
&= \mu(t_{l+1}) \cdot (\mu(t_1) \otimes \cdots \otimes \mu(t_l) \otimes I_n \otimes (\tilde{\mu}(t_{l+2})\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B \\
&= (\mu(t_1) \otimes \cdots \otimes \mu(t_l) \otimes \mu(t_{l+1}) \otimes (\tilde{\mu}(t_{l+2})\tilde{F}) \otimes \cdots \otimes (\tilde{\mu}(t_k)\tilde{F})) \cdot \mu(\sigma) \cdot B.
\end{aligned}$$

Continuing our inductive argument, for $l = k - 1$ we get that

$$\tilde{\mu}(t) \cdot \tilde{F} \cdot B = (\mu(t_1) \otimes \cdots \otimes \mu(t_k)) \cdot \mu(\sigma) \cdot B = \mu(t) \cdot B.$$

This completes the proof of (3.7) by induction.

Now since $\gamma \in \mathcal{B}$, for every $t \in T_\Sigma$ we have

$$\|\tilde{\mathcal{A}}\|(t) = \tilde{\mu}(t) \cdot \tilde{\gamma} = \tilde{\mu}(t) \cdot \tilde{F} \cdot \gamma \overset{\text{Eq. (3.7)}}{=} \mu(t) \cdot \gamma = \|\mathcal{A}\|(t).$$

Hence, MTAs $\tilde{\mathcal{A}}$ and $\mathcal{A}$ are equivalent. Minimality follows from Lemma 3.3. □

By a result of Bozapalidis and Alexandrakis [7, Proposition 4], all equivalent minimal multiplicity tree automata 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 such matrices. In this section we prove a key result, Proposition 3.10 below, suggesting a way to compute F and B, which we exploit in Sections 4.2 and 5.

Propositions 3.1 and 3.2 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 in Section 4.2 nor for a reduction to ACIT in Section 5.

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 Section 2.5. 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 characterisation 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 3.10. *Let Σ have rank r. Let $\mathcal{A} \times \mathcal{A} = (n^2, \Sigma, \mu', \gamma^{\otimes 2})$ be the product of $\mathcal{A}$ by $\mathcal{A}$. For every $l \in \mathbb{N}$, define*

$$f(l) := \sum_{t \in T_\Sigma^{<l}} \mu'(t) \in \mathbb{F}^{1 \times n^2} \text{ and } b(l) := \sum_{c \in C^{<l}_{\Sigma, T_\Sigma^{<n}}} \mu'(c) \in \mathbb{F}^{n^2 \times n^2}.$$

(a) *Let $F \in \mathbb{F}^{n \times n}$ be the matrix with $F_{i,j} = f(n) \cdot (e_i \otimes e_j)^\top$ for all $i, j \in [n]$. Let $B \in \mathbb{F}^{n \times n}$ be the matrix with $B_{i,j} = (e_i \otimes e_j) \cdot b(n) \cdot \gamma^{\otimes 2}$ for all $i, j \in [n]$. Then, $RS(F) = \mathcal{F}$ and $CS(B) = \mathcal{B}$.*

(b) *We have $f(1) = \sum_{\sigma \in \Sigma_0} \mu'(\sigma)$ and $b(1) = I_{n^2}$. For all $l \in \mathbb{N}$, it holds that*

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

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

Proof. First, we prove that $RS(F) = \mathcal{F}$ in part (a). Let $\widehat{F} \in \mathbb{F}^{T_\Sigma^{<n} \times [n]}$ be a matrix such that $\widehat{F}_t = \mu(t)$ for every $t \in T_\Sigma^{<n}$. From Proposition 3.1 (b) it follows that $RS(\widehat{F}) = \mathcal{F}$. By Lemma 2.2 we now have $RS(\widehat{F}^\top \widehat{F}) = RS(\widehat{F}) = \mathcal{F}$. 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 Kronecker product, we have for all $i, j \in [n]$:

$$\begin{aligned}
(\widehat{F}^\top \widehat{F})_{i,j} = (\widehat{F}^\top)_i \cdot (\widehat{F})^j &= \sum_{t \in T_\Sigma^{<n}} \mu(t)_i \cdot \mu(t)_j \\
&= \sum_{t \in T_\Sigma^{<n}} (\mu(t) \cdot e_i^\top) \otimes (\mu(t) \cdot e_j^\top) \\
&= \left(\sum_{t \in T_\Sigma^{<n}} (\mu(t) \otimes \mu(t)) \right) \cdot (e_i \otimes e_j)^\top \\
&\overset{\text{Prop. 2.7}}{=} \left(\sum_{t \in T_\Sigma^{<n}} \mu'(t) \right) \cdot (e_i \otimes e_j)^\top = f(n) \cdot (e_i \otimes e_j)^\top.
\end{aligned}$$

Next, we complete the proof of part (a) by proving that $CS(B) = \mathcal{B}$. To avoid notational clutter, in the following we write

$$C := C^{<n}_{\Sigma, T_\Sigma^{<n}}.$$

Define a matrix $\widehat{B} \in \mathbb{F}^{[n] \times C}$ such that $\widehat{B}^c = \mu(c) \cdot \gamma$ for all $c \in C$. From Proposition 3.2 (b) it follows that $CS(\widehat{B}) = \mathcal{B}$. By Lemma 2.2 we now have $CS(\widehat{B}\widehat{B}^\top) = CS(\widehat{B}) = \mathcal{B}$. Therefore in order to prove that $CS(B) = \mathcal{B}$, it suffices to show that $\widehat{B}\widehat{B}^\top = B$. Indeed, using the mixed-product property of Kronecker product, we have for all $i, j \in [n]$:

$$\begin{aligned}
(\widehat{B} \cdot \widehat{B}^\top)_{i,j} &= (\widehat{B})_i \cdot (\widehat{B}^\top)^j \\
&= \sum_{c \in C} (\mu(c)_i \cdot \gamma) \cdot (\mu(c)_j \cdot \gamma)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{c \in C} (e_i \cdot \mu(c) \cdot \gamma) \otimes (e_j \cdot \mu(c) \cdot \gamma) \\
&= \sum_{c \in C} (e_i \otimes e_j) \cdot (\mu(c) \otimes \mu(c)) \cdot (\gamma \otimes \gamma) \\
&= (e_i \otimes e_j) \cdot \left(\sum_{c \in C} (\mu(c) \otimes \mu(c)) \right) \cdot (\gamma \otimes \gamma) \\
&= (e_i \otimes e_j) \cdot \left(\sum_{c \in C} \mu'(c) \right) \cdot \gamma^{\otimes 2} && \text{(by Proposition 2.7 (ii))} \\
&= (e_i \otimes e_j) \cdot b(n) \cdot \gamma^{\otimes 2} && \text{(definition of } b(n)) \\
&= B_{i,j} .
\end{aligned}
$$

We turn to the proof of part (b). Here we do not use the fact that we are dealing with a product automaton. We first prove the statement on $f(l)$. The equality $f(1) = \sum_{\sigma \in \Sigma_0} \mu'(\sigma)$ follows directly from the definition. For all $l \in \mathbb{N}$,

$$
T_\Sigma^{<l+1} = \{\sigma(t_1, \ldots, t_k) : 0 \leq k \leq r,\ \sigma \in \Sigma_k,\ t_1, \ldots, t_k \in T_\Sigma^{<l}\} .
$$

Thus, by bilinearity of Kronecker product, it holds that

$$
\begin{aligned}
f(l+1) &= \sum_{t \in T_\Sigma^{<l+1}} \mu'(t) \\
&= \sum_{k=0}^{r} \sum_{\sigma \in \Sigma_k} \sum_{t_1 \in T_\Sigma^{<l}} \cdots \sum_{t_k \in T_\Sigma^{<l}} \big(\mu'(t_1) \otimes \cdots \otimes \mu'(t_k)\big) \cdot \mu'(\sigma) \\
&= \sum_{k=0}^{r} \left(\left(\sum_{t_1 \in T_\Sigma^{<l}} \mu'(t_1) \right) \otimes \cdots \otimes \left(\sum_{t_k \in T_\Sigma^{<l}} \mu'(t_k) \right) \right) \cdot \sum_{\sigma \in \Sigma_k} \mu'(\sigma) \\
&= \sum_{k=0}^{r} \left(\sum_{t \in T_\Sigma^{<l}} \mu'(t) \right)^{\otimes k} \sum_{\sigma \in \Sigma_k} \mu'(\sigma) \\
&= \sum_{k=0}^{r} f(l)^{\otimes k} \sum_{\sigma \in \Sigma_k} \mu'(\sigma) .
\end{aligned}
$$

Finally, we prove the statement on $b(l)$. The equality $b(1) = I_{n^2}$ follows from the definition. To avoid notational clutter we write $T := T_\Sigma^{<n}$ in the following. Recall that $f(n) = \sum_{t \in T} \mu'(t)$. We have for all $l \in \mathbb{N}$:

$$
\begin{aligned}
C_{\Sigma,T}^{<l+1} = \{\square\} \cup \{\sigma(t_1, \ldots, t_{j-1}, c_j, t_{j+1}, \ldots, t_k) : k \in [r],\ j \in [k],\ \sigma \in \Sigma_k, \\
c_j \in C_{\Sigma,T}^{<l},\ t_1, \ldots, t_{j-1}, t_{j+1}, \ldots, t_k \in T \Big\} .
\end{aligned}
$$

Thus, using bilinearity of Kronecker product, we get that

$$
b(l+1)
$$

$$\begin{aligned}
&= \sum_{c\in C^{<l+1}_{\Sigma,T}} \mu'(c) \\
&= \mu'(\square) + \sum_{k=1}^{r}\sum_{j=1}^{k}\sum_{\sigma\in\Sigma_k}\sum_{t_1,\dots,t_{j-1}\in T}\sum_{c_j\in C^{<l}_{\Sigma,T}}\sum_{t_{j+1},\dots,t_k\in T}(\mu'(t_1)\otimes\cdots\otimes\mu'(c_j) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\otimes\cdots\otimes\mu'(t_k))\cdot\mu'(\sigma) \\
&= I_{n^2} + \sum_{k=1}^{r}\sum_{j=1}^{k}\left(\left(\sum_{t_1\in T}\mu'(t_1)\right)\otimes\cdots\otimes\left(\sum_{c_j\in C^{<l}_{\Sigma,T}}\mu'(c_j)\right)\right. \\
&\qquad\qquad\left.\otimes\cdots\otimes\left(\sum_{t_k\in T}\mu'(t_k)\right)\right)\cdot\sum_{\sigma\in\Sigma_k}\mu'(\sigma) \\
&= I_{n^2} + \sum_{k=1}^{r}\sum_{j=1}^{k}\left(f(n)^{\otimes(j-1)}\otimes b(l)\otimes f(n)^{\otimes(k-j)}\right)\sum_{\sigma\in\Sigma_k}\mu'(\sigma)\,.
\end{aligned}$$

This completes the proof. □

Loosely speaking, Proposition 3.10 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 subsection we describe an implementation of the algorithm implicit in Section 3.2, and analyse the number of operations. We consider a multiplicity tree automaton $\mathcal{A} = (n, \Sigma, \mu, \gamma)$. We denote by r the rank of Σ. The algorithm has three steps, as follows:

4.1.1. *Step 1 "Forward".* The first step is to compute a matrix F such that $RS(F) = \mathcal{F}$. Seidl [34] outlines a saturation-based algorithm for this, and proves that the algorithm takes polynomial time assuming unit-cost arithmetic. Based on Proposition 3.1 (a) we now give in Table 1 an explicit version of Seidl's algorithm.

Our algorithm satisfies the following properties:

Lemma 4.1. *The algorithm in Table 1 returns a matrix $F \in \mathbb{Q}^{\overrightarrow{n}\times n}$ whose rows form a basis of the forward space $\mathcal{F}$. Each row of F equals $\mu(t)$ for some tree $t \in T^{<n}_{\Sigma}$. The algorithm executes $O\left(\sum_{k=0}^{r}|\Sigma_k|\cdot n^{2k+1}\right)$ operations.*

Proof. The fact that the rows of F span $\mathcal{F}$ follows from Proposition 3.1 (a). Moreover, it is clear from the algorithm that the rows of F are linearly independent.

```
Input: $\mathbb{Q}$-multiplicity tree automaton $(n, \Sigma, \mu, \gamma)$
Output: matrix $F$ whose rows form a basis of the forward space $\mathcal{F}$
    $i := 0$, $j := 0$
    while $i \leq j$ do
        forall $\sigma \in \Sigma$ do
            forall $(l_1, \ldots, l_{rk(\sigma)}) \in [i]^{rk(\sigma)} \setminus [i-1]^{rk(\sigma)}$ do
                $v := (F_{l_1} \otimes \cdots \otimes F_{l_{rk(\sigma)}}) \cdot \mu(\sigma)$
                if $v \notin \langle F_1, \ldots, F_j \rangle$
                    $j := j + 1$
                    $F_j := v$
    $i := i + 1$
    return matrix $F \in \mathbb{Q}^{j \times n}$
```

Table 1: Algorithm for computing a matrix F

A straightforward induction shows that for each row index $j \geq 1$, the row F_j equals $\mu(t)$ for some tree $t \in T_{\Sigma}^{<j}$. The returned matrix $F \in \mathbb{Q}^{\overrightarrow{n} \times n}$ has full row rank, and therefore $\overrightarrow{n} \leq n$. Hence, each row of F equals $\mu(t)$ for some tree $t \in T_{\Sigma}^{<n}$.

It remains to analyse the number of operations. Let us consider an iteration of the innermost "for" loop. The computation of $F_{l_1} \otimes \cdots \otimes F_{l_{rk(\sigma)}}$ requires $O(n^{rk(\sigma)})$ operations (by iteratively computing partial products). The vector

$$v = (F_{l_1} \otimes \cdots \otimes F_{l_{rk(\sigma)}}) \cdot \mu(\sigma)$$

is the product of a $1 \times n^{rk(\sigma)}$ vector with an $n^{rk(\sigma)} \times n$ matrix. Thus, computing v takes $O(n^{rk(\sigma)+1})$ operations. For the purpose of checking membership of v in the vector space $\mathcal{F}' := \langle F_1, \ldots, F_j \rangle$ it is useful to maintain a matrix F', which is upper triangular (up to a permutation of its columns) and whose rows form a basis of $\mathcal{F}'$. To check whether $v \in \mathcal{F}'$ we compute a vector v' as the result of performing a Gaussian elimination of v against F', which requires $O(j \cdot n)$ operations. If this membership test fails, we extend the matrix F' at the bottom by row v'. This preserves the upper-triangular shape of F'. Thus, an iteration of the innermost "for" loop takes $O(n^{rk(\sigma)+1})$ operations. For every $\sigma \in \Sigma$, this "for" loop is executed $O(n^{rk(\sigma)})$ times. Therefore, the algorithm executes $O\left(\sum_{k=0}^{r} |\Sigma_k| \cdot n^{2k+1}\right)$ operations. □

4.1.2. *Step 2 "Backward".* The next step suggested in Section 3.2 is to compute a matrix B such that $CS(B) = \mathcal{B}$. By Lemma 4.1, each row of the matrix F computed by the algorithm in Table 1 equals $\mu(t)$ for some tree $t \in T_{\Sigma}^{<n}$. Let S denote the set of those trees. Since $RS(F) = \mathcal{F}$, set $\{\mu(t) : t \in S\}$ spans $\mathcal{F}$. Thus by Proposition 3.2 (a), $\mathcal{B}$ is the smallest vector space $V \subseteq \mathbb{Q}^n$ such that $\gamma \in V$ and $M \cdot v \in V$ for all $M \in \mathcal{M} := \{\mu(c) : c \in C_{\Sigma,S}^{1}\}$ and $v \in V$. Tzeng [35] 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.*, [16]). This leads to the following lemma:

Lemma 4.2. *Given the matrix $F \in \mathbb{Q}^{\overrightarrow{n} \times n}$ which is the output of the algorithm in Table 1, a matrix B whose columns form a basis of the backward space $\mathcal{B}$ can be computed with $O\left(\sum_{k=1}^{r} |\Sigma_k| \cdot (kn^{2k} + kn^{k+2})\right)$ operations.*

Proof. Consider the computation of an arbitrary $M \in \mathcal{M} := \{\mu(c) : c \in C^1_{\Sigma,S}\}$. We have:

$$M = G \cdot \mu(\sigma) \ , \quad \text{where} \tag{4.1}$$

$$G = F_{l_1} \otimes \cdots \otimes F_{l_{i-1}} \otimes I_n \otimes F_{l_{i+1}} \otimes \cdots \otimes F_{l_{rk(\sigma)}} \in \mathbb{Q}^{n \times n^{rk(\sigma)}} \tag{4.2}$$

is such that $\sigma \in \Sigma \setminus \Sigma_0$, $i \in [rk(\sigma)]$, $l_1, \ldots, l_{i-1}, l_{i+1}, \ldots, l_{rk(\sigma)} \in [\overrightarrow{n}]$.

Exploiting the sparsity pattern in the matrix G as in (4.2), the computation of the non-zero entries of G takes $O(n^{rk(\sigma)})$ operations. Exploiting sparsity again, the computation of matrix M as in (4.1) then takes $O(n^{rk(\sigma)+1})$ operations. Since $\overrightarrow{n} \le n$, it follows from (4.1) and (4.2) that

$$|\mathcal{M}| \in O\left(\sum_{k=1}^{r} |\Sigma_k| \cdot k \cdot n^{k-1}\right) .$$

Thus, the number of operations required to compute $\mathcal{M}$ is $O\left(\sum_{k=1}^{r} |\Sigma_k| \cdot k \cdot n^{2k}\right)$. Given $\mathcal{M}$, computing a basis of $\mathcal{B}$ takes

$$O(|\mathcal{M}| \cdot n^3) = O\left(\sum_{k=1}^{r} |\Sigma_k| \cdot k \cdot n^{k-1} \cdot n^3\right)$$

operations, using, *e.g.*, the method from [16] that was mentioned above. Therefore, the total operation count for computing a matrix B is $O\left(\sum_{k=1}^{r} |\Sigma_k| \cdot (kn^{2k} + kn^{k+2})\right)$. □

4.1.3. *Step 3 "Solve".* The final step suggested in Section 3.2 has two substeps. The first substep is to compute a matrix $\tilde{F} \in \mathbb{Q}^{m \times n}$ with $m = \mathit{rank}(F \cdot B)$ and $RS(\tilde{F} \cdot B) = RS(F \cdot B)$. Such a 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 the matrix $F \cdot B$ into a triangular form using Gaussian elimination.

The second substep is to compute the minimal MTA $\tilde{\mathcal{A}} = (m, \Sigma, \tilde{\mu}, \tilde{\gamma})$. The vector $\tilde{\gamma} = \tilde{F} \cdot \gamma$ is easy to compute. Solving Equation (3.3) 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. Putting together the results of this subsection, we get:

Theorem 4.3. *There is an algorithm that transforms a given $\mathbb{Q}$-MTA $\mathcal{A} = (n, \Sigma, \mu, \gamma)$ 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 subsection, we consider the problem of minimising a given $\mathbb{Q}$-multiplicity word automaton. We prove the following result:

Theorem 4.4. *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 4.4 improves on two results of [25]. First, [25, Theorem 4.2] states that deciding whether a $\mathbb{Q}$-MWA is minimal is in NC. Second, [25, Theorem 4.5] states the same thing as our Theorem 4.4, but with NC replaced with *randomised* NC.

Proof of Theorem 4.4. The algorithm relies on Propositions 3.5 and 3.10. Let the given $\mathbb{Q}$-MWA be $\mathcal{A} = (n, \Sigma, \mu, \alpha, \gamma)$. In the notation of Proposition 3.10, 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 for all $l \in \mathbb{N}$:

$$b(l) = \sum_{k=0}^{l-1} \left(\sum_{\sigma \in \Sigma} \mu'(\sigma) \right)^k .$$

It follows for the matrix $B \in \mathbb{Q}^{n \times n}$ from Proposition 3.10 that for all $i, j \in [n]$:

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

Note that, since $\mathcal{A}$ is an MWA, we have $f(l) = b(l)$ for all $l \in \mathbb{N}$. We now have for the matrix $F \in \mathbb{Q}^{n \times n}$ from Proposition 3.10 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 and B can be computed in NC since sums and matrix powers can be computed in NC [15]. 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

$$\mathit{rank}(F_{[i],[n]} \cdot B) > \mathit{rank}(F_{[i-1],[n]} \cdot B).$$

This can be done in NC since the rank of a matrix can be determined in NC [23]. It remains to compute $\tilde{\gamma} := \tilde{F}\gamma$ and solve Equations (3.4) and (3.5) 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 5.1 below, that this problem is interreducible with the arithmetic circuit identity testing (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 nonnegative 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 5.1. *Minimisation is logspace interreducible with* ACIT.

We consider the lower and the upper bound separately.

5.1. 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 observe that there is a logspace reduction from ACIT to zeroness. Indeed, it is shown in [28] 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 $\|\mathcal{A}_1\|(t) = \|\mathcal{A}_2\|(t)$ for all trees t. By Proposition 2.7, one can reduce this problem to zeroness in logarithmic space. This implies ACIT-hardness of minimisation.

5.2. Upper Bound.

We prove:

Proposition 5.2. *There is a logspace reduction from minimisation to* ACIT.

Proof. Let $\mathcal{A} = (n, \Sigma, \mu, \gamma)$ be the given $\mathbb{Q}$-MTA, and let $d \in \mathbb{N}_0$ be 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 3.3, the dimension of a minimal MTA equivalent to $\mathcal{A}$ is $m := \mathit{rank}(F \cdot B)$ where F and B are matrices such that $\mathit{RS}(F) = \mathcal{F}$ and $\mathit{CS}(B) = \mathcal{B}$. Therefore, 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 3.10 allows us to compute in logarithmic space an arithmetic circuit for $F \cdot B$. Thus, the result follows from Lemma 5.3 below. $\square$

The following lemma follows easily from the well-known NC procedure for computing matrix rank [17].

Lemma 5.3. *Let $M \in \mathbb{Q}^{m \times n}$ and $d \in \mathbb{N}_0$. The problem of deciding whether $\mathit{rank}(M) \leq d$ is logspace reducible to* ACIT.

Proof. By the rank-nullity theorem, we have that $rank(M) \leq d$ if and only if $dim(ker(M)) \geq n - d$. Since $ker(M) = ker(M^\top M)$, this is equivalent to $dim(ker(M^\top M)) \geq n - d$. The matrix $M^\top M$ is Hermitian, therefore $dim(ker(M^\top M)) \geq n - d$ if and only if the $n - d$ lowest-order coefficients of the characteristic polynomial of $M^\top M$ are all zero [23]. But these coefficients are representable by arithmetic circuits with inputs from M (see [17]). □

We emphasise that our reduction to ACIT is a many-one reduction, thanks to Proposition 3.10: our reduction computes only a single instance of ACIT; there are no if-conditionals.

6. Minimal Consistent Multiplicity Automaton

Let $\mathbb{F}$ be an arbitrary field. 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 main result of this section concerns the computability of the above consistency problem for the field of rational numbers. More specifically, we consider a decision version of this problem, which we call *minimal consistency problem*, which asks whether there exists a $\mathbb{Q}$-MWA consistent with a set of input-output behaviours $S \subseteq \Sigma^* \times \mathbb{Q}$ and that has dimension at most some nonnegative integer bound n.

We show that the minimal consistency problem is logspace equivalent to the problem of deciding the truth of existential first-order sentences over the structure $(\mathbb{Q}, +, \cdot, 0, 1)$. The decidability of the latter is a longstanding open problem [31]. This should be compared with the result that the problem of finding the smallest deterministic finite automaton consistent with a set of accepted or rejected words is NP-complete [21].

The reduction of the minimal consistency problem to the decision problem for existential first-order sentences over the structure $(\mathbb{Q}, +, \cdot, 0, 1)$ is immediate. The idea is to represent a $\mathbb{Q}$-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, the consistency of automaton $\mathcal{A}$ with a given finite sample $S \subseteq \Sigma^* \times \mathbb{Q}$ can directly be written as an existential sentence.

We note in passing that the minimal consistency problem for weighted word and tree automata over the field $\mathbb{R}$ is in like manner reducible to the problem of deciding the truth of existential first-order sentences over the structure $(\mathbb{R}, +, \cdot, 0, 1)$, which is well known to be decidable in PSPACE [10].[1]

Conversely, we reduce the decision problem for existential first-order sentences over the structure $(\mathbb{Q}, +, \cdot, 0, 1)$ to the minimal consistency problem for $\mathbb{Q}$-MWA. In fact it suffices to consider sentences in the restricted form

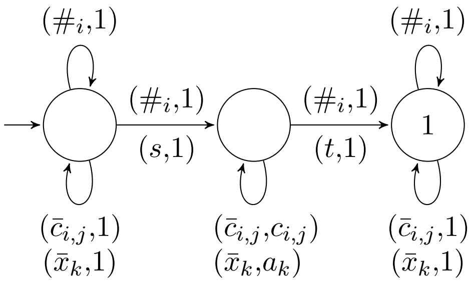
$$\exists x_1 \cdots \exists x_n \bigwedge_{i=1}^{m} f_i(x_1, \ldots, x_n) = 0\,, \tag{6.1}$$

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. We can make this simplification without loss of generality since a disjunction of atomic

[1] To consider this problem within the conventional Turing model, we assume that the set S of input-output behaviours is still a subset of $\Sigma^* \times \mathbb{Q}$. Of course, the dimension of the smallest MWA consistent with a given finite set of behaviours S depends on the weight field of the output automaton.

	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)

Figure 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 $\mathcal{A}$.

formulas $f = 0 \lor g = 0$, where f and g are polynomials, can be rewritten to

$$\exists x \left(x^2 - x = 0 \land x \cdot f = 0 \land (1-x) \cdot g = 0\right).$$

Moreover, the negation of an atomic formula $f \neq 0$ is equivalent to $\exists x\,(x \cdot f = 1)$.

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 the alphabet Σ we consider the 3-dimensional $\mathbb{Q}$-MWA $\mathcal{A}$, depicted in Figure 1 (b). The transitions in this automaton are annotated by label-weight pairs in $\Sigma \times \mathbb{Q}$. Recall that the weights $c_{i,j}$ are coefficients of the polynomial f_i. For each $k \in [n]$, the weight a_k is a fixed but arbitrary element of $\mathbb{Q}$.

Define $X, Y \subseteq \Sigma^*$ by $X = \{\varepsilon, s, st\}$ and $Y = \{st, t, \varepsilon\}$. Consider the fragment $\tilde{H} := H_{X \cup X\Sigma, Y}$, shown in Figure 1 (a), of the Hankel matrix H of $\|\mathcal{A}\|$. We know from Theorem 2.4 that $\mathit{rank}(H) \leq 3$. Since $\mathit{rank}(H_{X,Y}) = 3$, we have $\mathit{rank}(H_{X,Y}) = \mathit{rank}(H) = 3$. Now, from Remark 2.5 it follows that any 3-dimensional $\mathbb{Q}$-MWA $\mathcal{A}'$ that is consistent with $H_{X,Y}$ and $H_{X\Sigma,Y}$ (*i.e.*, consistent with $\tilde{H}$) is equivalent to $\mathcal{A}$.

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

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

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 $\|\mathcal{A}\|(w_i) = f_i(a_1, \ldots, a_n)$; the details are given below in the proof of Proposition 6.1.

We define a set of weighted words $S \subseteq \Sigma^* \times \mathbb{Q}$ 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

$S_2 := \{(w_i, 0) : i \in [m]\}$. That is, S_1 specifies all entries in the matrix $\tilde{H}$ except those that are in row $s\bar{x}_k$ and column t.

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

Proposition 6.1. *The sample S is consistent with a 3-dimensional $\mathbb{Q}$-MWA if and only if the sentence (6.1) is true in* $(\mathbb{Q}, +, \cdot, 0, 1)$.

Proof. We have already noted that any 3-dimensional $\mathbb{Q}$-MWA consistent with S must be equivalent to an automaton of the form $\mathcal{A}$ in Figure 1 (b) for some $a_1, \ldots, a_n \in \mathbb{Q}$. However, such an automaton is consistent with S if and only if it assigns weight 0 to each word w_i, $i \in [m]$. Now, we claim that this is the case if and only if $(a_1, \ldots, a_n)$ is a root of f_i for every $i \in [m]$, where a_k is the weight of the $\bar{x}_k$-labelled self-loop in the middle state, for every $k \in [n]$.

For every $i \in [m]$, the word w_i has l_i different accepting runs in $\mathcal{A}$, one for each monomial in f_i. The j^{th} such run, in which the block $\bar{c}_{i,j}\bar{x}_1^{k_{i,j,1}} \cdots \bar{x}_n^{k_{i,j,n}}$ is read in the middle state, has weight $c_{i,j}a_1^{k_{i,j,1}} \cdots a_n^{k_{i,j,n}}$, *i.e.*, the value of monomial $c_{i,j}x_1^{k_{i,j,1}} \cdots x_n^{k_{i,j,n}}$ evaluated at $(a_1, \ldots, a_n)$. Thus $\|\mathcal{A}\|(w_i) = f_i(a_1, \ldots, a_n)$. □

From Proposition 6.1 we derive the main result of this section:

Theorem 6.2. *The minimal consistency problem for $\mathbb{Q}$-MWAs is logspace equivalent to the decision problem for existential first-order sentences over* $(\mathbb{Q}, +, \cdot, 0, 1)$. □

7. Conclusions and Future Work

We have looked at the computational complexity of computing minimal multiplicity word and tree automata from several angles. Specifically, we have analysed the complexity of computing a minimal automaton equivalent to a given input automaton $\mathcal{A}$. We have considered also the corresponding decision problem, which asks whether there exists an automaton equivalent to $\mathcal{A}$ with a given number of states. Finally, we have considered the minimal consistency problem, in which the input is a finite set of word-weight pairs rather than a complete automaton.

Our complexity bounds have drawn connections between automaton minimisation and longstanding open questions in arithmetic complexity, including the complexity of polynomial identity testing and the decidability of Hilbert's tenth problem over the rationals, *i.e.*, the problem of deciding the truth of existential sentences over the structure $(\mathbb{Q}, +, \cdot, 0, 1)$.

Our algorithmic results exclusively concern automata over the fields of rational or real numbers, in which weights are allowed to be negative. The minimisation problems considered here all have natural analogues for the class of *probabilistic automata* over words and trees, in which the transition weights are probabilities. Recently, minimisation of probabilistic word automata was shown to be NP-hard [26]. A natural question is whether this minimisation problem lies in NP, and whether the corresponding problem for tree automata is even harder. Related to this is the following question: Given a multiplicity (word or tree) automaton with rational transition weights, need there always be a minimal equivalent automaton also with rational transition weights?

We have observed that the minimal consistency problem for word automata over the reals is in PSPACE, since it is directly reducible to the problem of deciding the truth of existential first-order sentences over the structure $(\mathbb{R}, +, \cdot, 0, 1)$. For tree automata this reduction is exponential in the alphabet rank, and we leave as an open question the complexity of the minimal consistency problem for tree automata over the reals.

In all cases, we have considered minimising automata with respect to the number of states. Another natural question is minimisation with respect to the number of transitions. This is particularly pertinent to the case of tree automata, where the number of transitions is potentially exponential in the number of states.

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. Marušić and Worrell gratefully acknowledge the support of the EPSRC.

References

[1] J. Albert and J. Kari. Digital image compression. In *Handbook of Weighted Automata*, pages 453–479. Springer, 2009.

[2] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[3] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[4] J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *Theoretical Computer Science*, 18(2):115–148, 1982.

[5] B. Borchardt. A pumping lemma and decidability problems for recognizable tree series. *Acta Cybernetica*, 16(4):509–544, 2004.

[6] S. Bozapalidis. Effective construction of the syntactic algebra of a recognizable series on trees. *Acta Informatica*, 28(4):351–363, 1991.

[7] S. Bozapalidis and A. Alexandrakis. Représentations matricielles des séries d'arbre reconnaissables. *RAIRO- Informatique Théorique et Applications*, 23(4):449–459, 1989.

[8] S. Bozapalidis and O. Louscou-Bozapalidou. The rank of a formal tree power series. *Theoretical Computer Science*, 27(1):211–215, 1983.

[9] W. S. Brainerd. The minimalization of tree automata. *Information and Control*, 13(5):484–491, 1968.

[10] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 460–467, 1988.

[11] J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1):26–40, 1971.

[12] J. Carme, R. Gilleron, A. Lemay, A. Terlutte, and M. Tommasi. Residual finite tree automata. In *Proceedings of the 7th International Conference on Developments in Language Theory (DLT)*, pages 171–182, 2003.

[13] R. Carrasco, J. Daciuk, and M. Forcada. An implementation of deterministic tree automata minimization. In *Proceedings of the 12th International Conference on Implementation and Application of Automata (CIAA)*, pages 122–129, 2007.

[14] S. Cho and D. T. Huynh. The parallel complexity of finite-state automata problems. *Information and Computation*, 97(1):1–22, 1992.

[15] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–22, 1985.

[16] C. Cortes, M. Mohri, and A. Rastogi. On the computation of some standard distances between probabilistic automata. In *Proceedings of the 11th International Conference on Implementation and Application of Automata (CIAA)*, volume 4094 of *LNCS*, pages 137–149. Springer, 2006.

[17] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.

[18] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

[19] J. Eisner. Simpler and more general minimization for weighted finite-state automata. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL HLT)*, volume 1, pages 64–71, 2003.

[20] M. Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 53:197–222, 1974.

[21] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[22] A. Habrard and J. Oncina. Learning multiplicity tree automata. In *Proceedings of the 8th International Colloquium on Grammatical Inference: Algorithms and Applications (ICGI)*, pages 268–280. Springer, 2006.

[23] O. H. Ibarra, S. Moran, and L. E. Rosier. A note on the parallel complexity of computing the rank of order n matrices. *Information Processing Letters*, 11(4/5):162, 1980.

[24] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

[25] S. Kiefer, A. Murawski, J. Ouaknine, B. Wachter, and J. Worrell. On the complexity of equivalence and minimisation for $\mathbb{Q}$-weighted automata. *Logical Methods in Computer Science*, 9(1), 2013.

[26] S. Kiefer and B. Wachter. Stability and complexity of minimising probabilistic automata. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8573 of *LNCS*, pages 268–279. Springer, 2014.

[27] A. Maletti. Minimizing deterministic weighted tree automata. *Information and Computation*, 207(11):1284–1299, 2009.

[28] I. Marušić and J. Worrell. Complexity of equivalence and learning for multiplicity tree automata. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS), Part I*, pages 414–425, 2014.

[29] M. Mohri, F. Pereira, and M. Riley. Weighted automata in text and speech processing. In *European Conference on Artificial Intelligence (ECAI), Workshop on Extended Finite State Models of Language*, 1996.

[30] A. Paz. *Introduction to Probabilistic Automata*. Academic Press, 1971.

[31] T. Pheidas. Hilbert's tenth problem for fields of rational functions over finite fields. *Inventiones mathematicae*, 103(1):1–8, 1991.

[32] M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2–3):245–270, 1961.

[33] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[34] H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3):424–437, 1990.

[35] W.-G. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.

[36] R. E. Zippel. Probabilistic algorithms for sparse polynominals. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (EUROSAM)*, volume 72 of *LNCS*, pages 216–226. Springer, 1979.