

Weighted Automata over Vector Spaces*

Nada Damljanović

University of Kragujevac, Faculty of Technical Sciences, Svetog Save 65, Čačak, Serbia

nada.damljanovic@ftn.kg.ac.rs

Miroslav Ćirić

Jelena Ignjatović

University of Niš, Faculty of Sciences and Mathematics, Višegradska 33, Niš, Serbia

miroslav.ciric@pmf.edu.rs

jelena.ignjatovic@pmf.edu.rs

In this paper we deal with three models of weighted automata that take weights in the field of real numbers. The first of these models are classical weighted finite automata, the second one are **crisp-deterministic weighted automata**, and the third one are **weighted automata over a vector space**. We explore the interrelationships between weighted automata over a vector space and other two models.

1 Introduction

Weighted automata belong to the fundamental models of computation in computer science. They can be understood as an extension of conventional automata in which the transitions and states carry numerical or other values called weights. These weights may model quantitative properties like the cost, the amount of resources needed for the execution of a transition, the reliability or probability of the successful execution of the transitions, or many other things. Different models of weighted automata differ in the algebraic structures within which the weights are taken, as well as in the way in which these weights are manipulated.

In this paper, we deal with weighted automata that take weights in the field of real numbers. Such automata have been the subject of study since the very beginning of the theory of weighted automata, since the seminal work of Schützenberger [23] who studied weighted automata over the field. Today, they are very popular due to their significant applications, primarily in formal specification and verification of systems, as well as in the field of machine learning, where they are successfully used as an alternative to recurrent neural networks. We discuss three models of weighted automata with weights taken in the field of real numbers.

The first of these models are classical *weighted finite automata*. The common way of viewing deterministic and nondeterministic finite automata as labelled graphs has also been used for weighted finite automata from the very beginning of their studying. From such a point of view, a weighted finite automaton is represented by a directed multi-graph whose edges carry two labells, the input letter and the weight, while nodes carry two weights, the initial and terminal weight. The computation along a path in the graph is performed by concatenation of the input letters and multiplication of the initial weight of the starting node, the weights of edges along the path, and the terminal weight of the final node, and then the sum of the weights of all paths labelled with the same input word is computed and assigned to this input word. This determines the behavior of the considered weighted finite automaton, that is, the word function computed by that automaton. Such an understanding of the behaviour can be called

*This research was supported by the Science Fund of the Republic of Serbia, Grant no 7750185, Quantitative Automata Models: Fundamental Problems and Applications - QUAM

the *dept-first semantics*. Another way of looking at weighted finite automata, through vector and matrix operations, has also been present since their very beginnings. From that point of view, the behaviour of a weighted finite automaton can be expressed as the product of the row vector representing the initial weights, matrices representing the weights of the transitions induced by input letters, and the column vector representing the terminal weights. Such a representation of a weighted finite automaton is called a *linear representation*, while such an understanding of the behaviour can be called the *breadth-first semantics*. In the case of weighted finite automata over a semiring these two semantics are the same. The linear algebraic approach proved to be extremely powerful and useful, especially in the study of simulations and bisimulations, as well as in the reduction of the number of states. That approach was successfully applied to nondeterministic finite automata [8], fuzzy finite automata [9, 10, 11, 25, 24] and weighted finite automata over an additively-idempotent semiring [13], and research is underway in which that approach is applied in the context of weighted automata over the max-plus semiring and the field of real numbers. The mentioned approach also plays a key role in this paper.

The second model of weighted automata that we deal with here are the so-called *crisp-deterministic weighted automata*. These are classical automata with a single initial state and deterministic transitions in which the set of terminal states is replaced by a function which assigns a terminal weight to each state. When such an automaton starts working from the initial state and performs a sequence of transitions conducted by a given input word, the weight assigned to that word is the terminal weight of the destination state. Those automata were studied for the first time in [16], in the context of fuzzy automata, and the most general definition of crisp-deterministic weighted automata was given in [17]. The name crisp-deterministic was introduced in [7] to distinguish it from a related type of automata for which the name deterministic weighted automata is used. An extensive study of crisp-deterministic weighted automata was carried out in [17], and in [7, 16, 18, 19, 20, 22] various procedures for converting a weighted finite automaton into an equivalent crisp-deterministic weighted automaton were provided. Such procedures are called *crisp-determinization*. If *we allow a crisp-deterministic weighted automaton to have an infinite set of states, as we do in this paper*, then any weighted finite automaton can be converted into an equivalent crisp-deterministic weighted automaton, and the basic problem is to perform such a conversion that will provide an equivalent crisp-deterministic weighted automaton with a finite number of states, as small as possible. For information on crisp-determinization of weighted tree automata we refer to [14, 15].

The main role in the crisp-determinization is played by the concept of the *Nerode automaton* assigned to the weighted finite automaton that is determinized. The construction of the Nerode automaton was first introduced in [16] as a counterpart to the *accessible subset construction* on which the determinization of classical nondeterministic finite automata is based. According to that construction, the states of a Nerode automaton are vectors with entries from the underlying structure of weights, but in the mentioned papers dealing with crisp-determinization, such nature of states was neglected, and the Nerode automaton was considered as an ordinary crisp-deterministic weighted automaton. If the vector nature of states is taken into account, this leads us to the third model of weighted automata that is considered here, to *weighted automata over a vector space* or *weighted automata with vector states*. Various forms of such automata were studied in [2, 3, 4, 5, 12, 21], and a related model of automata, called *automata with fuzzy states*, was studied within the framework of fuzzy automata theory (see [26] and sources cited there). The concept of a weighted automaton over a vector space discussed here differs slightly from the corresponding concepts studied in the cited articles. The first difference concerns the underlying vector space. Except in [12], in all the other mentioned articles, it is assumed that this vector space is finite-dimensional. Here we not only allow that space to be infinite-dimensional, but also introduce an extremely interesting weighted automaton over an infinite-dimensional space, the so-called *derivative*

automaton. The second difference concerns the set of states of these automata. In all the mentioned articles, except in [4], states are assumed to be all vectors from the underlying vector space V . However, in that case the set of states is always infinite and a huge number of states are unreachable from the initial state, and therefore redundant. For this reason, we take the set of states to be a subset of V , which can be both finite and infinite. The third difference relates to transition functions. In almost all cited articles, the transition functions induced by the input letters were required to be linear operators on V . In [21], a more general model of weighted automata over a vector space was proposed, where the transition functions do not have to be linear. This leads to the distinction between *linear* and *nonlinear* weighted automata over a vector space. Here we give a definition of a linear weighted automaton over a vector space which also includes the cases when the set of states is not the entire vector space and when the underlying vector space is infinite-dimensional.

This paper is the beginning of our extensive investigations of weighted automata with weights taken in the field of real numbers, and our aim here is to examine some general relations between weighted automata over vector spaces and other two models. First, by Theorem 4.1, we show that any crisp-deterministic weighted automaton can be naturally turned into a language-equivalent weighted automaton over a vector space, where the set of vector states can be any set of vectors that has the same cardinality as the set of states of that crisp-deterministic weighted automaton. Then by Theorem 4.2 we show that any finite-dimensional linear weighted automaton over a vector space can be turned into a completely language-equivalent weighted finite automaton, and conversely, any weighted finite automaton can be turned into a completely language-equivalent finite-dimensional linear weighted automaton over a vector space. Actually, we show that the previously mentioned Nerode automaton of a weighted finite automaton \mathcal{A} is a finite-dimensional linear weighted automaton over a vector space that is completely equivalent to \mathcal{A} . Theorem 4.3 gives us an elegant procedure for checking whether a given finite weighted automaton over a vector space is linear. At the end of the paper, we introduce the concept of the derivative automaton of a given word function and prove that it is a linear weighted automaton over a vector space that computes this word function and generates its prefix closure. In addition, we show that the derivative automaton is a minimal weighted automaton over a vector space which computes this word function.

2 Preliminaries

Throughout this paper, \mathbb{N} denotes the set of all natural numbers (without zero) and \mathbb{R} denotes the field of real numbers. For $i, j \in \mathbb{N}$ such that $i \leq j$ we use the notation $[i..j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$.

A *vector space* over \mathbb{R} is a triple $(V, +, \cdot)$ such that:

- * V is a non-empty set, whose members are called *vectors*;
- * $+: V \times V \rightarrow V$ given by $+: (\alpha, \beta) \mapsto \alpha + \beta$, for $\alpha, \beta \in V$, is the *vector addition* operation;
- * $\cdot: \mathbb{R} \times V \rightarrow V$ given by $\cdot: (r, \alpha) \mapsto r \cdot \alpha$, for $r \in \mathbb{R}$, $\alpha \in V$, is the *scalar multiplication* operation;
- * vector addition and scalar multiplication satisfy the following axioms:

(V1) $(V, +)$ is a commutative group,

(V2) $r \cdot (\alpha + \beta) = r \cdot \alpha + r \cdot \beta$,

(V3) $(r + s) \cdot \alpha = r \cdot \alpha + s \cdot \alpha$,

(V4) $(r \cdot s) \cdot \alpha = r \cdot (s \cdot \alpha)$,

(V5) $1 \cdot \alpha = \alpha$,

for all $r, s \in \mathbb{R}$ and $\alpha, \beta \in V$.

Note that a vector space over an arbitrary field can be defined in the same way.

The basic example of a vector space over \mathbb{R} is the vector space \mathbb{R}^n consisting of all n -tuples of real numbers, with vector addition and scalar multiplication defined coordinatewise. Another example of a vector space over \mathbb{R} that is important here is the vector space \mathbb{R}^T consisting of all functions from a set T into \mathbb{R} , with vector addition and scalar multiplication defined by $(\alpha + \beta)(t) = \alpha(t) + \beta(t)$ and $(r \cdot \alpha)(t) = r \cdot \alpha(t)$, for all $\alpha, \beta \in \mathbb{R}^T$, $r \in \mathbb{R}$ and $t \in T$. Such vector spaces are called *function spaces*.

Let V and W be vector spaces over \mathbb{R} . A function $h : V \rightarrow W$ is called a *homomorphism* or *linear transformation* of V into W if $h(\alpha + \beta) = h(\alpha) + h(\beta)$ and $h(r \cdot \alpha) = r \cdot h(\alpha)$, for all $\alpha, \beta \in V$ and $r \in \mathbb{R}$. If h is a bijective homomorphism, then it is called an *isomorphism* of V into W , and we say that V and W are *isomorphic* vector spaces. A vector space V over \mathbb{R} is said to be *finite-dimensional* if it is isomorphic to the vector space \mathbb{R}^n , for some $n \in \mathbb{N}$. In this case n is the unique natural number having this property and it is called the *dimension* of V . A vector space which is not finite-dimensional is called *infinite-dimensional*. A homomorphism (linear transformation) of a vector space V into itself is called a *linear operator* on V .

Let V be a vector space over \mathbb{R} . A *linear combination* of vectors $\alpha_1, \alpha_2, \dots, \alpha_k \in V$ is any expression of the form $r_1 \cdot \alpha_1 + r_2 \cdot \alpha_2 + \dots + r_k \cdot \alpha_k$, where $r_1, r_2, \dots, r_k \in \mathbb{R}$. For any set $S \subseteq V$, the set of all linear combinations of vectors from S is called the *span* of S and denoted by $\text{span}(S)$. In other words,

$$\text{span}(S) = \{\alpha \in V \mid (\exists k \in \mathbb{N})(\exists \alpha_1, \alpha_2, \dots, \alpha_k \in S)(\exists r_1, r_2, \dots, r_k \in \mathbb{R}) \alpha = r_1 \cdot \alpha_1 + r_2 \cdot \alpha_2 + \dots + r_k \cdot \alpha_k\}.$$

It is well-known that $\text{span}(S)$ is a vector space with vector addition and scalar multiplication inherited from V , i.e., it is a *subspace* of V .

Given natural numbers $m, n \in \mathbb{N}$. A *matrix* of type $m \times n$ with entries in the field of real numbers \mathbb{R} , or a real $m \times n$ -matrix, is defined as a mapping $M : [1..m] \times [1..n] \rightarrow \mathbb{R}$. For a pair $(i, j) \in [1..m] \times [1..n]$ the value $M(i, j)$ is called the (i, j) -entry of the matrix M . The set of all real matrices of type $m \times n$ is denoted by $\mathbb{R}^{m \times n}$. Similarly, a *vector* of length n with entries in \mathbb{R} , or real vector is defined as a mapping $v : [1..n] \rightarrow \mathbb{R}$. For each $i \in [1..n]$ the value $v(i)$ is called the i th entry or i th coordinate of the vector v . The set of all real vectors of length n is denoted by \mathbb{R}^n .

The zero matrix of type $m \times n$, denoted by $O_{m \times n}$, is a matrix of type $m \times n$ whose all entries are 0. Similarly, the zero vector of length n , denoted by o_n , is a vector of length n whose all entries are 0. For each $n \in \mathbb{N}$, a matrix of type $n \times n$ is called a *square matrix* of order n . The identity matrix of order n , denoted by I_n , is a square matrix of order n which satisfies $I_n(i, i) = 1$, for each $i \in [1..n]$, and $I_n(i, j) = 0$, for all $i, j \in [1..n]$ such that $i \neq j$. The transpose of a matrix M is denoted by M^\top . For a matrix $M \in \mathbb{R}^{m \times n}$ and $k \in [1..n]$, by $c_k(M)$ we denote the k th column vector of M .

For all pairs of matrices from $\mathbb{R}^{m \times n}$ the *matrix addition* is defined pointwise:

$$(M + N)(i, j) = M(i, j) + N(i, j), \quad (1)$$

for all $M, N \in \mathbb{R}^{m \times n}$, $i \in [1..m]$ and $j \in [1..n]$. It is an associative and commutative operation on $\mathbb{R}^{m \times n}$, and in particular, $(\mathbb{R}^{m \times n}, +, O_{m \times n})$ forms a commutative monoid. The *matrix product* is defined between matrices from $\mathbb{R}^{m \times n}$ and $\mathbb{R}^{n \times p}$ as follows: for $M \in \mathbb{R}^{m \times n}$ and $N \in \mathbb{R}^{n \times p}$ their product is a matrix $M \cdot N \in \mathbb{R}^{m \times p}$ with entries given by

$$(M \cdot N)(i, k) = \sum_{j=1}^n M(i, j) \cdot N(j, k), \quad (2)$$

for all $(i, k) \in [1..m] \times [1..p]$. The matrix product is associative whenever it is defined, i.e., $(M \cdot N) \cdot P = M \cdot (N \cdot P)$, for all $M \in \mathbb{R}^{m \times n}$, $N \in \mathbb{R}^{n \times p}$ and $P \in \mathbb{R}^{p \times q}$. In particular, $(\mathbb{R}^{n \times n}, +, \cdot, O_{n \times n}, I_n)$ is a semiring.

Given a matrix $M \in \mathbb{R}^{m \times n}$ and vectors $\mu \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^n$. When μ is treated as a matrix of type $1 \times m$ (row vector) and ν as a matrix of type $n \times 1$ (column vector), the *vector-matrix product* $\mu \cdot M$ and the *matrix-vector product* $M \cdot \nu$ are defined as matrix products. The *scalar product* or *dot product* of vectors $\mu, \nu \in \mathbb{R}^n$ is an element of \mathbb{R} given by

$$\mu \cdot \nu = \sum_{i=1}^n \mu(i) \cdot \nu(i). \quad (3)$$

A matrix $M \in \mathbb{R}^{m \times n}$ is said to be in the *row echelon form* if it satisfies the following properties:

- * If a row of M does not consist entirely of zeros, then the first nonzero entry in this row is 1. It is called a *leading 1*.
- * If there are any rows that consist entirely of zeros, then they are grouped together at the bottom of the matrix M .
- * In any two successive rows of M that do not consist entirely of zeros, the leading 1 in the lower row occurs farther to the right than the leading 1 in the higher row.

Moreover, M is said to be in the *reduced row echelon form* if, in addition to these three properties, it also satisfies the condition

- * Every column of M that contains a leading 1 has zeros everywhere else.

Every matrix $N \in \mathbb{R}^{m \times n}$ can be transformed to a row echelon form or a reduced row echelon form by applying some sequence of *elementary row operations* (multiplying a row by a nonzero scalar, interchanging two rows, and adding a multiple of one row to another). It should be noted that the reduced row echelon form of the matrix N is unique, in the sense that reducing the matrix N to the reduced row echelon form by applying any sequence of elementary row operations always yields the same matrix in the reduced row echelon form. This matrix will be denoted by $RREF(N)$. The *rank* of a matrix N , denoted by $\text{rank}(N)$, is defined as the number of nonzero rows in $RREF(N)$.

For matrices $M_1 \in \mathbb{R}^{m \times n_1}$, $M_2 \in \mathbb{R}^{m \times n_2}$, ..., $M_k \in \mathbb{R}^{m \times n_k}$, where $k, m, n_1, n_2, \dots, n_k \in \mathbb{N}$, by concatenating them from left to right we obtain a matrix $[M_1 \mid M_2 \mid \dots \mid M_k] \in \mathbb{R}^{m \times n}$, where $n = n_1 + n_2 + \dots + n_k$, which is called the *augmented matrix* (obtained from M_1, M_2, \dots, M_k).

For undefined notions and notation concerning vector spaces, vectors and matrices we refer to the book [1].

3 Three models of weighted automata

In terms of real matrices and their properties, we will investigate three models of weighted automata with weights in the field of real numbers.

3.1 Weighted finite automata

Let X be an alphabet. A *weighted finite automaton* over \mathbb{R} and X is defined as a tuple $\mathcal{A} = (A, \sigma, \delta, \tau)$, where A is a non-empty finite set, while $\sigma, \tau : A \rightarrow \mathbb{R}$ and $\delta : A \times X \times A \rightarrow \mathbb{R}$. The function δ is often replaced by the family of functions $\{\delta_x\}_{x \in X}$, where $\delta_x : A \times A \rightarrow \mathbb{R}$ is given by $\delta_x(a, b) = \delta(a, x, b)$, for all $a, b \in A$ and $x \in X$. We call A the *set of states*, σ the *initial weights function*, τ the *terminal weights function*, and δ and $\delta_x, x \in X$, the *transition weights functions*.

The behavior of the weighted finite automaton \mathcal{A} is a word function $\llbracket \mathcal{A} \rrbracket : X^* \rightarrow \mathbb{R}$ defined by

$$\llbracket \mathcal{A} \rrbracket(u) = \sum_{(a_0, a_1, \dots, a_k) \in A^{k+1}} \sigma(a_0) \cdot \delta(a_0, x_1, a_1) \cdot \delta(a_1, x_2, a_2) \cdots \delta(a_{k-1}, x_k, a_k) \cdot \tau(a_k), \quad (4)$$

for $u = x_1 x_2 \dots x_k \in X^+$, $x_1, x_2, \dots, x_k \in X$, and

$$\llbracket \mathcal{A} \rrbracket(\varepsilon) = \sum_{a \in A} \sigma(a) \cdot \tau(a). \quad (5)$$

We say that \mathcal{A} *computes* the function $\llbracket \mathcal{A} \rrbracket$.

Assume that n is the number of elements of A , i.e., $A = \{a_1, a_2, \dots, a_n\}$. In many situations, instead of as functions, it is more convenient to treat σ and τ as vectors in \mathbb{R}^n , and δ_x , $x \in X$, as $n \times n$ matrices with entries in \mathbb{R} . In other words, σ will be identified with a vector in \mathbb{R}^n whose i th coordinate is $\sigma(a_i)$, and τ will be identified with a vector in \mathbb{R}^n whose i th coordinate is $\tau(a_i)$. In order to clearly distinguish between matrices and vectors, we will use capital letters of the Latin alphabet to denote matrices, while vectors will be denoted by small letters of the Greek alphabet. Therefore, $\{M_x\}_{x \in X}$ will be a family of $n \times n$ matrices over \mathbb{R} such that the (i, j) -entry of M_x is equal to $\delta_x(a_i, a_j)$. A weighted finite automaton \mathcal{A} is then treated as a tuple $\mathcal{A} = (n, \sigma, \{M_x\}_{x \in X}, \tau)$, where we call n the *dimension*, σ the *initial weights vector*, τ the *terminal weights vector*, and M_x , $x \in X$, the *transition weights matrices* of \mathcal{A} . We call this tuple the *linear representation* of the weighted finite automaton \mathcal{A} .

When \mathcal{A} is given by the linear representation, its behavior is represented by

$$\llbracket \mathcal{A} \rrbracket(u) = \sigma \cdot M_{x_1} \cdot M_{x_2} \cdots M_{x_k} \cdot \tau = \sigma \cdot M_u \cdot \tau, \quad (6)$$

for $u = x_1 x_2 \dots x_k \in X^+$, $x_1, x_2, \dots, x_k \in X$, where $M_u = M_{x_1} \cdot M_{x_2} \cdots M_{x_k}$, and

$$\llbracket \mathcal{A} \rrbracket(\varepsilon) = \sigma \cdot \tau. \quad (7)$$

In applications of automata in the theory of discrete event systems, apart from the function $\llbracket \mathcal{A} \rrbracket$ computed by the automaton \mathcal{A} , another function plays an important role – the function $\llbracket \mathcal{A} \rrbracket_g$ generated by the automaton \mathcal{A} . For a weighted finite automaton $\mathcal{A} = (n, \sigma, \{M_x\}_{x \in X}, \tau)$ this function can be defined with:

$$\llbracket \mathcal{A} \rrbracket_g(u) = \|\sigma_u\|_\infty, \quad (8)$$

for every $u \in X^*$, where $\sigma_u = \sigma \cdot M_u$, for $u \in X^+$, and $\sigma_\varepsilon = \sigma$, while $\|\cdot\|_\infty$ denotes the *maximum norm* (called also a *uniform norm*) on \mathbb{R}^n that is given by

$$\|\alpha\|_\infty = \max_{i \in [1..n]} |\alpha_i|, \quad (9)$$

for each $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbb{R}^n$.

Let us note that in the case of conventional nondeterministic finite automata, $\llbracket \mathcal{A} \rrbracket_g$ is the language consisting of all words for which a transition is defined, i.e., of all words which "lead somewhere" (cf. [6]). A nondeterministic finite automaton can be considered as a weighted finite automaton over the two-element Boolean semiring, and then $\llbracket \mathcal{A} \rrbracket_g$ consists of all words u for which σ_u is a non-zero Boolean vector, i.e., for which

$$\max_{i \in [1..n]} |\alpha_i| = 1,$$

where $\sigma_u = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \{0, 1\}^n$. From conventional nondeterministic finite automata, such a definition was also extended to the case of fuzzy finite automata, where $\llbracket \mathcal{A} \rrbracket_g(u)$ is interpreted as the degree to which the word u leads somewhere. A similar interpretation can be given here as well, when it comes to weighted finite automata over the field of real numbers, where $\llbracket \mathcal{A} \rrbracket_g(u) = \|\sigma_u\|_\infty$ could be interpreted as the maximal probability of the existence of a transition determined by u , i.e. the probability that u leads somewhere. In order for this to be consistent with the conventional view of probability, the values for $\|\sigma_u\|_\infty$, which are always non-negative, can be translated by some monotone function (for example, by the function $1 - e^{-x}$), to values from the real unit interval $[0, 1]$.

3.2 Crisp-deterministic weighted automata

Another model of weighted automata is a *crisp-deterministic weighted automaton*, which is defined as a tuple $\mathcal{D} = (D, d_0, \Delta, \theta)$, where D is a non-empty set of states, $d_0 \in D$ is a distinguished state that is called the *initial state*, $\Delta : D \times X \rightarrow D$ is a function called the *transition function*, and $\theta : D \rightarrow \mathbb{R}$ is a function called the *terminal weights function*, or the *terminal weights vector*, if θ is considered as a vector in the space \mathbb{R}^D . Here it is not necessary that the set D is finite, so we will also allow the possibility that D is infinite. If the set of states D is finite, then \mathcal{D} is called a *finite crisp-deterministic weighted automaton*.

A finite crisp-deterministic weighted automaton can be considered as a special weighted finite automaton $\mathcal{A} = (D, \sigma, \delta, \tau)$ in which for every $a \in D$ and $x \in X$ there exists $a' = \Delta(a, x) \in D$ such that $\delta(a, x, a') = 1$, while $\delta(a, x, b) = 0$, for every $b \in D \setminus \{a'\}$, and there exists $a \in D$ such that $\sigma(a) = 1$, while $\sigma(b) = 0$, for every $b \in D \setminus \{a\}$ (then we assume that $d_0 = a$). In other words, a finite crisp-deterministic weighted automaton is a weighted finite automaton with a single initial state and a deterministic transition function, in which all weights are concentrated in the terminal weights vector.

The transition function Δ of a crisp-deterministic weighted automaton $\mathcal{D} = (D, d_0, \Delta, \theta)$ extends to a function $\Delta^* : D \times X^* \rightarrow D$ by putting $\Delta^*(a, \varepsilon) = a$ and $\Delta^*(a, ux) = \Delta(\Delta^*(a, u), x)$, for all $a \in D$, $u \in X^*$ and $x \in X$, and the behavior $\llbracket \mathcal{D} \rrbracket : X^* \rightarrow \mathbb{R}$ of \mathcal{D} is given by

$$\llbracket \mathcal{D} \rrbracket(u) = \theta(\Delta^*(d_0, u)), \quad (10)$$

for every $u \in X^*$.

From the transition function $\Delta^* : D \times X^* \rightarrow D$ we can extract a family of functions $\{\Delta_u\}_{u \in X^*}$, where $\Delta_u : D \rightarrow D$ is defined by $\Delta_u(a) = \Delta^*(a, u)$, for all $u \in X^*$ and $a \in D$. These functions will be also called *transition functions*. If $u = x_1 x_2 \dots x_k$, for $x_1, x_2, \dots, x_k \in X$, then

$$\Delta_u(a) = \Delta_{x_k}(\dots(\Delta_{x_2}(\Delta_{x_1}(a)))) ,$$

for every $a \in D$, that is, Δ_u is the composition $\Delta_u = \Delta_{x_1} \Delta_{x_2} \dots \Delta_{x_k}$ of transition functions $\Delta_{x_1}, \Delta_{x_2}, \dots, \Delta_{x_k}$.

3.3 Weighted automata over a vector space

Let V be a vector space over the field \mathbb{R} of real numbers. The third model of weighted automata is a *weighted automaton over a vector space (with vector states)*, defined as a tuple $\mathcal{A} = (S, \sigma, \delta, \Theta)$, where $S \subseteq V$ is a nonempty set of vectors, called the *set of vector states*, $\sigma \in S$ is a vector called the *initial vector state*, $\delta : S \times X \rightarrow S$ is a deterministic *transition function*, and $\Theta : S \rightarrow \mathbb{R}$ is a function called the *terminal weights function*. Here we also allow S to be infinite. Furthermore, in some sources S is taken to be the entire space V , but here we allow S to be only a subset of the space V to enable it to be finite. If the set S of vector states is finite, then \mathcal{A} is called a *finite weighted automaton over a vector space*, and

if V is a finite-dimensional vector space of dimension n , then \mathcal{A} is also said to be a *finite-dimensional weighted automaton over a vector space of dimension n* . If the cardinality of the set of states of \mathcal{A} is less than or equal to the cardinality of the set of states of any other weighted automaton over the vector space V , that we say that \mathcal{A} is a *minimal weighted automaton over the vector space V* .

As with crisp-deterministic weighted automata, δ can be extended to a function $\delta^* : S \times X^* \rightarrow S$ by $\delta^*(\alpha, \varepsilon) = \alpha$ and $\delta^*(\alpha, ux) = \delta(\delta^*(\alpha, u), x)$, for all $\alpha \in S$, $u \in X^*$, $x \in X$, and the function δ^* determines a family of functions $\{\delta_u\}_{u \in X^*}$, where $\delta_u : S \rightarrow S$ is defined by $\delta_u(\alpha) = \delta^*(\alpha, u)$, for all $u \in X^*$ and $\alpha \in S$. These functions are also called *transition functions*. If $u = x_1x_2 \dots x_k$, for $x_1, x_2, \dots, x_k \in X$, then

$$\delta_u(\alpha) = \delta_{x_k}(\dots(\delta_{x_2}(\delta_{x_1}(\alpha)))) ,$$

for every $\alpha \in V$, that is, δ_u is the composition $\delta_u = \delta_{x_1} \delta_{x_2} \dots \delta_{x_k}$ of transition functions $\delta_{x_1}, \delta_{x_2}, \dots, \delta_{x_k}$. Then \mathcal{A} can be equivalently represented as a tuple $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$. In addition, for every $u \in X^*$, with σ_u we denote a vector from S given by $\sigma_u = \delta^*(\sigma, u) = \delta_u(\sigma)$.

The behavior $\llbracket \mathcal{A} \rrbracket : X^* \rightarrow \mathbb{R}$ of the weighted automaton \mathcal{A} over a vector space V is given by

$$\llbracket \mathcal{A} \rrbracket(u) = \Theta(\delta^*(\sigma, u)), \quad (11)$$

for every $u \in X^*$. On the other hand, the function $\llbracket \mathcal{A} \rrbracket_g$ generated by \mathcal{A} is defined with

$$\llbracket \mathcal{A} \rrbracket_g(u) = \|\delta^*(\sigma, u)\| = \|\delta_u(\sigma)\|, \quad (12)$$

for each $u \in X^*$, where $\|\cdot\|$ denotes some norm on the vector space V . If V is a finite-dimensional space we will assume that $\|\cdot\|$ is the maximum norm $\|\cdot\|_\infty$ (see (9)), and if $V = \mathbb{R}^T$ is some function space (later we will consider the function space \mathbb{R}^{X^*}), then we will assume that $\|\cdot\|$ is the *supremum norm* $\|\cdot\|_\infty$ (also called the *uniform norm*) that is given by

$$\|f\|_\infty = \sup_{t \in T} |f(t)|,$$

for every $f : T \rightarrow \mathbb{R}$ (clearly, for $T = [1..n]$ we obtain (9), i.e., the maximum norm).

4 Relationships between different types of weighted automata

Let \mathcal{A} and \mathcal{B} be weighted automata of any of the three types discussed in the previous section, where they can be of different types. If $\llbracket \mathcal{A} \rrbracket = \llbracket \mathcal{B} \rrbracket$, then \mathcal{A} and \mathcal{B} are said to be *language-equivalent*. On the other hand, if each of these automata is a weighted finite automaton or a weighted automaton over a vector space, where they do not have to be of the same type, and if $\llbracket \mathcal{A} \rrbracket = \llbracket \mathcal{B} \rrbracket$ and $\llbracket \mathcal{A} \rrbracket_g = \llbracket \mathcal{B} \rrbracket_g$, then \mathcal{A} and \mathcal{B} are said to be *completely language-equivalent*.

Theorem 4.1 *Let $\mathcal{D} = (D, d_0, \Delta, \theta)$ be a crisp-deterministic weighted automaton over \mathbb{R} , let V be a vector space and let $S \subseteq V$ be a set of vectors which has the same cardinality as D .*

Then \mathcal{D} can be turned into a language-equivalent weighted automaton over the vector space V having S as its set of states.

Proof. Let $\phi : S \rightarrow D$ be an arbitrary bijective function between S and D . Then we can define an initial vector state $\sigma \in S$ by

$$\sigma = \phi^{-1}(d_0).$$

Let $\{\delta_x\}_{x \in X}$ be a family of transition functions defined in the following way: For each $x \in X$,

$$\delta_x(\alpha) = \phi^{-1}(\Delta(\phi(\alpha), x)), \quad \text{for every } \alpha \in S.$$

Let the terminal weights function $\Theta : S \rightarrow \mathbb{R}$ be defined by

$$\Theta(\alpha) = \theta(\phi(\alpha)), \quad \text{for every } \alpha \in S.$$

Then $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$ is a weighted automaton over the vector space V having S as its set of states.

Clearly, for each $\alpha \in S$ and $u \in X^*$ we have

$$\delta_u(\alpha) = \phi^{-1}(\Delta(\phi(\alpha), u)).$$

Furthermore, for every $u \in X^*$ the following holds

$$\begin{aligned} \llbracket \mathcal{A} \rrbracket(u) &= \Theta(\delta^*(\sigma, u)) = \Theta(\delta_u(\sigma)) = \theta(\phi(\delta_u(\sigma))) = \theta(\phi(\phi^{-1}(\Delta(\phi(\sigma), u)))) = \\ &= \theta(\phi(\phi^{-1}(\Delta(\phi(\phi^{-1}(d_0)), u)))) = \theta(\Delta(d_0, u)) = \llbracket \mathcal{D} \rrbracket(u) \end{aligned}$$

and therefore, \mathcal{A} and \mathcal{D} are language-equivalent. \square

T. Li, G. Rabusseau and D. Precup in [21] defined a *nonlinear weighted finite automaton* over the field of real numbers \mathbb{R} as a tuple $(\sigma, \{\delta_x\}_{x \in X}, \Theta)$, where $\sigma \in \mathbb{R}^n$ is a vector of initial weights, $\{\delta_x\}_{x \in X}$ is a family of transformations such that $\delta_x : \mathbb{R}^n \rightarrow \mathbb{R}^n$, for each $x \in X$, which are called transition functions, and $\theta : \mathbb{R}^n \rightarrow \mathbb{R}$ is a termination function. This definition is almost identical to our definition of a weighted automaton over a vector space. The difference is that the set S of vector states is not explicitly stated in the mentioned paper, but we can assume that S is the smallest set of vectors from \mathbb{R}^n that contains σ and is closed for all transformations from the family $\{\delta_x\}_{x \in X}$, that is, $S = \{\sigma_u\}_{u \in X^*}$, where $\sigma_u = \delta^*(\sigma, u) = \delta_u(\sigma)$, for each $u \in X^*$. Another difference is that the transformations $\{\delta_x\}_{x \in X}$ are defined on \mathbb{R}^n , but nothing changes significantly if we replace them with their restrictions on S . The third difference is that Lee, Rabusseau and Precup considered automata over the finite-dimensional vector space \mathbb{R}^n , while in our definition we provide the possibility that the underlying vector space V be also infinite-dimensional. Therefore, the concepts of a nonlinear weighted finite automaton and a weighted automaton over a vector space are almost the same. Let us note that the adjective "finite" in the name of nonlinear weighted finite automata does not refer to the finiteness of the set S of vector states, as in our definition, but to the finite dimension of the space \mathbb{R}^n . In addition, regardless of the adjective nonlinear in the name, in the nonlinear weighted finite automaton among the transformations δ_x , $x \in X$, there can be both linear and nonlinear ones.

Here, a weighted automaton $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$ over a vector space V is defined to be *linear* if for any $x \in X$ the function δ_x is the restriction of some linear operator $\delta'_x : \text{span}(S) \rightarrow \text{span}(S)$ to the set S , and also, Θ is the restriction of some linear functional (linear form) $\Theta' : \text{span}(S) \rightarrow \mathbb{R}$ to the set S , i.e.,

$$\delta'_x(s\alpha + t\beta) = s\delta'_x(\alpha) + t\delta'_x(\beta), \quad \Theta'(s\alpha + t\beta) = s\Theta'(\alpha) + t\Theta'(\beta), \quad (13)$$

for all $x \in X$, $\alpha, \beta \in \text{span}(S)$ and $s, t \in \mathbb{R}$. Otherwise, if some of the mappings δ_x , $x \in X$, and Θ can not be represented in this way, then \mathcal{A} is said to be a *nonlinear weighted automaton over a vector space*. If $V \subseteq \mathbb{R}^n$, for some $n \in \mathbb{N}$, then \mathcal{A} is linear if and only if for each $x \in X$ there is a matrix $M_x \in \mathbb{R}^{n \times n}$ such that $\delta_x(\alpha) = \alpha \cdot M_x$, for each $\alpha \in S$, and there is also a vector $\tau \in \mathbb{R}^n$ such that $\Theta(\alpha) = \alpha \cdot \tau$, for each $\alpha \in S$ (here $\alpha \cdot \tau$ denotes the scalar product of α and τ).

Let us note that our linear weighted automata over a vector space are almost identical to automata studied by Balle, Gourdeau and Panangaden in [3] (which are called there only weighted finite automata), the only difference is that there V was assumed to be a finite-dimensional space and $S = V$. However, even this small difference concerning the set of vector states can be significant. Namely, let $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$ be any linear weighted automaton over the vector space $V = \mathbb{R}^n$ with the set of vector states $S \subset V$ such that $\text{span}(S) \neq V$, for each $x \in X$ let δ_x be the restriction of some linear operator $\delta'_x : \text{span}(S) \rightarrow \text{span}(S)$ to the set S , and let Θ be the restriction of some linear functional $\Theta' : \text{span}(S) \rightarrow \mathbb{R}$ to the set S . Then we can easily extend any δ'_x to an operator $\delta''_x : V \rightarrow V$ which is not linear, for example, by taking δ''_x to coincide with δ'_x on $\text{span}(S)$ and

$$\delta''_x \begin{pmatrix} s_1 & s_2 & \dots & s_n \end{pmatrix} = \begin{pmatrix} s_1^2 & s_2^2 & \dots & s_n^2 \end{pmatrix},$$

for every $\begin{pmatrix} s_1 & s_2 & \dots & s_n \end{pmatrix} \in V \setminus \text{span}(S)$, and we can extend Θ' to a non-linear function $\Theta'' : V \rightarrow \mathbb{R}$. Therefore, $\mathcal{A}'' = (V, \sigma, \{\delta''_x\}_{x \in X}, \Theta'')$ is a non-linear weighted automaton over the vector space V with the set of vector states V , but if we assume that the set of vector states is S , then \mathcal{A}'' becomes linear.

The following theorem explains the connection between finite-dimensional linear weighted automata over a vector space and weighted finite automata.

Theorem 4.2 *Every finite-dimensional linear weighted automaton over a vector space can be turned into a completely language-equivalent weighted finite automaton.*

Conversely, every weighted finite automaton can be turned into a completely language-equivalent finite-dimensional linear weighted automaton over a vector space.

Proof. Let $\mathcal{A} = (S, \sigma, \delta, \Theta)$ be a finite-dimensional linear weighted automaton over a vector space V of dimension n . Since \mathcal{A} is linear, we have that for each $x \in X$ there exists a matrix $M_x \in \mathbb{R}^{n \times n}$ such that $\sigma \cdot M_x = \delta(\sigma, x)$, and moreover, there exists a vector $\tau \in \mathbb{R}^n$ such that $\Theta(\sigma) = \sigma \cdot \tau$. In this way, we obtain a weighted finite automaton \mathcal{A}' which is given by the linear representation $\mathcal{A}' = (n, \sigma, \{M_x\}_{x \in X}, \tau)$.

Now, for every $u \in X^+$ such that $u = x_1 x_2 \dots x_k$, where $x_1, x_2, \dots, x_k \in X$, we have

$$\llbracket \mathcal{A}' \rrbracket(u) = \sigma \cdot M_{x_1} \cdot M_{x_2} \cdot \dots \cdot M_{x_k} \cdot \tau = \sigma \cdot M_u \cdot \tau = \delta^*(\sigma, u) \cdot \tau = \Theta(\delta^*(\sigma, u)) = \llbracket \mathcal{A} \rrbracket(u),$$

and

$$\llbracket \mathcal{A}' \rrbracket(\varepsilon) = \sigma \cdot \tau = \delta^*(\sigma, \varepsilon) \cdot \tau = \Theta(\delta^*(\sigma, \varepsilon)) = \llbracket \mathcal{A} \rrbracket(\varepsilon).$$

Finally, we have that

$$\llbracket \mathcal{A}' \rrbracket_g(u) = \|\sigma_u\|_\infty = \|\sigma \cdot M_u\|_\infty = \|\delta^*(\sigma, u)\|_\infty = \llbracket \mathcal{A} \rrbracket_g(u),$$

for every $u \in X^*$.

Therefore, we have proved that the finite-dimensional linear weighted automaton \mathcal{A} is completely language-equivalent to the weighted finite automaton \mathcal{A}' .

Conversely, let $\mathcal{A} = (A, \sigma, \delta, \tau)$ be a weighted finite automaton with n states. We define a weighted automaton $\mathcal{A}_N = (S_N, \sigma^N, \delta^N, \Theta^N)$ over the vector space \mathbb{R}^n in the following way: we set

$$S_N = \{\sigma_u \mid u \in X^*\}, \quad \sigma^N = \sigma,$$

and we define functions $\delta^N : S_N \times X \rightarrow S_N$ and $\Theta^N : S_N \rightarrow \mathbb{R}$ by

$$\delta^N(\sigma_u, x) = \sigma_u \cdot \delta_x = \sigma_{ux}, \quad \Theta^N(\sigma_u) = \sigma_u \cdot \tau,$$

for every $u \in X^*$. The automaton \mathcal{A}_N is obviously well-defined and is called in [17] the *Nerode automaton* of the automaton \mathcal{A} . It remains to prove that the Nerode automaton \mathcal{A}_N is completely language-equivalent to the original weighted finite automaton \mathcal{A} .

For an arbitrary word $u \in X^*$ we have that

$$\llbracket \mathcal{A}_N \rrbracket(u) = \Theta^N((\delta^N)^*(\sigma, u)) = \Theta^N(\sigma_u) = \sigma_u \cdot \tau = \llbracket \mathcal{A} \rrbracket(u).$$

and also,

$$\llbracket \mathcal{A}_N \rrbracket_g(u) = \|(\delta^N)^*(\sigma, u)\|_\infty = \|\sigma_u\|_\infty = \llbracket \mathcal{A} \rrbracket_g(u).$$

Therefore, the Nerode automaton \mathcal{A}_N of \mathcal{A} is completely language-equivalent to \mathcal{A} . \square

Let $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$ be a finite weighted automaton over the vector space $V = \mathbb{R}^n$, and let us assume that $S = \{\alpha_1, \alpha_2, \dots, \alpha_m\}$ and $X = \{x_1, x_2, \dots, x_s\}$. Let us form $m \times n$ -matrices N and N_{x_i} , $i \in [1..s]$, and a column vector ($1 \times m$ -matrix) ϑ such that rows in N are $\alpha_1, \alpha_2, \dots, \alpha_m$, rows in N_{x_i} are $\delta_{x_i}(\alpha_1), \delta_{x_i}(\alpha_2), \dots, \delta_{x_i}(\alpha_m)$, and entries in ϑ are $\Theta(\alpha_1), \Theta(\alpha_2), \dots, \Theta(\alpha_m)$, in that order, i.e.

$$N = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix}, \quad N_{x_i} = \begin{bmatrix} \delta_{x_i}(\alpha_1) \\ \delta_{x_i}(\alpha_2) \\ \vdots \\ \delta_{x_i}(\alpha_m) \end{bmatrix}, \quad i \in [1..s], \quad \vartheta = \begin{bmatrix} \Theta(\alpha_1) \\ \Theta(\alpha_2) \\ \vdots \\ \Theta(\alpha_m) \end{bmatrix}$$

We will call N the *state matrix* and ϑ the *terminal vector*, while for each $i \in [1..s]$, we call N_{x_i} the *destination matrix* corresponding to the input letter x_i .

The following theorem provides a procedure for testing the linearity of a finite weighted automaton over a finite-dimensional vector space.

Theorem 4.3 *Let $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$ be a finite weighted automaton over the vector space $V \subseteq \mathbb{R}^n$. Then \mathcal{A} is linear if and only if the matrix N has the same rank as the augmented matrix*

$$[N \mid N_{x_1} \mid N_{x_2} \mid \dots \mid N_{x_s} \mid \vartheta].$$

Proof. First we prove that \mathcal{A} is linear if and only if each of the following equations is solvable

$$N \cdot X_1 = N_{x_1}, \quad N \cdot X_2 = N_{x_2}, \quad \dots, \quad N \cdot X_s = N_{x_s}, \quad N \cdot \chi = \vartheta, \quad (14)$$

where X_1, X_2, \dots, X_s are unknown $n \times n$ -matrices, and χ is an unknown vector taking values in \mathbb{R}^n .

Assume that \mathcal{A} is linear, i.e., that there are matrices $M_{x_i} \in \mathbb{R}^{n \times n}$, for each $i \in [1..s]$, and a vector $\tau \in \mathbb{R}^n$ such that

$$\delta_{x_i}(\alpha_j) = \alpha_j \cdot M_{x_i}, \quad \text{for all } i \in [1..s], \quad j \in [1..m], \quad \Theta(\alpha_j) = \alpha_j \cdot \tau, \quad \text{for each } j \in [1..m]. \quad (15)$$

According to the well-known row rule for matrix multiplication we obtain that

$$N \cdot M_{x_i} = \begin{bmatrix} \alpha_1 \cdot M_{x_i} \\ \alpha_2 \cdot M_{x_i} \\ \vdots \\ \alpha_m \cdot M_{x_i} \end{bmatrix} = \begin{bmatrix} \delta_{x_i}(\alpha_1) \\ \delta_{x_i}(\alpha_2) \\ \vdots \\ \delta_{x_i}(\alpha_m) \end{bmatrix} = N_{x_i}, \quad N \cdot \tau = \begin{bmatrix} \alpha_1 \cdot \tau \\ \alpha_2 \cdot \tau \\ \vdots \\ \alpha_m \cdot \tau \end{bmatrix} = \begin{bmatrix} \Theta(\alpha_1) \\ \Theta(\alpha_2) \\ \vdots \\ \Theta(\alpha_m) \end{bmatrix} = \vartheta, \quad (16)$$

for every $i \in [1..s]$, and we conclude that $X_1 = M_{x_1}, X_2 = M_{x_2}, \dots, X_s = M_{x_s}$ and $\chi = \tau$ are solutions of equations from (14).

Conversely, let any of the equations from (14) is solvable, and assume that $X_i = M_{x_i}$, for $i \in [1..s]$, and $\chi = \tau$, are solutions to these equations. From there we obtain that (16) holds, which further implies that (15) holds, and therefore, \mathcal{A} is a linear weighted automaton over a vector space V .

Next we prove that each of the equations from (14) is solvable if and only if the rank of N is equal to the rank of the augmented matrix $N^* = [N \mid N_{x_1} \mid N_{x_2} \mid \dots \mid N_{x_s} \mid \vartheta]$. Let R be the reduced row echelon form of the augmented matrix N^* , and let R be partitioned as follows:

$$R = [R_0 \mid R_1 \mid R_2 \mid \dots \mid R_s \mid \nu]$$

where $R_0, R_1, R_2, \dots, R_s$ are $m \times n$ -matrices and ν is a column vector of dimension m . Then R_0 is the reduced row echelon form of N , for each $i \in [1..s]$, $[R_0 \mid R_i]$ is the reduced row echelon form of $[N \mid N_{x_i}]$, and $[R_0 \mid \nu]$ is the reduced row echelon form of $[N \mid \vartheta]$.

For an arbitrary $i \in [1..s]$ we have that the equation $N \cdot X_i = N_{x_i}$ is solvable if and only if the equation $N \cdot c_k(X_i) = c_k(N_{x_i})$ is solvable for every $k \in [1..n]$. On the other hand, for every $k \in [1..n]$ we have that $N \cdot c_k(X_i) = c_k(N_{x_i})$ is solvable if and only if $\text{rank}(N) = \text{rank}([N \mid c_k(N_{x_i})])$, and this holds if and only if for every $j \in [1..m]$ for which the j th row of R_0 is the zero vector it follows that the j th coordinate of $c_k(N_{x_i})$ is equal to zero. Similarly, the equation $N \cdot \chi = \vartheta$ is solvable if and only if for every $j \in [1..m]$ for which the j th row of R_0 is the zero vector it follows that the j th coordinate of ϑ is equal to zero.

Therefore, we conclude that all equations in (14) are solvable if and only if for every $j \in [1..m]$ for which the j th row of R_0 is the zero vector, we have that all remaining entries in the j th row of the matrix $R = [R_0 \mid R_1 \mid R_2 \mid \dots \mid R_s \mid \nu]$ are equal to zero, and this is equivalent to $\text{rank}(N) = \text{rank}(N^*)$.

Let us note that if R_0 does not have zero rows then all equations in (14) are solvable if and only if

$$\text{rank}(N) = \text{rank}(N^*) = \text{rank}([N \mid N_{x_i}]) = \text{rank}([N \mid \vartheta]) = \text{rank}([N \mid c_k(N_{x_i})]) = m (\leq n),$$

for all $i \in [1..s]$ and $k \in [1..n]$. This completes the proof of the theorem. \square

Example 4.4 Let $\mathcal{A} = (S, \sigma, \{\delta_x\}_{x \in X}, \Theta)$ be a finite weighted automaton with three vector states over the vector space \mathbb{R}^2 and an alphabet $X = \{x, y\}$, where the vector states $\sigma = \alpha_1, \alpha_2$ and α_3 , and the terminal vector ϑ are given with

$$\sigma = \alpha_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \alpha_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad \alpha_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad \vartheta = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

while the transition graph is the one given in Figure 1.

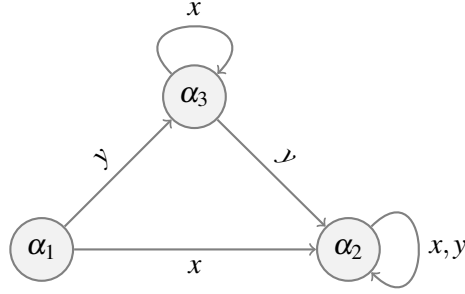
Then the augmented matrix $N^* = [N \mid N_x \mid N_y \mid \vartheta]$ and its reduced row echelon form $RREF(N^*)$ are given by

$$N^* = \left[\begin{array}{cc|cc|cc|c} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{array} \right], \quad RREF(N^*) = \left[\begin{array}{cc|cc|cc|c} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & -1 & -1 & 1 \end{array} \right],$$

and it is clear that $\text{rank}(N) = 2 \neq 3 = \text{rank}(N^*)$. From there we get that \mathcal{A} is nonlinear.

Let us also note that the equations (14) become

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Figure 1: The transition graph of the automaton \mathcal{A} from Example 4.4

It can be seen that none of these equations has a solution.

For a word function $f : X^* \rightarrow \mathbb{R}$ and a word $u \in X^*$, a word function $f_u : X^* \rightarrow \mathbb{R}$ defined by

$$f_u(v) = f(uv), \quad \text{for every } v \in X^*, \quad (17)$$

is called the *derivative* of f with respect to u . Derivatives of conventional languages are also known as *right quotients*, *quotients* or *residuals* of languages.

We define a weighted automaton $\mathcal{A}_f = (S_f, \sigma^f, \delta^f, \Theta^f)$ over a vector space \mathbb{R}^{X^*} as follows: the set S_f of vector states is given by $S_f = \{f_u \mid u \in X^*\}$, $\sigma^f = f$, and functions $\delta^f : S_f \times X \rightarrow S_f$ and $\Theta^f : S_f \rightarrow \mathbb{R}$ are given by

$$\delta^f(g, x) = g_x, \quad \Theta^f(g) = g(\varepsilon), \quad \text{for all } g \in S_f \text{ and } x \in X. \quad (18)$$

It is clear that \mathcal{A}_f is well-defined, and we will call it the *derivative automaton* of the word function f .

For a word function $f : X^* \rightarrow \mathbb{R}$, the *prefix closure* of f is a word function $\bar{f} : X^* \rightarrow \mathbb{R}$ defined by

$$\bar{f}(u) = \sup_{v \in X^*} |f(uv)| = \|f_u\|_\infty, \quad (19)$$

for every $u \in X^*$.

Theorem 4.5 *The derivative automaton \mathcal{A}_f of a word function $f \in \mathbb{R}^{X^*}$ is a linear weighted automaton over the vector space \mathbb{R}^{X^*} that computes f and generates the prefix closure \bar{f} of f .*

In addition, \mathcal{A}_f is a minimal weighted automaton over a vector space which computes f .

Proof. For the sake of simplicity, let us assume that $\mathbb{R}^{X^*} = V$.

First, we prove that \mathcal{A}_f is linear. To that end, define functions $\delta_x : V \rightarrow V$, for every $x \in X$, and $\Theta : V \rightarrow \mathbb{R}$ as follows: $\delta_x(g) = g_x$ and $\Theta(g) = g(\varepsilon)$, for all $g \in V$ and $x \in X$. Further, consider arbitrary $g, h \in V$ and $s, t \in \mathbb{R}$. Then for arbitrary $x \in X$ and $u \in X^*$ we have that

$$(sg + th)_x(u) = (sg + th)(xu) = sg(xu) + th(xu) = sg_x(u) + th_x(u) = (sg_x + th_x)(u),$$

so we conclude that $(sg + th)_x = sg_x + th_x$, and now we have that

$$\begin{aligned} \delta_x(sg + th) &= (sg + th)_x = sg_x + th_x = s\delta_x(g) + t\delta_x(h), \\ \Theta(sg + th) &= (sg + th)(\varepsilon) = sg(\varepsilon) + th(\varepsilon) = s\Theta(g) + t\Theta(h). \end{aligned}$$

Hence, δ_x , $x \in X$, and Θ are linear operators on V , and it is clear that δ_x^f (where $\delta_x^f(g) = \delta^f(g, x)$) is the restriction of δ_x to S_f , for each $x \in X$, and Θ^f is the restriction of Θ on S_f . This means that \mathcal{A}_f is a linear weighted automaton over the vector space V . Next, for an arbitrary $u \in X^*$ we have that

$$\begin{aligned} \llbracket \mathcal{A}_f \rrbracket(u) &= \Theta((\delta^f)^*(\sigma^f, u)) = \Theta((\delta^f)^*(f, u)) = \Theta(f_u) = f_u(\varepsilon) = f(u\varepsilon) = f(u), \\ \llbracket \mathcal{A}_f \rrbracket_g(u) &= \|(\delta^f)^*(\sigma^f, u)\|_\infty = \|(\delta^f)^*(f, u)\|_\infty = \|f_u\|_\infty = \bar{f}(u), \end{aligned}$$

which means that the automaton \mathcal{A}_f computes f and generates \bar{f} .

Let $\mathcal{A} = (S, \sigma, \delta, \Theta)$ be an arbitrary weighted automaton over a vector space that computes f , and let $S' = \{\sigma_u \mid u \in X^*\} \subseteq S$. Define a function $\phi : S' \rightarrow S_f$ by putting that $\phi(\sigma_u) = f_u$, for every $u \in X^*$. First we prove that ϕ is well-defined, i.e., that for any $u, v \in X^*$, from $\sigma_u = \sigma_v$ it follows $f_u = f_v$. Thus, consider $u, v \in X^*$ such that $\sigma_u = \sigma_v$, and an arbitrary $w \in X^*$. Then

$$\begin{aligned} f_u(w) &= f(uw) = \llbracket \mathcal{A} \rrbracket(uw) = \Theta(\delta^*(\sigma, uw)) = \Theta(\delta^*(\delta^*(\sigma, u), w)) \\ &= \Theta(\delta^*(\sigma_u, w)) = \Theta(\delta^*(\sigma_v, w)) = \Theta(\delta^*(\delta^*(\sigma, v), w)) \\ &= \Theta(\delta^*(\sigma, vw)) = \llbracket \mathcal{A} \rrbracket(vw) = f(vw) = f_v(w), \end{aligned}$$

so we conclude that $f_u = f_v$. Therefore, we get that ϕ is a well-defined function, and it is obvious that ϕ is surjective. This means that the cardinality of S_f is less than or equal to the cardinality of S' , which is less than or equal to the cardinality of S . From there, we conclude that \mathcal{A}_f is a minimal weighted automaton over a vector space which computes f . \square

References

- [1] Howard Anton & Robert C. Busby (2023): *Contemporary Linear Algebra*, 2nd edition. John Wiley & Sons.
- [2] Borja Balle, Pascale Gourdeau & Prakash Panangaden (2017): *Bisimulation metrics for weighted automata*. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn & Anca Muscholl, editors: *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, 80, Leibniz International Proceedings in Informatics (LIPIcs), pp. 103:1–103:14, doi:10.4230/LIPIcs.ICALP.2017.103.
- [3] Borja Balle, Pascale Gourdeau & Prakash Panangaden (2022): *Bisimulation metrics and norms for real-weighted automata*. *Information and Computation* 282, p. 104649, doi:10.1016/j.ic.2020.104649.
- [4] Filippo Bonchi, Marcello Bonsangue, Michele Boreale, Jan Rutten & Alexandra Silva (2012): *A coalgebraic perspective on linear weighted automata*. *Information and Computation* 211, pp. 77–105, doi:10.1016/j.ic.2011.12.002.
- [5] Michele Boreale (2009): *Weighted bisimulation in linear algebraic form*. In M. Bravetti & G. Zavattaro, editors: *CONCUR 2009 – Concurrency Theory, 20th Intern. Conference, Lecture Notes in Computer Science* 5710, Springer, Bologna, Italy, pp. 163–177, doi:10.1007/978-3-642-04081-8_12.
- [6] Christos G. Cassandras & Stéphane Lafortune (2008): *Introduction to Discrete Event Systems*. Springer, doi:10.1007/978-0-387-68612-7.
- [7] Miroslav Ćirić, Manfred Droste, Jelena Ignjatović & Heiko Vogler (2010): *Determinization of weighted finite automata over strong bimonoids*. *Information Sciences* 180, pp. 3497–3520, doi:10.1016/j.ins.2010.05.020.
- [8] Miroslav Ćirić, Jelena Ignjatović, Milan Bašić & Ivana Jančić (2014): *Nondeterministic automata: equivalence, bisimulations, and uniform relations*. *Information Sciences*, pp. 185–218, doi:10.1016/j.ins.2013.07.029.
- [9] Miroslav Ćirić, Jelena Ignjatović, Nada Damljanović & Milan Bašić (2012): *Bisimulations for fuzzy automata*. *Fuzzy Sets and Systems* 186, pp. 100–139, doi:10.1016/j.fss.2011.07.003.

- [10] Miroslav Ćirić, Jelena Ignjatović, Ivana Jančić & Nada Damljanović (2012): *Computation of the greatest simulations and bisimulations between fuzzy automata*. *Fuzzy Sets and Systems* 208, pp. 22–42, doi:10.1016/j.fss.2012.05.006.
- [11] Miroslav Ćirić, Aleksandar Stamenković, Jelena Ignjatović & Tatjana Petković (2010): *Fuzzy relation equations and reduction of fuzzy automata*. *Journal of Computer and System Sciences* 76, pp. 609–633, doi:10.1016/j.jcss.2009.10.015.
- [12] Thomas Colcombet & Daniela Petrisan (2017): *Automata and minimization*. *ACM SIGLOG News* 4(2), pp. 4–27, doi:10.1145/3090064.3090066.
- [13] Nada Damljanović, Miroslav Ćirić & Jelena Ignjatović (2014): *Bisimulations for weighted automata over an additively idempotent semiring*. *Theoretical Computer Science* 534, pp. 86–100, doi:10.1016/j.tcs.2014.02.032.
- [14] Manfred Droste, Zoltán Fülöp, Dávid Kószó & Heiko Vogler (2020): *Crisp-determinization of weighted tree automata over additively locally finite and past-finite monotonic strong bimonoids is decidable*. In Galina Jirásková & Giovanni Pighizzini, editors: *DCFS 2020, Lecture Notes in Computer Science* 12442, pp. 39–51, doi:10.1007/978-3-030-62536-8_4.
- [15] Zoltán Fülöp, Dávid Kószó & Heiko Vogler (2021): *Crisp-determinization of weighted tree automata over strong bimonoids*. *Discrete Mathematics and Theoretical Computer Science* 23(1), doi:10.46298/dmtcs.5943.
- [16] Jelena Ignjatović, Miroslav Ćirić & Stojan Bogdanović (2008): *Determinization of fuzzy automata with membership values in complete residuated lattices*. *Information Sciences* 178, pp. 164–180, doi:10.1016/j.ins.2007.08.003.
- [17] Jelena Ignjatović, Miroslav Ćirić, Stojan Bogdanović & Tatjana Petković (2010): *Myhill-Nerode type theory for fuzzy languages and automata*. *Fuzzy Sets and Systems* 161, pp. 1288–1324, doi:10.1016/j.fss.2009.06.007.
- [18] Zorana Jančić & Miroslav Ćirić (2014): *Brzozowski type determinization for fuzzy automata*. *Fuzzy Sets and Systems* 249, pp. 73–82, doi:10.1016/j.fss.2014.02.021.
- [19] Zorana Jančić, Jelena Ignjatović & Miroslav Ćirić (2011): *An improved algorithm for determinization of weighted and fuzzy automata*. *Information Sciences* 181, pp. 1358–1368, doi:10.1016/j.ins.2010.12.008.
- [20] Zorana Jančić, Ivana Micić, Jelena Ignjatović & Miroslav Ćirić (2016): *Further improvements of determinization methods for fuzzy finite automata*. *Fuzzy Sets and Systems* 301, pp. 79–102, doi:10.1016/j.fss.2015.11.019.
- [21] Tianyu Li, Guillaume Rabusseau & Doina Precup (2018): *Nonlinear weighted finite automata*. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research* 84, pp. 679–688.
- [22] Ivana Micić, Zorana Jančić, Jelena Ignjatović & Miroslav Ćirić (2015): *Determinization of fuzzy automata by means of the degrees of language inclusion*. *IEEE Transactions on Fuzzy Systems* 23(6), pp. 2144–2153, doi:10.1109/TFUZZ.2015.2404348.
- [23] Marcel-Paul Schützenberger (1961): *On the definition of a family of automata*. *Information and Control*, pp. 245–270, doi:10.1016/S0019-9958(61)80020-X.
- [24] Aleksandar Stamenković, Miroslav Ćirić & Milan Bašić (2018): *Ranks of fuzzy matrices. Applications in state reduction of fuzzy automata*. *Fuzzy Sets and Systems* 333, pp. 124–139, doi:10.1016/j.fss.2017.05.028.
- [25] Aleksandar Stamenković, Miroslav Ćirić & Jelena Ignjatović (2014): *Reduction of fuzzy automata by means of fuzzy quasi-orders*. *Information Sciences* 275, pp. 168–198, doi:10.1016/j.ins.2014.02.028.
- [26] Aleksandar Stamenković, Miroslav Ćirić & Jelena Ignjatović (2015): *Different models of automata with fuzzy states*. *Facta Universitatis, Series Mathematics and Informatics* 30(3), pp. 235–253.