Universiteit Leiden
The Netherlands

Opleiding Informatica

Extending methods for approximate determinization of weighted automata to other semirings

Jakob Wuhrer
s2045885

Supervisors:
Marcello Bonsangue (LIACS) & Peter Bruin (MI)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS) - liacs.leidenuniv.nl
Leiden Mathematisch Instituut (MI) - math.leidenuniv.nl

2022-08-21

Abstract

Weighted finite automata are a generalization of finite automata. Whereas finite automata can easily be determinized using the superset construction, determinization of weighted finite automata makes for a significant challenge. There are even weighted finite automata that cannot be determinized. In this thesis, we aim to provide a way to approximately determinize weighted automata over a variety of different semirings. We will attempt to generalize a preexisting algorithm that determinizes certain weighted automata over the tropical semiring.

Contents

1 Introduction **4**

2 Definitions **5**

2.1 Formal Languages 5
2.2 Monoids 6
2.3 Semirings 8
2.4 Weighted Subsets 10
2.5 Matrices 11
2.6 Ranges 13
2.7 Finite Automata 14
2.8 Weighted Finite Automata 16

3 Determinization **21**

3.1 Determinization of finite automata 21
3.2 Determinization of weighted finite automata 23

4 Approximate determinization over the tropical semiring **26**

4.1 Inner Workings 26
4.2 Conventions 26

5 Approximate determinization over other semirings **28**

5.1 General Results 30
5.2 Suitability 32
5.3 Finite Termination 37
5.4 Termination with τ-twins property 38

6 Conclusion **45**

7 Further Research **46**

7.1 Other termination conditions 46
7.2 Subsemirings 46
7.3 Alternate notions of approximation 46

1 Introduction

As mentioned in the abstract, weighted finite automata are a generalization of finite automata. Weighted automata find applications in various branches of scientific and scholarly research, and they can be used to model a wide range of problems. They are used in computational linguistics [MPR05; KM09], in image compression [AK09], to analyze probabilistic systems [BGC09], and for many other purposes.

A weighted automaton is defined over an alphabet Σ and a semiring S. In a weighted finite automaton, weights (elements of S) are assigned to each state and to each transition. Equivalently, when going from the mathematical representation of finite automata to that of weighted finite automata, subsets are replaced with S-weighted subsets. Thus, whereas a finite automaton computes a subset of the set of words over a given alphabet, a weighted finite automaton computes a weighted subset of words over a given alphabet.

Analogously to finite automata, any weighted automaton is either deterministic or nondeterministic. However, whereas there is a well-known method for determinizing finite automata, determinizing weighted finite automata is a non-trivial (and in some cases even impossible) task. In this thesis, we aim to define a notion of approximate determinization for a broad range of weighted automata. Furthermore, we aim to provide an algorithm to perform such approximate determinization and a proof of the correctness and adequacy of said algorithm.

This thesis was written as part of the bachelor programs Computer Science and Mathematics, under the supervision of Marcello Bonsangue (LIACS) and Peter Bruin (MI).

2 Definitions

In this section we will provide some standard definitions and results, mostly related to formal languages and automata theory. Most of these definitions can be found in [DKV09], others have been taken from or inspired by [AKL13] and [KM05].

2.1 Formal Languages

Definition 2.1. An **alphabet** Σ is in itself nothing but a finite set[1]. We will refer to elements $\sigma \in \Sigma$ of an alphabet Σ as **letters** or **symbols**.

We use the term alphabet to signify that a set Σ is to be used to construct a formal language. The lower case modern English alphabet $\{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$ will feature in many examples, but $\{\&, @, 9\}$ is as "valid" of an alphabet, as is any other finite set, regardless of whether it has a visual representation.

In the first example given above, "a" is a letter of the alphabet $\{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$. Given an alphabet Σ, a **word** over Σ is a finite sequence of elements of Σ. We will write ε to denote the empty sequence or the empty word, that is, the sequence/word that contains no elements/symbols.

Remark 2.2. We will often write a word $(\sigma_i)_{1 \leq i \leq n}, \sigma_i \in \Sigma$ of length n as $\sigma_1\sigma_2 \cdots \sigma_n$, by concatenating its symbols. Extending this notation, for $m \in \mathbb{N}$ we may write σ^m to denote m repetitions of a symbol σ, or $(\sigma_1\sigma_2 \cdots \sigma_n)^m$ to denote the sequence $\sigma_1\sigma_2 \cdots \sigma_n$ repeated m times, and finally, using this notation we have $(\sigma_1\sigma_2 \cdots \sigma_n)^0 = \varepsilon$ for any word $\sigma_1\sigma_2 \cdots \sigma_n$.

Definition 2.3. Given an alphabet Σ, a **formal language** or **language** L over Σ is a set of finite words over Σ.

These languages need not be assigned any particular orthography, grammar or meaning.

Example 2.4. If we again consider the alphabet $\Sigma := \{\mathrm{a}, \mathrm{b}, \cdots, \mathrm{z}\}$, then $\emptyset$ is the trivial empty language over Σ, and the set of all words over Σ is another language over Σ.

A couple of non-trivial languages are:

- $L_1 := \{\mathrm{a}^n\mathrm{b}^m \mid n, m \in \mathbb{N}\}$ (the set of all words consisting of any number of "a"'s followed by any number of "b"'s),
- $L_2 := \{(\mathrm{ab})^n \mid n \in \mathbb{N}\}$ (the set of all words consisting of the sequence "ab" repeated any number of times), and
- $L_3 := \{\mathrm{a}^n\mathrm{b}^n \mid n \in \mathbb{N}\}$ (the set of all words consisting of any number of "a"'s followed by an equal number of "b"'s).

Remark 2.5. Any language L over an alphabet Σ can be represented as a function $f : \Sigma^* \to \{0, 1\}$ that maps any word in L onto 1 and any other word onto 0.

[1] Infinite alphabets are possible but we will not consider them.

2.2 Monoids

Definition 2.6. A **monoid** $(M, \odot, 1)$ is a tuple of a set M, an associative binary operation $\odot : M \times M \to M$ and a neutral element $1 \in M$ such that for all $m \in M$ we have $1 \odot m = m \odot 1 = m$.

Thus, any group is trivially also a monoid[2] - the class of monoids is a strict superclass of the class of groups. If the operation and neutral element are clear from the context, we may choose to leave them unwritten, writing M instead of $(M, \odot, 1)$.

As an example, $(\mathbb{N}, +, 0)$, $(\mathbb{Z}, +, 0)$, $(\mathbb{N}, \times, 1)$, $(\mathbb{N} \setminus \{0\}, \times, 1)$, $(\mathbb{Q}, \times, 1)$, and $(\mathbb{Q} \setminus \{0\}, \times, 1)$ are all monoids. Of these, only $(\mathbb{Z}, +, 0)$ and $(\mathbb{Q} \setminus \{0\}, \times, 1)$ are groups.

Note that the operation on a monoid can be trivially and canonically extended to operate on finite lists of elements of the monoid, such that applying it to the empty list results in the neutral element.

Definition 2.7. A monoid $(M, \odot, 1)$ is **commutative** if $\forall m, n \in M, m \odot n = n \odot m$.

Commutative monoids are often written additively. All of the examples of monoids given beneath Definition 2.6 are commutative. The monoid $(\mathbb{R}^{2\times 2}, \cdot, \mathrm{Id}_2)$ is not commutative. Here $\mathbb{R}^{2\times 2}$ is the set of 2×2 matrices over $\mathbb{R}$, $\cdot$ is matrix multiplication, and Id_2 is the 2×2 identity matrix over $\mathbb{R}$.

Definition 2.8. Given any two monoids $(M, \odot_M, 1_M)$ and $(N, \odot_N, 1_N)$, a **morphism** from M to N is a function $h : M \to N$ such that for all $m, m' \in M$ we have $h(m \odot_M m') = h(m) \odot_N h(m')$ and $h(1_M) = 1_N$. In other words, a monoid morphism from M to N is a function $M \to N$ that preserves the monoid structure.

The next definition provides more examples of monoids that are not generally commutative.

Definition 2.9. Given any non-empty set Σ, the **free monoid** Σ^* generated by Σ is the set of all finite sequences over Σ, with concatenation as its operator and with the empty sequence as its neutral element.

When Σ is an alphabet, the free monoid Σ^* is the set of all finite words over Σ with an added structure of concatenation. For any alphabet Σ with at least two distinct letters, the free monoid Σ^* is a non-commutative monoid, since $ab \neq ba$.

Because Σ^* is a free monoid, any function $h : \Sigma \to M$ from an alphabet to a monoid $(M, \odot, 1)$ uniquely induces a monoid morphism $h^\# : \Sigma^* \to M$; under this morphism we have $h^\#(\varepsilon) = 1$ and $h^\#(\sigma_1\sigma_2 \cdots \sigma_n) = h(\sigma_1) \odot h(\sigma_2) \odot \cdots \odot h(\sigma_n)$. We will often implicitly extend such a function h and write h instead of $h^\#$.

Definition 2.10. A commutative monoid $(M, \odot, 1)$ is **cancellative** if we have $a \odot c = b \odot c \implies a = c$ for all $a, b, c \in M$.

Note that all of the monoids mentioned underneath Definition 2.6 are cancellative. If $(R, \oplus, \odot, 0, 1)$ is an non-trivial ring, $(R, \odot, 1)$ is a non-cancellative monoid since any non-trivial ring contains two distinct elements and since either of those elements multiplied by the ring's zero is equal to zero.

[2] A monoid $(M, \odot, 1)$ is a group if and only if $\forall a \in M$ there is a $b \in M$ such that $a \odot b = b \odot a = 1$.

2.2.1 Ordered Monoids

Definition 2.11. A commutative monoid $(M, +, 0)$ together with a partial order $\leq$ on M is called an **ordered monoid** if $\leq$ is preserved by addition, that is $\forall a, b, c, d \in M, (a \leq b \wedge c \leq d) \implies (a + b) \leq (c + d)$. When the inequality $0 \leq a$ holds for all elements of an ordered monoid, it is **positively ordered**, and when $\leq$ is a total order, M together with $\leq$ is called a **totally ordered monoid**.

All of the examples of monoids given beneath Definition 2.6 can be equipped with the usual total orders of the underlying sets, thus creating totally ordered monoids. Of these, only $(\mathbb{N}, +, 0)$ and $(\mathbb{N} \setminus \{0\}, \times, 1)$ are positively ordered.

Note that in any cancellative totally ordered monoid, we can conclude $a \leq b$ from $a \odot c \leq b \odot c$ - recall that the order of an ordered monoid is compatible with its operation, thus $a > b$ would imply that $a \odot c > b \odot c$. By applying the modus tollens we get our desired result.

Definition 2.12. The **natural relation** of a commutative monoid $(M, \oplus, 0)$ is the relation $\trianglelefteq$ such that for any two elements $a, b \in M$ we have $a \trianglelefteq b$ if and only if there is a $c \in M$ such that $a + c = b$.

Definition 2.13. A commutative monoid $(M, \oplus, 0)$ together with its natural relation $\trianglelefteq$ is **naturally ordered** if $\trianglelefteq$ is a partial order and if they, together, are an ordered monoid. We may then refer to $\trianglelefteq$ as M's **natural order**.

Note that whereas both $(\mathbb{N}, +, 0)$ and $(\mathbb{N} \setminus \{0\}, \times, 1)$ are naturally ordered, the natural order on $(\mathbb{N}, +, 0)$ does correspond to the usual order on $\mathbb{N}$ whereas the natural order on $(\mathbb{N} \setminus \{0\}, \times, 1)$ does not. Furthermore, for any set X, the monoid $(\mathcal{P}(X), \cup, \emptyset)$ is a naturally ordered monoid ordered by subset inclusion.

Finally, consider the case of a monoid consisting of a totally ordered set with a maximal element and whose operation is the dyadic minimum with respect to the aforementioned order. Such a monoid permits a natural order, but its natural order is the inverse of its original order.

Definition 2.14. The **Grothendieck group** of a commutative monoid $(M, \oplus, 0)$ is the group $(M \times M, \oplus_2, (0, 0))/\sim$ where $\sim$ is defined such that $(a_1, a_2) \sim (b_1, b_2)$ if and only if there is a $c \in M$ for which $a_1 \oplus b_2 \oplus c = a_2 \oplus b_1 \oplus c$ and where $\oplus_2$ is pointwise addition.

The conventional constructions of the integers from the natural numbers and of the rationals from the integers are instances of this construction. That is, the Grothendieck group of $(\mathbb{Z}_{\geq 0}, +, 0)$ is isomorphic to $(\mathbb{Z}, +, 0)$ and the Grothendieck group of $(\mathbb{Z} \setminus \{0\}, *, 1)$ is isomorphic to $(\mathbb{Q}, *, 1)$.

The (perhaps unfamiliar) addend c is required to ensure transitivity of $\sim$ in the case of non-cancellative monoids - since both the natural numbers and the non-negative reals are cancellative, it may be left out in those instances of the construction.

Note that there is a canonical injective monoid morphism from any commutative monoid into its Grothendieck group. Furthermore, note that the order of a totally ordered, commutative, cancellative monoid induces a total order on its Grothendieck group G, and that this order is compatible with the injection of M in G. This order is equivalent to the order on $S \times S$ defined such that $(a, b) \leq (c, d) \iff a + d \leq b + c$.

2.3 Semirings

Definition 2.15. A **semiring** $(S, \oplus, \odot, 0, 1)$ is a set S equipped with two binary operations $\oplus : (S \times S) \to S$ and $\odot : (S \times S) \to S$ such that:

- $(S, \oplus, 0)$ is a commutative monoid,
- $(S, \odot, 1)$ is a monoid,
- $\forall a \in S, 0 \odot a = a \odot 0 = 0$,
- $\forall a, b, c \in S, a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$,
- and $\forall a, b, c \in S, (b \oplus c) \odot a = (b \odot a) \oplus (c \odot a)$.

It is said to be **finite** if S is a finite set and **commutative** if $(S, \odot, 1)$ is a commutative monoid.

This is a "less strict" definition than the more familiar ring structure - any ring is trivially also a semiring, but not every semiring is a ring. If the operations and neutral elements are clear from the context, we may choose leave them unwritten, writing S instead of $(S, \oplus, \odot, 0, 1)$.

As an example, $(\mathbb{N}, +, \cdot, 0, 1)$, $(\mathbb{Q}, +, \cdot, 0, 1)$, $(\mathbb{R}_{\geq 0} \cup \infty, \min, +, \infty, 0)$, and $(\{0, 1\}, \text{or}, \text{and}, 0, 1)$ are all semirings. Of these, $(\mathbb{Q}, +, \cdot, 0, 1)$ is the only ring.

Furthermore, in our texts the second ("multiplicative") operation of a semiring generally has precedence over the first ("additive") operation of said semiring.

In this thesis we will frequently use the **boolean semiring** $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ as well as the **tropical semiring** $\mathbb{T} = (\mathbb{R}_{\geq 0} \cup \infty, \min, +, \infty, 0)$.

Definition 2.16. Given a semiring $(S, +, \cdot, 0, 1)$, a **subsemiring** of S is a subset $S' \subseteq S$ that contains $0, 1$ and that is itself a semiring with the (restricted) operations of S as its operations.

Any subset $S' \subseteq S$ uniquely **generates** a subsemiring of S, this generated subsemiring is the closure of S' under the operations of S. If the subsemirings generated by finite subsets $S' \subseteq S$ are all finite, then S is called **locally finite**.

In order to prove that a given subset $S' \subseteq S$ is a subsemiring of $(S, +, \cdot, 0, 1)$, it suffices to show that $0, 1 \in S'$ and that S' is closed under addition and multiplication, since the associativity, commutativity and distributivity properties are directly inherited from S.

As an example, $(\mathbb{N}, +, \cdot, 0, 1)$ is a sub-semiring of $(\mathbb{Q}, +, \cdot, 0, 1)$, but not of $((\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$.

Definition 2.17. An element $s \neq 0$ of a semiring $(S, \oplus, \odot, 0, 1)$ is a **zero-divisor** if a factor $s' \in S_{\neq 0}$ exists such that we have $s \odot s' = 0$. A semiring is **zero-divisor-free** if it contains no zero-divisors.

None of the semirings provided under Definition 2.15 contain zero-divisors. The semiring $(\mathbb{R}^{2\times 2}, +, \cdot, 0^{2\times 2}, I)$ (where I is the 2×2 identity matrix) does however contain zero-divisors. We will more closely examine matrices in Section 2.5.

Note that if a semiring $(S, \oplus, \odot, 0, 1)$ is multiplicatively cancellative (that is, the multiplicative monoid $(S \setminus \{0\}, \odot, 1)$ is cancellative), it is zero-divisor-free.

Definition 2.18. A semiring $(S, \oplus, \otimes, 0, 1)$ together with a partial order $\leq$ on S is a **partially ordered semiring** if both $(S, \oplus, 0)$ and $(S, \otimes, 1)$ are partially ordered monoids. Then, $\oplus$ and $\otimes$ are compatible with $\leq$. If this is the case and $\leq$ is a total order, $(S, \oplus, \otimes, 0, 1)$ together with $\leq$ is a **totally ordered semiring**.

Many of the semirings we have featured in examples above can be equipped with partial or even total orders that are compatible with the semiring structure. As an example, if we equip the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ with the canonical order on the natural numbers, we get a totally ordered semiring.

Do however note that in some cases, there may be several distinct possible orders for one and the same semiring - in the aforementioned example, equipping the natural numbers with the inverse of their canonical order still results in a totally ordered semiring, but a different one.
We took the following definition from [KM05], where several equivalent conditions are presented.

Definition 2.19. A totally ordered semiring $(S, \oplus, \odot, 0, 1)$ is a **min-semiring** if $\oplus$ is the minimum with respect to $\leq$.

In [Mah84], these semirings are referred to as **extremal** semirings. In [Pin98], Pin examines a special class of min-semirings: the min-plus semirings, sometimes referred to as tropical semirings. These closely resemble the semiring we call **the** tropical semiring (that is $\mathbb{T} = (\mathbb{R}_{\geq 0} \cup \infty, \min, +, \infty, 0)$).

Note that the natural order on $(S, \oplus, 0)$ is the inverted order of the order described in this definition. Furthermore, note that we may analogously define **max-semirings**, and that their natural orders *do* correspond to the order described in the analogous definition.

Since any min-semiring $(S, \oplus, \odot, 0, 1)$ contains a maximal element 0, we may say that subsets of S whose supremum is 0 are unbounded above, even though (strictly speaking) they do have an upper bound. We will extend this convention to lower bounds of subsets of max-semirings and to upper and lower bounds of sequences over min-semirings and max-semirings, respectively.

2.4 Weighted Subsets

In a fashion analogous to Remark 2.5, any subset Z' of an arbitrary set Z can be represented by its identifier function $1_{Z'} : Z \to \{0,1\}$, $z \mapsto \begin{cases} 1 & z \in Z' \\ 0 & z \notin Z' \end{cases}$. This representation gives rise to the following extension:

Definition 2.20. Given a set Z and a semiring S, an **S- weighted subset** of Z is a function $Z' : Z \to S$.

We will often use a weighted subset $Z' : Z \to S$ simply as a subset of Z, then, Z refers to the set $\{z \in Z \mid Z'(z) \neq 0\}$.

Definition 2.21. A **formal power series** over an alphabet Σ and a semiring S is an S-weighted subset of Σ^*. These formal power series are often written as formal sums, $\sum_{w \in \Sigma^*} f(w)w$. We will write $S\langle\langle \Sigma^* \rangle\rangle$ to refer to the set of all formal power series over a semiring S and an alphabet Σ. The **support** of a formal power series $r \in S\langle\langle \Sigma^* \rangle\rangle$ is the set $\{w \in \Sigma^* \mid r(w) \neq 0\}$. Those formal power series with a finite support are called **polynomials**, we will write $S\langle \Sigma^* \rangle$ to refer to the set of all polynomials over S and Σ^*.

In the notation $\sum_{w \in \Sigma^*} f(w)w$, symbols of Σ are somewhat analogous to formal variables while elements of S act like coefficients. However, it is important to recall that unlike the formal variables in conventional power series, Σ^* is not generally commutative.

Remark 2.22. This concept of a formal power series is an extension of formal languages. Specifically, it extends formal languages by extending the representation specified in Remark 2.5, which is itself a formal power series if we consider the boolean semiring structure $\mathbb{B}$ on $\{0,1\}$.

2.5 Matrices

In this section we will provide definitions for matrices over semirings and for common matrix operations. We represent matrices as functions from index sets to semirings.

Definition 2.23. Let P, Q be finite sets and let $(S, \oplus, \odot, 0, 1)$ be a semiring. A **matrix** indexed by P, Q with values in S is a function $f : P \times Q \to S$.

The **transpose** of a matrix $f : P \times Q \to S$ is the matrix $f^T : Q \times P \to S, (q, p) \mapsto f(p, q)$.

The **empty** $P \times Q$ **matrix** is the constant 0 matrix, that is, the matrix that is the constant 0 function. We may write $0^{P \times Q}$ to refer to it.

The $P \times P$ **identity matrix** is the matrix that is 1 whenever its two arguments are equal and 0 whenever they are not.

We may write $S^{P \times Q}$ to refer to the set of such matrices. We may write n to refer to the index set $\{m \in \mathbb{N} \mid 1 \leq m \leq n\}$ when the fact that we intend to refer to a set is clear from context.

To disambiguate functions that we intend to use as matrices from other functions, we may write matrix evaluation with square brackets, as in $f[p, q]$ rather than $f(p, q)$. Furthermore, for $q \in Q$, $f \in S^{1 \times Q}$ and $g \in S^{Q \times 1}$ we may write $f[1, q]$ as $f[q]$ and $g[q, 1]$ as $g[q]$. Finally, we will sometimes implicitly identify a 1×1 matrix $f \in S^{1 \times 1}$ with its sole element $f[1, 1]$.

Since we represent matrices as functions, the matrix generated by pointwise application of a unary function to the elements of a matrix can be written as function composition.

Definition 2.24. Let P, Q, R be finite sets and let $(S, \oplus, \odot, 0, 1)$ be a semiring. We will define the conventional **matrix multiplication** by

$$\cdot : S^{P \times Q} \times S^{Q \times R} \to S^{P \times R},$$

$$f \cdot g = ((p, r) \mapsto \bigoplus_{q \in Q} f[p, q] \odot g[q, r]).$$

Recall that multiplication in a semiring is not necessarily commutative and note that matrix multiplication is not necessarily commutative either, even over commutative semirings. It is, however, associative. Note that for any finite set Q, the tuple $(S^{Q \times Q}, \cdot, I)$ (where I is the $Q \times Q$ identity matrix) is a monoid.

Definition 2.25. Let P, Q be finite sets and let $(S, \oplus, \odot, 0, 1)$ be a semiring. We will use the addition operator $\oplus$ of S to refer to the pointwise sum of two matrices in $S^{P \times Q}$, that is: we define the conventional **matrix addition** as

$$\oplus : S^{P \times Q} \times S^{P \times Q} \to S^{P \times Q},$$

$$f \oplus g = ((p, q) \mapsto f[p, q] \oplus g[p, q]).$$

Note that for any two finite sets P, Q, the tuple $(S^{P \times Q}, \oplus, 0^{P \times Q})$ is a monoid.

Definition 2.26. Let P, Q be finite sets and let $(S, \oplus, \odot, 0, 1)$ be a semiring. The **left scalar product** of a matrix in $S^{P \times Q}$ with a scalar $s \in S$ is defined as

$$\odot : S \times S^{P \times Q} \to S^{P \times Q},$$

$$s \odot f = ((p, q) \mapsto s \odot f[p, q].$$

The right scalar product is defined analogously, but note that $s \odot f$ is not necessarily equal to $f \odot s$.

2.5.1 Factorizations

We will now describe the notion of "factorizations" of matrices, these are pairs of functions that split matrices into a common scalar factor and a "rest" matrix.

We have taken the definitions of (maximal) factorizations from [KM05] and have expanded them to 2-dimensional matrices rather than just vectors.

Factorization can be seen as a generalisation of one of the steps in the powerset construction for the determinization of finite automata (in particular, the step where edges are created). We will discuss this algorithm in Section 3.1.

Similarly, Aminof's algorithm (presented here as Algorithm 3) features a step that can be described as a (maximal) factorization over $\mathbb{R} \cup \infty$.

Definition 2.27. Let $(S, \oplus, \odot, 0, 1)$ be a semiring and let P, Q be finite sets. A **factorization** of $S^{P\times Q}$ is a pair of functions $f : S^{P\times Q} \setminus \{0^{P\times Q}\} \to S^{P\times Q}$ and $g : S^{P\times Q} \setminus \{0^{P\times Q}\} \to S$ such that $\forall u \in S^{P\times Q} \setminus \{0^{P\times Q}\}, u = g(u) \odot f(u)$.

We will, in a somewhat unrigorous manner, sometimes speak of factorizations of a semiring itself rather than of a particular set of matrices over said semiring.

Note that for all $u \in S^{P\times Q}\setminus\{0^{P\times Q}\}$ we have $f(u) \neq 0^{P\times Q}$ and $g(u) \neq 0$, since either of $f(u) = 0^{P\times Q}$ and $g(u) = 0$ would imply that $u = g(u) \odot f(u) = 0^{P\times Q}$.

As an example, for any set of matrices $S^{P\times Q}$ over any semiring $(S, \oplus, \odot, 0, 1)$, the pair consisting of the identity function and the constant 1 function is a factorization. We will refer to this factorization as the **trivial factorization** on $S^{P\times Q}$.

Note that this factorization is the only possible factorization in the case of matrices over the boolean semiring $\mathbb{B}$, since if (f, g) is a factorization of $\mathbb{B}^{P\times Q}$ for some finite sets P, Q, we must have $g(u) = 1$ for all $u \in \mathbb{B}^{P\times Q} \setminus \{0^{P\times Q}\}$.

We will later see that the fact that other semirings tend to permit more factorizations is relevant when constructing determinization algorithms of weighted finite automata.

Furthermore, if P, Q are finite sets, note that $(f, \min)$ is a factorization of $\mathbb{T}^{P\times Q}$ where min and f are defined such that

$$\min(v) = \min_{p\in P, q\in Q} v[p, q] \text{ and such that}$$
$$f(v) = (p, q \mapsto v[p, q] - \min(v)).$$

An analogous factorization may be defined over any semiring whose multiplicative monoid permits a natural order.

Definition 2.28. Let $(S, \oplus, \odot, 0, 1)$ be a semiring and let P, Q be finite sets. Let (f, g) be a factorization of $S^{P\times Q}$. The factorization (f, g) is **maximal** if $\forall u \in S^{P\times Q}, \forall s \in S$ we have that $s \odot u \neq 0 \implies f(u) = f(s \odot u)$.

Note that $S^{P\times Q}$ may or may not permit maximal factorization, and that it may permit several distinct maximal factorizations.

Note that the sole factorization of any set of matrices over the boolean semiring $\mathbb{B}$ is trivially a maximal factorization, again using the fact that $\mathbb{B}$ only has one non-zero scalar.

However, provided that P, Q are two non-empty finite sets, the trivial factorization of $\mathbb{T}^{P\times Q}$ is not maximal. In fact, this is the case for any semiring S with at least three elements.

Furthermore, note that the factorization $(f, \min)$ defined above is a maximal factorization.

2.6 Ranges

We will now use matrices to associate a range in a partially ordered semiring to each element of a given finite set.

Let Q be a finite set and let S be a partially ordered semiring. A matrix $f \in S^{\{l,u\}\times Q}$ can be seen as a mapping that associates a *l*ower and an *u*pper bound to each $q \in Q$. Thus, f associates the range $\{s \in S \mid f[l, q] \leq s \leq f[u, q]\}$ to an element $q \in Q$.

Note that such ranges are well-defined ranges only if $\forall q \in Q$ we have $f[l, q] \leq f[u, q]$, since they would otherwise be empty.

Definition 2.29. Let $(S, \oplus, \odot, 0, 1)$ be a partially ordered semiring and let Q be a finite set. Let two matrices $p, p' \in S^{\{l,u\}\times Q}$ be given. We say that p' **refines** p if $\forall q \in Q, p_{l,q} \leq p'_{l,q} \leq p'_{u,q} \leq p_{u,q}$.

These range matrices will feature in Algorithm 3 and Algorithm 4, where they are used as the states of the automata that these algorithms construct.

Next, we will define range functions: these functions can be used to assign an approximation range to each element of a partially ordered semiring. We require some distributivity and consistency properties.

Definition 2.30. Let $(S, \oplus, \odot, 0, 1)$ together with $\leq$ be a partially ordered semiring.

A function $\tau : S \to S$ is a **range function** if:

- no non-zero element is mapped onto zero, that is, if $\forall s \in S, \tau(s) = 0 \implies s = 0$,
- an element is always $\leq$ its image, that is, if $\forall s \in S, s \leq \tau(s)$,
- τ preserves order, that is, if $\forall s, s' \in S, s \leq s' \implies \tau(s) \leq \tau(s')$,
- τ distributes over $\oplus$, that is, if $\forall s, s' \in S, \tau(s \oplus s') = \tau(s) \oplus \tau(s')$,
- and τ distributes over $\odot$, that is, if $\forall s, s' \in S, \tau(s \odot s') = \tau(s) \odot \tau(s')$.

Such a function can be used to associate a well-defined range $[s, \tau(s)]$ to each element $s \in S$, such a range is always non-empty as it always contains s as well as $\tau(s)$.

Note that for any semiring S, the identity function on S is a range function, and note that the identity function is the only possible range function on the boolean semiring $\mathbb{B}$.

In the case of the tropical semiring $\mathbb{T}$, for any $t \in \mathbb{R}_{\geq 1}$ the function $\mathbb{T} \to \mathbb{T}, x \mapsto tx$ is a range function. We will later see that this allows us to consider t-approximation as a special case of τ-approximation.

2.7 Finite Automata

Finite automata are abstract models of computation that can be used to encode certain types of languages. They are modelled as finite, directed multigraphs with labelled transitions.

Definition 2.31. A **finite automaton** over an alphabet Σ is defined as a tuple (Q, I, F, E) consisting of a finite set of **states** Q (these correspond to the vertices of the graph), a set of **initial** states $I \subseteq Q$, a set of **accepting** or **final** states $F \subseteq Q$, and a set of **transitions** $E \subseteq Q \times \Sigma \times Q$.

Given $q, q' \in Q$ and $\sigma \in \Sigma$, a transition (q, σ, q') is an **outgoing** transition of q (its **start**), an **incoming** transition of q' (its **end**), and its **label** is σ.

We may write a transition (q_1, σ, q_2) as $q_1 \xrightarrow{\sigma} q_2$.

A **path** is a finite sequence of transitions such that for each consecutive pair $(p, \sigma, q), (r, \sigma', s)$ of transitions we have $q = r$. We will refer to the sequence of symbols associated with each of the edges of a path as its **label**. Note that the empty sequence is also a path, its label is the empty word ε.

Extending the aforementioned notation, we may write the path $((q_1, \sigma, q_2), (q_2, \sigma', q_3))$ as $q_1 \xrightarrow{\sigma} q_2 \xrightarrow{\sigma'} q_3$.

For any word $w \in \Sigma^*$, we will write $q_1 \overset{w}{\rightsquigarrow} q_2$ to refer to the set of paths starting in q_1, ending in q_2, and labelled with w. Similarly, given two subsets $Q', Q'' \subseteq Q$, $Q' \overset{w}{\rightsquigarrow} Q''$ refers to all paths starting in a state of Q', ending in a state of Q'', and labelled w.

Remark 2.32. As for the computational interpretation of finite automata: Finite automata are modelled in discrete time, such that at any time they are in precisely one of their states, starting out in one of the initial states. A finite automaton over an alphabet Σ takes a word in Σ^* as its input, processing it one symbol at a time. At any point in time, it is in exactly one state, and it starts out in one of the initial states. When the automaton is in a state $q \in Q$, it processes a symbol $\sigma \in \Sigma$ by following an outgoing transition $q \xrightarrow{\sigma} q'$, transferring the control to the state q', or blocking if there is no such state. When there are several possibilities, the control is transferred to one of them non-deterministically.

If at some point no symbols are left in the input, computation stops, the result of this run is either 1 (if the last state is in F) or 0 (if it is not).

If this point cannot be reached since at some point no adequate outgoing transition exists, computation stops and the result of the run is taken to be 0. In the case that for a certain input $\sigma_1\sigma_2\cdots\sigma_n$ a corresponding run that results in 1 exists, the finite automaton is said to **accept** $\sigma_1\sigma_2\cdots\sigma_n$. If this is not the case, the finite automaton **rejects** $\sigma_1\sigma_2\cdots\sigma_n$. Thus, the automaton accepts a word $w \in \Sigma^*$ if and only if $I \overset{w}{\rightsquigarrow} F$ is non-empty. Two finite automata are **equivalent** if they accept the same words.

Definition 2.33. A finite automaton (Q, I, F, E) over an alphabet Σ is **deterministic** if it has at most 1 initial state and for all $q \in Q$ and $\sigma \in \Sigma$, there is at most one outgoing transition from q that is labelled σ.

The computational interpretation described above can be ambiguous. Particularly, if an automaton is not deterministic, it may not always be clear which transition is to be taken, or which state to start from. If an automaton is indeed deterministic, the steps described in Remark 2.32 are unambiguous,

and only a single run is required to check whether the automaton accepts a given word. Furthermore, given any non-deterministic finite automaton, an equivalent, deterministic finite automaton can be constructed in polynomial time, we call this process **determinization**. A well-known algorithm to do so will be presented in Algorithm 1.

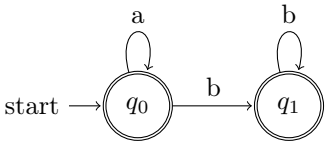
In Figure 4 and Figure 5 we have provided two examples of finite automata that are respectively nondeterministic and deterministic. Furthermore, note that both of the automata provided in Example 2.35 are deterministic. For an explanation of the visual notation used in these examples, see below.

Remark 2.34. Any finite automaton encodes a language, that is, the language consisting of all words accepted by the automaton. Such languages (that is, those languages for which a corresponding finite automaton exists) are called **regular** or **recognisable**.

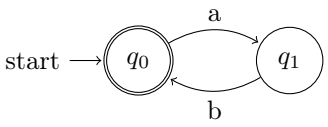
However, it is important to note that not all languages are regular. A finite automaton that corresponds to a regular language can be seen as an explicit procedure to compute the function that we defined in Remark 2.5. Since this concept of finite automata is rather limited, we will soon provide an extended definition, in a fashion similar to how formal power series are an extension of languages.

When representing finite automata visually, we will use the usual representation as a labelled graph. Initial states are marked with "start", and accepting/final states are drawn with a double line.

Example 2.35. The following directed graph is a visual representation of a finite automaton that encodes the language L_1 defined in Example 2.4.



And here is a finite state automaton that encodes L_2:



However, there is no finite automaton that encodes the language L_3 from that same example - thus, L_3 is not regular. We can construct finite automata that accept subsets or supersets of L_3, but a finite automaton that accepts no more or less than L_3 cannot be constructed.

2.8 Weighted Finite Automata

Weighted finite automata are an extension of finite automata which was first introduced in [Sch61] by M.P. Schützenberger.

One can envision this extension by replacing the subsets used to represent the edges, initial states and final states of finite automata by weighted subsets. This replacement also applies to the "output" of weighted finite automata compared to finite automata. Whereas a finite automaton models a language (which can be represented as a subset of Σ^* or as a function, see Remark 2.5), weighted finite automata model formal power series which can be represented as weighted subsets of Σ^* or equivalently as functions $\Sigma^* \to S$ where S is a semiring. Notably, any finite automaton can trivially be seen as a weighted finite automaton over the semiring $(\{0,1\}, \text{or}, \text{and}, 0, 1)$ on $\{0,1\}$.

Definition 2.36. A **weighted finite automaton (WFA)** over an alphabet Σ and a semiring S is defined as a tuple (Q, I, F, E) again consisting of a finite set of **states** Q, a weighted set of **initial** states $I : Q \mapsto S$, a weighted set of **accepting** or **final** states $F : Q \mapsto S$, and a weighted set of **transitions** $E : Q \times \Sigma \times Q \to S$.

We extend the notation we used for (labelled) paths in finite automata to weighted finite automata: we will again write $q \xrightarrow{\sigma} q'$ for the transition (q, σ, q'), and we will write $q \overset{w}{\rightsquigarrow} q'$ for the set of paths that start in q, end in q' and are labelled w. When representing an WFA visually, we will draw them in a manner similar to the visual representation for finite automata which can be seen in Example 2.35. In order to encode the weights, transitions in a visual representation of an WFA will be labelled both with their symbol and their weight. Furthermore, any state marked as initial/accepting has an initial/final weight of 1. Note that this visual representation cannot faithfully represent all weighted automata. We have chosen to use this simpler visual representation for the sake of simplicity and since non of our examples require non-unit initial or final weights. In Figure 1, Figure 2, and Figure 3 we have provided three examples of weighted finite automata in visual representations.

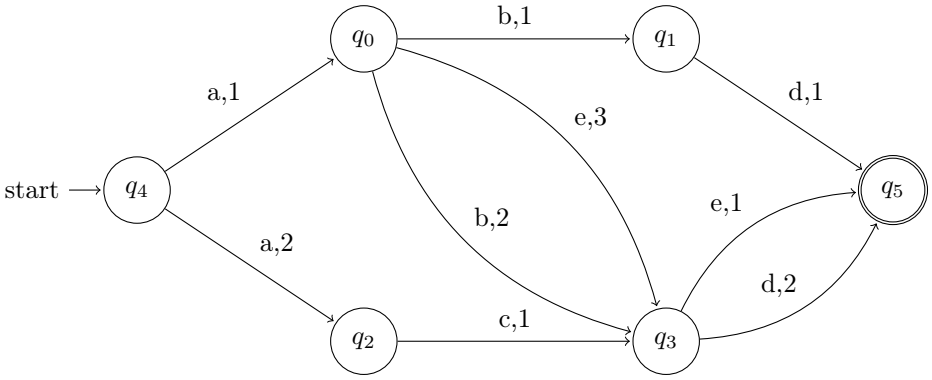


Figure 1: A WFA over the semiring $(\mathbb{Z}, +, *, 0, 1)$.

The notation and terminology related to transitions and (labelled) paths in finite automata can trivially be applied to weighted finite automata. We will reuse it without defining it again.

Do however note that if we interpret the weighted subset E of a weighted automaton (Q, I, F, E) over

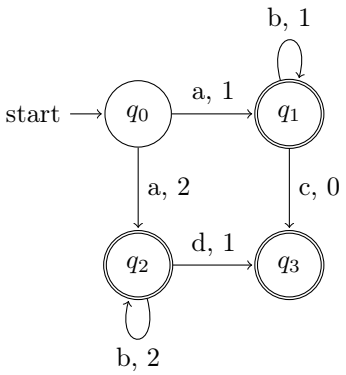


Figure 2: A WFA over the semiring $(\mathbb{Z}, +, *, 0, 1)$.

a semiring $(S,\ \oplus,\ \odot,\ 0,\ 1)$ and an alphabet Σ as a set, it consists of those pairs $(q_1, \sigma, q_2) \in Q \times \Sigma \times Q$ for which $E(q_1,\ \sigma,\ q_2)\ \neq\ 0$.

Therefore, the weights of transitions along a path are always (trivially) non-zero.

In some other representations, transitions may be weighted 0, but since such transitions can always be removed from a weighted automaton without altering the function it computes, we will use this somewhat restricted definition of transitions.

Remark 2.37. A WFA $\mathcal{A} = (Q, I, F, E)$ over an alphabet Σ and a semiring S assigns transition weights to edges, it assigns initial weights to states and it assigns final weights to states.

We will use these weight functions to construct some matrices:

- for $\sigma \in \Sigma$ we will write $\theta_{\mathcal{A}}(\sigma)$ to refer to the matrix $(Q \times Q) \to S, (q, q') \mapsto E(q, \sigma, q')$,
- for $\sigma_1\sigma_2 \cdots \sigma_n \in \Sigma^*$ we will write $\theta_{\mathcal{A}}(\sigma_1\sigma_2 \cdots \sigma_n) = \theta_{\mathcal{A}}(\sigma_1) \cdot \theta_{\mathcal{A}}(\sigma_2) \cdot \cdots \cdot \theta_{\mathcal{A}}(\sigma_n)$ (note that $\theta_{\mathcal{A}}(\varepsilon)$ is the identity matrix), this is the monoid morphism induced by the previous $\theta_{\mathcal{A}}$,
- we will write $\lambda_{\mathcal{A}}$ to refer to the matrix $(\{1\} \times Q) \to S, (1, q) \mapsto I(q)$,
- and we will write $\rho_{\mathcal{A}}$ to refer to the matrix $(Q \times \{1\}) \to S, (q, 1) \mapsto F(q)$.

Using these matrices, we will define some more functions:

- For $w \in \Sigma^*$, we will write $|\mathcal{A}|(w) = \lambda \odot \theta(w) \odot \rho$, we will refer to $|\mathcal{A}|(w)$ as the **weight** of w.
- Furthermore, let the function $\beta_{\mathcal{A}} : \Sigma^* \times Q \to S$ be given by $(w,\ q_2) \mapsto (\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(w))[q_2]$. Note that $\forall w \in \Sigma^*$, $\bigoplus_{q_2 \in Q} \beta_{\mathcal{A}}(w, q_2) \odot F(q_2) = |\mathcal{A}|(w)$. Given $w \in \Sigma^*$ and $q \in Q$, $\beta_{\mathcal{A}}(w, q)$ is sometimes referred to as the **partial weight** of w.
- Finally, we will write $\gamma_{\mathcal{A}}$ for the function from $S^{\{l,u\} \times Q \times \Sigma}$ to $S^{\{l,u\} \times Q}$ such that $\mathrm{row}_l(\gamma_{\mathcal{A}}(q', \sigma)) = \mathrm{row}_l(q') \cdot \theta_{\mathcal{A}}(\sigma)$ and such that $\mathrm{row}_u(\gamma_{\mathcal{A}}(q', \sigma)) = \mathrm{row}_u(q') \cdot \tau \circ \theta_{\mathcal{A}}(\sigma)$.

This definition of $|\mathcal{A}|$ is equivalent to stating that:

- the weight $|\mathcal{A}|(\alpha)$ of a **path** α in $\mathcal{A}$ is defined as the semiring product of the weights of its transitions,
- the weight of a **run** of a word w (a path labelled w) is the semiring product of
  - the initial weight of the first state of the run,
  - the weight of the run as a path,
  - and the final weight of the last state of the run.
- the weight $|\mathcal{A}|(w)$ of a word w is the semiring sum of all weights of all runs of w.

Two weighted finite automata $\mathcal{A}, \mathcal{A}'$ over an alphabet Σ and a semiring S are **equivalent** if for every word $w \in \Sigma^*$, $|\mathcal{A}|(w) = |\mathcal{A}'|(w)$.

Definition 2.38. Let $\mathcal{A} = (Q, I, F, E)$ be a WFA over an alphabet Σ and a semiring S be given arbitrarily. Furthermore, let a word $w = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$ as well as a pair $q_0, q_n \in Q$ of states be given arbitrarily. Then, a path $(q_0, q_1, \ldots, q_n) \in q_0 \overset{w}{\rightsquigarrow} q_n$ labelled w is **victorious** for $q_0 \overset{w}{\rightsquigarrow} q_n$ if $\theta(w)[q_0, q_n] = E(q_0, \sigma_1, q_1) \odot E(q_1, \sigma_2, \sigma_2) \odot \cdots \odot E(q_{n-1}, \sigma_n, \sigma_n)$.

Similarly, for two subsets $P, R \subseteq Q$ a path in $P \overset{w}{\rightsquigarrow} R$ is victorious if its weight is equal to the $\oplus$-sum of the weights of all paths in this semiring.

Definition 2.39. A WFA $\mathcal{A} = (Q, I, F, E)$ over an alphabet Σ and a semiring S is **deterministic** if the support of I consists of at most one state and if for every $q_1 \in Q$ and $\sigma \in \Sigma$, there is at most one $q_2 \in Q$ such that $E(\sigma)(q_1, q_2) \neq 0$.

In this case, we may refer to $\mathcal{A}$ as a deterministic weighted finite automaton (DWFA).

Again, we refer to the process of constructing a deterministic weighted finite automaton $\mathcal{A}'$ that is equivalent to a given weighted finite automaton $\mathcal{A}$ as **determinization**. However, it is important to note that for some weighted finite automata, no equivalent deterministic weighted finite automata exist.

If $\mathcal{A}$ is a DWFA, for any given word $w \in \Sigma^*$ there is at most one path labelled w that starts in the initial state of $\mathcal{A}$.

Note that neither of the automata in Figure 1 and Figure 2 are deterministic. However, in Figure 3 we find a deterministic weighted automaton that is equivalent to the automaton in Figure 1. Note that the automaton in Figure 2 is undeterminizable - that is, there is no DWFA that is equivalent to this automaton.

If $\mathcal{A} = (Q, I, F, E)$ is a DWFA over an alphabet Σ, let the function $\delta_\mathcal{A} : \Sigma^* \to Q$ be equal to the final state of the path labelled w that starts in the initial state of $\mathcal{A}$ wherever such a path exist, and let it be $0^{\{l,q\}\times Q}$ where such a path does not exist.

Note that in such an automaton, for any word $w \in \Sigma^*$, there is at most one $q_2 \in Q'$ such that $\beta_\mathcal{A}(w, q_2) \neq 0$. If such a q_2 exists, it is equal to $\delta_\mathcal{A}(w)$.
In [Cho77], Choffrut introduces the so-called **twins-property**. This property is a sufficient condition for determinizability of weighted automata over the tropical semiring (as proven by Mohri in [Moh97]), and can be checked in polynomial time (as proven by Allauzen and Mohri in [AM03]). We will use a more generic definition of the twins-property, taken from [KM05]. In this paper, Kirsten and Mäurer show that the twins-property is a sufficient condition for determinizability of weighted automata over min-semirings.

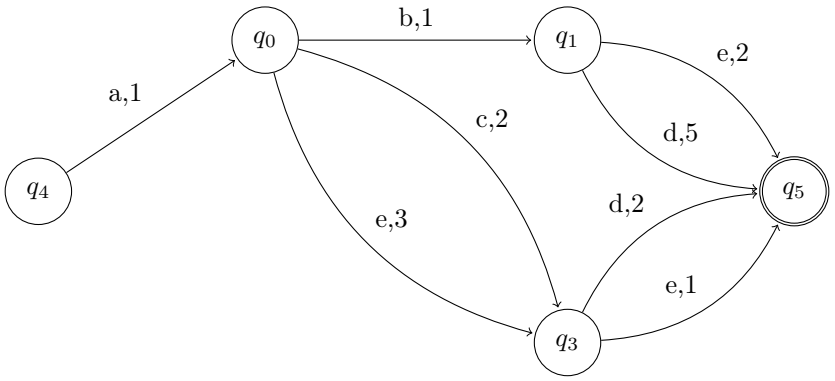


Figure 3: A DWFA over the semiring $(\mathbb{Z}, +, *, 0, 1)$. This automaton is equivalent to the automaton in Figure 1.

Definition 2.40. Let $\mathcal{A} = (Q, I, F, E)$ be a weighted automaton over an alphabet Σ and a semiring S. The automaton $\mathcal{A}$ has the **twins property** if $\forall u, v \in \Sigma^*$ and $\forall p, q \in Q$ we have $(I \overset{u}{\rightsquigarrow} p \overset{v}{\rightsquigarrow} p \neq \emptyset) \wedge (I \overset{u}{\rightsquigarrow} q \overset{v}{\rightsquigarrow} q \neq \emptyset) \implies \theta_{\mathcal{A}}(v)[p, p] = \theta_{\mathcal{A}}(v)[q, q]$.

Note that any weighted automaton $\mathcal{A}$ over the boolean semiring $\mathbb{B}$ and an alphabet Σ trivially has the twins-property. In many of the applications of WFAs, it is unnecessary or even infeasible (due to performance constraints) to calculate the exact weight a given automaton assigns to a given word. In [AKL13], Aminof, Kupferman, and Lampert define a precise notion of approximation for weighted automata over the tropical semiring $\mathbb{T}$.

Definition 2.41. Let $\mathcal{A} = (Q, I, F, E)$ and $\mathcal{A}' = (Q', I', F', E')$ be two weighted finite automata over the tropical semiring $\mathbb{T}$ and over an alphabet Σ. Let $t \in \mathbb{R}_{\geq 1}$ be given arbitrarily. We say that $\mathcal{A}'$ is a **t-approximation** of $\mathcal{A}$ if and only if $\forall w \in \Sigma^*, |\mathcal{A}|(w) \leq |\mathcal{A}'|(w) \leq t \cdot |\mathcal{A}|(w)$.

Note that $\mathcal{A}$ and $\mathcal{A}'$ are equivalent precisely when $\mathcal{A}'$ is a 1-approximation of $\mathcal{A}$, and that for $t \neq 1$, $\mathcal{A}'$ being a t-approximation of $\mathcal{A}$ does not necessarily imply that $\mathcal{A}$ is a t-approximation of $\mathcal{A}'$.

This notion of approximation gives rise to **t-determinization**: the process of constructing a deterministic weighted finite automaton that is a t-approximation of a given weighted finite automaton over the tropical semiring $(\mathbb{R}_{\geq 0} \cup \infty, \min, +, \infty, 0)$. In [AKL13], Aminof, Kupferman, and Lampert provide a weaker version of the twins property for weighted automata over the tropical semiring, and they prove that this so-called t-twins property is a sufficient condition for the termination of a new algorithm for the approximate determinization of weighted automata which they present in the same paper.

Definition 2.42. Let $\mathcal{A} = (Q, I, F, E)$ be a weighted automaton over an alphabet Σ and the tropical semiring. Let $t \in \mathbb{R}_{\geq 1}$ be given arbitrarily. Let $\cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ denote the multiplication on $\mathbb{R}$.

We say that $\mathcal{A}$ has the **t-twins property** if $\forall u, v \in \Sigma^*$ and $\forall p, q \in Q$ we have $(I \overset{u}{\rightsquigarrow} p \overset{v}{\rightsquigarrow} p \neq \emptyset \wedge I \overset{u}{\rightsquigarrow} q \overset{v}{\rightsquigarrow} q \neq \emptyset) \implies \theta_{\mathcal{A}}(v)[p, p] \leq t \cdot \theta_{\mathcal{A}}(v)[q, q]$.

Note that for $t = 1$, the t-twins property is equivalent to the regular twins-property.

Definition 2.43. Let $\mathcal{A} = (Q, I, F, E)$ and $\mathcal{A}' = (Q', I', F', E')$ be two weighted finite automata over a partially ordered semiring S and over an alphabet Σ. Let $\tau : S \to S$ be a range function.

We say that $\mathcal{A}'$ is a **τ-approximation** of $\mathcal{A}$ if and only if $\forall w \in \Sigma^*, |\mathcal{A}|(w) \leq |\mathcal{A}'|(w) \leq \tau(|\mathcal{A}|(w))$.

Note that if τ is the identity function, $\mathcal{A}$ and $\mathcal{A}'$ are equivalent precisely when $\mathcal{A}'$ is an τ-approximation of $\mathcal{A}$, and that for other functions, $\mathcal{A}'$ being an τ-approximation of $\mathcal{A}$ does not necessarily imply that $\mathcal{A}$ is a τ-approximation of $\mathcal{A}'$.

This notion of approximation gives rise to **τ-determinization**: the process of constructing a deterministic weighted finite automaton that is an τ-approximation of a given weighted finite automaton.

t-approximation and t-determinization are special cases of τ-approximation and τ-determinization, respectively, since (as stated before) for any $t \in \mathbb{R}_{\geq 1}$, the function $\mathbb{T} \to \mathbb{T}, x \mapsto xt$ is a range function.

Definition 2.44. Let $\mathcal{A} = (Q, I, F, E)$ be a weighted automaton over an alphabet Σ and a partially ordered semiring S. Let $\tau : S \to S$ be a range function.

We say that $\mathcal{A}$ has the **τ-twins property** if $\forall u, v \in \Sigma^*$ and $\forall p, q \in Q$ we have $(I \overset{u}{\rightsquigarrow} p \overset{v}{\rightsquigarrow} p \neq \emptyset \wedge I \overset{u}{\rightsquigarrow} q \overset{v}{\rightsquigarrow} q \neq \emptyset) \implies \theta_{\mathcal{A}}(v)[p, p] \leq \tau(\theta_{\mathcal{A}}(v)[q, q])$.

Definition 2.45. Let Σ be an alphabet, let S be a semiring and let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over Σ and S. Furthermore, let $w \in \Sigma^*$ and a set $\pi \subseteq Q \overset{w}{\rightsquigarrow} Q$ of paths in $\mathcal{A}$ labelled w be given.

We will say that π has a **synchronous loop** at i, j for some indices $0 \leq i < j \leq |w|$ if for all $\alpha \in \pi$ the i'th and j'th states of the path α are equal.

We may refer to the set of paths created by removing a synchronous loop i, j from a set of paths π, this is the following set: $\{(\alpha_1, \alpha_2, \cdots, \alpha_{i-1}, \alpha_j, \alpha_{j+1}, \ldots, \alpha_{|\alpha|}) \mid \alpha \in \pi\}$.

3 Determinization

In this section and in the following sections, we will present various algorithms for the (approximate) determinization of finite automata and weighted finite automata.

In these algorithms, we will frequently use abstract data types such as partial functions, queues, and sets. Note that these may generally be implemented in terms of arrays, lists, in terms of cons pairs, or using whichever implementations may be present in your language or paradigm of choice. Do, however, note that any queue we mention is implicitly first-in-first-out (FIFO). This may not be strictly necessary in some cases (such as in the case of Algorithm 1), but we do depend on this assumption when proving properties of Algorithm 4.

Furthermore, note that in these algorithms, we treat sets and functions as mutable mathematical objects, and that the operator := denotes assignment, whereas = denotes equality.

3.1 Determinization of finite automata

As we described previously, given any finite automaton (Q, I, F, E) over an alphabet Σ, one can construct an equivalent deterministic automaton (Q', I', F', E'). We refer to this process as "determinization", and we will provide an algorithm to determinize arbitrary finite automata.

This algorithm (Algorithm 1) is an implementation of the well-known powerset construction (which was first described by M. Rabin en D. Scott in [RS59]), and we have written it in a manner that closely resembles the approximate determinization algorithm from [AKL13], which we will present in Section 4.

We will not prove any particular properties of this algorithm for the determinization of finite automate, since it is included solely for illustrative purposes and since all relevant properties of the algorithm are well-known.

3.1.1 Inner workings

The algorithm constructs a new set of states such that each newly constructed state is a subset of the original set of states. In the algorithm below, we maintain a queue of newly created states that need to be further processed by the algorithm. Such a newly created state consists of all those states that the original automaton could be in, having read a given sequence of input characters. Since a nondeterministic finite automaton can be in any of its initial states when it has not yet read any characters, the new initial state consists of the set of all initial states of the original finite automaton. The queue is set up to contain this new initial element, and the queue is processed one by one.

When a new state is processed, for each character of the alphabet, the algorithm computes the set of states reachable by following outgoing transitions of the original states in the new states being processed. If necessary, these sets of states are enqueued and new transitions are added.

Since each new state can only be enqueued once and since a finite set only has a finite number of distinct subsets, the algorithm queue will eventually be empty, resulting in the termination of the algorithm. Once this occurs, (Q', I', F', E') will be a deterministic finite automaton that is equivalent to the input (Q, I, F, E).

Algorithm: FinAutDet

Input: A finite automaton (Q, I, F, E)

Output: An equivalent, finite deterministic automaton (Q', I', F', E')

$Q' := \emptyset$;
$I' := \{I\}$;
$F' := \emptyset$;
$E' := \emptyset$;
$\mathcal{Q}$:= empty queue;
Enqueue$(\mathcal{Q}, I)$;
while $\mathcal{Q}$ *not empty* **do**
    $p :=$ Dequeue$(\mathcal{Q})$;
    $Q' := Q' \cup \{p\}$;
    if $p \cap F \neq \emptyset$ **then**
        $F' := F' \cup \{p\}$;
    end
    foreach $\sigma \in \Sigma$ **do**
        $p' := \bigcup_{q \in p} \{q' \in Q \mid (q, \sigma, q') \in E\}\}$;
        $E' := E' \cup \{p, \sigma, p'\}$;
        if $p' \notin Q' \cup \mathcal{Q}$ **then**
            Enqueue$(\mathcal{Q}, p')$;
        end
    end
end
return (Q', I', F', E');

Algorithm 1: Algorithm to determinize any finite automaton over an alphabet Σ - adapted from [RS59].

Note that applying this algorithm to a deterministic finite automaton results in an automaton whose states are singletons corresponding to (and containing) the reachable states of the original automaton (the automaton may also contain an "empty" state, consisting of the empty set).

In Algorithm 1, we have provided the aforementioned algorithm in pseudo-code. Recall that we treat sets and functions as mutable mathematical objects, and that the operator := denotes assignment, whereas = denotes equality.

In Figure 4, we have provided an example of a simple non-deterministic finite automaton (FA). In Figure 5, we have provided a deterministic finite automaton (DFA) (that is, a deterministic finite automaton). This automaton is equivalent to the aforementioned automaton in Figure 4 and is the output that Algorithm 1 would produce if provided with Figure 5 as its input.

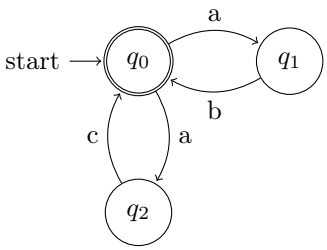


Figure 4: A FA that is non-deterministic.

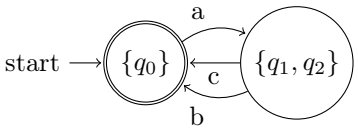


Figure 5: A DFA that is equivalent to the automaton in Figure 4.

3.2 Determinization of weighted finite automata

Attempting to adapt the powerset construction to weighted finite automata poses some challenges. First of all, there is a wealth of different semirings with all sorts of different properties, generalizing over the entire class of semirings may prove difficult. For that reason, we will solely concern ourselves with commutative min-semirings, even though some of the properties and results discussed here may transfer to weighted automata over other semirings. With that out of the way, we will describe a few different approaches to (approximate) determinization of weighted automata.

In the powerset construction, the algorithm constructs a new state-space consisting of subsets of the original state-space. When adapting this algorithm, it is natural to replace these subsets with weighted subsets. These weighted subsets can be represented as vectors $S^{1\times Q}$ indexed by the original space state. This change, when applied consistently, gives rise to a class of algorithms that determinize weighted automata. The algorithms in this class differ with regards to the specific matrices they add to the generated state space and the exact weights they assign to edges. In Algorithm 2, we have provided a pseudo-code implementation of these algorithms.

Algorithm: WFA-Det

Input: A weighted finite automaton (Q, I, F, E) over S

Output: A deterministic WFA (Q', I', F', E') that is equivalent to $\mathcal{A}$

Let (f, g) be a factorization of $S^{1\times Q}$;
$Q' := \emptyset$;
$F' := S^{1\times Q} \multimap S$;
$E' := S^{1\times Q} \times \Sigma \times S^{1\times Q} \multimap S$;
Let q'_0 be a matrix in $S^{1\times Q}$;
$q'_0 := \lambda_{\mathcal{A}}$;
$\mathcal{Q} :=$ empty queue;
Enqueue$(\mathcal{Q}, f(q'_0))$;
while $\mathcal{Q}$ *not empty* **do**
    $p :=$ Dequeue$(\mathcal{Q})$;
    $Q' := Q' \cup \{p\}$;
    $F'(p) := p \cdot \rho_{\mathcal{A}}$;
    foreach $\sigma \in \Sigma$ **do**
        $p_2 := f(p \cdot \theta(\sigma))$;
        $c := g(p \cdot \theta(\sigma))$;
        if *there is* $r \in \mathcal{Q} \cup Q'$ *such that* $r = p_2$ **then**
            $E'(p, \sigma, r) := c$;
        else if *there is* $r \in \mathcal{Q} \cup Q'$ *such that* r *refines* p_2 **then**
            $E'(p, \sigma, r) := c$;
        else if $p_2 \neq 0^{1\times Q}$ **then**
            $E'(p, \sigma, p_2) := c$;
            Enqueue$(\mathcal{Q}, p_2)$;
        end
    end
end
$I' := E' \mapsto S, q \mapsto \begin{cases} g(q'_0) & q = f(q'_0), \\ 0 & \text{else} \end{cases}$;
$E' := E' \vee 0$;
return (Q', I', F', E');

Algorithm 2: Algorithm to determinize a weighted automaton over an alphabet Σ with weights in a suitable semiring S.

3.2.1 Termination

Recall that any FA can be determinized by the well-known superset construction. One may intuitively see that this hinges on the fact that the powerset of any finite set is itself finite. The analogous claim for weighted automata is however clearly untrue: Whether or not the set of *weighted* subsets of a finite set is finite depends on the semiring of weights being used. Thus, termination is an important concern for this class of algorithms.

For any factorization (f, g) of $S^{1\times Q}$, one such algorithm exists. If such an algorithm terminates, its output is a DWFA that is equivalent to the input WFA. Note, however, that these algorithms do not necessarily terminate, and that whether they terminate on a given input depends on the choice of the factorization.

In [KM05], Kirsten and Mäurer prove that maximal factorizations are optimal with regards to termination: if (f, g) is a maximal factorization and any factorization (f', g') exist such that the (f', g') algorithm terminates on a given input, then the (f, g) algorithm also terminates on this input.

While it may be appealing to use the trivial factorization of a semiring S, since it is always available. This would imply that the edge weights are all set to the multiplicative identity 1 of S. However, while this approach does work on some automata, it may fail to terminate even on trivial, deterministic inputs.

In conclusion, some care may be needed when selecting a factorization, and if a maximal factorization is available, it may be a good choice.

3.2.2 Inner Workings

Now, let us recall: when determinizing finite automata using the conventional superset construction, the state that the output DFA arrives at after processing a word w is the subset of the states of the input consisting of those states that can be reached by a partial run labeled w.

However, in the case of a weighted automaton, keeping track of which states can be reached by processing w is insufficient - we also need to know the weights of these paths, that is, for any word w and state q our DWFA should encode the sum of the weights of all partial runs $I \overset{w}{\rightsquigarrow} q$.

Let $w \in \Sigma^*$ be arbitrarily given, and let us write $r' \in S^Q$ for the vector mapping a state $q \in Q$ of the input onto the sum of the weights of all partial runs $I \overset{w}{\rightsquigarrow} q$.

We ensure that r' can be "read" or "recovered" from the generated DWFA by ensuring that the weight of the sole partial run of w in the output is equal to $g(r')$ and that its final state is the matrix $f(r')$ if we are using the factorization (f, g). That way, their scalar product is equal to r'.

Furthermore, we set the final weight of any generated state $q' \in S^Q$ to be the dot product of $f(r')$ and the vector mapping a state $q \in Q$ on its final weight in the input automaton. This way, if the algorithm terminates, the output DWFA is equivalent to the input WFA.

4 Approximate determinization over the tropical semiring

In [AKL13], Aminof, Kupferman, and Lampert present an algorithm to approximately determinize a certain class of weighted automata over the tropical semiring. This algorithm, adapted to our mathematical representation, can be found here as Algorithm 3.

It takes a WFA over the tropical semiring $\mathbb{T}$ as well as an approximation factor $t \in \mathbb{R}_{\geq 1}$ as its input and attempts to construct a DWFA that is t-equivalent to its input WFA.

If this algorithm terminates, the result is a DWFA that t-approximates the given WFA, and the t-twins property is a sufficient condition for termination. For proofs of these claims, we refer to the original paper [AKL13].

Furthermore, as mentioned before, in [AKL13], the authors show that one can determine within polynomial time whether a given WFA has the t-twins property.

4.1 Inner Workings

Whereas the state-space of the output of Mohri's and Kirsten's algorithms consists of vectors indexed by the states of the input, in Aminof's case the state-space of the output consist of matrices indexed by the symbolic set $\{l, u\}$ as well as the states of the input.

These matrices can be interpreted such that each state of the output automaton assigns a lower and an upper bound in $\mathbb{R} \cup \infty$ to each state of the input automaton.

4.2 Conventions

In order to make the algorithm more readable, we will introduce the following conventions relating to partial functions:

- we use $A \rightharpoonup B$ to denote partial functions from A to B,
- partial functions are initialized to be undefined everywhere,
- and given a partial function $f : A \rightharpoonup S$ and an element $s \in S$, we will write $f \vee s$ to refer to the function that is equal to f where f is defined and equal to s where f is not defined.

Furthermore, in the following section, we will refer to the tropical semiring $(\mathbb{R} \cup \infty, \min, +, \infty, 0)$ as $\mathbb{T}$.

Algorithm: TWFA-AppDet

Input: A WFA (Q, I, F, E) over $\mathbb{T}$ and an approximation factor $t \in \mathbb{R}_{\geq 1}$

Output: A deterministic WFA (Q', I', F', E') that t-approximates (Q, I, F, E)

$Q' := \emptyset$;

Let F' be a partial function $\mathbb{T}^{\{l,u\}\times Q} \rightharpoonup \mathbb{T}$ that is undefined everywhere;

Let E' be a partial function $\mathbb{T}^{\{l,u\}\times Q} \times \Sigma \times \mathbb{T}^{\{l,u\}\times Q} \rightharpoonup \mathbb{T}$ that is undefined everywhere;

$i' := t * \min_{q\in Q}(I(q))$;

Let q'_0 be a matrix in $\mathbb{T}^{\{l,u\}\times Q}$;

foreach $q \in Q$ **do**

  $q'_0[l, q] = I(q) - i'$;

  $q'_0[u, q] = t * I(q) - i'$;

end

$\mathcal{Q}$:= empty queue;

Enqueue$(\mathcal{Q}, q'_0)$;

while $\mathcal{Q}$ *not empty* **do**

  $p :=$ Dequeue$(\mathcal{Q})$;

  $Q' := Q' \cup \{p\}$;

  $F'(p) := \min_{q\in Q}(p[l, q] + F(q))$;

  foreach $\sigma \in \Sigma$ **do**

    $c' := \min_{(q_1,q_2)\in Q\times Q}(p[u, q_1] + t * E(q_1, \sigma, q_2))$;

    Let p_2 be a matrix in $\mathbb{T}^{\{l,u\}\times Q}$;

    foreach $q_2 \in Q$ **do**

      $p_2[l, q_2] = \min_{q_1\in Q}(p[l, q_1] + E(q_1, \sigma, q_2)) - c'$;

      $p_2[u, q_2] = \min_{q_1\in Q}(p[u, q_1] + t * E(q_1, \sigma, q_2)) - c'$;

    end

    if *there is* $r \in \mathcal{Q} \cup Q'$ *such that* r *refines* p_2 **then**

      $E'(p, \sigma, r) := c'$;

    else if $p_2 \neq \infty^{\{l,u\}\times Q}$ **then**

      $E'(p, \sigma, p_2) := c'$;

      Enqueue$(\mathcal{Q}, p_2)$;

    end

  end

end

$I' := Q' \to \mathbb{T}, q \mapsto \begin{cases} i' & q = q_0, \\ \infty & \text{else} \end{cases}$;

$E' := E' \vee \infty$;

return (Q', I', F', E');

Algorithm 3: Algorithm to approximately determinize a weighted automaton over an alphabet Σ with weights in the tropical semiring $\mathbb{T}$ - adapted from [AKL13].

5 Approximate determinization over other semirings

The previously presented algorithm is somewhat limited in its applicability: it is only defined for weighted automata over the tropical semiring.

We have adapted it to a larger set of semirings using factorizations as Kirsten and Mäurer did in [KM05].

To be precise, our algorithm can be applied to any commutative totally ordered semiring $(S, \leq)$ that allows maximal factorization and where the following pair of logical equivalences holds:

For any $c \in S$ with $c \neq 0$ we have $\forall a, b, c \in S, a \leq b \iff a \odot c \leq b \odot c$ as well as $\forall a, b, c \in S, a \leq b \iff c \odot a \leq c \odot b$.

The output states of this algorithm again each associate ranges of weights to the input states, but do note that Aminof's algorithm is not a (direct) instance of this algorithm due to the fact that its output space-state may use the entirety of $\mathbb{R} \cup \infty$ in ranges rather than solely elements of the tropical semiring $\mathbb{T} = \mathbb{R}_{\geq 0} \cup \infty$ itself.

In the case of weighted automata over the boolean semiring $\mathbb{B}$ (equipped with the conventional order such that $0 < 1$), our algorithm is equivalent to the standard powerset construction since the trivial factorization is the only factorization of $\mathbb{B}$ and since the identity function is the only range function on $\mathbb{B}$.

The algorithm can be found in Algorithm 4.

It takes a WFA over any suitable semiring S as well as a range function $\tau : S \to S$ as its input and attempts to construct a DWFA that is τ-equivalent to its input WFA.

If this algorithm terminates, the result is a DWFA over S that τ-approximates the given WFA (see Theorem 5.13).

In Theorem 5.14 and Theorem 5.14 we will provide sufficient conditions for the termination of Algorithm 4 for certain input over particular semirings.

Algorithm: WFA-AppDet

Input: A weighted finite automaton (Q, I, F, E) over S as well as a range function $\tau : S \to S$

Output: A deterministic WFA (Q', I', F', E') that τ-approximates (Q, I, F, E)

Let (f, g) be a maximal factorization of $S^{\{l,u\}\times Q}$;
$Q' := \emptyset$;
$F' := S^{\{l,u\}\times Q} \multimap S$;
$E' := S^{\{l,u\}\times Q} \times \Sigma \times S^{\{l,u\}\times Q} \multimap S$;
Let q'_0 be a matrix in $S^{\{l,u\}\times Q}$;
$\mathrm{row}_l(q'_0) := \lambda_{\mathcal{A}}$;
$\mathrm{row}_u(q'_0) := (\tau \circ \lambda_{\mathcal{A}})$;
$\mathcal{Q} :=$ empty queue;
Enqueue$(\mathcal{Q}, f(q'_0))$;
while $\mathcal{Q}$ *not empty* **do**
    $p :=$ Dequeue$(\mathcal{Q})$;
    $Q' := Q' \cup \{p\}$;
    $F'(p) := \mathrm{row}_l(p)^T \cdot \rho_{\mathcal{A}}$;
    foreach $\sigma \in \Sigma$ **do**
        $p_2 := f(\gamma_{\mathcal{A}}(p, \sigma))$;
        $c := g(\gamma_{\mathcal{A}}(p, \sigma))$;
        if *there is* $r \in \mathcal{Q} \cup Q'$ *such that* $r = p_2$ **then**
            $E'(p, \sigma, r) := c$;
        else if *there is* $r \in \mathcal{Q} \cup Q'$ *such that* r *refines* p_2 **then**
            $E'(p, \sigma, r) := c$;
        else if $p_2 \neq 0^{\{l,u\}\times Q}$ **then**
            $E'(p, \sigma, p_2) := c$;
            Enqueue$(\mathcal{Q}, p_2)$;
        end
    end
end
$I' := E' \mapsto S, q \mapsto \begin{cases} g(q'_0) & q = f(q'_0), \\ 0 & \text{else} \end{cases}$;
$E' := E' \vee 0$;
return (Q', I', F', E');

Algorithm 4: Algorithm to approximately determinize a weighted automaton over an alphabet Σ with weights in a suitable semiring S.

5.1 General Results

We will now provide a few relatively generic results relating to weighted automata. The following lemma was inspired by a step in the termination proof in [KM05]. We have expressed it in terms of weighted automata here, but it can be applied to be graphs in general.

Lemma 5.1. Let Σ be an alphabet, let S be a semiring and let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over Σ and S. Let $w \in \Sigma^*$ and a set $\pi \subseteq Q \overset{w}{\rightsquigarrow} Q$ of paths in $\mathcal{A}$ labelled w be given such that π has no synchronous loops. Then $|w| \leq |Q|^{|\pi|}$.

We will prove this using a combinatorial argument.

Proof. Let Σ be an alphabet, let S be a semiring and let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over Σ and S. Let $w \in \Sigma^*$ and a set $\pi \subseteq Q \overset{w}{\rightsquigarrow} Q$ of paths in $\mathcal{A}$ labelled w be given such that π has no synchronous loops.

Note that for each $0 \leq i < j \leq |w|$, the tuples $(\alpha_i \mid \alpha \in \pi)$ and $(\alpha_j \mid \alpha \in \pi)$ consisting of the i'th and j'th states visited by each of the paths must be distinct.

Furthermore, note that there are exactly $|Q|^{|\pi|}$ different tuples of $|\pi|$ states, and note that each such combination can occur at most once as the list of states visited by each of the paths at a specific index i in these paths.

Therefore, the number of tuples of states visited by π is less than $|Q|^{|\pi|}$, and thus $w \leq |Q|^{|\pi|}$. □

Note that this implies that any set of n paths each of length $m > n$ has at least one synchronous loop, and that one can reduce any such set to paths of length at most n by removing synchronous loops. The next lemma was also taken from [KM05], for a proof we refer to said paper.

Lemma 5.2. Let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over a min-semiring S and an alphabet Σ. Let a word $w \in \Sigma^*$ as well as two states $q_0, q_n \in Q$ be given arbitrarily. Then, if $\theta(w)[q_0, q_n] \neq 0$, there is a victorious path for $q_0 \overset{w}{\rightsquigarrow} q_n$.

Note that this implies that for any two sets $P, R \subseteq Q$ of states, if there is a path $P \overset{w}{\rightsquigarrow} R$ that has non-zero weight, there is a victorious path for $P \overset{w}{\rightsquigarrow} R$.

Lemma 5.3. Let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over a min-semiring S and an alphabet Σ.

Then, for any word $w = \sigma_1\sigma_2 \dots \sigma_n \in \Sigma^*$, for any victorious path α labeled w and for any index i such that $0 \leq i \leq n$, there is a victorious path α' such that it coincides with α at the 0'th, the i'th and the n'th state and such that the subpaths $\alpha'_0, \alpha'_1, \dots, \alpha'_i$ and $\alpha'_i, \alpha'_{i+1}, \dots, \alpha'_n$ are respectively victorious for $\alpha_0 \overset{\sigma_1\sigma_2\dots\sigma_i}{\rightsquigarrow} \alpha_i$ and $\alpha_i \overset{\sigma_i\sigma_{i+1}\dots\sigma_n}{\rightsquigarrow} \alpha_n$.

Proof. Let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over a min-semiring S and an alphabet Σ. Let a word $w = \sigma_1\sigma_2 \dots \sigma_n \in \Sigma^*$ as well as a pair of subsets $P_0, P_n \subseteq Q$ be given such that there is a non-zero path in $P_0 \overset{w}{\rightsquigarrow} P_n$. Let α be a victorious path for $P_0 \overset{w}{\rightsquigarrow} P_n$, and let an index i be given such that $0 \leq i \leq n$.

Then, by Lemma 5.2, there are victorious paths for $\alpha_0 \overset{\sigma_1\sigma_2\dots\sigma_i}{\rightsquigarrow} \alpha_i$ and for $\alpha_i \overset{\sigma_i\sigma_{i+1}\dots\sigma_n}{\rightsquigarrow} \alpha_n$.

Let such victorious paths be given and let α' be their concatenation. Note that α and α' coincide at the 0'th, the i'th and the n'th state, and that α' is therefore a path in $P_0 \overset{w}{\rightsquigarrow} P_n$.

Furthermore, note that the weight of path α' is less or equal to the weight of path α, since it is composed of two parts whose weights are less or equal than the weights of the corresponding parts of α.

Thus, α' is victorious for $P_0 \overset{w}{\rightsquigarrow} P_n$. □

Lemma 5.4. Let $\mathcal{A} = (Q, I, F, E)$ be a DWFA over a min-semiring S and an alphabet Σ.

Then, for any word $w = \sigma_1\sigma_2 \dots \sigma_n \in \Sigma^*$, for any victorious path α labeled w and for any sequence $(i_j)_{0 \leq j \leq m}$ of indices such that $0 = i_0 \leq i_1 \leq \cdots i_m = n$ there is a victorious path α' such that it coincides with α at the i_j'th state for all j and such that the subpaths $\alpha'_{i_j}, \alpha'_{i_j+1}, \dots, \alpha'_{i_{j+1}}$ are each victorious for $\alpha_{i_j} \overset{\sigma_{i_j}\sigma_{i_j+1}\dots\sigma_{i_{j+1}}}{\rightsquigarrow} \alpha_{i_{j+1}}$.

Proof. This proof is wholly analogous to the proof of Lemma 5.3. □

Lemma 5.5. If (f, g) is maximal, then for any state q' added to Q' during the execution of Algorithm 4, there is a word $w = \sigma_1\sigma_2 \cdots \sigma_n \in \Sigma^*$ and a matrix $r' \in S^{\{l,u\} \times Q}$ such that $q' = f(r')$, such that $\mathrm{row}_l(r') = \lambda_\mathcal{A} \cdot \theta_\mathcal{A}(w)$, and such that $\mathrm{row}_u(r') = \tau \circ \mathrm{row}_l(r')$.

Proof. Let a state q' that is added to Q' during the execution of Algorithm 4 be given arbitrarily. Note that there is exactly one word $w \in \Sigma^*$ such that there is a path $f(q'_0) \overset{w}{\rightsquigarrow} q'_0$ such that all of its transitions are added to E' in Line 23 of Algorithm 4. Let $w = \sigma_1\sigma_2 \cdots \sigma_n \in \Sigma^*$ be this word. By inspecting the algorithm, we find that

$$q' = f(\gamma_\mathcal{A}(f(\gamma_\mathcal{A}(\cdots f(\gamma_\mathcal{A}(f(\gamma_\mathcal{A}(f(q'_0), \sigma_1)), \sigma_2)), \cdots)), \sigma_n)).$$

Since (f, g) is a maximal factorization, for any non-zero constants $c_1, c_2, \dots, c_n)$ we have

$$q' = f(c_n\gamma_\mathcal{A}(f(c_{n-1}\gamma_\mathcal{A}(\cdots f(c_2\gamma_\mathcal{A}(f(c_1\gamma_\mathcal{A}(f(q'_0), \sigma_1)), \sigma_2)), \cdots)), \sigma_n)).$$

Then, since $\gamma_\mathcal{A}$ is linear in its first argument, we find that

$$q' = f(\gamma_\mathcal{A}(c_n f(\gamma_\mathcal{A}(c_{n-1} \cdots c_3 f(\gamma_\mathcal{A}(c_2 f(\gamma_\mathcal{A}(c_1 f(q'_0), \sigma_1)), \sigma_2)), \cdots)), \sigma_n)).$$

By inspecting the definition of $\gamma_\mathcal{A}$, we find the following equalities:

$$\begin{aligned}
\mathrm{row}_l(r') &= \mathrm{row}_l({q'}_0) \cdot \theta_\mathcal{A}(\sigma_1) \cdot \theta_\mathcal{A}(\sigma_2) \cdot \cdots \cdot \theta_\mathcal{A}(\sigma_n) \\
&= {q'}_0 \cdot \theta_\mathcal{A}(w) \\
&= \lambda_\mathcal{A} \cdot \theta_\mathcal{A}(w) \\
\mathrm{row}_u(r') &= \mathrm{row}_u({q'}_0) \cdot (\tau \circ \theta_\mathcal{A}(\sigma_1)) \cdot (\tau \circ \theta_\mathcal{A}(\sigma_2)) \cdot \cdots \cdot (\tau \circ \theta_\mathcal{A}(\sigma_n)) \\
&= (\tau \circ \lambda_\mathcal{A}) \cdot (\tau \circ \theta_\mathcal{A}(w)) \\
&= \tau \circ (\lambda_\mathcal{A} \cdot \theta_\mathcal{A}(w))
\end{aligned}$$

This gives the desired result. □

Lemma 5.6. If S is a min-semiring and (f, g) is maximal, then for any state q' added to Q' during the execution of Algorithm 4, there is a word $w = \sigma_1\sigma_2 \dots \sigma_n \in \Sigma^*$ and a matrix $r' \in S^{\{l,u\}\times Q}$ such that $q' = f(r')$ and such that for any $q_n \in Q$ with $r'[l, q_n] \neq 0$ there is a partial run of w in $\mathcal{A}$ ending in q_n such that its weight is $r'[l, q_n]$.

Proof. This follows by combining the result from Lemma 5.5 with Lemma 5.2, since for any word $w \in \Sigma^*$ and for any state $q_n \in Q$, the sum of the weights of all partial runs of w that end in q_n is equal to $(\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(w))[q_n]$. □

5.2 Suitability

We will now derive some properties of the output $\mathcal{A}' = (Q', I', F', E')$ of Algorithm 4 under the assumption that it terminates on some particular input $\mathcal{A} = (Q, I, F, E)$.

First of all, we will formalize this assumption:

Assumption ∗. Let Σ be an alphabet.

Let $(S, \oplus, \odot, 0, 1)$ together with a relation $\leq$ on S be a totally ordered commutative semiring such that S allows maximal factorization and such that for any $c \in S$ with $c \neq 0$ $\forall a, b, c \in S, a \leq b \iff a \odot c \leq b \odot c$ and that $\forall a, b, c \in S, a \leq b \iff c \odot a \leq c \odot b$.

Let $\tau : S \to S$ be a range function on S and let $\mathcal{A} = (Q, I, F, E)$ be a WFA over S and Σ such that Algorithm 4 terminates on the input $(\mathcal{A}, \tau)$. Let $\mathcal{A}' = (Q', I', F', E')$ be the associated output.

Theorem 5.7. *Under Assumption ∗, the output $\mathcal{A}'$ is a DWFA.*

Proof. Let $q'_1 \in Q$ and $\sigma \in \Sigma$ be given arbitrarily.

First of all, due to the definition of I' in Line 28 of Algorithm 4, there is exactly one $q' \in Q'$ such that $I'(q') \neq 0$. Thus, this state is the sole initial state of $\mathcal{A}'$ and is equal to $f(q'_0)$.

Furthermore, since any state refines itself, a given matrix $q \in S^{\{l,u\}\times Q}$ is enqueued at most once. This is in particular also true for q'_1 - to be more precise, it was enqueued precisely once.

Thus, the loop body of the main loop was also executed precisely once for each q'_1. Therefore, the inner loop body is executed precisely once for the pair q'_1, σ and thus, there is exactly one $q'_2 \in Q'$ such that $E'(q'_1, \sigma, q'_2)$ is defined.

Since $E' \vee 0$ is equal to 0 whenever E' is undefined, there is at most one $q'_2 \in Q'$ such that $(E' \vee 0)(q'_1, \sigma, q'_2$ is non-zero. Thus, if the algorithm terminates, the resulting WFA is deterministic. □

Lemma 5.8. Under Assumption ∗, the states of the resulting DWFA are all reachable.

Proof. It is quite apparent that the initial state of the DWFA is reachable by definition.

Furthermore, since the states of the resulting DWFA are precisely the states that are enqueued into $\mathcal{Q}$ by the algorithm, it suffices to show that all states that are enqueued are reachable.

Whenever a state other than the initial state is enqueued, a non-zero edge is created from a state that was previously enqueued to the newly enqueued state.

Thus, if Algorithm 4 terminates, all of the states of the resulting DWFA are reachable. □

Now, we will show that if Algorithm 4 terminates, the states of the resulting DWFA each map states of the original WFA onto well-defined non-empty ranges in S.

Lemma 5.9. Under Assumption $*$, for any $q' \in Q'$ and for any $q \in Q$, the inequality $q'[l, q] \leq q'[u, q]$ holds.

Proof. Since the states of the resulting DWFA are precisely the states that are enqueued into $\mathcal{Q}$ by Algorithm 4, it suffices to show that all states that are enqueued induce ranges.

First, let us consider the initial state of the output automaton $\mathcal{A}' = (Q', I', F', E')$. As we have seen in Theorem 5.7, the initial state of the output is $f(q'_0)$.

Due to Line 7 of Algorithm 4, we know that for any $q \in Q$, $q'_0[l, q] = \lambda_A[q]$ and, similarly, due to Line 8 of Algorithm 4, we know that for any $q \in Q$, $q'_0[u, q] = (\tau \circ \lambda_A)[q]$.

Thus, we find:

$$\begin{aligned} q'_0[l, q] &= \lambda_{\mathcal{A}}[q] && \text{(Due to Line 7 of Algorithm 4)} \\ &\leq \tau(\lambda_{\mathcal{A}}[q]) && \text{(Since } \forall x \in S, x \leq \tau(x)\text{)} \\ &= (\tau \circ \lambda_{\mathcal{A}})[q] \\ &= q'_0[u, q] && \text{(Due to Line 8 of Algorithm 4)} \end{aligned}$$

By factorizing q'_0 on both sides of this inequality, we can derive

$$f(q'_0)[l, q] \odot g(q'_0) \leq f(q'_0)[u, q] \odot g(q'_0).$$

Then, since multiplying by a nonzero factor such as $g(q'_0)$ preserves order (as we assumed in Assumption $*$), for any $q \in Q$ we find:

$$f(q'_0)[l, q] \leq f(q'_0)[u, q].$$

Thus, if $q' \in Q'$ is the initial state, $\forall q \in Q$ the inequality $q'[l, q] \leq q'[u, q]$ holds.

Next, we will show that the equality holds for any non-initial state $q' \in Q'$ if it holds for all states enqueued before q'. Let a non-initial state $p_2 \in Q'$ be given such that the inequality $q'[l, q] \leq q'[u, q]$ for all $q' \in Q'$ enqueued before p_2 and $\forall q \in Q$. Let $\sigma \in \Sigma$ and $p_1 \in Q'$ be given such that p_2 is enqueued while "following" the edges labelled σ from p_1.

Then, since states are processed after being enqueued, we have $p_1[l, q] \leq p_2[u, q]$ for all $q \in Q$. We can derive the following inequalities, where each implies the ones that follow:

$$\begin{aligned} p_1[l, q_1] &\leq p_1[u, q_1], && \forall q_1 \in Q \\ p_1[l, q_1] \odot E(q_1, \sigma, q_2) &\leq p_1[u, q] \odot \tau(E(q_1, \sigma, q_2)), && \forall q_1, q_2 \in Q \\ \bigoplus_{q_1 \in Q} p_1[l, q_1] \odot E(q_1, \sigma, q_2) &\leq \bigoplus_{q_1 \in Q} p_1[u, q] \odot \tau(E(q_1, \sigma, q_2)), && \forall q_2 \in Q \\ (\mathrm{row}_l(p_1) \cdot \theta_{\mathcal{A}}(\sigma))[q_2] &\leq (\mathrm{row}_u(p_1) \cdot \tau \circ \theta_{\mathcal{A}}(\sigma))[q_2], && \forall q_2 \in Q \\ \gamma_{\mathcal{A}}(p_1)[l, q_2] &\leq \gamma_{\mathcal{A}}(p_1)[u, q_2], && \forall q_2 \in Q \\ p_2[l, q_2] \odot g(\gamma_{\mathcal{A}}(p_1)) &\leq p_2[u, q_2] \odot g(\gamma_{\mathcal{A}}(p_1)), && \forall q_2 \in Q \\ p_2[l, q_2] &\leq p_2[u, q_2], && \forall q_2 \in Q \end{aligned}$$

Thus, since the states are enqueued sequentially, the inequality holds for all states of Q' by natural induction on the order in which they are enqueued. □

Lemma 5.10. Under Assumption $*$, for any $w \in \Sigma^*$ and for any $q_2 \in Q$, the inequality $\beta_{\mathcal{A}}(w, q_2) \leq \delta_{\mathcal{A}'}(w)[l, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w))$ holds.

Proof. Let us prove this using induction on the length $n = |w|$ of words $w \in \Sigma^*$. First of all, for the base case $n = 0$ we need only consider the empty word ε. Let $q_2 \in Q$ be given arbitrarily. Then we find:

$$\begin{aligned}
\beta_{\mathcal{A}}(\varepsilon, q_2) &= (\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(\varepsilon))[q_2] \\
&= \lambda_{\mathcal{A}}[q_2] \\
&= I(q_2) \\
&= q_0'[l, q_2] \\
&= (f(q_0') \odot g(q_0'))[l, q_2] \\
&= (q_0' \odot \beta_{\mathcal{A}'}(\varepsilon, q_0'))[l, q_2] \\
&= (\delta_{\mathcal{A}'}(\varepsilon) \odot \beta_{\mathcal{A}'}(\varepsilon, \delta_{\mathcal{A}'}(\varepsilon)))[l, q_2] \\
&= \delta_{\mathcal{A}'}(\varepsilon)[l, q_2] \odot \beta_{\mathcal{A}'}(\varepsilon, \delta_{\mathcal{A}'}(\varepsilon))
\end{aligned}$$

Thus, the claim holds for all $q_2 \in Q$ and $w \in \Sigma^*$ such that $|w| = 0$.

Now, for the induction step, let us assume that the claim holds for all $q_2 \in Q$ and all words of length at most n. Let $q_2 \in Q$ and a word $w \in \Sigma^*$ of length $n + 1$ be arbitrarily given. Next, let $\sigma \in \Sigma$ be the symbol and $v \in \Sigma^*$ be the word such that $v\sigma = w$. Then:

$$\begin{aligned}
&\delta_{\mathcal{A}'}(w)[l, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \\
&= \delta_{\mathcal{A}'}(w)[l, q_2] \odot E'(\delta_{\mathcal{A}'}(v), \sigma, \delta_{\mathcal{A}'}(w)) \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to the def. of } \beta_{\mathcal{A}'}\text{)} \\
&= \delta_{\mathcal{A}'}(w)[l, q_2] \odot g(\gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma)) \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to Line 21 of Algorithm 4)} \\
&\geq f(\gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma))[l, q_2] \odot g(\gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma))) && \text{(Due to Line 21 and Line 23)} \\
&\quad \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) \\
&= \gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma)[l, q_2] \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Since } (f, g) \text{ is a factorization)} \\
&= (\mathrm{row}_l(\delta_{\mathcal{A}'}(v)) \cdot \theta_{\mathcal{A}}(\sigma))[q_2] \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) \\
&= \bigoplus_{q_1 \in Q} \delta_{\mathcal{A}'}(v)[l, q_1] \odot E(q_1, \sigma, q_2) \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to the def. of matrix mult.)} \\
&\geq \bigoplus_{q_1 \in Q} \beta_{\mathcal{A}}(v, q_1) \odot E(q_1, \sigma, q_2) && \text{(Due to the induction hypothesis)} \\
&= \bigoplus_{q_1 \in Q} (\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(v))[q_1] \odot E(q_1, \sigma, q_2) && \text{(Due to the def. of } \beta_{\mathcal{A}}\text{)} \\
&= (\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(v) \cdot \theta_{\mathcal{A}}(\sigma))[q_2] && \text{(Due to the def. of matrix mult.)} \\
&= (\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(w))[q_2] \\
&= \beta_{\mathcal{A}}(w, q_2).
\end{aligned}$$

By induction, the lemma follows. □

Lemma 5.11. Under Assumption $*$, for any $w \in \Sigma^*$ and for any $q_2 \in Q$, the inequality $\delta_{\mathcal{A}'}(w)[u, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \leq \tau(\beta_{\mathcal{A}}(w, q_2))$ holds.

Proof. Let us prove this using induction on the length $n = |w|$ of words $w \in \Sigma^*$, more or less analogously to the proof of Lemma 5.10. First of all, for the base case $n = 0$ we need only consider the empty word ε. Let $q_2 \in Q$ be given arbitrarily. Then we find:

$$\begin{aligned}
\tau(\beta_{\mathcal{A}}(\varepsilon, q_2)) &= \tau((\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(\varepsilon))[q_2]) \\
&= \tau(\lambda_{\mathcal{A}}[q_2]) \\
&= \tau(I(q_2)) \\
&= q_0'[u, q_2] \\
&= (f(q_0') \odot g(q_0'))[u, q_2] \\
&= (q_0' \odot \beta_{\mathcal{A}'}(\varepsilon, q_0'))[u, q_2] \\
&= (\delta_{\mathcal{A}'}(\varepsilon) \odot \beta_{\mathcal{A}'}(\varepsilon, \delta_{\mathcal{A}'}(\varepsilon)))[u, q_2] \\
&= \delta_{\mathcal{A}'}(\varepsilon)[u, q_2] \odot \beta_{\mathcal{A}'}(\varepsilon, \delta_{\mathcal{A}'}(\varepsilon))
\end{aligned}$$

Thus, the claim holds for all $q_2 \in Q$ and $w \in \Sigma^*$ such that $|w| = 0$.

Now, for the induction step, let us assume that the inequality holds for all $q_2 \in Q$ and all words of length at most n. Let $q_2 \in Q$ and a word $w \in \Sigma^*$ of length $n + 1$ be arbitrarily given. Then let $\sigma \in \Sigma$ be the letter and $v \in \Sigma^*$ be the word for which $v\sigma = w$. Then:

$$\begin{aligned}
&\delta_{\mathcal{A}'}(w)[u, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \\
&= \delta_{\mathcal{A}'}(w)[u, q_2] \odot E'(\delta_{\mathcal{A}'}(v), \sigma, \delta_{\mathcal{A}'}(w)) \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to the def. of } \beta_{\mathcal{A}'}\text{)} \\
&= \delta_{\mathcal{A}'}(w)[u, q_2] \odot g(\gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma)) \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to Line 21 of Algorithm 4)} \\
&\leq f(\gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma))[u, q_2] \odot g(\gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma))) \\
&\quad \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to Line 21 and Line 23)} \\
&= \gamma_{\mathcal{A}}(\delta_{\mathcal{A}'}(v), \sigma)[u, q_2] \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Since } (f, g) \text{ is a factorization)} \\
&= (\mathrm{row}_u(\delta_{\mathcal{A}'}(v)) \cdot (\tau \circ \theta_{\mathcal{A}}(\sigma)))[q_2] \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) \\
&= \bigoplus_{q_1 \in Q} \delta_{\mathcal{A}'}(v)[u, q_1] \odot \tau(E(q_1, \sigma, q_2)) \odot \beta_{\mathcal{A}'}(v, \delta_{\mathcal{A}'}(v)) && \text{(Due to the def. of matrix mult.)} \\
&\leq \bigoplus_{q_1 \in Q} \tau(\beta_{\mathcal{A}}(v, q_1)) \odot \tau(E(q_1, \sigma, q_2)) && \text{(Due to the induction hypothesis)} \\
&= \tau\left(\bigoplus_{q_1 \in Q} (\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(v))[q_1] \odot E(q_1, \sigma, q_2)\right) && \text{(Due to the def. of } \beta_{\mathcal{A}}\text{)} \\
&= \tau\left((\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(v) \cdot \theta_{\mathcal{A}}(\sigma))[q_2]\right) && \text{(Due to the def. of matrix mult.)} \\
&= \tau((\lambda_{\mathcal{A}} \cdot \theta_{\mathcal{A}}(w))[q_2]) \\
&= \tau(\beta_{\mathcal{A}}(w, q_2))
\end{aligned}$$

By induction, the lemma follows. □

Theorem 5.12. *Under Assumption $*$, the resulting WFA τ-approximates the input.*

Proof. By applying Lemmas 5.9, 5.10, and 5.11, we find that $\forall w \in \Sigma^*, \forall q_2 \in Q$ the following series of inequalities hold:

$$\beta_{\mathcal{A}}(w, q_2) \leq \delta_{\mathcal{A}'}(w)[l, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \leq \tau(\beta_{\mathcal{A}}(w, q_2)). \tag{1}$$

By multiplying the expressions in Equation (1) with $F(q_2)$ and $\tau(F(q_2))$, we can derive the following series of inequalities:

$$\begin{aligned} \beta_{\mathcal{A}}(w, q_2) \odot F(q_2) &\leq \delta_{\mathcal{A}'}(w)[l, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \odot F(q_2) \\ &\leq \tau(\beta_{\mathcal{A}}(w, q_2)) \odot F(q_2) \\ &\leq \tau(\beta_{\mathcal{A}}(w, q_2)) \odot \tau(F(q_2)). \end{aligned} \tag{2}$$

And by summing over all states in Q we find:

$$\begin{aligned} \bigoplus_{q_2 \in Q} \beta_{\mathcal{A}}(w, q_2) \odot F(q_2) &\leq \bigoplus_{q_2 \in Q} \delta_{\mathcal{A}'}(w)[l, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \odot F(q_2) \\ &\leq \bigoplus_{q_2 \in Q} \tau(\beta_{\mathcal{A}}(w, q_2)) \odot \tau(F(q_2)). \end{aligned} \tag{3}$$

Next, we will simplify each of the expressions in Equation (3). First, note that for all $w \in \Sigma^*$ we have $\bigoplus_{q_2 \in Q} \beta_{\mathcal{A}}(w, q_2) \odot F(q_2) = |A|(w)$. Since τ is compatible with $\odot$ as well as with $\oplus$, we find $\bigoplus_{q_2 \in Q} \tau(\beta_{\mathcal{A}}(w, q_2)) \odot \tau(F(q_2)) = \tau(|A|(w))$. Furthermore,

$$\begin{aligned} \bigoplus_{q_2 \in Q} \delta_{\mathcal{A}'}(w)[l, q_2] \odot \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \odot F(q_2) &= \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \odot \bigoplus_{q_2 \in Q} \delta_{\mathcal{A}'}(w)[l, q_2] \odot F(q_2) \\ &= \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \odot \mathrm{row}_l(\delta_{\mathcal{A}'}(w))^T \cdot \rho_{\mathcal{A}} \\ &= \beta_{\mathcal{A}'}(w, \delta_{\mathcal{A}'}(w)) \odot F'(\delta_{\mathcal{A}'}(w)) \\ &= |\mathcal{A}'|(w). \end{aligned} \tag{4}$$

Now, we can simplify the expressions in Equation (3) using Equation (4) and the equalities stated above, and thus, we find that $\forall w \in \Sigma^*$, $|\mathcal{A}|(w) \leq |\mathcal{A}'|(w) \leq \tau(|\mathcal{A}|(w))$.

Thus, if Algorithm 4 terminates, the resulting WFA τ-approximates the given input. □

Now, to summarize:

Theorem 5.13. *Let Σ be an alphabet.*

Let $(S, \oplus, \odot, 0, 1)$ together with a relation $\leq$ on S be a totally ordered, commutative semiring such that S allows maximal factorization and such that for any $c \in S$ with $c \neq 0$ $\forall a, b, c \in S, a \leq b \iff a \odot c \leq b \odot c$ and that $\forall a, b, c \in S, a \leq b \iff c \odot a \leq c \odot b$.

Let $\tau : S \to S$ be a range function on S and let $\mathcal{A} = (Q, I, F, E)$ be a WFA over S and Σ.

Then, if the algorithm terminates on the input $(\mathcal{A}, \tau)$, its output $\mathcal{A}'$ is a τ-approximation of $\mathcal{A}$.

Proof. This follows directly from Theorem 5.12. □

Thus, we have found sufficient conditions for the suitability of our algorithm. However, our algorithm may produce suitable output even in situations where some of the conditions in Theorem 5.13 are not met. This set of conditions simply forms the largest class for which we have been able to prove suitability.

5.3 Finite Termination

We will now prove (somewhat trivially) that the algorithm terminates for all weighted finite automata over finite semirings.

Recall, however, that we require a few additional conditions for the suitability of the output of the algorithm. These may be found immediately above, in Theorem 5.13.

Theorem 5.14. *If the semiring S is finite (that is, if it has a finite number of elements), Algorithm 4 will terminate.*

Proof. Let us assume that S is finite. In this case, the set of matrices $S^{\{l,u\} \times Q}$ is finite too.

Every state of $\mathcal{A}'$ is an element of $S^{\{l,u\} \times Q}$. Therefore, at most finitely many distinct states are added to the queue.

No state is added to the queue twice, therefore Algorithm 4 will terminate if S is finite. □

5.4 Termination with τ-twins property

Now, we will address the situation where S is infinite. We have adapted a proof from [AKL13] to prove a sufficient condition for the termination of our algorithm in the case of automata over a semiring that shares some of the properties of the integers.

We will do so by first assuming that the algorithm does not terminate. Under this assumption, we will derive some properties of the computation and its (intermediary) output. Finally, by means of contradiction, we will provide a sufficient condition for the termination of Algorithm 4 on certain infinite semirings.

5.4.1 Required Properties

For this proof, we will make some assumptions about the semiring, about the WFA itself, and about the factorization used in our algorithm.

First of all, our semiring S needs to be totally ordered, commutative, and for any $a, b \in S$ we need to have $a \oplus b \in \{a, b\}$. This implies that there is a total order on S such that $\oplus$ is the minimum with respect to this order. However, this order is not necessarily equal to the order which we will be using.

Furthermore, the multiplicative monoid $(S \setminus \{0\}, \odot, 1)$ needs to be cancellative.

Finally, it needs to have the property that any bounded sequence in S has an infinite, constant subsequence. This is the case if and only if any open range in S is finite.

Our factorization (f, g) needs to be maximal, but we also require an additional property somewhat resembling the maximality property: for any matrix $M \in S^{\{l,u\} \times Q\}}$ whose upper row is not entirely 0, we need to have $\min_{q \in Q} f(q)[u, q] = 1$. That is, f needs to scale the minimum of the upper row of any matrix with a non-zero upper row to fixed point 1.

Note that the semiring $(\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$ (the min-plus semiring on the integers) has all the required properties. Furthermore, note that these requirements are quite strict, so strict even that we have not managed to find a semiring that meets them and is not isomorphic to the min-plus semiring on the integers.

5.4.2 Infinite Graphs

In the case that Algorithm 4 does not terminate, infinitely many states as well as infinitely many edges are added to the set Q' during its execution. Note, however, that the queue $\mathcal{Q}$ of states that are to be added to Q' is always finite.

While an edge may be added to the (weighted) edge-set before both of the states it connects are added to Q', these states will always be added to Q' within finitely many iterations of the main loop.

Equivalently, at any point during the execution of Algorithm 4, we can consider an "intermediary" graph consisting of all vertices that have thus far been added to Q' and $\mathcal{Q}$ as well as all edges that have been added to E'.

We will refer to the "limit" of this sequence of graphs as G. Furthermore, we will refer to its (spanning) subgraph containing only those edges that were added in Line 23 of Algorithm 4 as H. Note that H is a directed tree (with $f(q'_0)$ as its root) and note that H is a spanning subgraph of G.

Lemma 5.15. The graphs G and H each contain an infinitely long path of distinct vertices.

Proof. Since H is a directed tree with root $f(q'_0)$, for any vertex v of H there is a unique path of finite length from $f(q'_0)$ to v, we refer to the length of this path as the level of v.

Next, since any vertex has at most one outgoing edge per symbol of Σ, for any $n \in \mathbb{Z}_{\geq 0}$, there are at most finitely many vertices of level n (to be precise, at most $(\#\Sigma)^n$).

Furthermore, since we have assumed that the algorithm does not terminate, for any $n \in \mathbb{Z}_{\geq 0}$, there are at least 1 vertex of level n.

Finally, for any $n \in \mathbb{Z}_{\geq 0}$, for any vertex w of level $n+1$ there is a vertex v of level n such that there is an edge from v to w.

Thus, due to Kőnig's lemma [Kőn27], H contains an infinitely long path of distinct vertices.

Since H is a subgraph of G, this path is trivially also a path in G, therefore G contains an infinitely long path of distinct vertices. □

Now, recall that the vertices in G and H are $S^{\{l,u\}\times Q}$ matrices. The lower bounds of such states induce weighted subsets of Q which in turn induce regular subsets of Q. Thus, for any vertex $q' \in Q'$ of G we will consider the subset $\{q \in Q \mid q'[l,q] \neq 0\}$. These sets are (roughly) equivalent to the subsets of the original state space used in the powerset construction. We will write (q') to refer to this subset.

Thus, we may find the following, small result:

Lemma 5.16. For any infinitely long path α in G, there is a subset $P_0 \subseteq Q$ such that there are infinitely many distinct indices i for which $(\alpha_i) = P_0$.

Proof. Since Q has only finitely many elements, there are only finitely many distinct subsets of Q. By the pigeon hole principle, where the indices i are pigeons and the subsets of Q are the holes, we may conclude that there must be a hole containing an infinite number of pigeons.

Thus, there is a subset $P_0 \subseteq Q$ such that there are infinitely many distinct indices i for which $(\alpha_i) = P_0$. □

5.4.3 Bounds

Now, let α be an infinite path of distinct vertices of H. Furthermore, let a subset $P_0 \subseteq Q$ be given such that there are infinitely many indices i for which $(\alpha_i) = P_0$. Note that $P_0 \neq \emptyset$, since there is at most one state $q' \in Q'$ of which all lower bounds are 0. Finally, let α' be the (infinite) subsequence of α consisting of only those states for which $(\alpha_i) = P_0$.

We will derive some properties of α as well as of α' and we will construct subsequences of α with certain properties.

Lemma 5.17. There is a state $q \in P_0$ such that the sequence $\alpha'_i[l, q]$ is unbounded above or such that the sequence $\alpha'_i[u, q]$ is unbounded below.

We will prove this by contradiction.

Proof. Let us assume that for all states $q \in P_0$, the sequences $\alpha'_i[l, q]$ are bounded above and the sequences $\alpha'_i[u, q]$ are bounded below.

Let $b_u \in S$ and $b_l \in S$ be given such that for all $q \in P_0$ and for all indices i, we have $\alpha'_i[l, q] \leq b_u$ as well as $\alpha'_i[u, q] \geq b_l$.

Recall that:

- every bounded sequence has an infinite, constant subsequence (by the requirements in Section 5.4.1), that
- every sequence which is unbounded above has a strictly increasing subsequence, and that
- every sequence which is unbounded below has a strictly decreasing subsequence.

Thus, α' has a subsequence π such that for every $q \in P_0$ we have:

- either $\pi_i[u, q]$ monotonically decreases and is unbounded or $\pi_i[u, q]$ is constant and
- either $\pi_i[l, q]$ monotonically increases and is unbounded or $\pi_i[l, q]$ is constant.

This is impossible, since in this case for any two indices i, j with $i < j$ the state π_j is refined by π_i and is added to Q' after π_j, even though Line 20 of Algorithm 4 ensures that this never occurs.

Therefore, there is a $q \in P_0$ such that the sequence $\alpha'_i[l, q]$ is unbounded above or such that the sequence $\alpha'_i[u, q]$ is unbounded below. □

Lemma 5.18. There is a state $p \in P_0$ such that the sequence $\alpha'_i[l, p]$ is unbounded above.

Proof. Note that for any state q' added to Q' there is a matrix q'' such that $q' = f(q'')$ and such that $\text{row}_u(q'') = \tau \circ \text{row}_l(q'')$. Let such q'' be given. Note that $\text{row}_u(q'')$ is not the constant 0-vector, since this would imply that all entries of q'' are 0 and since such a matrix is never added to the state set Q' of the automaton being generated.

Therefore, we know that for any state $q' \in Q'$ we have $\min_{q \in Q} q'[u, q] = 1$, and thus, there is no $q \in Q$ for which the sequence $\alpha'_i[u, q]$ is unbounded below.

Thus, combined with the previous result, we may conclude that there is a state $p \in P_0$ such that the sequence $\alpha'_i[l, p]$ is unbounded above. □

Lemma 5.19. Let γ be a sequence of states in Q'. Then, there is a state $q \in Q$ such that there are infinitely many distinct indices i for which $\gamma'_i[u, q] = 1$.

Proof. Note that the upper rows of any element of γ is not entirely 0, by an argument analogous to the previous proof.

By our assumption, for any $q' \in Q'$ there is a $q \in Q$ such that $q'[u, p] = 1$. Therefore, for any index i there is a $q \in Q$ such that $\gamma'_i[u, q] = 1$.

Since there are infinitely many distinct indices and finitely many distinct states in Q, by the pigeonhole principle there is a state $q \in Q$ such that there are infinitely many distinct indices i for which $\gamma'_i[u, q] = 1$. □

Note that the natural relations of the commutative monoids $(\mathbb{Z}_{\geq 0}, +, 0)$, $(\mathbb{T}, +, 0)$ and $(\mathbb{Z}, +, 0)$ are each super-relations of their respective usual total orders. This means that one can "subtract" any lesser element from a greater element (or, when viewed multiplicatively, that one can "divide" any greater element by any lesser element).

In [AKL13], Aminof, Kupferman, and Lampert use this property of the non-negative integers to subtract t times the cost of a run of the cost of another run.

We do not want to restrict ourselves only to monoids where we can perform such subtractions, therefore we will instead use the Grothendieck group of our multiplicative monoid $(S \setminus \{0\}, \odot, 1)$.

For any two monoid elements $c, d \in (S \setminus \{0\}, \odot, 1)$, we will write $\frac{c}{d}$ to refer to their difference in the Grothendieck group of $(S \setminus \{0\}, \odot, 1)$.

Lemma 5.20. Let a word $w \in \Sigma^*$ and two runs $\alpha, \beta \in Q \overset{w}{\rightsquigarrow} Q$ of w be given. Let c, d be the weights of α and β, respectively. Then, there is a word v of length at most $|Q|^2$ along with two runs $\gamma, \delta \in Q \overset{v}{\rightsquigarrow} Q$ of v such that

$$\frac{\overline{e}}{\overline{\tau(f)}} \geq \frac{\overline{c}}{\overline{\tau(d)}}$$

where e, f are the weights of γ and δ, respectively.

Proof. If $w = \sigma_1 \sigma_2 \dots \sigma_n$ has length at most $|Q|^2$, we need not prove anything.

Let us therefore assume that w has length greater than $|Q|^2$.

By Lemma 5.1, the set $\{\alpha, \beta\}$ of runs has at least one synchronized loop. Let i, j be given such that $\{\alpha, \beta\}$ has a synchronized loop at (i, j).

We will show that the ratio between the cost of α and τ of the cost of β does not decrease if we remove the synchronized loop at (i, j) from both α and β.

Let c and d be the respective costs of the runs α and β. We will "split" the costs c and d into three parts each. Let:

- $c_1 = I(\alpha_0) \bigodot_{m=0}^{i-1} E(\alpha_m, \sigma_{m+1}, \alpha_{m+1})$,
- $c_2 = \bigodot_{m=i}^{j-1} E(\alpha_m, \sigma_{m+1}, \alpha_{m+1})$, and
- $c_3 = F(\alpha_n) \bigodot_{m=j}^{n-1} E(\alpha_m, \sigma_{m+1}, \alpha_{m+1})$.

Let d_1, d_2, and d_3 be defined analogously. Note that $c = c_1 \odot c_2 \odot c_3$ and that $d = d_1 \odot d_2 \odot d_3$. Furthermore, note that $c_1 \odot c_3$ and $d_1 \odot d_3$ are the respective weights of the runs that are created by removing the synchronous loop at (i, j) from α and b. Note that these are runs of the same word of length less than w.

Since the injection of our multiplicative monoid into its Grothendieck completion is a monoid morphism, we find the following equalities:

$$\begin{aligned}
\frac{\overline{\tau(d)}}{1} &= \frac{\overline{\tau(d_1) \odot \tau(d_2) \odot \tau(d_3)}}{1} \\
\frac{\overline{\tau(d)}}{\overline{\tau(d_1) \odot \tau(d_3)}} &= \frac{\overline{\tau(d_1) \odot \tau(d_2) \odot \tau(d_3)}}{\tau(d_1) \odot \tau(d_3)} \\
&= \frac{\overline{\tau(d_2)}}{1}
\end{aligned}
\qquad\qquad
\begin{aligned}
\frac{\overline{c}}{1} &= \frac{\overline{c_1 \odot c_2 \odot c_3}}{1} \\
\frac{\overline{c}}{\overline{c_1 \odot c_3}} &= \frac{\overline{c_1 \odot c_2 \odot c_3}}{c_1 \odot c_3} \\
&= \frac{\overline{c_2}}{1}.
\end{aligned}$$

Since our automaton has the τ-twins property, and since the injection of our multiplicative monoid into its Grothendieck completion is compatible with the order on our semiring, we find the following inequalities:

$$\begin{aligned}
c_2 &\leq \tau(d_2) && \text{(Due to the } \tau\text{-twins property)} \\
\frac{\overline{c_2}}{1} &\leq \frac{\overline{\tau(d)}}{1} && \text{(Since the injection into the Groth. compl. preserves order)} \\
\frac{\overline{c}}{\overline{c_1 \odot c_3}} &\leq \frac{\overline{\tau(d)}}{\overline{\tau(d_1) \odot \tau(d_3)}} && \text{(By filling in the earlier equalities)} \\
\frac{\overline{c}}{\overline{\tau(d)}} &\leq \frac{\overline{c_1 \odot c_3}}{\overline{\tau(d_1) \odot \tau(d_3)}} && \text{(By multiplying and cancelling out)} \\
\frac{\overline{c}}{\overline{\tau(d)}} &\leq \frac{\overline{c_1 \odot c_3}}{\overline{\tau(d_1 \odot d_3)}} && \text{(Since } \tau \text{ is compatible with } \odot\text{)}
\end{aligned}$$

Now, note that by removing the synchronous loop at (i, j) we have found a word $v = \sigma_1\sigma_2 \dots \sigma_i\sigma_{j+1} \dots \sigma_n$ that is shorter than w and for which there are two runs α' and β' with respective costs $c' = c_1 \odot c_3$ and $d' = d_1 \odot d_3$ such that

$$\frac{\overline{c}}{\overline{\tau(d)}} \leq \frac{\overline{c'}}{\overline{\tau(d')}}.$$

Thus, by natural induction on the length of w, there must be a word u of length at most $|Q|^2$ along with two runs γ and δ of u with respective weights e and f such that

$$\frac{\overline{c}}{\overline{\tau(d)}} \leq \frac{\overline{e}}{\overline{\tau(f)}}.$$

□

Note that for any word w, there are only finitely many runs of $\mathcal{A}$ labelled w. Furthermore, note that there are only finitely many words of length at most $|Q|^2$. Thus, the set of pairs of equally labelled paths of length at most $|Q|^2$ is finite.

Then, the set of the value that the expression $\overline{\frac{x_1}{\tau(x_2)}}$ takes on where x_1 and x_2 are weights of partial runs of a word w of length at most $|Q|^2$ is a totally ordered, finite set. Thus, it has a maximal element. Let the maximal element x and the corresponding x_1 and x_2 be given.

Lemma 5.21. Let $w = \sigma_1\sigma_2\dots\sigma_n \in \Sigma^*$ be an arbitrary word, and let the path $\alpha, \beta \in I \overset{w}{\rightsquigarrow} Q$ be partial runs of w in $\mathcal{A}$. Furthermore, let $c, d \in S$ be the respective weights of α and β.

Then, $\overline{\frac{c}{\tau(d)}}$ (that is the equivalence class of $(c, \tau(d))$ in the Grothendieck group of S) is lesser than x.

Proof. Let $w = \sigma_1\sigma_2\dots\sigma_n \in \Sigma^*$ be an arbitrary word, and let the path $\alpha, \beta \in I \overset{w}{\rightsquigarrow} Q$ be partial runs of w in $\mathcal{A}$.

Furthermore, let $c, d \in S$ be the respective weights of α and β.

By Lemma 5.20 and by the definition of x, we have

$$\overline{\frac{c}{\tau(d)}} \leq x.$$

□

Lemma 5.22. There is a word $w \in \Sigma^*$ such that there are partial runs of w in $\mathcal{A}$ with weights c, d respectively such that $\overline{\frac{c}{\tau(d)}} > x$.

Proof. We will consider four different sequences of lower bounds in α''.

The first sequence is $(\alpha''_i[l,p])_{i\in\mathbb{Z}_{\geq 0}}$. Recall that it is unbounded above and monotonically increasing.

Our second sequence is $(\tau(\alpha''_i[l,p]))_{i\in\mathbb{Z}_{\geq 0}}$. Since τ is a range function, this sequence is also unbounded above. It is, however, not necessarily monotonically increasing.

Thus, let an infinite subsequence α''' of α'' be given such that $(\tau(\alpha'''_i[l,p]))_{i\in\mathbb{Z}_{\geq 0}}$ increases monotonically. This is our third sequence.

Our fourth and final sequence is the third sequence scaled by x_2: $x_2 \odot \tau(\alpha''_i[l,p]))_{i\in\mathbb{Z}_{\geq 0}}$. Since multiplication by a constant preserves order, our fourth sequence is a non-decreasing sequence. Furthermore, note that the elements of the third sequence are pairwise distinct and that the elements of the scaled sequence are thus also pairwise distinct, since our multiplicative monoid $(S \setminus \{0\}, \cdot, 1)$ is cancellative.

Thus, the fourth sequence is unbounded above, since it is bounded below by $x_1 \cdot \alpha'''_0[l,p]$ and since any bounded sequence over S has an element that repeats infinitely.

Therefore, there is an i such that

$$x_1 \cdot \tau(\alpha''_i[l,q]) < \alpha''_i[l,p] \cdot x_2.$$

In other words, there is an i such that

$$\overline{\frac{x_1}{x_2}} < \overline{\frac{\alpha''_i[l,p]}{\tau(\alpha''_i[l,q])}}.$$

Thus, due to Lemma 5.6, there is a word $w \in \Sigma^*$ such that there are partial runs of w in $\mathcal{A}$ with weights c, d respectively such that $\overline{\frac{c}{\tau(d)}} > x$. □

Theorem 5.23. *Thus, if our semiring and factorization have the described properties, the algorithm terminates on any WFA that has the τ-twins property.*

Proof. Since we have found two contradicting results in Lemma 5.22 and Lemma 5.21, we may conclude that our assumption was incorrect. □

And, to summarize:

Theorem 5.24. *Let Σ be an alphabet. Let $(S, \oplus, \odot, 0, 1)$ be a totally ordered, commutative semiring such that for any $a, b \in S$ we have $a \oplus b \in \{a, b\}$, such that the multiplicative monoid $(S \setminus \{0\}, \odot, 1)$ is cancellative, such that any bounded sequence in S has an infinite, constant subsequence, and such that it permits maximal factorization of its matrices.*

Let (Q, I, F, E) be a WFA over S and Σ and let $\tau : S \to S$ be a range-function on S.

Finally, let us assume that there is a maximal factorization (f, g) of $S^{\{l,u\} \times Q}$ such that for any matrix $M \in S^{\{l,u\} \times Q\}}$ whose upper row is not entirely 0, we have $\min_{q \in Q} f(q)[u, q] = 1$. Let such a factorization be given arbitrarily.

Then, Algorithm 4 terminates on the input $((Q, I, F, E), \tau)$ if the aforementioned factorization is used.

Proof. This follows directly from Theorem 5.23. □

6 Conclusion

We defined a general notion of approximate determinization of weighted automata by generalizing definitions from [AKL13]. In Algorithm 4, we have provided an algorithm to approximately determinize weighted automata.

In Theorem 5.13, we have proven that our algorithm is correct for certain inputs (that is, it produces correct output if it terminates).

Furthermore, we have shown that our algorithm terminates in the case that the weighted automata in question are defined over a finite automaton (see Theorem 5.14).

Finally, in Theorem 5.24 we have provided another sufficient condition for the termination of our algorithm in the case of weighted automata over semirings with certain properties. However, we have not found an infinite semiring that has the required properties and is not isomorphic to the min-plus semiring on the integers.

7 Further Research

While writing this thesis, we stumbled across several potentially interesting areas of future research. We will describe a selection in this section.

7.1 Other termination conditions

It is fairly easy to construct weighted automata that do not meet either of our termination conditions such that our algorithm still terminates and produces a correct output on these automata. This suggests that there may be other classes of weighted automata on which our algorithm terminates.

Particularly, it may be possible to laxen the conditions set on the semiring or the factorization by strengthening the conditions set on the automaton (and vice versa).

7.2 Subsemirings

Additionally, while our conditions for termination and adequacy are quite strict, it is possible to apply our algorithm to weighted automata over semirings that do not meet our conditions in the case that the weights of such an automaton are part of a sub-semiring that does meet our conditions. In [AKL13], Aminof, Kupferman, and Lampert use this "trick" as part of their proof of a sufficient condition for the termination of their algorithm.

It may be interesting to rigorously describe when such a "trick" is possible.

7.3 Alternate notions of approximation

Finally, exploring alternate definitions of approximate equivalence may give rise to interesting, similar algorithms, or to wholly different ones.

References

[AK09] Jürgen Albert and Jarkko Kari. "Digital Image Compression". In: *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Springer Berlin Heidelberg, 2009, pp. 453–479. DOI: 10.1007/978-3-642-01492-5_11 (cit. on p. 4).

[AKL13] Benjamin Aminof, Orna Kupferman, and Robby Lampert. "Rigorous approximated determinization of weighted automata". In: *Theoretical Computer Science* 480 (Apr. 2013), pp. 104–117. DOI: 10.1016/j.tcs.2013.02.005 (cit. on pp. 5, 19, 21, 26, 27, 38, 41, 45, 46).

[AM03] Cyril Allauzen and Mehryar Mohri. "Efficient Algorithms for Testing the Twins Property". In: *Journal of Automata, Languages and Combinatorics* 8 (2003), pp. 117–144. DOI: 10.25596/JALC-2003-117 (cit. on p. 18).

[BGC09] Christel Baier, Marcus Größer, and Frank Ciesinski. "Model Checking Linear-Time Properties of Probabilistic Systems". In: *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Springer Berlin Heidelberg, 2009, pp. 519–570. DOI: 10.1007/978-3-642-01492-5_13 (cit. on p. 4).

[Cho77] Christian Choffrut. "Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles". en. In: *Theoretical Computer Science* 5.3 (Dec. 1977), pp. 325–337. ISSN: 03043975. DOI: 10.1016/0304-3975(77)90049-4. (Visited on 09/11/2021) (cit. on p. 18).

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler, eds. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 9783642014918 9783642014925. DOI: 10.1007/978-3-642-01492-5. (Visited on 03/09/2021) (cit. on p. 5).

[KM05] Daniel Kirsten and Ina Mäurer. "On the Determinization of Weighted Automata". en. In: *Journal of Automata* Languages and Combinatorics (2005), pp. 287–312. DOI: 10.25596/JALC-2005-287. URL: http://jalc.de/issues/2005/issue_10_2-3/jalc-2005-287-312.php (cit. on pp. 5, 9, 12, 18, 25, 28, 30).

[KM09] Kevin Knight and Jonathan May. "Applications of Weighted Automata in Natural Language Processing". In: *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. Springer Berlin Heidelberg, 2009, pp. 571–596. DOI: 10.1007/978-3-642-01492-5_14 (cit. on p. 4).

[Kőn27] Dénes Kőnig. "Über eine Schlussweise aus dem Endlichen ins Unendliche". In: *Acta Scientiarum Mathematicarum (Szeged)* 3.2-3 (1927), pp. 96–106 (cit. on p. 39).

[Mah84] B. Mahr. "Iteration and Summability in Semirings". In: *Algebraic and Combinatorial Methods in Operations Research, Proceedings of the Workshop on Algebraic Structures in Operations Research*. North-Holland Mathematics Studies. Elsevier, 1984, pp. 229–256. DOI: 10.1016/s0304-0208(08)72965-7 (cit. on p. 9).

[Moh97] Mehryar Mohri. "Finite-state transducers in language and speech processing". In: *Computational Linguistics* 3.2 (1997), pp. 269–311. URL: https://dl.acm.org/doi/10.5555/972695.972698 (cit. on p. 18).

[MPR05] Mehryar Mohri, Fernando C Pereira, and Michael Riley. "Weighted Automata in Text and Speech Processing". In: *ArXiv* abs/cs/0503077 (2005). URL: https://arxiv.org/abs/cs/0503077 (cit. on p. 4).

[Pin98] Jean-Eric Pin. "Tropical semirings". In: *Idempotency*. Cambridge University Press, 1998, pp. 50–69. DOI: 10.1017/cbo9780511662508.004 (cit. on p. 9).

[RS59] M. O. Rabin and D. S. Scott. "Finite automata and their decision problems". In: *IBM Journal of Research and Development* 3.2 (1959), pp. 114–125. DOI: 10.1147/rd.32.0114 (cit. on pp. 21, 22).

[Sch61] M.P. Schützenberger. "On the Definition of a Family of Automata". In: *Information and Control* 4.2-3 (1961), pp. 245–270. DOI: 10.1016/s0019-9958(61)80020-x (cit. on p. 16).