What's Decidable About Weighted Automata?*

Shaull Almagor[1] Udi Boker[2] Orna Kupferman[3]

[1] Computer Science Department, Technion, Israel.
[2] Interdisciplinary Center (IDC) Herzliya, Israel.
[3] Hebrew University, School of Engineering and Computer Science, Jerusalem, Israel.

Abstract

Weighted automata map input words to numerical values. Applications of weighted automata include formal verification of quantitative properties, as well as text, speech, and image processing. In the 90's, Krob studied the decidability of problems on rational series, which strongly relate to weighted automata. In particular, it follows from Krob's results that the universality problem (that is, deciding whether the values of all words are below some threshold) is decidable for weighted automata over the tropical semiring with weights in $\mathbb{N} \cup \{\infty\}$ and undecidable when the weights are in $\mathbb{Z} \cup \{\infty\}$.

We continue the study of the borders of decidability in weighted automata over the tropical semiring, and further analyze the complexity bounds of the decidable problems. We give a complete picture of the decidability and complexity of the fundamental decision problems of deterministic and nondeterministic weighted automata over the tropical semiring with weights in $\mathbb{N} \cup \{\infty\}$, $\mathbb{Z} \cup \{\infty\}$, and $\mathbb{Q} \cup \{\infty\}$. The problems we consider are non-emptiness, universality, $\exists$-exact, $\forall$-exact, upper boundedness, absolute boundedness, equality, and containment. To this end, we provide alternative and direct proofs of the undecidability results, and tighten them further. Unlike the proofs of Krob, which are algebraic in their nature, our proofs stay in the terrain of state machines, and the reduction is from the halting problem of a two-counter machine. This enables us to strengthen the results to apply already to a very simple class of automata: all the states are accepting, there are no initial nor final weights, and all the weights are from the set $\{-1, 0, 1\}$.

The fact we work directly with automata enables us to tighten also the complexity bounds of the decidable problems. We provide a toolbox of algorithms and techniques for weighted automata, on top of which we establish the complexity bounds. Some of the tools we provide formalize and generalize known techniques, while others are novel.

*A preliminary version appeared in the 9th International Symposium on Automated Technology for Verification and Analysis (ATVA 2011).

1 Introduction

Traditional (Boolean) automata accept or reject their input, and are therefore Boolean. A *weighted finite automaton* (WFA, for short) has numeric weights on its transitions and maps each word to a numeric value. Applications of weighted automata include formal verification, where they are used for the verification of quantitative properties [9, 10, 14, 30, 39], for reasoning about probabilistic systems [5], and for reasoning about the competitive ratio of on-line algorithms [3], as well as text, speech, and image processing, where the weights of the automaton are used in order to account for the variability of the data and to rank alternative hypotheses [12, 34]. They have also been used to prove theorems regarding Boolean automata [20].

Technically, a weighted automaton is defined with respect to a semiring. The value of a run is the semiring-product of the weights along the transitions traversed (and the initial and final weights). The value of a word is the semiring-sum of the values of the accepting runs on it. A formalism that is equivalent to the one of weighted automata is the one of *rational series* [41]. There too, the series is defined with respect to a semiring, and maps words to values from the domain of the semiring. For the well developed theory on weighted automata, we refer the reader to [18, 37, 29, 24, 15].

Of special interest is the *tropical semiring* over the rational numbers (together with ∞), whose sum operator is min (with ∞ being the identity element) and whose product operator is $+$ (with 0 being the identity element). We only consider, unless stated differently, weighted automata over the tropical semiring with the domain of the natural numbers, integers, or rational numbers. Considering decidability questions and complexity questions that do not go below PTIME, all of our results are the same over the integers and over the rational numbers, while often very different over the natural numbers. We thus present all the results with respect to the integers and the natural numbers.

The rich structure of weighted automata makes them intriguing mathematical objects. Fundamental problems that have been solved decades ago for Boolean automata are still open or known to be undecidable in the weighted setting. For example, while in the Boolean setting, nondeterminism does not add to the expressive power of the automata, not all weighted automata can be determinized, and the problem of deciding whether a given nondeterministic weighted automaton can be determinized, is still open, in the sense we do not even know whether it is decidable.

The standard decision problems in the Boolean setting, namely non-emptiness, universality, containment, and equivalence, have natural analogues for weighted automata. For example, in the Boolean setting, the universality problem asks, given a nondeterministic automaton (NFA) $\mathcal{A}$, whether all the words in Σ^* are accepted by $\mathcal{A}$. In the weighted setting, the "goal" of the automaton is not just to accept words, but also to do it with a minimal value. Accordingly, the universality problem for WFAs asks, given a WFA $\mathcal{A}$ and a threshold ϑ, whether $\mathcal{A}$ assigns a value that is smaller than ϑ to all words in Σ^*. The other problems admit similarly flavored analogues (see Section 2.3).

In addition, the weighted setting gives rise to other decision problems, which do not have a qualitative analogue. For example, given a WFA $\mathcal{A}$, the $\exists$-*exact* problem asks whether there exists a word w that is accepted with value exactly ϑ. Another example, when $\mathcal{A}$ is defined with respect to an *ordered* semiring, in particular the tropical semiring, is the upper-boundedness problem, where we ask whether the values that words are mapped to by $\mathcal{A}$ are bounded from above. We introduce these problems and others in Section 2.3.

In the Boolean setting, the complexity of the problems we mentioned is well understood. Specifically, the complexity of universality, containment, and equivalence of NFAs are PSPACE-complete [32], and are NLOGSPACE-complete for deterministic automata, whereas the non-emptiness problem is NLOGSPACE-complete already for NFAs. In contrast, for weighted automata, already the non-emptiness problem may become undecidable, even for simple semirings [36]. As we shall see in this paper, for weighted automata over the tropical semiring, the picture is involved. Thus, the complexity and decidability of the problems at hand depend both on the determinism of the automata, as well as on the underlying semiring.

The problems we study are of great practical interest: in the automata-theoretic approach to reasoning about systems and their specifications, non-emptiness, universality, and equivalence amount to satisfiability, validity, and equivalence of specifications, respectively, and containment amounts to correctness of systems with respect to their specifications. The same motivation applies also for weighted systems, with the specifications being quantitative [9], and these also motivate the other weighted problems.

In [27], Krob proved that the universality problem for rational series is undecidable for the tropical semiring with domain $\mathbb{Z} \cup \{\infty\}$, and that this implies undecidability of the containment and equivalence problems for the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$. Moreover, in [28], Krob proved that universality for rational series defined with respect to the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$ is decidable. The equivalence between rational series and weighted automata implies the same results for the universality and containment problems for weighted automata.

In this paper we refine the borders of decidability by restricting the conditions for undecidability on one hand, and by giving complexity bounds for the decidable fragments, on the other. We now elaborate on these two aspects. On the undecidability frontier, we describe alternative and direct proofs of the above results. Our clean reduction enables us to strengthen the result to a weaker model of automata, and to make the proof generalizable to additional decision problems.

Our proofs offer the following advantages. First, unlike the undecidability proofs of Krob, which refer to rational series and are therefore algebraic in their nature, our proofs stay in the terrain of state machines. [1] In particular, while

[1]Readers familiar with both rational series and weighted automata may find this contribution straightforward, as algebraic operations like min, sum and Hadamard product of rational series have analogous constructions in weighted automata. Given, however, the intricacy of Krob's proof, in particular the fact these operations and constructions cannot be taken as

Krob's reduction is from Hilbert's 10th problem (solving a Diophantine equation), ours is from the halting problem of a two-counter machine. This enables us to significantly simplify Krob's reasoning and make the undecidability result accessible to the automata-theoretic community. Second, the clean reduction enables us to strengthen the result and show that undecidability applies already to a very simple class of automata. For example, universality is already undecidable when the weights of the automaton are in $\{-1, 0, 1\}$, it has no initial nor final weights, and all its states are accepting. Third, the pure algebraic view of rational series has the drawback that it cannot be generalized to some natural extensions of the weighted setting, in which the automata-theoretic definition is useful [4, 7, 8, 9, 10, 11, 13, 45].

Our reductions from the counter-machine halting problem use ideas similar to those presented in [13]. Given a two-counter machine $\mathcal{M}$, we define a weighted automaton $\mathcal{A}$ whose alphabet is the set of $\mathcal{M}$'s operations. We show that $\mathcal{A}$ assigns a positive value to a word w if and only if w describes the actual run of $\mathcal{M}$ and this run is halting with both counters having value 0. Intuitively, this enable us to "simulate" the run of $\mathcal{M}$ using $\mathcal{A}$. Our undecidability results for the various problems then follow, either directly or with some additional small modifications. Our proofs generalize the undecidability results of [27], and we believe that they may also be useful when investigating other models of weighted automata.

On the decidability frontier, we further study the complexity of each problem. To this end, we improve and generalize known techniques, as well as provide some novel tools and constructions. In particular, we present a standalone toolbox of constructions and algorithms for weighted automata. Our results are summarized in Table 1, providing a complete picture of the decidability and complexity of the fundamental problems for WFAs.

2 Preliminaries

In this section we define the models of automata we use throughout the paper, and the decision problems we study.

2.1 Boolean Automata

We start by briefly giving the notations we use for Boolean automata. We refer the reader to e.g., [44] for the complete definitions. A (Boolean) nondeterministic finite automaton (NFA) is a 5-tuple $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, F \rangle$ where Σ is a finite alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to 2^Q$ is a nondeterministic transition function, $Q_0 \subseteq Q$ is a set of initial states, and F is a set of accepting states. When $|Q_0| = 1$ and $|\delta(q, \sigma)| = 1$ for every $q \in Q$ and $\sigma \in \Sigma$, we say that $\mathcal{A}$ is a deterministic finite automaton (DFA). The language of $\mathcal{A}$ is denoted by $L(\mathcal{A})$.

black boxes, its adoption to the formalism of weighted automata is not trivial.

	Deterministic		Nondeterministic	
	$\mathbb{N}$-WFA	$\mathbb{Z}$-WFA	$\mathbb{N}$-WFA	$\mathbb{Z}$-WFA
Non-emptiness	PTIME Prop. 6.1			
Universality	PTIME Prop. 6.2		PSPACE-C Thm. 6.3	Undecidable Thm. 4.1; [27]
$\exists$-*Exact*	NP-C Thm. 6.7		PSPACE-C Thm. 6.10	Undecidable Thm. 4.5
$\forall$-*Exact*	PTIME Prop. 6.11		PSPACE-C Prop. 6.12	Thm. 6.13
Upper Boundedness	PTIME Prop. 6.21		PSPACE-C Thm. 6.20	Undecidable Thm. 4.6; [1]
Absolute Boundedness	PTIME Prop. 6.23		PSPACE-C Thm. 6.22	
Equality	DD: PTIME Prop. 6.15; DN, ND: PSPACE-C Thm. 6.16		NN: Undecidable Thm. 4.3	
Containment	DD, ND: PTIME Prop. 6.14		DN, NN: Undecidable Thm. 4.2	

Table 1: Complexity of solving the decision problems. In the equality and containment problems, the same results hold for $\mathbb{N}$-WFAs and $\mathbb{Z}$-WFAs, and the D/N notation indicates whether the involved automata are deterministic or nondeterministic.

2.2 Weighted Automata

We turn to describe the model of weighted automata we study. Let $\mathbb{Q}_\infty, \mathbb{Z}_\infty$, and $\mathbb{N}_\infty$ denote $\mathbb{Q} \cup \{\infty\}, \mathbb{Z} \cup \{\infty\}$, and $\mathbb{N} \cup \{\infty\}$, respectively.

While Boolean automata map words in Σ^* to either "accept" or "reject", weighted automata may be viewed as functions from Σ^* to $\mathbb{Q}_\infty$.

Formally, a *weighted finite automaton* (WFA, for short) is a 5-tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin} \rangle$, where Σ is a finite input alphabet, Q is a finite set of states, $\Delta : Q \times \Sigma \times Q \to \mathbb{Q}_\infty$ is a weighted transition function, $\mathtt{init} : Q \to \mathbb{Q}_\infty$ is an initial-weight function, and $\mathtt{fin} : Q \to \mathbb{Q}_\infty$ is a final-weight function.

A transition $\langle q, a, p \rangle \in Q \times \Sigma \times Q$ can be taken by $\mathcal{A}$ when reading the input letter a in the state q, and it causes $\mathcal{A}$ to move to the state p with *cost* $\Delta(q, a, p)$.

Note that the range of Δ, $\mathtt{init}$, and $\mathtt{fin}$ includes the value ∞. Intuitively, once the weight ∞ is traversed (either along a run, in the initial state, or in a final state), the relevant run of the automaton is "doomed". In the Boolean setting, this corresponds to non-existing transitions, a non-initial state, or a rejecting state, respectively. By abusing notation, we add the following definitions. For states $q, p \in Q$, and for a letter $a \in \Sigma$, we write $(q, a, p) \in \Delta$ if $\Delta(q, a, p) < \infty$. Similarly, we write $q \in \mathtt{init}$ if $\mathtt{init}(q) < \infty$ and $q \in \mathtt{fin}$ if $\mathtt{fin}(q) < \infty$. Also, when referring to the weights in a WFA, we only refer to weights that are not ∞.

Note that a WFA $\mathcal{A}$ may be nondeterministic in the sense that it may have many initial states, and that for some $q \in Q$ and $a \in \Sigma$, it may have $(q, a, p_1) \in \Delta$ and $(q, a, p_2) \in \Delta$, with $p_1 \neq p_2$. We say that $\mathcal{A}$ is *deterministic* if $|\{q : q \in \texttt{init}\}| = 1$ and for every $q \in \mathcal{Q}$ and $a \in \Sigma$ there exists up to a single state $p \in Q$ such that $(q, a, p) \in \Delta$. We say that $\mathcal{A}$ is *complete* if for every state $q \in Q$ and letter $a \in \Sigma$, there is at least one state $p \in Q$ such that $(q, a, p) \in \Delta$.

For a word $w = w_1 \dots w_n \in \Sigma^*$, a *run* of $\mathcal{A}$ on w is a sequence $r = r_0 r_1 \dots r_n \in Q^+$, where $r_0 \in \texttt{init}, r_n \in \texttt{fin}$, and for all $1 \leq i \leq n$, we have $d_i = \langle r_{i-1}, w_i, r_i \rangle \in \Delta$. The *cost* of the run r is $\mathsf{cost}_{\mathcal{A}}(r) = \texttt{init}(r_0) + \sum_{i=1}^{n} \Delta(d_i) + \texttt{fin}(r_n)$ (we omit the subscript $\mathcal{A}$ when it is clear from the context). Note that if $\mathcal{A}$ is nondeterministic, it may have several runs on w. The *cost* of w in $\mathcal{A}$ is $\mathcal{A}(w) = \min\{\mathsf{cost}(r) : r \text{ is a run of } \mathcal{A} \text{ on } w\}$. If the minimum is taken over an empty set, then $\mathcal{A}(w) = \infty$.

A state $q \in Q$ is called *live* if it lies on a (finite-cost) run on some word. We shall assume that all states in a WFA are live, as "dead" states can be removed in polynomial time from the automaton, obtaining an equivalent WFA.

We define $\mathsf{Dom}(\mathcal{A}) = \{w \in \Sigma^* : \mathcal{A}(w) < \infty\}$. Observe that $\mathsf{Dom}(\mathcal{A})$ is a regular language, and can be recognized by an NFA with the same structure as $\mathcal{A}$.

We note that in general, a WFA may be defined with respect to a semiring $\langle \mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1} \rangle$. The cost of a run is then the semiring product of the initial weight of the first state, the weights along the run, and the final weight of the last state. The cost of an accepted word is the semiring sum over the costs of all accepting runs on it. In this work, we focus on weighted automata defined with respect to the *min-sum semiring*, $\langle \mathbb{Q}_\infty, \min, +, \infty, 0 \rangle$, sometimes called the *tropical* semiring, as defined above.

When speaking of the complexity of solving a problem, we assume that all weights are given in binary. (A rational value is represented by a pair of integers, corresponding to its reduced fraction.) That is, we consider the *size* of a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \texttt{init}, \texttt{fin} \rangle$ to be the maximum between $|\Sigma|, |Q|^2$, and the maximal binary representation of a weight in $\Delta, \texttt{init}$, and $\texttt{fin}$.[2] Similarly, thresholds are given in binary representation.

We distinguish between subclasses of weighted automata based on the underlying domain of their weights. Specifically, if the weights are all integers, we refer to the automaton as a $\mathbb{Z}$-WFA, and if the weights are all naturals, as an $\mathbb{N}$-WFA (recall that we also allow ∞ weights). Note that for $K \in \{\mathbb{N}_\infty, \mathbb{Z}_\infty\}$ we have that $\langle K, \min, +, \infty, 0 \rangle$ is a semiring.

We can obtain from every WFA $\mathcal{A}$ a $\mathbb{Z}$-WFA $\mathcal{B}$ by multiplying rational weights by their common denominator c, such that for every word $w \in \Sigma^*$, we have that $c \cdot \mathcal{A}(w) = \mathcal{B}(w)$. For the purpose of this paper, it suffices to work with $\mathcal{B}$, as the description length of $\mathcal{B}$ is polynomial in that of $\mathcal{A}$ (since c is at most singly exponential in the denominators of the weights in $\mathcal{A}$). Therefore, in the following we only consider $\mathbb{Z}$-WFAs and $\mathbb{N}$-WFAs, unless stated otherwise.

[2]We take $|Q|^2$ as an upper bound on the number of transitions in the automaton per letter of the alphabet.

2.3 Decision Problems for WFAs

Recall that in the Boolean setting, the "goal" of an automaton is, intuitively, to accept a given word. That is, to have a run that reaches an accepting state on the word. In the weighted setting, the "goal" of the automaton is not just to accept a word, but also to do it with a minimal weight. This intuition gives rise to quantitative analogues of standard decision problems for Boolean automata, as well as to some new decision problems. For example, in the Boolean setting, the non-emptiness problem is to decide, given an automaton, whether it accepts some word. Thus, in the weighted setting, the non-emptiness problem asks whether some word is accepted below a given threshold.

We are going to study the following decision problems. [3]

- The *non-emptiness* problem is to decide, given a WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{Z}$, whether there exists a word $w \in \Sigma^*$ such that $\mathcal{A}(w) < \vartheta$.
- The *universality* problem is to decide, given a WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{Z}$, whether for every word $w \in \Sigma^*$, it holds that $\mathcal{A}(w) < \vartheta$.
- The $\exists$-*exact* problem is to decide, given a WFA $\mathcal{A}$ and a value $\vartheta \in \mathbb{Z}$, whether there exists a word $w \in \Sigma^*$ such that $\mathcal{A}(w) = \vartheta$.
- The $\forall$-*exact* problem is to decide, given a WFA $\mathcal{A}$ and a value $\vartheta \in \mathbb{Z}$, whether for every word $w \in \mathsf{Dom}(\mathcal{A})$, it holds that $\mathcal{A}(w) = \vartheta$.
- The *upper boundedness* problem is to decide whether there exists $M \in \mathbb{N}$ such that $\mathcal{A}(w) < M$, for every $w \in \Sigma^*$.
- The *absolute boundedness* problem is to decide whether there exists $M \in \mathbb{N}$ such that $-M < \mathcal{A}(w) < M$, for every $w \in \Sigma^*$.
- The *equality* problem is to decide, given two WFAs $\mathcal{A}$ and $\mathcal{B}$, whether for every word $w \in \Sigma^*$, it holds that $\mathcal{A}(w) = \mathcal{B}(w)$.
- The *containment* problem is to decide, given two WFAs $\mathcal{A}$ and $\mathcal{B}$, whether for every word $w \in \Sigma^*$, it holds that $\mathcal{A}(w) \geq \mathcal{B}(w)$.

Two comments about the problems. First, regarding the $\mathcal{A}(w) \geq \mathcal{B}(w)$ condition in the containment problem. For our confused readers, the $\geq$ there is not a typo: recall that the "goal" of a WFA is to accept a given word w with a minimal value. When $\mathcal{A}$ is contained in $\mathcal{B}$, then for every word w, the WFA $\mathcal{B}$ can achieve its goal with w at least as well as $\mathcal{A}$ can. In the Boolean setting, this amounts to $L(\mathcal{A})$ being a subset of $L(\mathcal{B})$. In the weighted setting, this amounts to the values that words are mapped to in $\mathcal{B}$ being smaller than (or equal to) the values to which they are mapped in $\mathcal{A}$. Second, observe that for the $\forall$-exact

[3]We consider only one variant of each problem, corresponding to one inequality notion. For example, the non-emptiness problem with respect to the strictly-smaller-than notion $<$. As weights are integers, it is easy to reduce one variant of a problem to another, for example our non-emptiness problem to the one with the $\leq$ notion.

problem, we restrict attention to words in $\mathsf{Dom}(\mathcal{A})$. Our results can be easily adapted to hold also for the version of the problem that requires all words in Σ^* to attain value ϑ.

In order to demonstrate the flavor of the problems, consider the universality problem. In the Boolean setting, the universality problem asks, given a nondeterministic automaton (NFA) $\mathcal{A}$, whether $L(\mathcal{A}) = \Sigma^*$. Thus, the automaton has to accept all the words in Σ^*. In the weighted setting, the "goal" is not just to accept, but to do so with a minimal value. Accordingly, the universality problem for WFAs asks, given a WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{Z}$, whether $\mathcal{A}(w) < \vartheta$ for all $w \in \Sigma^*$. We denote the latter fact by $\mathcal{A} < \vartheta$. The rest of the problems follow a similar flavor.

It is easy to see that an upper bound on the complexity of the containment problem implies upper bounds on the complexity of the equality problem (by checking two-sided containment) and the universality problem. Likewise, a lower bound on the universality problem implies a lower bound on the containment and the equality problems. In the Boolean setting, the complexities for the three problems coincide, and are PSPACE-complete [32]. As we shall see in this paper, in the weighted setting the picture is more involved, and depends on the domain of the weights in the WFA.

3 Simulation of Counter Machines with $\mathbb{Z}$-WFAs

In this section we present our main technical construction. We show how we can simulate, in a sense, a two-counter machine using a WFA. We then apply this construction in order to show the undecidability of various problems.

In fact, our construction requires considering only complete automata where all weights are in $\{-1, 0, 1\}$, initial weights are in $\{0, \infty\}$, final weights are 0, and all states are accepting, i.e., $\{q : q \in \mathtt{fin}\} = Q$.

We note that an extension of the construction below is described, in the algebraic approach, in [16], Theorem 8.6.

3.1 From Counter Machines to $\mathbb{Z}$-WFAs

Our construction uses ideas similar to those presented in [13]. A two-counter machine $\mathcal{M}$ is a sequence $(l_1, \ldots, l_n)$ of commands involving two counters x and y. We refer to $\{1, \ldots, n\}$ as the *locations* of the machine. There are five possible forms of commands:

$$\textsc{inc}(c),\ \textsc{dec}(c),\ \textsc{goto}\ l_i,\ \textsc{if}\ c{=}0\ \textsc{goto}\ l_i\ \textsc{else goto}\ l_j,\ \textsc{halt},$$

where $c \in \{x, y\}$ is a counter and $1 \leq i, j \leq n$ are locations. The counters are initially set to 0. Since we can always check whether $c = 0$ before a $\textsc{dec}(c)$ command, we assume that the machine never reaches $\textsc{dec}(c)$ with $c = 0$. That is, the counters never have negative values. Given a counter machine $\mathcal{M}$, deciding whether $\mathcal{M}$ halts is known to be undecidable [33]. Given $\mathcal{M}$, deciding whether $\mathcal{M}$ halts with both counters having value 0 is also undecidable. Indeed, given a counter machine $\mathcal{M}$, we can replace every HALT command with code that clears the counters before halting. Thus, the halting problem can be reduced to the latter problem, termed the 0-*halting problem.*

Given a two-counter machine $\mathcal{M}$, we are going to construct a $\mathbb{Z}$-WFA that can be used to characterize whether $\mathcal{M}$ 0-halts.

Theorem 3.1 *Given a two-counter machine $\mathcal{M}$, we can compute a $\mathbb{Z}$-WFA $\mathcal{A}$ with the following properties:*

1. *$\mathcal{A}(w) \leq 1$ for every word w.*
2. *$\mathcal{M}$ 0-halts iff there exists a word w such that $\mathcal{A}(w) = 1$.*
3. *$\mathcal{A}$ is complete, all its initial weights are in $\{0, \infty\}$, all final weights are 0 (and there are no ∞ final weights), and all weights in Δ are in $\{-1, 0, 1\}$.*

Proof: Let $\mathcal{M}$ be a two-counter machine with commands $(l_1, \ldots, l_n)$. A *run* of a two-counter machine with commands from the set $L = \{l_1, \ldots, l_n\}$ is a sequence $\rho = \rho_1, \ldots, \rho_m \in (L \times \mathbb{N} \times \mathbb{N})^*$ such that the following hold.

1. $\rho_1 = \langle l_1, 0, 0\rangle$.
2. For all $1 < i \leq m$, let $\rho_{i-1} = (l_k, \alpha, \beta)$ and $\rho_i = (l', \alpha', \beta')$. Then, the following hold.
   - If l_k is a INC(x) command (resp. INC(y)), then $\alpha' = \alpha + 1$, $\beta' = \beta$ (resp. $\beta = \beta + 1$, $\alpha' = \alpha$), and $l' = l_{k+1}$.
   - If l_k is a DEC(x) command (resp. DEC(y)), then $\alpha' = \alpha - 1$, $\beta' = \beta$ (resp. $\beta = \beta - 1$, $\alpha' = \alpha$), and $l' = l_{k+1}$.
   - If l_k is a GOTO l_s command, then $\alpha' = \alpha$, $\beta' = \beta$, and $l' = l_s$.
   - If l_k is an IF x=0 GOTO l_s ELSE GOTO l_t command, then $\alpha' = \alpha$, $\beta' = \beta$, and $l' = l_s$ if $\alpha = 0$, and $l' = l_t$ otherwise.
   - If l_k is a IF y=0 GOTO l_s ELSE GOTO l_t command, then $\alpha' = \alpha, \beta' = \beta$, and $l' = l_s$ if $\beta = 0$, and $l' = l_t$ otherwise.
   - If l' is a HALT command, then $i = m$. That is, a run does not continue after HALT.

If, in addition, we have that $\rho_m = \langle l_k, \alpha, \beta\rangle$ such that l_k is a HALT command, we say that ρ is a *halting run.*

Observe that the machine $\mathcal{M}$ is deterministic. We say that a machine $\mathcal{M}$ 0-halts if its run is halting and ends in $\langle l, 0, 0\rangle$.

We say that a sequence of commands $\tau \in L^*$ *fits* a run ρ, if τ is the projection of ρ on its first component.

The *command trace* $\pi = \pi_1, \ldots, \pi_m$ of a run $\rho = \rho_1, \ldots, \rho_m$ is defined as follows. For every $1 \leq i \leq m$, if the command taken in ρ_i is not of the form IF c=0 GOTO l_k ELSE GOTO $l_{k'}$, then $\pi_i = l_i$. Otherwise, $\pi_i =$ GOTO l_s, where s is the location of the command in ρ_{i+1}.

We start by explaining the intuition behind the construction. We construct a WFA $\mathcal{A}$ such that $\mathcal{M}$ 0-halts iff there exists $w \in \Sigma^*$ such that $\mathcal{A}(w) = 1$. The alphabet of $\mathcal{A}$ consists of the following $n + 5$ letters:

$$\Sigma = \{\textsc{inc}(x), \textsc{dec}(x), \textsc{inc}(y), \textsc{dec}(y), \textsc{halt}\} \cup \{\textsc{goto} \quad l_i : i \in \{1, \ldots, n\}\}.$$

When $\mathcal{A}$ reads a sequence of commands w, it tries to simulate the run of $\mathcal{M}$ that induces the command trace w. If the sequence of commands fits the actual run, and this run 0-halts, then all the runs of $\mathcal{A}$ have cost exactly 1. Thus, the word w is such that $\mathcal{A}(w) = 1$. If, however, the sequence of commands does not fit the actual run, then the violation is detected and $\mathcal{A}$ has a run on w with non-positive cost.

We now construct the WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \texttt{init}, \texttt{fin} \rangle$. In Section 3.2 we describe an example of the reduction.

We set $\texttt{fin}(q) = 0$ for every $q \in Q$. That is, there are no final costs.

We designate a state q_{freeze} such that for all $\sigma \in \Sigma$, there is the transition $\Delta(q_{\mathsf{freeze}}, \sigma, q_{\mathsf{freeze}}) = 0$. There is also a state q_{halt} with the transition $\Delta(q_{\mathsf{halt}}, \sigma, q_{\mathsf{freeze}}) = -1$ for all $\sigma \in \Sigma$ (see Figure 1).

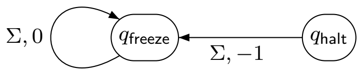

Figure 1: q_{freeze} and q_{halt}.

In order to define $\mathcal{A}$, we first define a "skeleton" ComCheck, which is an underspecified WFA. We then compose $\mathcal{A}$ from variants of ComCheck.

The skeleton ComCheck consists of states $q_1, \ldots, q_n$ that correspond to the commands $l_1, \ldots, l_n$. For two locations i and j, there is a transition from q_i to q_j iff l_j can *locally follow* l_i in a run of $\mathcal{M}$. That is, either $j = i + 1$ and l_i is an $\textsc{inc}$ or $\textsc{dec}$ command, l_i is a $\textsc{goto}$ l_j command, or l_i is an $\textsc{if}$ $c{=}0$ $\textsc{goto}$ l_k $\textsc{else goto}$ l'_k command, with $j \in \{k, k'\}$. The letter labeling the transition from q_i to q_j corresponds to the command trace. That is, the letter is l_i, except the case l_i is an $\textsc{if}$ $c{=}0$ $\textsc{goto}$ l_k $\textsc{else goto}$ l'_k command with $j \in \{k, k'\}$, in which case the letter is $\textsc{goto}$ l_j. The weights on the transitions, as well as additional transitions, are specified below in every variant of ComCheck.

The WFA $\mathcal{A}$ is composed of 5 gadgets, each responsible for checking a certain type of violation in the description of a 0-halting run of $\mathcal{M}$. The gadgets are obtained from ComCheck as described below.

Command Checker. The first gadget we construct is the *command checker*. This gadget checks for local violations of successive commands. That is, it makes sure that the letter w_i represents a command that can follow the command represented by w_{i-1} in $\mathcal{M}$. For example, if the command in location l_2 is $\textsc{inc}(x)$, then from state q_2, which is associated with l_2, we move with the letter $\textsc{inc}(x)$ to q_3, which is associated with l_3. The test is local, as this gadget does not check for violations involving illegal jumps due to the values of the counters. The command checker consists of a ComCheck in which all the weights are 0. In addition, we add transitions labeled by HALT from every state q_i such that $l_i = \text{HALT}$ to q_{halt}. These transitions have cost 1. Every other transition that is not specified in ComCheck leads to q_{freeze} with weight 0. For example, reading a command that does not correspond to l_i in q_i leads to q_{freeze} with weight 0. Note that indeed, if a word represents the command trace of a halting run, it ends with a HALT letter from a state q_i such that $l_i = \text{HALT}$. Thus, the last transition has weight 1. Otherwise, the run of the command checker on w ends with a 0 weight transition.

Positive Jump Checker. The second gadget we need is the *positive jump checker*, which is defined for each counter $c \in \{x, y\}$. This gadget checks for violations in conditional jumps. In every IF c=0 GOTO l_j ELSE GOTO l_k command, it makes sure that if the jump GOTO l_k is taken, then the value of c is indeed greater than 0.

This gadget is a variant of ComCheck in which the weights are defined as follows. Every transition that is taken upon reading INC(c) has weight 1, and every transition that is taken upon reading DEC(c) has weight -1. In every state q_i such that l_i = IF c=0 GOTO l_j ELSE GOTO l_k, we add a transition $\langle q_i, \text{GOTO } l_k, q_{\mathsf{freeze}}\rangle$ with weight -1. We add an initial state q_0 (i.e. we set $\texttt{init}(q_0) = 0$) that, intuitively, has an ϵ transition with weight 1 to q_1 in ComCheck. Since we do not allow ϵ transitions, we remove the transition by connecting q_0 to the appropriate descendants of q_1. All the other transitions induced by ComCheck have weight 0. In addition, for every state q in ComCheck we add a transition $\langle q, \text{HALT}, q_{\mathsf{freeze}}\rangle$ with weight 0 (See Figure 2).

The intuition behind this gadget is as follows. Along the run, the cost of the run reflects the value of the counter c plus 1. Whenever a conditional jump is taken, $\mathcal{A}$ nondeterministically moves to q_{freeze}, accumulating a weight of -1. If the jump is legal, then the value of the counter is at least 1, so the cost of the run so far is at least $1 + 1 = 2$. Thus, the nondeterministic run that follows this route has weight at least 1 when it reaches q_{freeze}. Otherwise, the value of the counter is 0, so the cost of the run is 1, and the nondeterministic move to q_{freeze} induces a run with cost 0, thus "detecting" the violation.

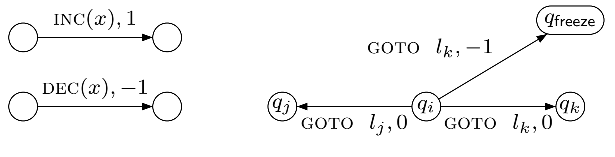

Figure 2: Positive Jump Checker for x, where l_i : IF x=0 GOTO l_j ELSE GOTO l_k.

Zero Jump Checker. Dually to the positive jump checker, we define the gadget *zero jump checker* for each counter $c \in \{x, y\}$.

This gadget checks for the dual violations in conditional jumps. Thus, in every command of the form IF c=0 GOTO l_j ELSE GOTO l_k, it makes sure that if the jump GOTO l_j is taken, then the value of c is indeed 0.

This gadget is a variant of ComCheck in which the weights are as follows. Every transition that is taken upon reading INC(c) has weight -1, and every transition that is taken upon reading DEC(c) has weight 1. In every state q_i such that l_i = IF c=0 GOTO l_j ELSE GOTO l_k, we add a transition $\langle q_i, \text{GOTO } l_j, q_{\mathsf{freeze}}\rangle$ with weight 0. We add an initial state q_0 exactly as in the positive jump checker. All the other transitions in ComCheck have weight 0. In addition, for every state q in ComCheck we have a transition $\langle q, \text{HALT}, q_{\mathsf{freeze}}\rangle$ with weight 0 (See Figure 3).

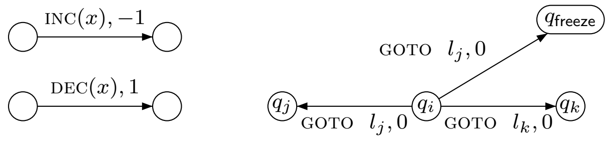

Figure 3: Zero Jump Checker for x, where l_i : IF x=0 GOTO l_j ELSE GOTO l_k.

To complete the definition of the automaton, we define `init` to give weight 0 to the states corresponding to l_1 in the command checker gadget and the q_0 states defined for the jump checkers for each counter $c \in \{x, y\}$.

It is easy to check that Property 3 in the theorem statement holds. That is, $\mathcal{A}$ is complete, all its initial weights are in $\{0, \infty\}$, all final weights are 0, and all weights in Δ are in $\{-1, 0, 1\}$.

It remains to prove that the following properties hold:

1. $\mathcal{A}(w) \leq 1$ for every $w \in \Sigma^*$.
2. $\mathcal{M}$ 0-halts iff there exists $w \in \Sigma^*$ such that $\mathcal{A}(w) \geq 1$.

Property 1 is easy to see: every run in the command checker accumulates weight 0 until seeing HALT. Then, it either accumulates weight 1, and reaches q_{halt}, or immediately reaches q_{freeze} with weight -1 (which does not change further). From q_{halt}, it may still reach q_{freeze}, but that makes it cost 0 again. It follows that the maximal possible weight for any word is 1.

We now turn to prove Property 2. Observe that the runs of $\mathcal{A}$ consist of all the runs in the underlying gadgets. Thus, it is enough to prove that $\mathcal{M}$ 0-halts iff there exists $w \in \Sigma^*$ such that all the runs of all the gadgets of $\mathcal{A}$ on w have cost of at least 1.

We start with the easier direction. Assume that $\mathcal{M}$ 0-halts. Let ρ be the 0-halting run of $\mathcal{M}$, and let w be the command trace of ρ. We prove that w is assigned a cost of at least 1 in all gadgets. Consider first the command checker. Since $\mathcal{M}$ 0-halts, all the transitions are "legal" in $\mathcal{M}$. Also, ComCheck is deterministic. Accordingly, all the commands except for the final HALT command accumulate cost 0, and the final HALT command moves to q_{halt} with weight 1. It follows that the command checker contributes a single run with weight 1.

Next, consider the positive jump checker for counter c. The ComCheck part of this gadget acts the same as the command checker in the sense that the transitions never reach q_{freeze} or q_{halt} until the HALT command. However, by the definition of the gadget, the accumulated cost of the single run on a prefix $w[1,\dots,k]$ is $val_k(c)+1$, where $val_k(c)$ is the value of the counter c after the commands $w_1,\dots,w_k$ have been executed. (The $+1$ is from the transition from the initial state q_0). Since w is the trace of a legal run, then when a line l_k of the form IF c=0 GOTO l_i ELSE GOTO l_j is encountered in the run of $\mathcal{M}$, the corresponding GOTO command in w is legal for $val_k(c)$. According to the definition of the transitions, if $val(c)=0$, and hence the next letter in the input is GOTO l_i, then the run continues with GOTO l_i in the ComCheck component. Otherwise, the next letter in the input is GOTO l_j, and the nondeterministic choice on GOTO l_j enables the run to also continue to q_{freeze} with weight -1. Since we assume that the value of the counters is never negative, the fact that $val_k(c)\neq 0$ implies that $val(c)+1\geq 2$. Thus, the cost accumulated in the run that goes to q_{freeze} is at least $2+(-1)=1$. Finally, the runs that remain in the ComCheck component go, upon reading HALT, to q_{freeze}, with whatever cost they have. Since this is a 0-halting run, this cost is $0+1\geq 1$.

Dually, consider the zero jump checker for counter c. Here, the accumulated cost in ComCheck is $-val(c)+1$. If a jump is to l_j in a location where the command is of the form IF c=0 GOTO l_i ELSE GOTO l_j, the run continues in ComCheck. If the run takes the l_i jump, then $val(c)=-val(c)=0$, so the nondeterministic choice to q_{freeze} induces a run with weight $0+1\geq 1$. As for the runs that remain in ComCheck, upon reading halt their accumulated cost is 0, and as in the positive jump checker, they move to q_{freeze} with weight 1.

Thus, all the runs have cost of at least 1, so $\mathcal{A}(w)\geq 1$.

We proceed to prove the harder direction. Assume there exists $w\in\Sigma^*$ such that $\mathcal{A}(w)\geq 1$. We claim that w is the command trace that is induced by a 0-halting run of $\mathcal{M}$.

Consider the run of the command checker on w. Assume by way of contradiction that the sequence of commands described by w is not the command trace of the run of $\mathcal{M}$. Thus, there is some violation in the run induced by the word w. Let k be the minimal index in w where a violation occurs.

If the violation is not in a conditional jump, it must be that the successive command of w does not fit the current location of $\mathcal{M}$. In this case, the command checker goes to q_{freeze} with accumulated cost 0. Thus, there exists at least one run of $\mathcal{A}$ on w with accumulated cost 0, so $\mathcal{A}(w)\leq 0$, contradicting the assumption that $\mathcal{A}(w)\geq 1$. Furthermore, if w does comply with $\mathcal{M}$ along the entire run, but does not halt, then the run stays in ComCheck, with cost 0. If w has a halt command, but then has additional letters, the command checker reaches q_{halt} with cost 1, and then moves to q_{freeze} with accumulated cost of $1-1=0$. From this we get that if $\mathcal{A}(w)\geq 1$, then w is the command trace of the run of $\mathcal{M}$, up to violations in conditional jumps or a violation of halting with non-zero counters.

Consider again the first violation (which occurs in w_k). By the above argument, either w takes a conditional jump with the wrong jumping condition, or w halts with a (strictly) positive counter.

Assume that w takes an illegal conditional jump as the first violation. Let the corresponding command in $\mathcal{M}$ be l_i = IF c=0 GOTO l_j ELSE GOTO l_k be. Two scenarios are possible. Either w_k is GOTO l_j and $val_k(c) > 0$, or w_k is GOTO l_k and $val_k(c) = 0$. In the former case, consider the runs of the zero jump checker on w. The cost accumulated by the run that stays in the ComCheck component is $-val_k(c)+1 \leq 0$. Thus, reading GOTO l_j, the run moves to q_{freeze} with accumulated cost of at most 0, and the run stays there with the same accumulated cost contradicting the assumption that $\mathcal{A}(w) \geq 1$. In the latter case, consider the runs of the positive jump checker. The value of the run that stays in the ComCheck component at the corresponding letter is $val_k(c)+1 = 1$. Thus, reading GOTO l_j moves to q_{freeze} with accumulated weight of $1 - 1 = 0$, which is again a contradiction.

Finally, assume that w halts with a strictly positive counter c. Consider again the run of the zero jump checker (for c) on w. The cost accumulated by the run that stays in the ComCheck component is $-val_k(c) + 1$. If $val_k(c) \geq 1$ then this run costs at most 0. Upon reading HALT the run moves to q_{freeze} with accumulated cost 0, contradicting the assumption that $\mathcal{A}(w) \geq 1$.

We conclude that w is the actual trace of the 0-halting run of $\mathcal{M}$. Therefore, $\mathcal{M}$ 0-halts, and we are done. □

Remark 3.2 Observe that the alphabet of the WFA constructed in the proof of Theorem 3.1 is dependent on the number of commands in the machine. However, we can easily modify the construction to work with a binary alphabet $\{a, b\}$. Indeed, one only needs to encode each letter in the original construction by a string $a^i b$ for distinct i. Then, we modify the transition function as follows: for each transition of the form (q, σ, r) with weight W for states q, r and letter σ, if σ is encoded as $a^i b$, we introduce states $q^1, \ldots, q^i$ with transitions (q, a, q^1) and (q^j, a, q^{j+1}) for every $1 \leq j < i$ with weight 0. Finally, we have a transition (q^i, b, r) with weight W.

It is not hard to verify that the behavior of the modified automaton maintains the desired properties of the reductions. Thus, undecidability of universality holds already for automata over a binary alphabet.

3.2 An Example

In this section we describe an example of the reduction presented in Theorem 3.1. Consider the following two-counter machine $\mathcal{M}$.

l_1 : IF x=0 GOTO l_5 ELSE GOTO l_2
l_2 : DEC(x)
l_3 : INC(y)
l_4 : GOTO l_1
l_5 : HALT

The single run of $\mathcal{M}$ is $\langle l_1, 0, 0\rangle, \langle l_5, 0, 0\rangle$, and its command trace is GOTO l_5, HALT. Note, however, that $\mathcal{M}$ contains many "potential violations", which would make it an interesting machine to consider. Figure 4 describes the command checker for $\mathcal{M}$. The gray arrows are transitions that are taken on every letter that is unspecified in the gadget, all with cost 0.

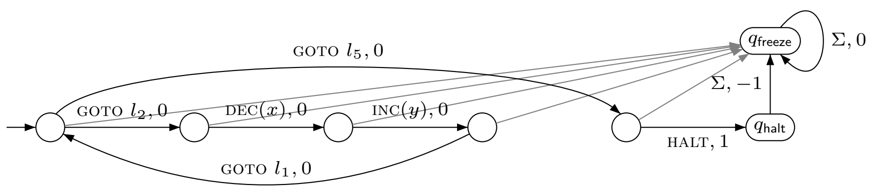

Figure 4: Example Command Checker.

Note that the cost of GOTO l_5, HALT is 1. Note also that some words that do not fit the actual run of $\mathcal{M}$ also have cost 1, for example GOTO l_2, DEC(x), INC(y), GOTO l_1, GOTO l_5, HALT, . This is, however, not a problem, as the command checker does not attempt to detect violations that have to do with conditional jumps in which a wrong jump has been taken – such violations are going to be detected by the jump checkers. On the other hand, the command checker assigns a cost of 0 to words like GOTO l_2, INC(y), which do not follow $\mathcal{M}$, or to words like GOTO l_5 or GOTO l_5, HALT, GOTO l_1, which are too short or too long.

Figure 5 describes the positive jump checker for x. For clarity, we use an ϵ-transition. Formally, this transition is removed by replacing it with two edges with cost 1 to the states reachable from q_1.

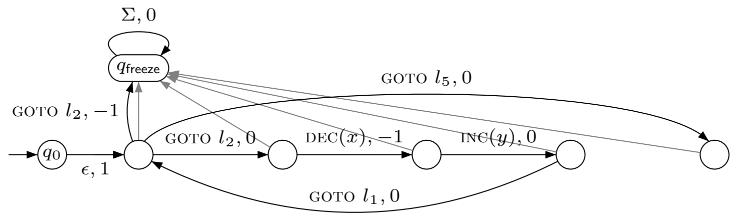

Figure 5: Example Positive Jump Checker for x.

Recall the trace GOTO l_2, DEC(x), INC(y), GOTO l_1, GOTO l_5, HALT, which is illegal in $\mathcal{M}$, but went undetected in the command checker. In the positive jump checker, this trace has a run with cost 0. Indeed, reading the GOTO l_2 command, the gadget has accumulated cost 1 and proceeds to q_{freeze}, with accumulated cost $1 + (-1) = 0$. Thus, the violation is detected.

Finally, Figure 6 describes the zero jump checker for x. The jump checkers for y are similar and are therefore not depicted here. In fact, in our case of $\mathcal{M}$,

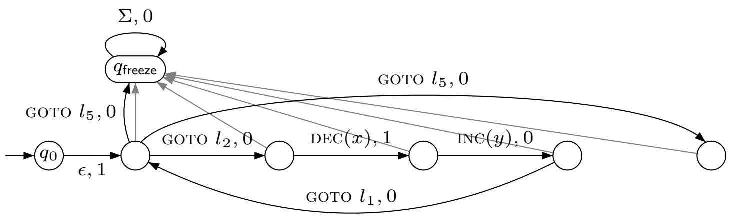

Figure 6: Example Zero Jump Checker for x.

no word that survives the command checker may violated the conditions that the zero jump checker attempts to detect. Indeed, the only possible violation that is not detected by the command checker is a wrong jump in l_1, which is a positive (rather than zero) jump, or a halt with the counters not being cleared, which again cannot happen in words that survive the other checks. Still, it is important to observe that the zero jump checker does not detect false violations. For example, the word GOTO l_5, HALT has two runs in this gadget, both with cost 1.

4 Undecidability Results

Theorem 3.1 enables us to show undecidability of several decision problems for WFAs. We remark that several of these problems (namely universality, containment, and equality) were already shown to be undecidable in [27]. Nonetheless, our proofs here contribute to two aspects. First, all the reductions are based on our construction in Theorem 3.1. This circumvents the algebraic approach taken in [27], which is often inaccessible to the automata community. Second, it allows us to carefully analyze properties of the models for which the undecidability results apply. Indeed, in Section 6 we complement the undecidability results with decidability results for the remaining models.

We start by showing the undecidability result of the universality problem.

Theorem 4.1 *The universality problem for nondeterministic* $\mathbb{Z}$-WFAs *is undecidable, even for complete* WFAs *with weights in* $\{-1, 0, 1\}$ *and with threshold* 1.

Proof: We show a reduction from the 0-halting problem of two-counter machines to the complement of the universality problem, namely the problem of deciding, given a $\mathbb{Z}$-WFA $\mathcal{A}$ and a value ϑ, whether there exists a word w such that $\mathcal{A}(w) \geq \vartheta$. In particular, we use the threshold $\vartheta = 1$.

The reduction proceeds as follows: given a two-counter machine $\mathcal{M}$, we apply the construction in Theorem 3.1 to obtain a $\mathbb{Z}$-WFA $\mathcal{A}$, and the output of the reduction is $\mathcal{A}$ and the threshold 1. By Theorem 3.1, $\mathcal{M}$ 0-halts iff there exists a word w for which $\mathcal{A}(w) \geq 1$, and we are done.

By the construction in Theorem 3.1, we have that $\mathcal{A}$ is complete, and the weights in $\mathcal{A}$ are from $\{-1, 0, 1\}$.

Since the set of decidable languages is closed under complementation, undecidability applies also to the universality problem. $\square$

Recall that the containment problem is to decide, given two WFAs $\mathcal{A}$ and $\mathcal{B}$, whether $\mathcal{A}(w) \geq \mathcal{B}(w)$ for every word w. As in the Boolean case, the complexity (or decidability) of this problem depends on whether $\mathcal{A}$ and/or $\mathcal{B}$ are deterministic or nondeterministic. We thus refer to four cases of the problem, namely $(D, D), (D, N), (N, D)$, and (N, N), where the first and second components indicate whether $\mathcal{A}$ and $\mathcal{B}$, respectively, are deterministic (D) or nondeterministic (N). For example, the (D, N) containment problem gets as input a deterministic WFA $\mathcal{A}$ and a nondeterministic WFA $\mathcal{B}$.

Theorem 4.2 *The* (D, N) *and* (N, N) *containment problems are undecidable for* $\mathbb{N}$-WFAs *(and in particular for* $\mathbb{Z}$-WFAs*).*

Proof: From Theorem 4.1 we have that it is undecidable, given a $\mathbb{Z}$-WFA $\mathcal{A}$ with weights in $\{-1, 0, 1\}$, whether $0 \geq \mathcal{A}(w)$ for every word w. Since the function that assigns to every word the value 0 can be realized by a deterministic $\mathbb{N}$-WFA $\mathcal{Z}ero$, this implies that the (D, N) and (N, N) containment problems for $\mathbb{Z}$-WFAs are undecidable.

In order to lift this to $\mathbb{N}$-WFAs, we make the following observation. Consider a WFA $\mathcal{A}$, and let $\mathcal{A}^{+1}$ be the WFA obtained from $\mathcal{A}$ by adding 1 to all the weights, then for every word w we have that $\mathcal{A}^{+1}(w) = \mathcal{A}(w) + |w|$.

Applying this to the $\mathbb{Z}$-WFA $\mathcal{A}$ above, as well as to the $\mathbb{N}$-WFA $\mathcal{Z}ero$, we have that $0 \geq \mathcal{A}(w)$ for every word w iff $\mathcal{Z}ero^{+1}(w) \geq \mathcal{A}^{+1}(w)$ for every word w. Moreover, observe that $\mathcal{A}^{+1}$ is a WFA whose weights are all in $\{0, 1, 2\}$, and in particular, is an $\mathbb{N}$-WFA. We conclude that the (D, N) and (N, N) containment problems for $\mathbb{N}$-WFAs are undecidable. $\square$

We now turn to study the equality problem – given two WFAs $\mathcal{A}$ and $\mathcal{B}$, whether $\mathcal{A}(w) = \mathcal{B}(w)$ for every word w. As in the containment problem, we refer to the four cases.

Theorem 4.3 *The* (N, N) *equality problem is undecidable for* $\mathbb{N}$-WFAs *(and in particular for* $\mathbb{Z}$-WFAs*).*

Proof: We show a reduction from the (N, N) containment problem for $\mathbb{N}$-WFAs, which is undecidable by Theorem 4.2.

The reduction proceeds as follows. Given two $\mathbb{N}$-WFAs $\mathcal{A}$ and $\mathcal{B}$, we construct an $\mathbb{N}$-WFA $\mathcal{C}$ which is obtained by taking the union of $\mathcal{A}$ and $\mathcal{B}$. That is, $\mathcal{C}$ reads a word by nondeterministically choosing between $\mathcal{A}$ and $\mathcal{B}$. It follows that for every word w, $\mathcal{C}(w) = \min\{\mathcal{A}(w), \mathcal{B}(w)\}$. The reduction then outputs $\mathcal{C}$ and $\mathcal{B}$.

Observe that for every word w, $\mathcal{C}(w) = \mathcal{B}(w)$ iff $\mathcal{A}(w) \geq \mathcal{B}(w)$, which concludes the correctness of the reduction. $\square$

Remark 4.4 We note that Theorem 8.6 in [16] gives an alternative proof (reminiscent to that in Theorem 3.1) that the (N, N) containment and equality problems for $\mathbb{N}$-WFAs are undecidable, which also follows from [27]. Furthermore, it is shown in [16] that these problems remain undecidable even if one of the automata is a (specific) fixed WFA. This subsumes our result in Theorem 4.3. The result in Theorem 4.2 shows that the containment problem remains undecidable even if the "left" WFA is fixed, and that moreover – this fixed WFA can be taken to be a very simple and deterministic WFA.

We proceed to the $\exists$-exact problem. The following is obtained from Theorem 3.1 in the same manner as Theorem 4.1 is obtained from it.

Theorem 4.5 *The $\exists$-exact problem for nondeterministic* $\mathbb{Z}$-WFAs *is undecidable, even for complete* WFAs *with weights in* $\{-1, 0, 1\}$ *and with threshold* 1*.*

Finally, we study the upper-boundedness problem for nondeteministic $\mathbb{Z}$-WFAs. This problem was shown to be undecidable in [1]. We bring an alternative proof here.

Theorem 4.6 *The upper-boundedness problem is undecidable for nondeterministic* $\mathbb{Z}$-WFAs*.*

Proof: We show a reduction from the 0-halting problem of two-counter machines to the complement of the upper-boundedness problem, namely the problem of deciding, given a WFA $\mathcal{A}$, whether for every $\vartheta \in \mathbb{N}$ there exists a word w such that $\mathcal{A}(w) \geq \vartheta$.

The reduction proceeds as follows: given a two-counter machine $\mathcal{M}$, we apply the construction in Theorem 3.1 to obtain a $\mathbb{Z}$-WFA $\mathcal{A}$. We then modify $\mathcal{A}$ as follows. Intuitively, we allow $\mathcal{A}$ to reset from q_{halt} back to the initial states. Thus, if a word gets value (at least) 1, it can be repeated and its weight accumulated.

Formally, this is done as follows. We introduce a new letter reset and a new state $q_\perp$, with the following transitions: $\Delta(q_{\mathsf{freeze}}, \mathsf{reset}, q_0) = \Delta(q_{\mathsf{halt}}, \mathsf{reset}, q_0) = 0$ for every state $q_0 \in \mathtt{init}$, $\Delta(q, \mathsf{reset}, q_\perp) = 0$ for every state $q \notin \{q_{\mathsf{freeze}}, q_{\mathsf{halt}}\}$, and $\Delta(q_\perp, \sigma, q_\perp) = -1$ for every letter σ. We refer to the new WFA as $\mathcal{B}$.

We claim that $\mathcal{B}$ is not upper bounded iff there exists a word w with $\mathcal{A}(w) \geq 1$. Observe that for every word w, if $\mathcal{A}(w) = 1$, then all the runs of $\mathcal{A}$ on w end up in either q_{halt} or q_{freeze}. Consider the word $u_n = (w \cdot \mathsf{reset})^{n-1} w$ for some $n \in \mathbb{N}$. It follows that $\mathcal{B}(u_n) = n$, so $\mathcal{B}$ is not upper bounded. Conversely, if $\mathcal{A}(w) \leq 0$ for every word w, it is easy to see that any application of reset cannot result in a word with positive weight, so $\mathcal{B}$ is upper bounded, which concludes the proof. $\square$

Remark 4.7 The construction in the proof of Theorem 4.6 is a variant of well-known constructions, which are analogous to the Kleene-star operator on regular languages [16, 15]. These constructions, which are typically presented in the algebraic view of weighted automata, namely with rational series, can be used to prove Theorem 4.6 by directly reducing from universality of WFAs with threshold 1, which is undecidable by Theorem 4.1.

5 A Toolbox for WFAs

Having established undecidability results in Section 4, the remainder of the paper is dedicated to establishing decidability results, with corresponding complexity bounds. Before approaching the problems at hand, we develop several general tools.

5.1 The Product Construction – Sum and Difference of WFAs

In the Boolean setting, given two NFAs $\mathcal{A}$ and $\mathcal{B}$, we can use the standard product construction to obtain an NFA whose language is $L(\mathcal{A}) \cap L(\mathcal{B})$. In the tropical setting, intersection corresponds to summation. Fortunately, it is well known [17, 15] that the construction can be adapted easily to WFAs. For completeness, we detail the construction below, as we use it later.

Consider WFAs $\mathcal{A}_i = \langle \Sigma, Q_i, \Delta_i, \mathtt{init}_i, \mathtt{fin}_i \rangle$, for $i \in \{1, 2\}$. We construct their *sum* WFA $\mathcal{B} = \langle \Sigma, Q', \Delta', \mathtt{init}', \mathtt{fin}' \rangle$ as follows. The set of states is $Q' = Q_1 \times Q_2$. For states $(q_1, q_2), (s_1, s_2) \in Q'$ and for $\sigma \in \Sigma$, we define $\Delta'((q_1, q_2), \sigma, (s_1, s_2)) = \Delta_1(q_1, \sigma, s_1) + \Delta_2(q_2, \sigma, s_2)$. Similarly, for every state $(q_1, q_2) \in Q'$ we set $\mathtt{init}'(q_1, q_2) = \mathtt{init}_1(q_1) + \mathtt{init}_2(q_2)$ and $\mathtt{fin}'(q_1, q_2) = \mathtt{fin}_1(q_1) + \mathtt{fin}_2(q_2)$.

The following proposition is easy to prove by induction.

Proposition 5.1 *In the notation above, for every word $w \in \Sigma^*$, we have that $\mathcal{B}(w) = \mathcal{A}_1(w) + \mathcal{A}_2(w)$.*

In the Boolean setting, the product construction is often used to reason about containment and equality of automata. Indeed, we have that e.g., $L(\mathcal{A}) \subseteq L(\mathcal{B})$ iff $L(\mathcal{A}) \cap \overline{L(\mathcal{B})} = \emptyset$. Crucially, however, this requires the ability to complement automata.

In the weighted setting, complementation corresponds to taking the negation of a WFA. Unfortunately, WFAs are, in general, not closed under negation. [9, 2] Deterministic WFAs, however, can be negated, in the following sense.

Consider a deterministic $\mathbb{Z}$-WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin} \rangle$, and let $\overline{\mathcal{A}}$ be the WFA obtained from $\mathcal{A}$ by negating all non-infinity weights. Then, we have the following.

Proposition 5.2 *For every $w \in \Sigma^*$, we have that*

$$\overline{\mathcal{A}}(w) = \begin{cases} -\mathcal{A}(w) & \text{if } w \in \mathsf{Dom}(\mathcal{A}), \\ \infty & \text{if } w \notin \mathsf{Dom}(\mathcal{A}). \end{cases}$$

Combining Propositions 5.1 and 5.2, we conclude with the following.

Proposition 5.3 *Consider a* WFA $\mathcal{A}$ *and a deterministic* WFA $\mathcal{B}$*, and let $\mathcal{C}$ be the* WFA *obtained by taking the sum* WFA *of $\mathcal{A}$ and $\overline{\mathcal{B}}$. Then for every word*

$w \in \Sigma^*$, we have that

$$\mathcal{C}(w) = \begin{cases} \mathcal{A}(w) - \mathcal{B}(w) & \textit{if } w \in \mathsf{Dom}(\mathcal{B}), \\ \infty & \textit{if } w \notin \mathsf{Dom}(\mathcal{B}). \end{cases}$$

5.2 Weighted Subset Construction

In the Boolean setting, nondeterministic finite automata (NFAs) can be determinized to equivalent deterministic finite automata (DFAs) using the *subset construction*, which keeps track of all the reachable states after reading each letter of the word. This construction suffices for determinization, as this set, also known as a *configuration*, contains all the necessary information of the runs of the NFA.

It is well known that WFAs are in general, not determinizable. For example, the function that assigns a word in $\{a,b\}^*$ the minimum between the number of a's and the number of b's can be realized by a nondeterministic WFA, but not by a deterministic WFA [9, 2].

Intuitively, the reason that WFAs cannot be determinized is because their configurations must also account for the weight accumulated along the different runs, resulting in an infinite state space. Nonetheless, the subset-construction approach described above can be used to gain insight into the working of a WFA, as we demonstrate in this section.

Consider a $\mathbb{Z}$-WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin} \rangle$. The *weighted subset construction* of $\mathcal{A}$ is an infinite-state deterministic structure $\textsc{wss}(\mathcal{A}) = \langle \Sigma, \mathcal{E}, \delta \rangle$ defined as follows. $\mathcal{E}$ is a set of states comprising all functions from Q to $\mathbb{Z}_\infty$. That is, $\mathcal{E} = \mathbb{Z}_\infty^Q$. Intuitively, a reachable state $E : Q \to \mathbb{Z}_\infty$ assigns, for each state $q \in Q$, the value of the minimal run of $\mathcal{A}$ that ends in q on the word that has been read so far (ignoring final weights). The transition function $\delta : \mathcal{E} \times \Sigma \to \mathcal{E}$ is deterministic and is defined as follows: for states $E, E' \in \mathcal{E}$ and a letter $\sigma \in \Sigma$, we have that $\delta(E, \sigma) = E'$ iff for every $q' \in Q$ it holds that $E'(q') = \min_{q \in Q} \{E(q) + \Delta(q, \sigma, q')\}$.

Observe that $\mathtt{init}, \mathtt{fin} \in \mathcal{E}$. The following proposition formalizes the way $\textsc{wss}(\mathcal{A})$ captures the behavior of $\mathcal{A}$.

Proposition 5.4 *Consider a word $w \in \Sigma^*$, and let $E' \in \mathcal{E}$ be the state that is reached by $\textsc{wss}(\mathcal{A})$ on its run on w when starting in $\mathtt{init}$. Then, for every state $q \in Q$, the minimal weight of a run of $\mathcal{A}$ on w that ends in q (ignoring final weights) is $E'(q)$. In particular, $\mathcal{A}(w) = \min_{q \in Q} \{E'(q) + \mathtt{fin}(q)\}$.*

In general, the weighted subset construction is infinite. However, in problems where a threshold is available, we can often restrict attention to a finite subset, as we now demonstrate.

Consider an $\mathbb{N}$-WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin} \rangle$ (note that while Proposition 5.4 applies for $\mathbb{Z}$-WFAs, we now restrict attention to $\mathbb{N}$-WFAs), and let $\vartheta \in \mathbb{N}$ be a threshold. Let $\textsc{wss}(\mathcal{A})$ be its weighted subset construction. We define an equivalence relation $\sim_\vartheta$ on $\mathcal{E}$ as follows: for $E, E' \in \mathcal{E}$ we say that $E \sim_\vartheta E'$ iff for every $q \in Q$ the following hold:

1. $E(q) < \vartheta$ iff $E'(q) < \vartheta$, and if $E(q) < \vartheta$ then $E(q) = E'(q)$.
2. $E(q) = \infty$ iff $E'(q) = \infty$.

Intuitively, we ignore any values over ϑ, except ∞. We identify the equivalence class of a function $E : Q \to \mathbb{Z}_\infty$ with the function $E|_\vartheta : Q \to \{0, \dots, \vartheta, \infty\}$, defined by

$$E|_\vartheta(q) = \begin{cases} E(q) & \text{if } E(q) < \vartheta, \\ \infty & \text{if } E(q) = \infty, \\ \vartheta & \text{otherwise.} \end{cases}$$

Thus, there are at most $(\vartheta + 2)^Q$ equivalence classes of $\sim_\vartheta$. We naturally lift the transition function δ to the equivalence classes by setting $\delta(E|_\vartheta, \sigma) = \delta(E, \sigma)|_\vartheta$. Since the weights are non-negative, this is well defined.

We can now obtain a Boolean deterministic automaton (DFA) $\mathcal{B} = \langle \Sigma, S, \delta, s_0, F \rangle$ as follows. The alphabet is Σ, the states are $S = \mathcal{E}/\sim_\vartheta$, i.e., the equivalence classes of $\sim_\vartheta$, the transition function is δ, lifted to the equivalence classes as described above, and the initial state is $s_0 = \texttt{init}|_\vartheta$. We may naturally extend δ to words. We omit the definition of F, as it is irrelevant for the following lemma, whose proof is immediate.

Lemma 5.5 *Using the notations above, consider a word $w \in \Sigma^*$ and let $E|_\vartheta = \delta(\texttt{init}|_\vartheta, w)$. Then, the following hold.*

1. *If $\mathcal{A}(w) < \vartheta$ then $\mathcal{A}(w) = \min_{q \in Q} \{E|_\vartheta(q) + \texttt{fin}(q)\}$.*
2. *If $\infty > \mathcal{A}(w) \geq \vartheta$ then $\infty > \min_{q \in Q} \{E|_\vartheta(q) + \texttt{fin}(q)\} \geq \vartheta$.*
3. *$\mathcal{A}(w) = \infty$ iff $E|_\vartheta(q) + \texttt{fin}(q) = \infty$ for all $q \in Q$.*

Lemma 5.5 shows that we can construct a Boolean automaton that intuitively captures any bounded fragment of a function described by an $\mathbb{N}$-WFA. We refer to $\mathcal{B}$ above as the *ϑ-bounded weighted determinization of $\mathcal{A}$*.

5.3 $\mathbb{Z}$-WFAs Without Negative Cycles

Recall that by the semantics of WFAs, the cost of a word w is the cost of the minimal run of the automaton on w. Therefore, if a $\mathbb{Z}$-WFA $\mathcal{A}$ has a negative cycle (i.e., a cycle in the underlying weighted graph whose sum of weights is negative) then for every $M \in \mathbb{Z}$ there is a word w such that $\mathcal{A}(w) < M$.

In several decision problems, we wish to consider $\mathbb{Z}$-WFAs without such negative cycles. Consider such a WFA $\mathcal{A}$, then there exists some $M \in \mathbb{Z}$ such that $\mathcal{A} \geq M$. Moreover, if $M < 0$, then consider the WFA $\mathcal{A}'$ obtained from $\mathcal{A}$ by, intuitively, adding a transition with cost $-M$ “before” the initial state. Then, $\mathcal{A}' \geq 0$. It is proved in [28] that a $\mathbb{Z}$-WFA whose image is non-negative has an equivalent $\mathbb{N}$-WFA. Unfortunately, the proof described in [28] only states that there exists an equivalent $\mathbb{N}$-WFA of bounded size, and the procedure to find it is by brute-force search, which takes exponential time.

Since $\mathcal{A}'$ has no negative cycles, and moreover - no negative-valued runs, it seems that "morally", we should be able to redistribute the weights to obtain an equivalent $\mathbb{N}$-WFA. As we now show, this is indeed the case. The main tool to achieve this is Johnson's algorithm [23] for reweighing on graphs.

A *weighted graph* is a directed graph $G = \langle V, E\rangle$ equipped with a weight function $\omega : E \to \mathbb{Z}$. The cost of a path $p = v_0, v_1, \ldots, v_k$ is $\omega(p) = \sum_{i=0}^{k-1} \omega(v_i, v_{i+1})$. We say that p is a *negative cycle* in G if $v_k = v_0$ and $\omega(p) < 0$.

Theorem 5.6 (Johnson's Algorithm [23]) *Consider a weighted graph $G = \langle V, E\rangle$ with weight function $\omega : E \to \mathbb{Z}$, such that G has no negative cycles according to ω. We can compute in polynomial time functions $h : V \to \mathbb{Z}$ and $\omega' : E \to \mathbb{N}$ such that for every path $p = v_0, v_1, \ldots, v_k$ in G it holds that $\omega'(p) = \omega(p) + h(v_0) - h(v_k)$.*

Remark 5.7 Theorem 5.6 is stated for graphs. However, for our applications we need to apply it to multi-graphs (namely the underlying graphs of WFAs). Fortunately, the theorem readily applies to multi-graphs as well: given a multi-graph G, we obtain from G a (simple) graph H by keeping, for every ordered pair of vertices (u, v), only the edge of minimal weight from u to v (if there is an edge between them).

Then, applying Theorem 5.6 to H yields a new weight function ω' with the property that $\omega'(u, v) = \omega(u, v) + h(u) - h(v)$. That is, we view u, v as a path in H.

We can now lift ω' to all the edges in the multi graph G, by increasing the weight of every edge from u to v by $h(u) - h(v)$. Since H used the minimal-weight edges in G, we now have that all weights in the re-weighted multigraph are positive.

We can now use Theorem 5.6 to eliminate negative costs from $\mathbb{Z}$-WFAs without negative cycles.

Lemma 5.8 *Given a $\mathbb{Z}$-WFA $\mathcal{A}$ with no negative cycles, we can compute in polynomial time an $\mathbb{N}$-WFA $\mathcal{A}^+$ and $M \in \mathbb{N}$ such that $\mathcal{A}^+(w) = \mathcal{A}(w) + M$ for every w. Moreover, $\mathcal{A}^+$ differs from $\mathcal{A}$ only by the weights.*

Proof: Consider a $\mathbb{Z}$-WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin}\rangle$. We can view $\mathcal{A}$ as a weighted multi-graph with weights prescribed by Δ, and ignoring $\mathtt{init}$, $\mathtt{fin}$, and infinite weights. We obtain from $\mathcal{A}$ a $\mathbb{Z}$-WFA $\mathcal{A}' = \langle \Sigma, Q, \Delta', \mathtt{init}', \mathtt{fin}'\rangle$ by first applying Theorem 5.6 (and Remark 5.7) to the graph, and defining Δ' according to ω'. (An infinite-weight transition of Δ remains the same in Δ'.) Next, for every $q \in Q$ we define $\mathtt{init}'(q) = \mathtt{init}(q) - h(q)$ and $\mathtt{fin}'(q) = \mathtt{fin}(q) + h(q)$, where h is as per Theorem 5.6.

We claim that for every word $w \in \Sigma^*$, we have that $\mathcal{A}'(w) = \mathcal{A}(w)$. Indeed, consider a word $w = \sigma_1, \ldots, \sigma_n$ and a run $r = q_0, q_1, \ldots, q_n$ of $\mathcal{A}$ on w. Then,

$$\begin{aligned}
&\mathsf{cost}_{\mathcal{A}'}(r) = \mathtt{init}'(q_0) + \sum_{i=0}^{n-1} \Delta'(q_i, \sigma_{i+1}, q_{i+1}) + \mathtt{fin}'(q_n) \\
&= \mathtt{init}(q_0) - h(q_0) + \sum_{i=0}^{n-1} \Delta'(q_i, \sigma_{i+1}, q_{i+1}) + \mathtt{fin}(q_n) + h(q_n) \\
&= \mathtt{init}(q_0) - h(q_0) + \sum_{i=0}^{n-1} \Delta(q_i, \sigma_{i+1}, q_{i+1}) + h(q_0) - h(q_n) + \mathtt{fin}(q_n) + h(q_n) \\
&= \mathtt{init}(q_0) + \sum_{i=0}^{n-1} \Delta(q_i, \sigma_{i+1}, q_{i+1}) + \mathtt{fin}(q_n) = \mathsf{cost}_{\mathcal{A}}(r)
\end{aligned}$$

where in the third equality we use the reweighing property of Theorem 5.6.

Since all the runs maintain their values in $\mathcal{A}$ and $\mathcal{A}'$, we conclude our claim.

Finally, note that in $\mathcal{A}'$, the only negative weights occur in $\mathtt{init}'$ and $\mathtt{fin}'$. Let W be the minimal weight in $\mathtt{init}'$ and $\mathtt{fin}'$, then if $W < 0$ we can obtain from $\mathcal{A}'$ the $\mathbb{N}$-WFA $\mathcal{A}^+$ by adding $-W$ to each entry of $\mathtt{init}'$ and of $\mathtt{fin}'$. Let $M = 2W$, then for every word w we have that $\mathcal{A}^+(w) = \mathcal{A}'(w) + M = \mathcal{A}(w) + M$. □

6 Decidability Results

In this section we complement the results of Section 4 by showing decidability, and analyzing the complexity, of all problems in Section 2.3 that were not proven to be undecidable.

6.1 The non-emptiness problem

The min-sum semantics of WFAs means that the value of a word is the value of the minimal run of the automaton on it. In the non-emptiness problem, we try to find a word whose value is below some given threshold. It is thus enough to find a run of the automaton that has a minimal value. Therefore, the labels on the transitions have no real effect, and the problem reduces to finding a shortest path in a weighted graph. Using standard algorithms such as Dijkstra or Bellman-Ford, we can easily decide non-emptiness.

Proposition 6.1 *The non-emptiness problem is decidable in polynomial time for nondeterministic* $\mathbb{Z}$-WFAs.

Proof: Consider a $\mathbb{Z}$-WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{Z}$. Observe that if $\mathbb{Z}$ has a negative cycle, then there exists a word w such that $\mathcal{A}(w) < \vartheta$. Indeed, a run that repeats the negative cycle enough times can reduce the cost arbitrarily, and since all states are live, we can augment such a run to end in a state $q \in \mathtt{fin}$, thus inducing a run with an arbitrarily low value. This, in turn, induces a word with arbitrarily low value. Since finding negative cycles can be done in polynomial time, this concludes the proof in this case.

If $\mathcal{A}$ does not have a negative cycle, then there exists a word w for which $\mathcal{A}(w) < \vartheta$ iff there exists a cycle-free run of $\mathcal{A}$ with cost less than ϑ. We can determine whether such a run exists by finding the shortest paths from the initial states $\{q : q \in \texttt{init}\}$ to the final states $\{q : q \in \texttt{fin}\}$ in the underlying graph of $\mathcal{A}$, and checking whether any of these paths, with the addition of the initial and final weights prescribed in $\mathcal{A}$, has cost less than ϑ. This can be done in polynomial time e.g., by finding all-pairs shortest paths. □

6.2 The universality problem

We split our analysis of the universality problem to deterministic and non-deterministic WFAs.

For a deterministic WFA $\mathcal{A}$, we start by checking whether $\mathsf{Dom}(\mathcal{A}) = \Sigma^*$. This can be done in polynomial time by constructing a DFA for $\mathsf{Dom}(\mathcal{A})$ and checking its universality. If $\mathsf{Dom}(\mathcal{A}) \neq \Sigma^*$, we reject (see Remark 6.4 below). Next, we obtain the WFA $\overline{\mathcal{A}}$ as per Proposition 5.2. Since $\mathsf{Dom}(\mathcal{A}) = \Sigma^*$, we have that $\overline{\mathcal{A}}(w) = -\mathcal{A}(w)$ for every word $w \in \Sigma^*$. Then, $\mathcal{A} < \vartheta$ iff there does not exist a word w such that $\mathcal{A}(w) \geq \vartheta$, which takes place iff $\overline{\mathcal{A}}(w) \leq -\vartheta$ as well as iff $\overline{\mathcal{A}}(w) < -\vartheta + 1$. Thus, we can reduce the universality problem in this case to the complement of the non-emptiness problem, and from Proposition 6.1 we have the following.

Proposition 6.2 *The universality problem is decidable in polynomial time for deterministic* $\mathbb{Z}$-WFAs.

For nondeterministic WFAs, things are more involved. By Theorem 4.1, the problem is undecidable for $\mathbb{Z}$-WFAs, and therefore we restrict attention to $\mathbb{N}$-WFAs.

Theorem 6.3 *The universality problem for* $\mathbb{N}$-WFAs *is PSPACE-complete.*

Proof: We start by showing that the problem is in PSPACE. Consider an $\mathbb{N}$-WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{N}$. By Lemma 5.5, there exists a word w such that $\mathcal{A}(w) \geq \vartheta$ iff there exists a state $E|_\vartheta$ for which $E|_\vartheta(q) + \texttt{fin}(q) \geq \vartheta$ for all $q \in Q$ that is reachable in the ϑ-bounded weighted determinization of $\mathcal{A}$. In particular, such a state is reachable iff it is reachable by a word of length at most $(\vartheta + 2)^{|Q|}$ (which is an upper bound on the number of states in the construction). We conclude that $\mathcal{A} < \vartheta$ iff there does not exist a word w of length at most $(\vartheta + 2)^{|Q|}$ such that $\mathcal{A}(w) \geq \vartheta$.

Checking whether such a word w exists can be easily done in NPSPACE by guessing each letter of w and keeping track of the accumulated weight. Since NPSPACE=PSPACE [38], and since PSPACE=co-PSPACE, we conclude that universality is in PSPACE.

Finally, it is not hard to see that the universality problem is at least as hard as the universality problem for Boolean nondeterministic automata, which is known to be PSPACE-hard [32]. □

Remark 6.4 (Domain-restricted universality) In the universality problem, the requirement that $\mathcal{A} < \vartheta$ implies, in particular, that $\mathsf{Dom}(\mathcal{A}) = \Sigma^*$. This is desirable, since we want the universality in the weighted setting to extend universality in the Boolean setting. However, there are cases where we want to relax this, and only require that $\mathcal{A}(w) < \vartheta$ for all $w \in \mathsf{Dom}(\mathcal{A})$ (e.g., Proposition 6.11 below).

For deterministic WFAs, adapting the solution is easy - we only drop the initial check that $\mathsf{Dom}(\mathcal{A}) = \Sigma^*$. For nondeterministic WFAs, observe that the ϑ-bounded weighted determinization differentiates between runs of value ∞ to runs of finite value. Thus, we can adapt the proof of Theorem 6.3 to search for a state $E|_\vartheta$ for which $\infty > \min_{q \in Q} \{E|_\vartheta(q) + \mathtt{fin}(q)\} \geq \vartheta$. Lemma 5.5 guarantees the correctness of this procedure.

6.3 The ∃-exact problem

Recall that by Theorem 4.5, the ∃-exact problem is undecidable for nondeterministic $\mathbb{Z}$-WFAs. We now turn to study the remaining cases. We start with deterministic WFAs.

We provide below an integer linear programming solution to the ∃-exact problem of deterministic $\mathbb{Z}$-WFAs, which implies that its complexity is in NP. Then, using a reduction from the *subset-sum* problem, we show that the problem is NP-hard already for deterministic $\mathbb{N}$-WFAs, getting NP-completeness for both $\mathbb{N}$-WFAs and $\mathbb{Z}$-WFAs.

We start with the upper bound. Consider a deterministic $\mathbb{Z}$-WFA $\mathcal{A}$ and a target value $\vartheta \in \mathbb{N}$. The ∃-exact problem asks for the existence of a word w, such that $\mathcal{A}(w) = \vartheta$. Since $\mathcal{A}$ is deterministic, the existence of such a word is equivalent to the existence of a path π along the graph of $\mathcal{A}$, such that the total cost of π, including the initial and final weights, is exactly ϑ.

The ability to only consider the weights of transitions, ignoring their alphabet letters, is what makes this problem decidable, as opposed to the case of the same problem over nondeterministic WFAs. Yet, unlike the non-emptiness problem, which also ignores the alphabet letters, in the case of the ∃-exact problem, we cannot use polynomial algorithms that find an optimal path, such as Dijkstra or Bellman-Ford, as we look for a path with a specific value.

We resolve the question of whether there exists a path whose value is exactly ϑ, by following known techniques for defining graph properties by Presburger arithmetic formulas, in the more general cases, and integer linear programming in special cases, such as ours [6, 19, 26, 35, 40]. Intuitively, we define a variable x_e for every transition e of $\mathcal{A}$ whose weight is not ∞, and construct a set of integer linear equations and inequalities, such that its solution assigns to each variable x_e the number of times that e is repeated in a path whose value is exactly ϑ.

Lemma 6.5 *The ∃-exact problem for deterministic* $\mathbb{Z}$-WFAs *is in NP.*

Proof: Consider a deterministic $\mathbb{Z}$-WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin} \rangle$ and a target value $\vartheta \in \mathbb{N}$. Guess the set of states $Q' \subseteq Q$ that will participate in the path, as well as an order of them, namely denote them by $q_1, q_2, \ldots, q_n$. Also guess the initial state $s = q_1 \in \mathtt{init} \cap Q'$, and a final state $t \in \mathtt{fin} \cap Q'$. (It need not be that $t = q_n$.) Let Δ' be the restriction of Δ to Q'.

We shall define a set of linear equations and inequalities, such that it has integer solutions iff there exists a path π from s to t that visits exactly the states of Q' and whose value is exactly ϑ. Our set of equations and inequalities follows similar ones in the literature, and specifically the one given in [40].

For every transition $e \in \Delta'$, define the variable x_e, which intuitively stands for the number of repetitions of e in π, and the inequality $x_e \geq 0$. To ensure that the value of the corresponding path is ϑ, we sum up the transition weights, multiplied by the number of times that they are repeated. That is, we have the equation $\mathtt{init}(s) + \mathtt{fin}(t) + \sum_{e \in \Delta'} x_e \cdot \Delta'(e) = \vartheta$.

The next step is to ensure that the assigned transition repetitions indeed correspond to a connected path. This is done in two parts. The first part ensures that cycle-wise, the path is proper, in the sense that it completes entire cycles. This is done by adding "flow equations", also known as "Kirchhoff's circuit laws", which ensure that every state, except for the initial and final states, is entered and left the same number of times.

Formally, for every state $q \in Q'$, let $In(q) = \{e \in \Delta' \mid e \text{ is an incoming transition of } q\}$ and $Out(q) = \{e \in \Delta' \mid e \text{ is an outgoing transition of } q\}$. Then, for every state $q \in Q' \setminus \{s, t\}$, define the equation $\sum_{e \in In(q)} x_e = \sum_{e \in Out(q)} x_e$, for s the equation $1 + \sum_{e \in In(s)} x_e = \sum_{e \in Out(s)} x_e$, and for t the equation $\sum_{e \in In(t)} x_e = 1 + \sum_{e \in Out(t)} x_e$.

The above flow equations ensure that the transition repetitions correspond to some connected paths, yet not necessarily to a single connected path. For example, they allow for a path that starts in s and ends in t, accompanied by another path that makes a complete cycle that is unconnected to the first path. For ensuring a single connected path, we require that every state $q \in Q'$ participates in the path by the inequality $\sum_{e \in In(q)} x_e + \sum_{e \in Out(q)} x_e > 0$, and that every state except for s, namely every $q_i \in \{q_2, \ldots, q_n\}$, has a participating incoming transition from a state of a smaller order by the inequality $\sum_{1 \leq j < i} \sum_{e \in Out(q_j) \cap In(q_i)} x_e > 0$. $\square$

Lemma 6.6 *The $\exists$-exact problem for deterministic $\mathbb{N}$-WFAs is NP-hard.*

Proof: We show NP-hardness by a reduction from the *subset-sum* problem. Recall that an instance of *subset-sum* consists of a set of numbers $x_1, \ldots, x_n, t \in \mathbb{N}$ given in binary, and the problem is to decide whether there exists $I \subseteq [1..n]$ such that $\sum_{i \in I} x_i = t$.

The reduction is depicted in Figure 7. Intuitively, given an instance as above, we construct a deterministic $\mathbb{N}$-WFA $\mathcal{A}$ with states $s_0, \ldots s_n$, such that at every state s_i we can either choose to add x_{i+1}, using the letter 'a', or to add 0, using the letter 'b'. After going over all the states, we reach a final state s_n. Thus, every word of length n induces a sum of a subset of the elements, and there exists a word of cost exactly t iff there is a subset whose sum is t.

Formally, $\mathcal{A} = \langle \Sigma, Q, \Delta, \mathtt{init}, \mathtt{fin} \rangle$, with $\Sigma = \{a, b\}$ and $Q = \{s_0, \ldots, s_n\}$. The initial weights are $\mathtt{init}(s_0) = 0$ and $\mathtt{init}(q) = \infty$ for $q \neq s_0$, and the final weights are $\mathtt{fin}(s_n) = 0$ and $\mathtt{fin}(q) = \infty$ for $q \neq s_n$. The transition function is defined as follows. For $i \in [0..n-1]$ we have $\Delta(s_i, a, s_{i+1}) = x_{i+1}$ and $\Delta(s_i, b, s_{i+1}) = 0$. All other transitions have weight ∞.

Observe that $\mathsf{Dom}(\mathcal{A}) = \{w \in \{a,b\}^* : |w| = n\}$. Indeed, words longer or shorter than n either use a transition of cost ∞, or do not reach s_n. For every word w, let $w|_a$ be the set of indices in w where the letter a appears. It is easy to see that $\mathcal{A}(w) = \sum_{i \in w|_a} x_i$. Therefore, there exists $I \subseteq [1..n]$ such that $\sum_{i \in I} x_i = t$ iff there exists a word w such that $\mathcal{A}(w) = t$, and we are done. $\square$

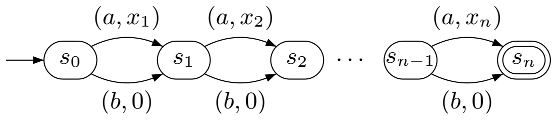

Figure 7: The reduction from *subset-sum* to $\exists$-exact. The transitions are labeled with (letter,weight).

From Lemmas 6.5 and 6.6, we conclude the NP-completeness of the problem.

Theorem 6.7 *The $\exists$-exact problem of deterministic* $\mathbb{N}$-WFAs *and* $\mathbb{Z}$-WFAs *is NP-complete.*

It remains to study the $\exists$-exact problem for nondeterministic $\mathbb{N}$-WFAs. We show that the problem is in PSPACE by taking a similar approach to that taken for universality in Theorem 6.3.

Lemma 6.8 *The $\exists$-exact problem for* $\mathbb{N}$-WFAs *is in PSPACE.*

Proof: Let $\mathcal{A}$ be an $\mathbb{N}$-WFA and $\vartheta \in \mathbb{N}$ be the given target. Consider the $\vartheta + 1$-bounded weighted determinization of $\mathcal{A}$, described in Section 5.2. By Lemma 5.5 and Proposition 5.4, there exists a word w such that $\mathcal{A}(w) = \vartheta$ (and in particular, $\mathcal{A}(w) < \vartheta + 1$) iff there exists a reachable state $E|_\vartheta$ such that $\min_{q \in Q} \{E|_\vartheta(q) + \mathtt{fin}(q) = \vartheta\}$.

In particular, such a state is reachable iff it is reachable by a word of length at most $(\vartheta + 1 + 2)^{|Q|} = (\vartheta + 3)^{|Q|}$ (which is an upper bound on the number of states in the construction). We conclude that there exists a word w with $\mathcal{A}(w) = \vartheta$ iff there exists such a word w of length at most $(\vartheta + 3)^{|Q|}$.

As in the proof of theorem 6.3, checking the latter condition can be easily done in NPSPACE by guessing each letter of w and keeping track of the accumulated weights in all the runs. Since NPSPACE=PSPACE [38], we conclude the proof. $\square$

Finally, it remains to show that the problem is PSPACE-hard.

Lemma 6.9 *The $\exists$-exact problem for* $\mathbb{N}$-WFAs *is PSPACE-hard.*

Proof: We show a reduction from the universality problem for NFAs to the complement of the $\exists$-exact problem.

Given an NFA $\mathcal{A}$, we obtain an $\mathbb{N}$-WFA from it as follows. Assume $\mathcal{A}$ is complete, that is, for every state and every letter there is at least one defined transition (this can be achieved by adding transitions to a rejecting sink). Then, we obtain an $\mathbb{N}$-WFA $\mathcal{B}$ by giving weight 0 to all the transitions in $\mathcal{A}$, and setting the final weight of accepting states in $\mathcal{A}$ to be 0, and of non-accepting states to be 1.

The output of the reduction is the $\mathbb{N}$-WFA $\mathcal{B}$ with threshold 1.

Then, if $L(\mathcal{A}) = \Sigma^*$ we have $\mathcal{B}(w) = 0$ for every $w \in \Sigma^*$, and if $L(\mathcal{B}) \neq \Sigma^*$, there exists a word x such that $\mathcal{B}(x) = 1$, which concludes the correctness of the reduction. $\square$

Combining Lemmas 6.8 and 6.9, we have the following.

Theorem 6.10 *The $\exists$-exact problem for* $\mathbb{N}$-WFAs *is PSPACE-complete.*

6.4 The $\forall$-exact problem

Recall that the $\forall$-exact problem asks, given a WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{Z}$, whether $\mathcal{A}(w) = \vartheta$ for every word $w \in \mathsf{Dom}(\mathcal{A})$. Since $\mathcal{A}(w) = \infty$ for $w \notin \mathsf{Dom}(\mathcal{A})$, an equivalent formulation is to ask whether the following conditions hold:

1. $\mathcal{A}(w) \geq \vartheta$ for every $w \in \Sigma^*$,
2. $\mathcal{A}(w) < \vartheta + 1$ for every $w \in \mathsf{Dom}(\mathcal{A})$.

Observe that the first condition is the complement of the non-emptiness problem with threshold ϑ, while the second condition is the universality problem, restricted to $\mathsf{Dom}(\mathcal{A})$ in the sense of Remark 6.4. Thus, from Propositions 6.1, 6.2 and Remark 6.4, we have the following.

Proposition 6.11 *The $\forall$-exact problem is decidable in polynomial time for deterministic* $\mathbb{Z}$-WFAs *(and hence for* $\mathbb{N}$-WFAs*).*

Similarly, from Proposition 6.1, Theorem 6.3, and Remark 6.4 we can conclude that the $\forall$-exact problem is in PSPACE for nondeterministic $\mathbb{N}$-WFAs. In addition, we notice that the reduction described in the proof of Lemma 6.9 can be adapted to show PSPACE-hardness for $\forall$-exact, by changing the threshold from 2 to 1. We thus conclude with the following.

Proposition 6.12 *The $\forall$-exact problem is PSPACE-complete for nondeterministic* $\mathbb{N}$-WFAs.

It remains to address the case of nondeterministic $\mathbb{Z}$-WFAs. Recall from Theorem 4.1 that universality for $\mathbb{Z}$-WFAs is undecidable, and therefore the result does not follow immediately as above. Nonetheless, it turns out that the main difficulty that exists for universality, namely negative cycles, does not hinder us in the case of $\forall$-exact.

Theorem 6.13 *The $\forall$-exact problem is PSPACE-complete for nondeterministic $\mathbb{Z}$-WFAs.*

Proof: Consider a $\mathbb{Z}$-WFA $\mathcal{A}$ and a threshold $\vartheta \in \mathbb{Z}$. If there is a negative cycle in $\mathcal{A}$, then $\mathcal{A}$ is unbounded from below, and in particular, words attain values other than ϑ and ∞, so we can reject. Note that this check can be done in polynomial time.

If $\mathcal{A}$ does not have a negative cycle, then by Lemma 5.8 we can compute in polynomial time an $\mathbb{N}$-WFA $\mathcal{A}^+$ and $M \in \mathbb{N}$ such that for every word $w \in \Sigma^*$ we have that $\mathcal{A}^+(w) = \mathcal{A}(w) + M$. Then, it suffices to solve the $\forall$-exact problem for $\mathcal{A}^+$ with the threshold $\vartheta + M$, using Proposition 6.12.

PSPACE-hardness follows from the hardness for $\mathbb{N}$-WFAs in Proposition 6.12. $\square$

6.5 The containment problem

Recall that by Theorem 4.2, the (D, N) and (N, N) containment problems are undecidable for both $\mathbb{N}$ and $\mathbb{Z}$-WFAs. Thus, it remains to study the (N, D) and (D, D) containment problems.

Proposition 6.14 *The (N, D) and (D, D) containment problems are decidable in polynomial time for $\mathbb{Z}$-WFAs (and hence for $\mathbb{N}$-WFAs).*

Proof: It's enough to consider the (N, D) containment problem. Consider a nondeterministic $\mathbb{Z}$-WFA $\mathcal{A} = \langle \Sigma, Q_1, \Delta_1, \texttt{init}_1, \texttt{fin}_1 \rangle$ and a deterministic $\mathbb{Z}$-WFA $\mathcal{B} = \langle \Sigma, Q_2, \Delta_2, \texttt{init}_2, \texttt{fin}_2 \rangle$.

Note that if $\mathcal{A}(w) \geq \mathcal{B}(w)$ for every word w, then in particular $\mathsf{Dom}(\mathcal{A}) \subseteq \mathsf{Dom}(\mathcal{B})$. We thus start by checking, in polynomial time, whether $\mathsf{Dom}(\mathcal{A}) \subseteq \mathsf{Dom}(\mathcal{B})$ by constructing an NFA for $\mathsf{Dom}(\mathcal{A})$ and a DFA for $\mathsf{Dom}(\mathcal{B})$. If the latter does not hold, we reject. Note that this check takes polynomial time, since the right-hand automaton is deterministic.

Next, we construct a WFA $\mathcal{C}$ as per Proposition 5.3, such that $\mathcal{C}(w) = \mathcal{A}(w) - \mathcal{B}(w)$ for every word $w \in \mathsf{Dom}(\mathcal{B})$, and $\mathcal{C}(w) = \infty$ otherwise. Observe that now, $\mathcal{A}(w) \geq \mathcal{B}(w)$ for every word w iff there does not exist a word w for which $\mathcal{C}(w) < 0$. Indeed, if $\mathcal{A}(w) \geq \mathcal{B}(w)$ for every word $w \in \Sigma^*$, then for every word $w \in \mathsf{Dom}(\mathcal{B})$ we have that $\mathcal{C}(w) \geq 0$, and for every word $w \notin \mathsf{Dom}(\mathcal{B})$ we have $\mathcal{C}(w) = \infty$. Conversely, if $\mathcal{C}(w) \geq 0$ for every word w, then for every word $w \in \mathsf{Dom}(\mathcal{B})$ we have $\mathcal{A}(w) \geq \mathcal{B}(w)$ and for every word $w \notin \mathsf{Dom}(\mathcal{B})$ we have that $w \notin \mathsf{Dom}(\mathcal{A})$ (by the Boolean containment check above), so $\mathcal{A}(w) = \mathcal{B}(w) = \infty$

Thus, it suffices to check the non-emptiness of $\mathcal{C}$ with the threshold 0, which can be done in polynomial time by Proposition 6.1. $\square$

6.6 The equality problem

Recall that by Theorem 4.3, (N, N) equality is undecidable for $\mathbb{N}$-WFAs (and hence for $\mathbb{Z}$-WFAs). It thus remains to handle $(D, N), (N, D)$, and (D, D) equality.

Observe that for (D, D)-equality, we can simply check two-sided containment. By Proposition 6.14, we have the following.

Proposition 6.15 *The (D, D) equality problem is decidable in polynomial time for* $\mathbb{Z}$-WFAs *(and hence for* $\mathbb{N}$-WFAs*).*

It remains to handle the (N, D) equality problem (which is symmetrical to the (D, N) equality problem).

Theorem 6.16 *The (N, D) equality problem is PSPACE complete for* $\mathbb{Z}$-WFAs *and for* $\mathbb{N}$-WFAs*.*

Proof: We start by showing that the problem is in PSPACE for $\mathbb{Z}$-WFAs (and hence for $\mathbb{N}$-WFAs). Let $\mathcal{A} = \langle \Sigma, Q_1, \Delta_1, \mathtt{init}_1, \mathtt{fin}_1 \rangle$ be a nondeterministic $\mathbb{Z}$-WFA, and let $\mathcal{B} = \langle \Sigma, Q_2, \Delta_2, \mathtt{init}_2, \mathtt{fin}_2 \rangle$ be a deterministic $\mathbb{Z}$-WFA.

Note that if $\mathcal{A}(w) = \mathcal{B}(w)$ for every word w, then in particular $\mathsf{Dom}(\mathcal{A}) = \mathsf{Dom}(\mathcal{B})$. We thus start by checking, in polynomial space, whether $\mathsf{Dom}(\mathcal{A}) = \mathsf{Dom}(\mathcal{B})$ by constructing NFAs for $\mathsf{Dom}(\mathcal{A})$ and for $\mathsf{Dom}(\mathcal{B})$. If the latter does not hold, we reject.

Next, we construct a WFA $\mathcal{C}$ as per Proposition 5.3, such that $\mathcal{C}(w) = \mathcal{A}(w) - \mathcal{B}(w)$ for every word $w \in \mathsf{Dom}(\mathcal{B})$, and $\mathcal{C}(w) = \infty$ otherwise. Observe that now, $\mathcal{A}(w) = \mathcal{B}(w)$ for every word w iff $\mathcal{C}(w) = 0$ for every word $w \in \mathsf{Dom}(\mathcal{B}) = \mathsf{Dom}(\mathcal{A})$.

The latter condition can be viewed as an instance of the $\forall$-exact problem, which can be solved in PSPACE by Theorem 6.13.

Next, it is easy to see that the (N, D)-equality problem is PSPACE-hard for $\mathbb{N}$-WFAs, by reducing from the universality problem for NFAs in a similar manner as the proof of Lemma 6.9. $\square$

6.7 The upper-boundedness problem

The upper-boundedness problem asks, given a WFA $\mathcal{A}$, whether there exists $M \in \mathbb{N}$ such that $\mathcal{A}(w) < M$ for every $w \in \Sigma^*$. By Theorem 4.6, the problem is undecidable for nondeterministic $\mathbb{Z}$-WFAs. We now turn to complete the picture by showing that the remaining cases are decidable.

The upper-boundedness problem was studied in [21, 31, 43] for *distance automata*, namely $\mathbb{N}$-WFAs with weights in $\{0, 1, \infty\}$. Using very careful analysis and methods such as the *tree factorization forest* of [42], they show the following.

Theorem 6.17 ([21, 31, 43]) *The upper-boundedness problem for distance automata is in PSPACE.*

Consider now an arbitrary $\mathbb{N}$-WFA $\mathcal{A}$, we can obtain from it a distance automaton by changing every weight $c \neq 0$ in $\mathcal{A}$ to weight 1. It is easy to see that $\mathcal{A}$ is upper bounded iff the obtained distance automaton is upper bounded. We thus have the following.

Lemma 6.18 *The upper-boundedness problem for nondeterministic* $\mathbb{N}$-WFAs *is in PSPACE.*

We proceed to present a matching lower bound.

Lemma 6.19 *The upper-boundedness problem for nondeterministic* $\mathbb{N}$-WFAs *is PSPACE-hard.*

Proof: We show a reduction from the universality problem for NFAs. Given an NFA $\mathcal{A}$, we obtain from it an $\mathbb{N}$-WFA $\mathcal{B}$ as follows. First, as in Lemma 6.9, we assume $\mathcal{A}$ is complete. We add a new letter # to the alphabet of $\mathcal{A}$, and two new states f and g, having self loops of weight 1 and 0, respectively, on every letter. Then, when reading # in an accepting state, we proceed to g, and in a non-accepting state, proceed to f. The final weight of f and g is 0, as well as the weight for all other accepting states. The initial weights are 0 for the initial states of $\mathcal{A}$, and all the transitions in $\mathcal{A}$ have weight 0. All other weights are ∞.

We claim that $L(\mathcal{A}) = \Sigma^*$ iff $\mathcal{B}$ is upper bounded. Indeed, the only way for a run to accumulate nonzero cost is by reaching f, from which we can accumulate unbounded cost. Thus, $\mathcal{B}$ is not upper bounded iff there exists a word w for which all the runs reach non-accepting states, iff $L(\mathcal{A}) \neq \Sigma^*$. $\square$

Combining Lemmas 6.18 and 6.19, we have the following.

Theorem 6.20 *The upper-boundedness problem for nondeterministic* $\mathbb{N}$-WFAs *is PSPACE-complete.*

We now turn to the case of deterministic WFAs. Consider a deterministic $\mathbb{Z}$-WFA $\mathcal{A}$. It is easy to see that $\mathcal{A}$ is upper-bounded iff there does not exist a positive cycle. Since detecting positive cycles can be easily done in polynomial time (using, e.g., Bellman-Ford), we have the following.

Proposition 6.21 *The upper-boundedness problem for deterministic* $\mathbb{Z}$-WFAs *is decidable in polynomial time.*

6.8 The absolute-boundedness problem

Recall that the absolute-boundedness problem asks, given a WFA $\mathcal{A}$, whether there exists $M \in \mathbb{N}$ such that $|\mathcal{A}(w)| < M$ for every $w \in \Sigma^*$.

We first observe that for $\mathbb{N}$-WFAs, this is equivalent to upper-boundedness, since $\mathbb{N}$-WFAs are always bounded from below by 0. We thus turn our attention to $\mathbb{Z}$-WFAs.

Consider a $\mathbb{Z}$-WFA $\mathcal{A}$. As we observed in our study of the non-emptiness problem, $\mathcal{A}$ is unbounded from below iff it has a negative cycle (both for deterministic and nondeterministic WFAs). Moreover, if $\mathcal{A}$ does not have a negative cycle, then by Lemma 5.8 we can obtain from $\mathcal{A}$ the $\mathbb{N}$-WFA $\mathcal{A}^+$, and notice that $\mathcal{A}$ is upper-bounded iff $\mathcal{A}^+$ is upper bounded.

Combining this with Theorem 6.20 and with Proposition 6.21, we have the following.

Theorem 6.22 *The absolute-boundedness problem for nondeterministic* $\mathbb{Z}$-WFAs *is PSPACE-complete.*

Proposition 6.23 *The absolute-boundedness problem for deterministic* $\mathbb{Z}$-WFAs *is decidable in polynomial time.*

7 Discussion and Future Research

We consider several fundamental decision problems for weighted automata. We provide alternative proofs of known undecidability results, and strengthen the results by restricting the class of automata for which they apply. We complete the picture by studying the decidability frontier, and by providing complexity bounds for the decidable problems. In order to establish these bounds, we provide a toolbox of algorithms and techniques for weighted automata. In addition, we employ Johnson's algorithm (Section 5.3) to obtain a novel efficient construction for translating $\mathbb{Z}$-WFAs without negative cycles into $\mathbb{N}$-WFAs, which in turn yields efficient algorithms for relevant problems.

The undecidability of many problems for WFAs, as well as the open status of the determinizability, has lead researchers to focus on fragments of WFAs that relate to their *ambiguity* – the number of possible runs with finite weight on a given word. Restricting the ambiguity can have significant implications on the decidability and complexity of decision problems. For instance, it is shown in [25] that determinizability is decidable for polynomially-ambiguous WFAs, and in [22] that the equivalence problem is decidable for finitely-ambiguous WFAs.

It is not hard to see that the construction we present in Section 3 yields a linearly-ambiguous WFA, implying that our undecidability results hold already for linearly-ambiguous WFAs. In the future, we plan to refine our study and include bounds for different ambiguity classes.

References

[1] S. Almagor, M. Cadilhac, F. Mazowiecki, and G. A. Pérez. Weak cost register automata are still powerful. In *Proc. of 22nd DLT*, volume 11088 of *LNCS*, pages 83–95. Springer, 2018.

[2] S. Almagor and O. Kupferman. Max and sum semantics for alternating weighted automata. In *9th Int. Symp. on Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 13–27. Springer, 2011.

[3] B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.

[4] D. Andersson. An improved algorithm for discounted payoff games. In *ESSLLI Student Session*, pages 91–98, 2006.

[5] C. Baier, N. Bertrand, and M Grösser. Probabilistic automata over infinite words: Expressiveness, efficiency, and decidability. In *Proc. 11th International Workshop on Descriptional Complexity of Formal Systems*, pages 3 – 16, 2006.

[6] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):27:1–27:25, 2014.

[7] U. Boker and T. A. Henzinger. Exact and approximate determinization of discounted-sum automata. *Logical Methods in Computer Science*, 10(1), 2014.

[8] U. Boker, T. A. Henzinger, and J. Otop. The target discounted-sum problem. In *Proc. of 30th LICS*, pages 750–761. IEEE Computer Society, 2015.

[9] K. Chatterjee, L. Doyen, and T. Henzinger. Quantitative languages. In *Proc. 17th Annual Conf. of the European Association for Computer Science Logic*, volume 5213 of *LNCS*, pages 385–400. Springer, 2008.

[10] K. Chatterjee, L. Doyen, and T. Henzinger. Alternating weighted automata. In *Proc. 17th International Symposium on Fundamentals of Computation Theory*, volume 5699, pages 3–13. Springer, 2009.

[11] K. Chatterjee, L. Doyen, and T. A. Henzinger. Expressiveness and closure properties for quantitative languages. *Logical Methods in Computer Science*, 6(3), 2010.

[12] K. Culik and J. Kari. Digital images and formal languages. *Handbook of formal languages, vol. 3: beyond words*, pages 599–616, 1997.

[13] A. Degorre, L. Doyen, R. Gentilini, J. Raskin, and S. Torunczyk. Energy and mean-payoff games with imperfect information. In *Proc. 19th Annual Conf. of the European Association for Computer Science Logic*, volume 6247 of *LNCS*, pages 260–274. Springer, 2010.

[14] M. Droste and P. Gastin. Weighted automata and weighted logics. *Theoretical Computer Science*, 380(1-2):69–86, 2007.

[15] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[16] M. Droste and D. Kuske. Weighted automata. In *Handbook of Automata Theory (J.-E. Pin, ed)*, chapter 4. European Mathematical Society, in press., 2019.

[17] M. Droste and I. Meinecke. Weighted automata and weighted MSO logics for average and long-time behaviors. *Inf. Comput.*, 220:44–59, 2012.

[18] S. Eilenberg. *Automata, languages, and machines*. Academic press, 1974.

[19] S. Ginsburg and E.H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[20] K. Hashiguchi. Limitedness theorem on finite automata with distance functions. *Journal of computer and system sciences*, 24(2):233–244, 1982.

[21] K. Hashiguchi. New upper bounds to the limitedness of distance automata. *Theoretical Computer Science*, 233(1-2):19–32, 2000.

[22] K. Hashiguchi, K. Ishiguro, and S. Jimbo. Decidability of the equivalence problem for finitely ambiguous finance automata. *International Journal of Algebra and Computation*, 12(03):445–461, 2002.

[23] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, 24(1):1–13, 1977.

[24] J. Berstel Jr and C. Reutenauer. *Rational series and their languages.* Springer-Verlag, 1988.

[25] D. Kirsten. A Burnside approach to the termination of Mohri's algorithm for polynomially ambiguous min-plus-automata. *RAIRO-Theoretical Informatics and Applications*, 42(3):553–581, 2008.

[26] S. Rao Kosaraju and G. F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In *STOC*, pages 398–406. ACM, 1988.

[27] D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3):405–425, 1994.

[28] D. Krob. Some consequences of a Fatou property of the tropical semiring. *Journal of Pure and Appllied Algebra*, 93(3):231–249, 1994.

[29] W. Kuich and A. Salomaa. *Semirings, automata, languages*, volume 5. Springer Science & Business Media, 2012.

[30] D. Kuperberg. Linear temporal logic for regular cost functions. In *Proc. 28th Symp. on Theoretical Aspects of Computer Science*, volume 9 of *LIPIcs*, pages 627–636. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[31] H. Leung and V. Podolskiy. The limitedness problem on distance automata: Hashiguchi's method revisited. *Theoretical Computer Science*, 310(1-3):147–158, 2004.

[32] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972.

[33] M.L. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, 1 edition, 1967.

[34] M. Mohri, F.C.N. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[35] R. J. Parikh. On context-free languages. *J. of the ACM*, 13(4):570–581, 1966.

[36] A. Paz. *Introduction to probabilistic automata.* Computer science and applied mathematics. Academic Press, 1971.

[37] A. Salomaa and M. Soittola. *Automata-theoretic aspects of formal power series.* Springer Science & Business Media, 2012.

[38] W.J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *Journal of Computer and Systems Science*, 4:177–192, 1970.

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

[40] H. Seidl, T. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *ICALP*, volume 3142 of *LNCS*, pages 1136–1149. Springer, 2004.

[41] I. Simon. Recognizable sets with multiplicitives in the tropical semiring. In *13th Int. Symp. on Mathematical Foundations of Computer Science*, volume 324 of *LNCS*, pages 107–120. Springer, 1988.

[42] I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.

[43] I. Simon. On semigroups of matrices over the tropical semiring. *RAIRO-Theoretical Informatics and Applications*, 28(3-4):277–294, 1994.

[44] M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[45] U. Zwick and M.S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.