What's Decidable About Weighted Automata?

Shaull Almagor[1], Udi Boker[1,2], and Orna Kupferman[1]

[1] Hebrew University, School of Engineering and Computer Science, Jerusalem, Israel.
[2] IST, Austria.

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 with weights in $\mathbb{N} \cup \{\infty\}$, and that the equality problem is undecidable when the weights are in $\mathbb{Z} \cup \{\infty\}$.

In this paper we continue the study of the borders of decidability in weighted automata, describe alternative and direct proofs of the above 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 significantly simplify Krob's reasoning and 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 decidability results and to show that the universality problem for weighted automata with weights in $\mathbb{N} \cup \{\infty\}$, and in fact even with weights in $\mathbb{Q}^{\geq 0} \cup \{\infty\}$, is PSPACE-complete. Our results thus draw a sharper picture about the decidability of decision problems for weighted automata, in both the front of equality vs. universality and the front of the $\mathbb{N} \cup \{\infty\}$ vs. the $\mathbb{Z} \cup \{\infty\}$ domains.

1 Introduction

Traditional 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 [5, 6, 10, 13, 17], for reasoning about probabilistic systems [3], and for reasoning about the competitive ratio of on-line algorithms [1], 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 [8, 16].

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.

Other two problems of great interest in the context of automata are the *universality* and *containment* problems. 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 words is not just to get accepted, but also to do it with a minimal value. Accordingly, the universality problem for WFAs asks, given a WFA $\mathcal{A}$ and a threshold v, whether $\mathcal{A}$ assigns a value that is smaller than v to all words in Σ^*. Similarly, the containment problem in the weighted setting naturally extends the Boolean one by asking, given two WFAs $\mathcal{A}$ and $\mathcal{B}$, whether for all words $w \in \Sigma^*$, the value of w in $\mathcal{B}$ is less than or equal to its value in $\mathcal{A}$. In the Boolean setting, the complexity for the two problems coincide, and is PSPACE-complete [14]. As we shall see in this paper, in the weighted setting the picture is more involved. Both problems are of great practical interest: in the automata-theoretic approach to reasoning about systems and their specifications, universality amounts to validity of specifications, 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 [5].

Recall that weighted automata map words to numerical values. Technically, each weighted automaton is defined with respect to an algebraic semiring. For example, $\langle \mathbb{N} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ is a semiring whose sum operator is min (with ∞ being the identity element) and whose product operator is $+$ (with 0 being the identity element). Such a min-sum semiring is called a *tropical 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 analogous to the one of weighted automata is the one of *rational series* [18]. There too, the series is defined with respect to a semiring, and maps words to values from the domain of the semiring.

In [11], 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 problem for the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$. Moreover, in [12], Krob proved that universality for rational series defined with respect to the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$ is decidable. The analogy between rational series and weighted automata implies the same results for the universality and containment problems for weighted automata.

In this paper 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 automata over infinite words.

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: while 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: the weights of the automaton are in $\{-1, 0, 1\}$, it has no initial nor final weights, and all its states are accepting. We note that Krob's reduction does not capture this weaker class of automata.

Third, the pure algebraic view of rational series has the drawback that it cannot be generalized to some natural extensions of the weighted setting. For example, rational series cannot capture weighted automata on infinite words (where one cannot speak about final states or final weights), nor can it capture discounted-sum automata over finite and infinite words [19, 2, 5, 4]. For these cases, the non-algebraic, automata-theoretic definition, is useful [5, 9, 6, 7].

Our proof uses ideas similar to those presented in [9]. 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. Hence, we have that $\mathcal{M}$ halts if and only if $\mathcal{A}$ is not universal with respect to the threshold 1. A direct corollary is that the containment problem is also undecidable. Our proof generalizes the undecidability result of [11], and we believe that it may also be useful when investigating other models of weighted automata.

Recall that when rational series are defined with respect to the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$, universality becomes decidable [12]. The fact that we work directly with the automata enables us to tighten this result too. By bounding the length of the shortest witness to non-universality we are able to show that the universality problem for weighted automata defined with respect to the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$ is PSPACE-complete. We extend this good news also to weighted automata defined with respect to the tropical semiring with domain $\mathbb{Q}^{\geq 0} \cup \{\infty\}$. On the other hand, we show that restricting to the domain $\mathbb{N} \cup \{\infty\}$ is not helpful for the containment problem, which is undecidable. We conclude that, unlike the Boolean case, the universality and containment problems do not have the same complexity in the weighted setting, and are in fact on different sides of the border of decidability. Moreover, this border crucially depends on whether the weights of the weighted automaton are all of the same polarity (all in $\mathbb{N} \cup \{\infty\}$ or all in $-\mathbb{N} \cup \{-\infty\}$) or are mixed (as in $\mathbb{Z} \cup \{\infty\}$).

2 Preliminaries

While standard automata map words in Σ^* to either "accept" or "reject", weighted automata may be viewed as partial functions (defined only for accepted words) from Σ^* to $\mathbb{Q} \cup \{\infty\}$. Formally, a *weighted finite automaton* (WFA, for short) is a 8-tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f \rangle$, where Σ is a finite input alphabet, Q is a finite set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, $c : \Delta \to \mathbb{Q}$ is a cost function, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, $i : Q_0 \to \mathbb{Q} \cup \{\infty\}$ is an initial-weight function, and $f : F \to \mathbb{Q} \cup \{\infty\}$ is a final-weight function. A

transition $d = \langle q, a, p \rangle \in \Delta$ (also written as $\Delta(q, a, p)$) 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* $c(d)$. 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 $\Delta(q, a, p_1)$ and $\Delta(q, a, p_2)$, with $p_1 \neq p_2$. We say that $\mathcal{A}$ is *complete* if Δ is total; that is, for every state $q \in Q$ and letter $a \in \Sigma$, there is at least one state $p \in Q$ such that $\Delta(q, a, p)$.

For a word $w = w_1 \ldots w_n \in \Sigma^*$, and states $q, q' \in Q$, a *run* of $\mathcal{A}$ on w is a sequence $r = r_0 r_1 \ldots r_n \in Q^+$, where $r_0 \in Q_0, r_n \in F$, 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 $c(r) = i(r_0) + \sum_{i=1}^{n} c(d_i) + f(r_n)$. Note that if $\mathcal{A}$ is nondeterministic, it may have several runs on w. The *cost* of w in $\mathcal{A}$ is $L_{\mathcal{A}}(w) = \min \{c(r) : r$ is a run of $\mathcal{A}$ on $w \}$. If the minimum is taken over an empty set, then w is not in the range of $L_{\mathcal{A}}$.

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*, $\langle \mathbb{Q} \cup \{\infty\}, \min, +, \infty, 0 \rangle$, sometimes called the *tropical* semiring, as defined above.

Recall that in the binary setting, the *universality* problem asks, given a nondeterministic automaton (NFA) $\mathcal{A}$, whether $L(\mathcal{A}) = \Sigma^*$. Thus, all the words in Σ^* have to be accepted by the automaton. In the weighted setting, the "goal" of words is not just to get accepted, but also to do it with a minimal value. Accordingly, the universality problem for WFAs asks, given a WFA $\mathcal{A}$ and a threshold $v \in \mathbb{Q}$ given in binary, whether $L_{\mathcal{A}}(w) < v$ for all $w \in \Sigma^*$. We denote the latter fact by $L_{\mathcal{A}} < v$. The *containment* and *equality* problems for NFAs are lifted to the weighted setting in a similar manner: Given two WFAs $\mathcal{A}$ and $\mathcal{B}$, the containment problem is to decide whether $L_{\mathcal{A}}(w) \geq L_{\mathcal{B}}(w)$ for all $w \in \Sigma^*$. We refer to $\perp$ as being greater than ∞, thus if $L_B(w) = \perp$ then $L_A(w) = \perp$ too. Thus, the domain of $\mathcal{A}$ has to be contained in the domain of $\mathcal{B}$. [1] Similarly, the equality problem is to decide whether $L_{\mathcal{A}}(w) = L_{\mathcal{B}}(w)$ for all $w \in \Sigma^*$. In particular, the domains of $L_{\mathcal{A}}$ and $L_{\mathcal{B}}$ coincide. It is easy to see that an upper bound on the containment problem implies upper bounds on the equality and the universality problems. Also, a lower bound on the universality problem implies a lower bound on the containment and the equality problems. In the Boolean setting, the complexity for the three problems coincide, and is PSPACE-complete [14]. 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. Studying the universality problem, it is more convenient to consider its dual, namely the *non-universality* problem. There, given

---

[1] For our confused readers, the $\geq$ in the $L_{\mathcal{A}}(w) \geq L_{\mathcal{B}}(w)$ condition is not a typo: recall that the goal of words is to get accepted, and with a minimal value. When $\mathcal{A}$ is contained in $\mathcal{B}$, it is more challenging for words to satisfy their goal in $\mathcal{A}$ rather than in $\mathcal{B}$. 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 $\mathcal{A}$ being greater than the values to which they are mapped in $\mathcal{B}$.

$\mathcal{A}$ and v, we ask whether there is a word $w \in \Sigma^*$ such that $L_{\mathcal{A}}(w) \geq v$. Thus, the non-universality problem asks whether there exists a word for which all the runs of $\mathcal{A}$ have value of at least v.

3   Weighted Automata With Integer Weights

In this section we show that the universality problem, and therefore also the containment problem, are undecidable for WFAs with weights in $\mathbb{Z}$. In fact, even when only considering complete automata where all states are final, and without initial or final weights, in which the weights are only in $\{-1, 0, 1\}$, the problems remain undecidable.

We show this by a reduction from the halting problem for two-counter (Minsky) machines. Our proof uses ideas similar to those presented in [9]. 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:

$$\text{INC}(c), \ \text{DEC}(c), \ \text{GOTO } l_i, \ \text{IF } c{=}0 \text{ GOTO } l_i \text{ ELSE GOTO } l_j, \ \text{HALT},$$

where $c \in \{x, y\}$ is a counter and $1 \leq i, j \leq n$ are locations. Since we can always check whether $c = 0$ before a $\text{DEC}(c)$ command, we assume that the machine never reaches $\text{DEC}(c)$ 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 [15]. 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*.

We are going to reduce the 0-halting problem to the non-universality problem for complete WFAs with weights in {-1,0,1}, without initial weights or final weights, in which all the states are final.

Theorem 1. *The universality problem for complete* WFAs *over the semiring* $\langle \mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ *with weights in {-1,0,1}, without initial weights or final weights, in which all the states are final, is undecidable.*

Proof. We show a reduction from the 0-halting problem for two-counter machines to the non-universality problem. Let $\mathcal{M}$ be a two-counter machine with commands $(l_1, \ldots, l_n)$. A *halting 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 $\text{INC}(x)$ command (resp. $\text{INC}(y)$), then $\alpha' = \alpha + 1$, $\beta' = \beta$ (resp. $\beta = \beta + 1$, $\alpha' = \alpha$), and $l' = l_{k+1}$.
    - If l_k is a $\text{DEC}(x)$ command (resp. $\text{DEC}(y)$), then $\alpha' = \alpha - 1$, $\beta' = \beta$ (resp. $\beta = \beta - 1$, $\alpha' = \alpha$), and $l' = l_{k+1}$.

5

- 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.

3. $\rho_m = \langle l_k, \alpha, \beta \rangle$ such that l_k is a HALT command.

Observe that the machine $\mathcal{M}$ is deterministic. We say that a machine $\mathcal{M}$ 0-halts if its run 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 reduction. We construct a WFA $\mathcal{A}$ such that $\mathcal{M}$ 0-halts iff there exists $w \in \Sigma^*$ such that $L_{\mathcal{A}}(w) \geq 1$. The alphabet of $\mathcal{A}$ consists of the following $n + 5$ letters:

$$\Sigma = \{\text{INC}(x), \text{DEC}(x), \text{INC}(y), \text{DEC}(y), \text{HALT}\} \cup \{\text{GOTO } l_i : i \in \{1, \ldots, n\}\}.$$

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 $\mathcal{A}$ cost at least 1. Thus, the word w is such that $L_{\mathcal{A}}(w) \geq 1$. 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 $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0 \rangle$. Observe that we omit F, i and f, as all the states are accepting, and there are no initial nor final weights. In Section 3.1 we describe an example of the reduction.

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

$\Sigma, 0$ q_{freeze} $\Sigma, -1$ q_{halt}

Fig. 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 INC or

DEC command, l_i is a GOTO l_j command, or l_i is an IF $c{=}0$ GOTO l_k ELSE GOTO l'_k command, with $j \in \{k, k'\}$. The letters 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 IF $c{=}0$ GOTO l_k ELSE GOTO l'_k command with $j \in \{k, k'\}$, in which case the letter is 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 succesive 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 $\mathcal{M}$. The test is local, as this gadget does not check for violations involving illegal jumps due to the value 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 = $ HALT to q_{halt}. These transitions 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 = $ 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 $\langle q_i, \text{GOTO } l_k, q_{freeze}\rangle$ with weight -1. We add an initial state q_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 $\langle q, \text{HALT}, q_{freeze}\rangle$ 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.

Zero Jump Checker. Dually to the positive jump checker, we define the gadget *zero jump checker* for each counter $c \in \{x, y\}$.
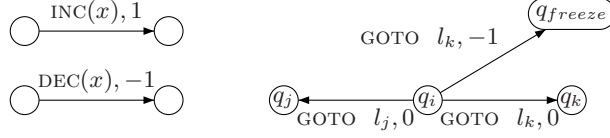
Fig. 2. Positive Jump Checker for x, where $l_i :$ IF $x{=}0$ GOTO l_j ELSE GOTO l_k.

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, $ GOTO $l_j, q_{freeze}\rangle$ 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, $ HALT$, q_{freeze}\rangle$ with weight 0 (See Figure 3).
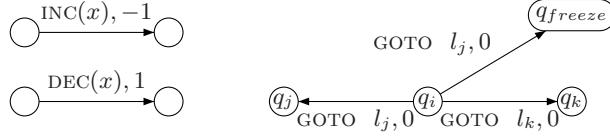

Fig. 3. Zero Jump Checker for x, where $l_i :$ IF $x{=}0$ GOTO l_j ELSE GOTO l_k.

To complete the definition of the automaton, we define Q_0 to include the states corresponding to l_1 in the command checker gadget and the q_0 states defined for the jump checkers for each counter $c \in \{x, y\}$.

We claim that $\mathcal{M}$ 0-halts iff there exists $w \in \Sigma^*$ such that $L_{\mathcal{A}}(w) \geq 1$. 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 $w[1, \ldots, k]$

is $val_k(c) + 1$, where $val_k(c)$ is the value of the counter c after the commands $w_1, \ldots, w_k$ 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 in a location where the command is of the form IF $c=0$ GOTO l_i ELSE GOTO l_j is to 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 $L_\mathcal{A}(w) \geq 1$.

We proceed to prove the harder direction. Assume there exists $w \in \Sigma^*$ such that $L_\mathcal{A}(w) \geq 1$. 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 succesive 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 $L_\mathcal{A}(w) \leq 0$, contradicting the assumption that $L_\mathcal{A}(w) \geq 1$. 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 $L_\mathcal{A}(w) \geq 1$, 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 $l_i = $ IF $c=0$ GOTO l_j ELSE GOTO l_k be the corresponding command in $\mathcal{M}$. 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

9

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 $L_{\mathcal{A}}(w) \geq 1$. 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 $L_{\mathcal{A}}(w) \geq 1$.

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

3.1 An Example

In this section we describe an example of the reduction presented in Theorem 1. Consider the following two-counter machine $\mathcal{M}$.

$$l_1 : \text{IF } x{=}0 \text{ GOTO } l_5 \text{ ELSE GOTO } l_2$$
$$l_2 : \text{DEC}(x)$$
$$l_3 : \text{INC}(y)$$
$$l_4 : \text{GOTO } l_1$$
$$l_5 : \text{HALT}$$

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.
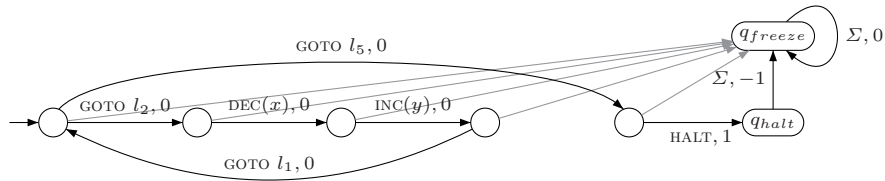

Fig. 4. Example Command Checker.

Note that the cost of GOTO l_5, HALT is 1. Note also that some words that do not fit the actual run of $\mathcal{M}$, for example GOTO l_2, DEC(x), INC(y), GOTO l_1, GOTO l_5, HALT, also have cost 1. 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.
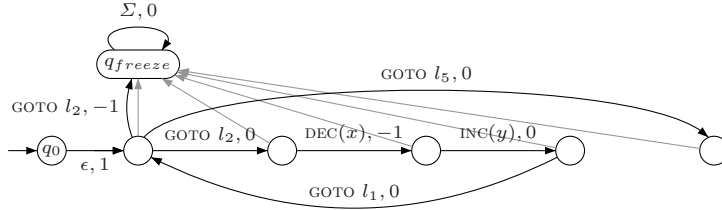

Fig. 5. Example Positive Jump Checker for x.

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 $\mathcal{M}$, no word
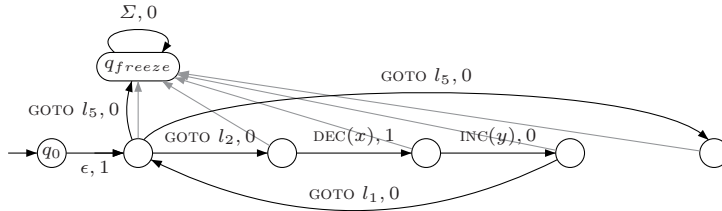

Fig. 6. Example Zero Jump Checker for x.

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   Weighted Automata With Positive Weights

In many models, the complexity of the universality problem and of the containment problem coincide. This is the case with Boolean automata, in which they are both PSPACE-complete [14], as well as with weighted automata over integer weights, for which the previous section shows undecidability. In this section we show that the model of weighted automata over positive integers is different: while the universality problem is PSPACE-complete, the containment problem is undecidable.

4.1   Universality is PSPACE-Complete

In this section we prove that the universality problem for WFAs defined over the tropical semiring with domain $\mathbb{N} \cup \{\infty\}$, and in fact even $\mathbb{Q}^{\geq 0} \cup \{\infty\}$, is decidable, and is PSPACE-complete. Intuitively, the fact the weights are all positive enables us to bound the length of a shortest witness to non-universality, and reason about a bounded unwinding of the WFA into a deterministic WFA.

Theorem 2. *The universality problem for WFAs defined with respect to the semiring $\langle \mathbb{N} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ is PSPACE-complete.*

Proof. Consider a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f \rangle$ and a threshold $v \in \mathbb{N}$. Given a word $w = w_1 \cdots w_n$, we associate with w the *weighted subset run* $E_0, E_1, \ldots, E_n$ of $\mathcal{A}$ on w as follows. Each E_i is a function $E_i : Q \to \mathbb{N} \cup \{\infty, \bot\}$. We initialize E_0 by setting

$$E_0(q) = \begin{cases} i(q) & \text{if } q \in Q_0, \\ \bot & \text{if } q \notin Q_0. \end{cases}$$

For every letter w_i, we then define the function E_i as follows. For every $q \in Q$, we define $E_i(q) = \min\{\Delta(q', w_i, q) + E_{i-1}(q') : E_{i-1}(q') \neq \bot\}$. That is, $E_i(q)$ is the minimal cost of a run of $\mathcal{A}$ on $w_1 \cdots w_i$, excluding the final weight. If the minimum is taken over the empty set, we define $E_i(q) = \bot$, which means that q is not reachable after reading $w_1 \cdots w_i$.

   We first prove that if there exists some word $w \in \Sigma^*$ such that $L_{\mathcal{A}}(w) \geq v$ (a "witness"), then there also exists such a word of length bounded by $(v+2)^{|Q|}$.

   Consider a word $w = w_1 \cdots w_n \in \Sigma^*$, and let $E_0, \ldots, E_n$ be its weighted subset run. We associate with every set E_i its *v-restriction* E_i^v, defined by

$$E_i^v(q) = \begin{cases} v & \text{if } E_i(q) \geq v, \\ E_i(q) & \text{if } E_i(q) < v, \\ \bot & \text{if } E_i(q) = \bot. \end{cases}$$

We say that E_i and E_j *v-coincide* if $E_i^v = E_j^v$. It is easy to see that there are only $(v+2)^{|Q|}$ different v-restrictions. Indeed, a v-restriction corresponds to a vector of size $|Q|$ with entries in $\{0, \ldots, v, \bot\}$. Assume that w is a shortest witness for the non-universality. Then, we claim that no two weighted subsets in $E_0, \ldots, E_n$ can v-coincide.

12

Assume by way of contradiction that E_i and E_j v-coincide, for $i < j$. We shrink w to obtain $w' = w[1, \ldots, i] \cdot w[j + 1, \ldots, n - 1]$. We claim that w' is a (shorter) witness for the non-universality. Indeed, let $F_0, \ldots, F_i, F_{j+1}, \ldots, F_n$ be the weighted subset run of w'. Clearly $F_k = E_k$ for all $k \leq i$. Since E_i and E_j v-coincide, then for every q, either $E_i(q) = E_j(q) = \bot$, or $E_i(q) = E_j(q) \in \{0, \ldots, v\}$, or both $E_i(q) > v$ and $E_j(q) > v$. In the first case, q is not reachable, so no runs continue through q. In the second case, every run that visits q after reading $w[1, \ldots, i]$ can continue the same way it continues from q after it reads $w[1, \ldots, j]$; thus the cost of these runs is the same in w' as in w. In the third case, every run through q has already accumulated weight v. Since all the weights are positive, all the relevant runs reach F_n with weight at least v. Since $L_{\mathcal{A}}(w) \geq v$, then every run of $\mathcal{A}$ on w costs at least v. It follows that every run on w' also costs at least v, so $L_{\mathcal{A}}(w') \geq v$, which is a contradiction to the minimality of the length of w.

Thus, every v-restriction of a weighted subset construction run on a shortest witness can appear at most once. It follows that the length of a shortest witness is bounded by $(v + 2)^{|Q|} = 2^{|Q|\log(v+2)}$.

We proceed to describe how we can solve the non-universality problem. Observe that given w, we can check whether $L_{\mathcal{A}}(w) \geq v$ by calculating the functions E_i for all $0 \leq i \leq n$, add $f(q)$ for all $q \in F$ in E_n, and take the minimal cost, which is exactly $L_{\mathcal{A}}(w)$.

Since the length of the shortest witness is single-exponential in $|Q|$ and $\log v$, which is the size of the input to the problem, we can solve the problem by guessing a run of this length on-the-fly, thus deciding the problem in NPSPACE=PSPACE. An important note is that calculating E_i from E_{i-1} involves arithmetic operations on numbers, which can all be done in linear time in the representation of the numbers.

Since PSPACE=co-PSPACE, it follows that the universality problem for WFAs is in PSPACE as well. Finally, it is not hard to see that the universality problem is at least as hard as the universality problem for NFW, which is known to be PSPACE-hard [14]. □

The PSPACE proof above makes use of the discrete nature of $\mathbb{N}$. Nevertheless, even when the domain with respect to which the automaton is defined is dense, there are only finitely many different "configurations" one has to consider. Formally, we have the following.

Theorem 3. *The universality problem for WFAs defined with respect to the semiring $\langle \mathbb{Q}^{\geq 0} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ is PSPACE-complete.*

Proof. The upper bound is similar to the one described in the proof of Theorem 2, except that we start by multiplying all the weights of $\mathcal{A}$ as well as v by a common denominator α, so that they are all natural. The denominator α is at most the multiplication of all the denominators, which is single-exponential in the weights and v. It follows that the bound on the shortest witness is $(\alpha v + 2)^{|Q|}$. Thus, the problem can be solved in NSPACE$(|Q| \log(\alpha v + 2))$, which is polynomial in $|Q|, v$. □

4.2 Containment is Undecidable

We now show that the containment problem is undecidable for WFAs with weights in $\mathbb{N}$. In fact, the problem is undecidable already for complete WFAs with weights in $\{0,1,2\}$, without initial or final weights, in which all the states are final.

The decidability result for the universality problem used the monotonicity of weights accumulated in weighted automata with weights in $\mathbb{N}$. One may wonder why a similar approach cannot work for the containment problem. The reason is that the containment problem relates to the difference between two WFAs. Consequently, the underlying function, which is the difference in the weight accumulated in the two WFAs, is not monotonic even when the automata have only positive weights.

The undecidability proof is by a reduction from the containment problem for WFAs defined with respect to the domain $\mathbb{Z}$. It follows an analogous lemma in [11], according to which, two WFAs with domain $\mathbb{Z}$ are equal iff so are WFAs that they induce, and that are with domain $\mathbb{N}$. Intuitively, the induced WFAs are obtained by increasing all the weights in the original WFAs. Formally, we have the following.

Theorem 4. *The containment and equality problems for complete* WFAs *over the semiring* $\langle \mathbb{N} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ *with weights in* $\{0,1,2\}$, *without initial or final weights, in which all the states are final, is undecidable.*

Proof. We start by defining a "weight-increase" operation on WFAs. Consider a number $k \in \mathbb{N}$ and a WFA $\mathcal{A}$ over $\mathbb{Z}$ with a cost function c. We define the k-*increase* of $\mathcal{A}$, denoted $\mathcal{A}^{+k}$, to be a WFA with a cost function c^{+k} that is equivalent to $\mathcal{A}$, except for having all weights increased by k; that is, for every transition d of $\mathcal{A}$, we have that $c^{+k}(d) = c(d) + k$.

We claim that for every word w, we have that $L_{\mathcal{A}^{+k}}(w) = L_{\mathcal{A}}(w) + k|w|$. Indeed, consider a run r of $\mathcal{A}$ on w, such that $c(r) = L_{\mathcal{A}}(w)$. Since $\mathcal{A}^{+k}$ has the same transitions as $\mathcal{A}$, there is a run r' of $\mathcal{A}^{+k}$ on w that follows the same transitions as r. Thus, $c(r') = c(r) + k|w|$, and therefore $L_{\mathcal{A}^{+k}}(w) \leq L_{\mathcal{A}}(w) + k|w|$. Analogously, we have that $L_{\mathcal{A}}(w) \leq L_{\mathcal{A}^{+k}}(w) - k|w|$, choosing the same run for $\mathcal{A}$ as the one used for $\mathcal{A}^{+k}$. Hence, $L_{\mathcal{A}^{+k}}(w) = L_{\mathcal{A}}(w) + k|w|$.

Now, consider two automata, $\mathcal{A}$ and $\mathcal{B}$, over $\mathbb{Z}$. Let k be the maximal absolute value of a weight in the transitions of $\mathcal{A}$ and $\mathcal{B}$. It is easy to see that all the weights in $\mathcal{A}^{+k}$ and $\mathcal{B}^{+k}$ are positive, thus they are defined with respect to the domain $\mathbb{N}$. We claim that $L_{\mathcal{A}} \leq L_{\mathcal{B}}$ iff $L_{\mathcal{A}^{+k}} \leq L_{\mathcal{B}^{+k}}$. Indeed, for every word w, $L_{\mathcal{A}^{+k}}(w) \leq L_{\mathcal{B}^{+k}}(w)$ iff $L_{\mathcal{A}^{+k}}(w) + k|w| \leq L_{\mathcal{B}^{+k}}(w) + k|w|$. Hence, the containment problem of WFAs over $\mathbb{Z}$ can be reduced to the containment problem of WFAs over $\mathbb{N}$, which is undecidable by Theorem 1. Furthermore, as the automata in Theorem 1 can be restricted to have weights in $\{-1, 0, 1\}$, their corresponding automata over $\mathbb{N}$ can be restricted to have weights in $\{0, 1, 2\}$.

We now reduce the containment problem to the equality problem, showing that the latter is undecidable as well. For WFAs $\mathcal{A}$ and $\mathcal{B}$, observe that $L_{\mathcal{A}} \leq L_{\mathcal{B}}$ iff $L_{\mathcal{A}} = min\{L_{\mathcal{A}}, L_{\mathcal{B}}\}$. Since we can easily construct a WFA for $min\{L_{\mathcal{A}}, L_{\mathcal{B}}\}$, then we can indeed reduce the containment problem to the equality problem. $\qquad \square$

References

1. B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.
2. D. Andersson. An improved algorithm for discounted payoff games. In *ESSLLI Student Session*, pages 91–98, 2006.
3. 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.
4. U. Boker and T. A. Henzinger. Determinizing discounted-sum automata. In *Proc. 20th Annual Conf. of the European Association for Computer Science Logic*, 2011.
5. K. Chatterjee, L. Doyen, and T. Henzinger. Quantative languages. In *Proc. 17th Annual Conf. of the European Association for Computer Science Logic*, pages 385–400, 2008.
6. 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, 2009.
7. K. Chatterjee, L. Doyen, and T. A. Henzinger. Expressiveness and closure properties for quantitative languages. *Logical Methods in Computer Science*, 6(3), 2010.
8. K. Culik and J. Kari. Digital images and formal languages. *Handbook of formal languages, vol. 3: beyond words*, pages 599–616, 1997.
9. 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*, pages 260–274, 2010.
10. M. Droste and P. Gastin. Weighted automata and weighted logics. In *Proc. 32nd Int. Colloq. on Automata, Languages, and Programming*, pages 513–525, 2005.
11. 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.
12. D. Krob. Some consequences of a fatou property of the tropical semiring. *Journal of Pure and Appllied Algebra*, 93(3):231–249, 1994.
13. D. Kuperberg. Linear temporal logic for regular cost functions. In *Proc. 28th Symp. on Theoretical Aspects of Computer Science*, pages 627–636, 2011.
14. A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972.
15. M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1 edition, 1967.
16. 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.
17. M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.
18. I. Simon. Recognizable sets with multiplicitives in the tropical semiring. In *13th Int. Symp. on Mathematical Foundations of Computer Science*, volume 324 of *Lecture Notes in Computer Science*, pages 107–120, 1988.
19. U. Zwick and M.S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.