Max and Sum Semantics for Alternating Weighted Automata

Shaull Almagor and Orna Kupferman

Hebrew University, School of Engineering and Computer Science, Jerusalem, Israel.

Abstract. In the traditional Boolean setting of formal verification, alternating automata are the key to many algorithms and tools. In this setting, the correspondence between disjunctions/conjunctions in the specification and nondeterministic/universal transitions in the automaton for the specification is straightforward. A recent exciting research direction aims at adding a quality measure to the satisfaction of specifications of reactive systems. The corresponding automata-theoretic framework is based on *weighted automata*, which map input words to numerical values. In the weighted setting, nondeterminism has a minimum semantics – the weight that an automaton assigns to a word is the cost of the cheapest run on it. For universal branches, researchers have studied a (dual) *maximum* semantics. We argue that a *summation* semantics is of interest too, as it captures the intuition that one has to pay for the cost of all conjuncts.

We introduce and study alternating weighted automata on finite words in both the max and sum semantics. We study the duality between the min and max semantics, closure under max and sum, the added power of universality and alternation, and arithmetic operations on automata. In particular, we show that universal weighted automata in the sum semantics can represent all polynomials.

1 Introduction

Formal verification is the study of algorithms and tools for the development of correct hardware and software systems. Traditional formal verification is Boolean: the system may either satisfy its specification or not satisfy it. A recent exciting research direction aims at adding a quality measure to the satisfaction of specifications of reactive systems, and using it in order to formally define and reason about quality of systems and in order to improve the quality of automatically synthesized systems.

The *automata-theoretic* approach uses the theory of automata as a unifying paradigm for system specification, verification, and synthesis [17, 19]. By viewing computations as words (over the alphabet of possible assignments to variables of the system), we can view both the system and its specification as languages, and reduce problems like model checking, satisfiability, and synthesis, to questions about automata. The automata-theoretic approach has proven to be very versatile and fruitful.

Traditional automata accept or reject their input, and are therefore Boolean. A *weighted automaton* maps each word to a value from some semiring [13]. We focus on the *tropical* semiring. There, each transition of the automaton has a *cost* in $\mathbb{R}$, and the value of a run is the sum of the costs of the transitions taken along the run. A nondeterministic automaton $\mathcal{A}$ may have several runs on a word, and the *weight* of a word w in $\mathcal{A}$ is the value of the cheapest run on it. Applications of weighted automata over the tropical semiring include formal verification, where WFAs are used for the verification of quantitative properties [3, 4, 6, 9, 15] for reasoning about probabilistic systems [2], and for reasoning about the competitive ratio of on-line algorithms [1], as well as text, speech, and image processing, where the costs of the automaton are used in order to account for the variability of the data and to rank alternative hypotheses [5, 14].

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. This includes the problem of deciding whether a given nondeterministic weighted automaton can be determinized, and the problem of deciding whether the language of one automaton is contained (in the weighted sense) in the language of another automaton [8].

In the Boolean setting, the model of *alternating* automata has proven to be especially useful in the context of formal verification. While in a nondeterministic automaton the transition function specifies only existential requirements on the run, in an alternating automaton it specifies both existential and universal requirements. The universal requirements correspond to conjunctions in the specifications, making the translation of temporal-logic formulas to alternating automata simple and linear [7, 18], as opposed to the exponential translation to nondeterministic automata [19]. The linear translation of temporal logic to alternating automata is essential in automata-based algorithms for model checking of branching temporal logic formulas [12], and is useful for further minimization of the automata [16], for handling of incomplete information [11], for algorithms that avoid determinization [10], and more.

In the Boolean setting, the semantics of both disjunctions and conjunctions is straightforward. Recall that in a nondeterministic weighted automaton, the weight of a word is the value of the cheapest run on it. This meets our intuition of a "minimum semantics" for disjunctions in the weighted setting. It is less clear what the semantics of conjunctions should be. If we want to maintain the traditional helpful dualization between disjunctions and conjunctions, then an appropriate semantics for conjunction is a *maximum* semantics. The maximum semantics is also suitable for an analysis in which the weights correspond to a confidence or a truth-level indication. However, if the analysis is used in order to study the *cost* of the satisfaction, then an appropriate semantics is a *summation* semantics, in which the cost of satisfying a conjunction $\varphi_1 \wedge \varphi_2$ is not the maximal cost of satisfying φ_1 or φ_2, but rather the sum of these costs. Note that for the motivation of quantitative specifications, where one wants to replace Boolean satisfaction by a quantitative value that describes the quality of the satisfaction, both the maximum and the summation semantics are of interest. The two possible semantics for conjunctions in the weighted setting induce two different semantics for universal branches in alternating weighted automata.

We study alternating weighted automata on finite words, in both the maximum and summation semantics. We refer to the automata by MAX-AWAs and SUM-AWAs, respectively. We start with the max semantics. We study the expressive power of MAX-AWAs, their closure properties with respect to the operators min, max, and negation of weighted languages, the power of alternation with respect to nondeterminism, and arithmetics with MAX-AWAs. We also formalize the duality between the min semantics of existential transitions and the max semantics of universal transitions by means of a negation operator on weighted languages. Alternating automata with the max semantics are studied in [4]. The automata there are on infinite words.[1] The fact we work with finite words, where the value of a sequence of costs follows the tropical semiring, enables us to get a clear picture on the effect of adding alternation on top of nondeterminism. Indeed, in the case of infinite words, several max semantics are possible for an infinite sequence of costs (c.f., limit average, discounted sum, and more). This makes the setting more involved and yields a less uniform picture [4]. Nevertheless, the picture we obtain, and in particular the fact alternation cannot be removed, are similar to the general picture obtained for the different variants of the max semantics in the setting of infinite words.

We continue and study the sum semantics. A key difference between MAX-AWAs and SUM-AWAs is the fact that in the sum semantics the weight of a word may be exponentially larger than its length (even when all costs in the automaton are bounded by a constant). One immediate implication of this is that alternation cannot be removed in SUM-AWA. We also study closure properties for SUM-AWA, and the added expressive power of alternation even in languages in which the weight of a word does not go beyond its length.

An interesting feature of SUM-AWA is their ability to represent polynomials. We say an automaton $\mathcal{A}$ over a singleton alphabet $\{a\}$ represents a function $f : \mathbb{N} \setminus \{0\} \to \mathbb{R}$, if for all $n \in \mathbb{N} \setminus \{0\}$, we have that $L_{\mathcal{A}}(a^n) = f(n)$. We show that SUM-AWA (in fact, we even do not need nondeterminism) can represent all polynomials. Moreover, when the coefficients of the polynomial are non-negative, we can construct $\mathcal{A}$ so that it has only non-negative costs. It is interesting to compare these results with the fact that regular automata on finite words cannot recognize polynomials (for example, the language of all words of the form a^{n^2} is not regular).

Due to the lack of space, some proofs are omitted and can be found in the full version, in the authors' home pages.

[1] More work on weighted automata on infinite words, all in the nondeterministic setting, include different semantics of the value of a run (for example, in B-automata [9], the value depends on counters whose values are manipulated during the run), and the relation to quantitative variants of LTL or MSO [6].

2 Alternating Weighted Automata

For a finite alphabet Σ, a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$ is a finite sequence of letters from Σ. We use Σ^* to denote the set of all finite words over the alphabet Σ. A *nondeterministic finite weighted automaton* (NWA) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta \rangle$, where Σ is a finite non-empty alphabet, Q is a finite non-empty set of *states*, $Q_0 \subseteq Q$ is a non-empty set of *initial states*, and $\delta : Q \times \Sigma \to 2^{Q \times \mathbb{R}}$ is a *weighted transition function.* Intuitively, when the automaton is in state q and it reads the letter σ, it moves to state q' at *cost* c, for some $\langle q', c \rangle \in \delta(q, \sigma)$.

A *run* r of $\mathcal{A}$ on a finite word $w = \sigma_1 \cdots \sigma_n \in \Sigma^*$ is a sequence $r = (q_0, c_0), (q_1, c_1), \ldots, (q_n, c_n)$ of $n+1$ pairs in $Q \times \mathbb{R}$ such that $q_0 \in Q_0$, $c_0 = 0$, and for all $0 \leq i < n$, we have $(q_{i+1}, c_{i+1}) \in \delta(q_i, \sigma_{i+1})$. We associate the run r with the two sequences $S(r) = q_0, ..., q_n$ and $C(r) = c_0, ..., c_n$. We define $val(r) = c_0 + c_1 + ... + c_n$ to be the *value* of the run r. Thus, the costs of the transitions taken along the run are accumulated, and induce the value of the run.[2]

A weighted automaton $\mathcal{A}$ assigns weights to words in Σ^*. The weight of $w \in \Sigma^*$, denoted $L_{\mathcal{A}}(w)$, is the value of the cheapest run of $\mathcal{A}$ on w. Formally, $L_{\mathcal{A}}(w) = \min\{val(r) : r \text{ is a run of } \mathcal{A} \text{ on } w\}$. If there are no runs of $\mathcal{A}$ on w, then $L_{\mathcal{A}}(w)$ is undefined. The function $L_{\mathcal{A}}$ is called the *(weighted) language* of $\mathcal{A}$, and we say that $\mathcal{A}$ *recognizes* $L_{\mathcal{A}}$. We use $dom(L_{\mathcal{A}})$ to denote the domain of $L_{\mathcal{A}}$.

In an *alternating* automaton, the transition function may specify not only existential choices, but also *universal* ones. Below we define alternating weighted automata formally. For a given set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas **true** and **false**. For $Y \subseteq X$, we say that Y *satisfies* a formula $\theta \in \mathcal{B}^+(X)$ iff the truth assignment that assigns *true* to the members of Y and assigns *false* to the members of $X \setminus Y$ satisfies θ. The set Y *minimally satisfies* θ if Y satisfies θ and no set that is contained in Y satisfies θ. For example, the sets $\{x_1, x_3\}$ and $\{x_2, x_3\}$ both minimally satisfy the formula $(x_1 \vee x_2) \wedge x_3$, while the set $\{x_1, x_2\}$ does not satisfy this formula, and the set $\{x_1, x_2, x_3\}$ satisfies it but not minimally.

A *weighted alternating automaton* (AWA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta \rangle$, where Σ and Q are as in nondeterministic automata, $q_0 \in Q$ is an initial state[3] and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q \times \mathbb{R})$ is a weighted transition function. In order to define runs of alternating automata, we first have to define trees and weighted labeled

[2] Other common models for weighted automata include initial and final costs on states, transitions with an infinite cost, and accepting/non-accepting states. Our model here is simpler, yet our results can be easily extended to other models. Also, in general, an NWA may be defined with respect to any semiring $\langle \mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1} \rangle$. The value of a run is then the semiring product of the costs along the run. The weight of a 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*, $\langle \mathbb{R} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ (sometimes called the *tropical* semiring), as defined above.

[3] We could have assumed an initial foruma in $\mathcal{B}^+(Q)$ instead.

trees. A *tree* is a prefix closed set $T \subseteq \mathbb{N}^*$ (i.e., if $x \cdot c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$). The elements of T are called *nodes*. For every $x \in T$, the nodes $x \cdot c$, with $c \in \mathbb{N}$, are the *successors* of x. A node is a *leaf* if it has no successors. We sometimes refer to the length $|x|$ of x as its *level* in the tree. A *path* π of a tree T is a prefix-closed set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. A path is *full* if it contains a leaf.

An *edge* in T is a pair $\langle x, x \cdot c\rangle \in T \times T$. The set of edges in T is denoted $Edge(T)$. We sometimes refer to a path π as a sequence of edges. Then, we say that an edge $\langle x, x \cdot c\rangle$ is in π iff both x and $x \cdot c$ are in π. Given an alphabet Σ, a *weighted Σ-labeled tree* is a triple $\langle T, V, C\rangle$, where T is a tree, $V : T \to \Sigma$ maps each node of T to a letter in Σ, and $C : Edge(T) \to \mathbb{R}$ maps each edge of T to a cost in $\mathbb{R}$.

A run of a nondeterministic weighted automaton on a word can be thought of as a Q-labeled weighted tree with branching degree 1. Extending this notion, a run of an alternating automaton is a “real” Q-labeled weighted tree. Formally, given a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$, a run of $\mathcal{A}$ on w is a Q-labeled weighted tree $\tau = \langle T_r, r, \rho\rangle$, such that the following hold:

– $\varepsilon \in T_r$ and $r(\varepsilon) = q_0$.
– Consider a node $x \in T_r$ with $r(x) = q$ and $\delta(q, \sigma_{|x|+1}) = \theta$. There is a (possibly empty) set $S = \{(q_1, c_1), \ldots, (q_k, c_k)\} \subseteq 2^{Q \times \mathbb{R}}$ such that S minimally satisfies θ and for all $1 \leq d \leq k$, we have that $x \cdot d \in T_r$, $r(x \cdot d) = q_d$, and $\rho(x, x \cdot d) = c_d$.

For example, if $\delta(q_0, \sigma_1) = ((q_1, 2) \vee (q_2, 3)) \wedge ((q_3, 0) \vee (q_4, -2))$, then possible runs of $\mathcal{A}$ on σ_1 have a root labeled q_0, have one node in level 1 labeled q_1 (and the weight of the edge to it is 2) or q_2 (with edge weight 3), and have another node in level 1 labeled q_3 (with edge weight 0) or q_4 (with edge weight -2). Note that if $\theta = \textbf{true}$, then x need not have children. This is the reason why T_r may have leaves before level n. Also, since there exists no set S as required for $\theta = \textbf{false}$, we cannot have a run that takes a transition with $\theta = \textbf{false}$.

An AWA in which all the transitions are disjunctions is simply an NWA. An automaton in which all the transitions are conjunction is *universal*. An automaton that is both universal and nondeterministic (that is, $\delta(q, \sigma) \in Q \times \mathbb{R}$ for all q and σ) is *deterministic*.

As in the nondeterministic case, we want to define $L_{\mathcal{A}}$ to assign weights to words in Σ^*. To be consistent with the nondeterministic case, we want to define $L_{\mathcal{A}}(w) = \min\{(val(\tau)) : \tau \text{ is a run of } \mathcal{A} \text{ on } w\}$. If $\mathcal{A}$ does not have runs on w, then w is not in the domain of $L_{\mathcal{A}}$. For this definition to be complete, we need to define $val(\tau)$ for a run τ. As discussed in Section 1, we suggest and study two semantics for val. Let $\tau = \langle T, r, \rho\rangle$.

– In the *max* semantics, the value of every path in the tree is the sum of costs along the path, and the value of a run is the maximal value of a full path. Formally, MAX-$val(\tau) = \max\{\sum_{e \in \pi}(\rho(e)) : \pi \text{ is a path in } T\}$.

– In the *sum* semantics, the value of a run is the sum over all the costs in the edges of τ. Formally, $\text{SUM-}val(\tau) = \sum_{e \in Edge(T)} \rho(e)$.

Note that the two semantics are relevant only for universal transitions (that is, for transitions with $\wedge$). Note also that the sum semantics involves some non-trivial technical issues. To see this, note for example our requirement for the set S in the definition of a run tree to minimally satisfy the transition function. In the Boolean setting, as well as in the max semantics, we can remove the minimality requirement, as runs are monotonic: the more branches we have in the run tree, the more difficult it is for the run to be accepting or to have a minimal value. On the other hand, in the sum semantics, since $\mathcal{A}$ may have transitions with negative costs, sending more copies may actually reduce the value of a run. Moreover, even when all costs are positive, in the Boolean or the max semantics, it is clear that we have no reason to have multiple copies of the same state in the same level of the run tree. This is why S is a set of states, and no multiple occurrences of the same state are possible. In the sum semantics, one could argue that such multiple occurrences should be allowed, as they reflect the fact that some mission has to be fulfilled (and payed for) several times. We preferred not to proceed with this multiple-occurrence semantics, as it can be simulated by our sum semantics (by duplicating states).

We abbreviate the different types of automata by acronyms in $\{\text{MAX-}, \text{SUM-}\} \times \{A, U, N, D\} \times \{WA\}$. For example, MAX-UWA refers to a universal weighted automaton in the max semantics. For nondeterministic or deterministic automata we omit the semantics prefix and simply use NWA and DWA, respectively.

For two weighted languages L_1 and L_2, and $c \in \mathbb{R}$, we use $-L_1$, $L_1 + L_2$, $\max\{L_1, L_2\}$, and $c \cdot L_1$ to denote the weighted languages that negate L_1, sum L_1 and L_2, take their maximum, and multiply L_1 by c. Thus, for every word $w \in \Sigma^*$, we have that $(-L_1)(w) = -L_1(w)$, $(L_1 + L_2)(w) = L_1(w) + L_2(w)$, $\max\{L_1, L_2\}(w) = \max\{L_1(w), L_2(w)\}$, and $(c \cdot L_1)(w) = c \cdot L_1(w)$. Note that $(L_1 + L_2)(w)$ and $\max\{L_1, L_2\}(w)$ are defined only if both $L_1(w)$ and $L_2(w)$ are defined.

We say that two AWAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent* if for all $w \in \Sigma^*$ it holds that $L_{\mathcal{A}_1}(w) = L_{\mathcal{A}_2}(w)$. For two classes of automata γ_1 and γ_2, we say that γ_2 is *more expressive* than γ_1, denoted $\gamma_1 \leq \gamma_2$, if every language L that is recognized by an automaton in γ_1 can also be recognized by an automaton in γ_2. We also use the notations $\gamma_1 \not\leq \gamma_2$, $\gamma_1 < \gamma_2$, and $\gamma_1 \neq \gamma_2$, derived as expected from $\gamma_1 \leq \gamma_2$.

3 The Max Semantics

In this section we study the max semantics. As we mentioned above, one motivation for the max semantics is the duality with the min semantics of nondeterminism. We first formalize this duality and then study other properties of the max semantics.

3.1 The duality between the min and the max semantics

For a formula $\theta \in \mathcal{B}^+(Q \times \mathbb{R})$, let $\widetilde{\theta}$ be the formula obtained from θ by switching $\wedge$'s and $\vee$'s, switching **true**'s and **false**'s, and negating all the costs in the atoms of θ. If, for example, $\theta = (p, 3) \vee (\textbf{true} \wedge (q, -5))$, then $\widetilde{\theta} = (p, -3) \wedge (\textbf{false} \vee (q, 5))$. For a transition function δ, let $\widetilde{\delta}$ be the transition function obtained from by dualizing δ. That is, for all $q \in Q$ and $\sigma \in \Sigma$, we have that $\widetilde{\delta}(q, \sigma) = \widetilde{\delta(q, \sigma)}$. Given an AWA $\mathcal{A} = \langle Q, \Sigma, q_0, \delta \rangle$, its *dual* AWA is $\widetilde{\mathcal{A}} = \langle Q, \Sigma, q_0, \widetilde{\delta} \rangle$.

Dualizing an alternating automaton in the unweighted setting complements the language of the automaton. Intuitively, it follows from the fact dualization amounts to switching the roles between the two players in the two-player game that the automaton models. In the case of MAX-AWAs, dualization is more involved and corresponds to negating the language of the automaton. Formally, we have the following.

Lemma 1. *Let $\mathcal{A}$ be a* MAX-AWA*. Then, $L(\widetilde{\mathcal{A}}) = -L(\mathcal{A})$.*

Proof. We first prove that for every word $w \in \Sigma^*$, we have that $L_{\widetilde{\mathcal{A}}}(w)$ is the maximal value of a minimal path in a run of $-\mathcal{A}$ on w, where $-\mathcal{A}$ is the MAX-AWA obtained from $\mathcal{A}$ by multiplying all the costs by -1. Proofs of duality claims such as this are usually technical. We give the main idea of the proof. The value of a word w in $L_{\mathcal{A}}$ can be thought of as the outcome of the following two-player game. The set of states in the game is Q. The game starts in the state q_0. In every round, Player 1 (the *maximizer*) chooses a set $E \subseteq Q$ that satisfies $\delta(q_i, w_i)$. Player 2 (the *minimizer*) then chooses a state $q_{i+1} \in E$, and the game continues in the same manner from q_{i+1}, reading w_{i+1}, and so on. The game ends when the last letter in w is read, and the value of the game is the sum of values along the selected transitions. The goal of Player 1 is to maximize the value, and the goal of Player 2 is to minimize it.

When the same game is played on $\widetilde{\mathcal{A}}$, the roles of the players are interchanged. Thus, Player 1 is now the minimizer, and Player 2 is the maximizer. The path induced by this game corresponds to a minimal path in a maximal run of $\mathcal{A}$ on w. Indeed, Player 1 determines which path is taken in every run of $\mathcal{A}$, and Player 2 determines which run is taken. This implies that the value of every $w \in \Sigma^*$ in $L_{\widetilde{\mathcal{A}}}$ is the value of the minimal path in a maximal run of $\mathcal{A}$ on w.

Now, for every word $w \in \Sigma^*$, we have that $L_{\widetilde{\mathcal{A}}}(w) = \max\{\min\{\sum_{e \in \pi}(-\rho(e)) : \pi \text{ is a path in } \tau\} : \tau \text{ is a run of } \mathcal{A} \text{ on } w\} = \max\{\min\{-\sum_{e \in \pi}(\rho(e)) : \pi \text{ is a path in } \tau\} : \tau \text{ is a run of } \mathcal{A} \text{ on } w\} = \max\{-\max\{\sum_{e \in \pi}(\rho(e)) : \pi \text{ is a path in } \tau\} : \tau \text{ is a run of } \mathcal{A} \text{ on } w\} = -\min\{\max\{\sum_{e \in \pi}(\rho(e)) : \pi \text{ is a path in } \tau\} : \tau \text{ is a run of } \mathcal{A} \text{ on } w\} = -L_{\mathcal{A}}(w)$.

A special case of Lemma 1 is when the automaton $\mathcal{A}$ is an NWA, in which case $\widetilde{\mathcal{A}}$ is a MAX-UWA. We then have the following.

Corollary 1. *A weighted language L is recognizable by an* NWA *iff $-L$ is recognizable by a* MAX-UWA*.*

3.2 Expressive power

In the Boolean setting, natural questions to ask in the context of expressive power include closure to Boolean operators, and the added power of each of the branching modes. It is well known that in the Boolean setting, alternation and nondeterminism do not add to the expressive power of deterministic automata, and the latter are closed under union, intersection, and complementation. In the weighted setting, the operators that correspond to the Boolean ones are min, max, and negation. As we have seen above, MAX-AWAs are closed under negation. As in the Boolean setting, closure under min and max is easy, and follows from the semantics of the transitions of MAX-AWAs. Likewise, MAX-UWAs are closed under max.

In the Boolean setting, nondeterministic automata are closed under intersection. Indeed, the "product construction" enables us to trace several automata in parallel. Moreover, the "subset construction" enables us to trace even an unbounded type of intersection. Consequently, in the Boolean setting, alternation can be removed. We now turn to study the added expressive power of universal branches in the max semantics. We show that NWAs are not closed under max, and conclude that alternation in MAX-AWAs cannot be removed.

Let $\Sigma = \{a, b\}$. For $\sigma \in \Sigma$, let L_σ be the language that maps $w \in \Sigma^*$ to the number of occurrences of σ in w. In [4], the languages L_a and L_b are used in order to show that DWAs are not closed under min. Here we follow similar ideas and use them in the study of closure under max.

Theorem 1. NWAs *and* DWAs *are not closed under* max*.*

Proof. Consider the language $L = \max\{L_a, L_b\}$. The language L_a can be defined by a DWA with a single self loop that has cost 1 to a and cost 0 to b, and similarly for L_b. In the full version, we prove that no NWA can recognizes L. Essentially, it follows from the fact that all the reachable a-cycles in an NWA for L must have a strictly positive cost, which implies that runs on the word $a^{n+1}b^{n+1}$, where n is the number of states in the NWA, suggest runs with value strictly smaller than $n + 1$ to words of the form $a^j b^{n+1}$, for $j < n + 1$.

Since MAX-UWAs dualize NWAs, we can dualize Theorem 1 as follows.

Theorem 2. MAX-UWAs *and* DWAs *are not closed under* min*.*

Proof. We start with MAX-UWAs. Assume by way of contradiction that $\mathcal{A}$ is a MAX-UWA that recognizes $L = \min\{L_a, L_b\}$. From Lemma 1, we get that $\widetilde{\mathcal{A}}$ is an NWA such that $L_{\widetilde{\mathcal{A}}} = -L_{\mathcal{A}}$. Consider the NWA $\mathcal{B}$ obtained from $\widetilde{\mathcal{A}}$ by adding 1 to the cost of every transition. It is easy to verify that for every word w, we have that $L_{\mathcal{B}}(w) = |w| - L_{\mathcal{A}}(w)$. On the other hand, $\max\{L_a(w), L_b(w)\} = |w| - \min\{L_a(w), L_b(w)\} = |w| - L_{\mathcal{A}}(w)$. It follows that $\mathcal{B}$ is an NWA that recognizes $\max\{L_a, L_b\}$, which contradicts the proof of Theorem 1. Finally, observe that if $\mathcal{A}$ is a DWA, then $\mathcal{B}$ is a DWA as well, again contradicting the proof of Theorem 1.

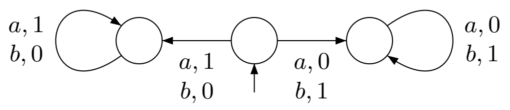

Fig. 1. A MAX-UWA for $\max\{L_a, L_b\}$.

Since MAX-UWAs are closed under max (in particular, Fig. 1 describes a MAX-UWA for $\max\{L_a, L_b\}$), Theorem 1 implies that alternation cannot be removed in the weighted setting with the max semantics. Combining this with dualization, we conclude with the following.

Corollary 2. MAX-UWA $\neq$ NWA*,* MAX-AWA $>$ NWA*, and* MAX-AWA $>$ MAX-UWA*.*

3.3 Arithmetics

In the weighted setting, additional interesting properties of automata are related to the fact they manipulate real values. In this section we consider the closure of max automata under addition and multiplication by a scalar.

Theorem 3. DWAs*,* NWAs*,* MAX-UWAs*, and* MAX-AWAs *are closed under addition.*

Proof. The proof is by construction: Given automata $\mathcal{A}_1$ and $\mathcal{A}_2$, we construct an automaton $\mathcal{A}$ of the same class such that $L_{\mathcal{A}} = L_{\mathcal{A}_1} + L_{\mathcal{A}_2}$. We describe the construction in detail in the full version. Essentially, the state space of $\mathcal{A}$ is the product of the state spaces of $\mathcal{A}_1$ and $\mathcal{A}_2$, and the transitions from $\langle q_1, q_2\rangle$ sum the corresponding transitions from q_1 and q_2.

Next, we consider scalar multiplication. Clearly, multiplying all the costs of an AWA by a scalar c causes the value of every run to be multiplied by c (since the costs are summed along a path in the run tree). If $c \geq 0$, then multiplying by c is a monotonic (increasing) function. That is, if r_1 and r_2 are two runs, and $val(r_1) \leq val(r_2)$, then $c \cdot val(r_1) \leq c \cdot val(r_2)$. Therefore, the cheapest run stays the cheapest run, implying the following theorem.

Theorem 4. MAX-AWAs*,* MAX-UWAs*,* NWAs *and* DWAs *are closed under multiplication by a positive scalar.*

The case of a negative scalar is different, as multiplication is an anti-monotonic (decreasing) function. Dualization, together with Th. 4, imply that MAX-AWAs are closed under multiplication by a negative scalar. For the other classes, the fact that such a multiplication causes the cheapest run to become the most expensive run is crucial:

Theorem 5. NWAs *and* MAX-UWAs *are not closed under multiplication by a negative scalar.*

Proof. In the proof of Theorem 1, we saw that there is no NWA for $\max\{L_a, L_b\}$. On the other hand, there is an NWA $\mathcal{A}$ for $\min\{L_a, L_b\}$. Assume by way of contradiction that NWAs are closed under multiplication by a negative scalar. Then, by multiplying $\mathcal{A}$ by -1 we can obtain an NWA $\mathcal{A}'$ for $-\min\{L_a, L_b\}$. For every word w, we have that $\max\{L_a(w), L_b(w)\} = |w| - \min\{L_a(w), L_b(w)\}$. It is easy to construct an automaton $\mathcal{B}$ the recognizes the language $L(w) = |w|$. By Lemma 3, NWAs are closed under addition. By adding $\mathcal{A}'$ and $\mathcal{B}$ we obtain an NWA $\mathcal{C}$ such that $L_{\mathcal{C}}(w) = |w| - \min\{L_a(w), L_b(w)\} = \max\{L_a(w), L_b(w)\}$. So, $\mathcal{C}$ is an NWA that recognizes $\max\{L_a, L_b\}$, contradicting the fact that no such NWA exists. The proof for MAX-UWAs is dual.

4 The Sum Semantics

Recall that in a SUM-AWA $\mathcal{A}$, the value of a run is the sum of all the costs in the run tree. This corresponds to the intuition that $\mathcal{A}$ spawns copies that fulfill different tasks and has to pay for the costs of all tasks. The first observation we make about SUM-AWAs, which makes them an interesting arithmetic tool, is that they can recognize languages in which the weight of a word is not linear in its length.

4.1 Exponential weights

In NWAs, and in fact even in MAX-AWAs, the value of a run of $\mathcal{A}$ on a word w of length n is bounded from above by $c_{\max} \cdot n$, where $c_{\max}$ is the maximal cost in $\mathcal{A}$. Indeed, even if several copies of the automaton run on the same prefix of the word, only one copy contributes to the value of the run, which is therefore linear in the length of the word. When we allow summing over all the weights in a tree, this is no longer true, and the weight of a word may become exponential. Consider for example the automaton $\mathcal{A}_{\exp}$ in Fig. 2 (we draw UWAs the same way we draw NWAs. but keep in mind that in UWAs, transitions with the same label are conjunctively related). The single run tree of $\mathcal{A}_{\exp}$ on the word a^n is a complete binary tree of depth n. Each level doubles the number accumulated so far (starting with 2), and so $L_{\mathcal{A}_{\exp}}(a^n) = 2^n$.

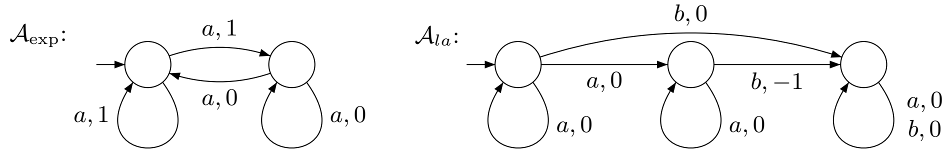

Fig. 2. The SUM-UWAs $\mathcal{A}_{exp}$ and $\mathcal{A}_{la}$.

Since the number of edges in a run tree on a word w is bounded by $d^{|w|+1}$, where d is the branching degree of the tree, and d is bounded by the number of states of the automaton, we cannot go beyond an exponential weight, either positive of negative:

Lemma 2. *Let $\mathcal{A}$ be a* SUM-AWA *with n states. For every word $w \in \Sigma^*$, we have that* $\min\{0, c_{min} \cdot n^{|w|+1}\} \leq L_{\mathcal{A}}(w) \leq c_{\max} \cdot n^{|w|+1}$, *where c_{min} and $c_{\max}$ are the minimal and maximal costs in $\mathcal{A}$, respectively.*

4.2 Expressive power

One could argue that it is not "fair" to compare SUM-AWAs with NWA, due to the ability of the first to assign super-linear weights. As we show now, alternation cannot be removed even if we restrict attention to SUM-UWAs with linear-bounded language. To see this, consider the "leading a's" SUM-UWA $\mathcal{A}_{la}$ in Fig. 2. One can verify that for all $w \in \{a,b\}^*$, we have that

$$L_{\mathcal{A}_{la}}(w) = \begin{cases} -k & \text{if } w \in a^k \cdot b \cdot (a+b)^* \text{ for } k \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

It is not hard to prove that there is no NWA for $L_{\mathcal{A}_{la}}$. Thus, the power of SUM-AWAs goes beyond the ability to assign super-linear weights.

We now turn to study closure properties for SUM-AWAs. As in the Boolean setting, closure under min and summation is straightforward, as they correspond to existential and universal transitions. Also, as has been the case in the max semantics, closure under multiplication by a positive scalar c is easy, as we only have to multiply the costs of $\mathcal{A}$ by c. Such a multiplication would work also with a negative scalar in case the automaton is universal. Indeed, all the costs in the single run are multiplied by the scalar. The general case, of a negative scalar and SUM-AWAs is less clear, and is related to the problem of closure under max. For this problem, we have examples to the lack of closure for following two fragments of SUM-AWAs.

Theorem 6. SUM-UWAs *and* SUM-AWAs *with non-negative costs are not closed under* max.

Proof. Consider the language $L = \max\{L_a, L_b\}$. Recall that L_a and L_b are recognized by DWAs with non-negative costs, which are a special case of SUM-AWAs. We prove that there is no SUM-UWA nor SUM-AWA with non-negative costs for L.

Consider a SUM-UWA $\mathcal{A}$ with n states, and consider the run τ of $\mathcal{A}$ on $w = a^{n+1}b^{n+1}$. Each level of τ consists of states of $\mathcal{A}$, where every state appears 0 or more times. We characterize the levels by vectors in $\mathbb{N}^Q$, which we represent as $\mathbb{N}^n$. For example, the vector $(2,0,1)$ means that in this level there are 2 copies of q_0, 1 copy of q_2 and no copies of q_1. Thus, τ can be thought of as a sequence of vectors. We say that a vector β is a *configuration* of $\mathcal{A}$.

Let β be a configuration, and consider what $\mathcal{A}$ does when it reads b. Every state $q_i \in Q$ sends out β_i copies of states in $\delta(q_i, b)$. That is, there are $d_1^i, ..., d_n^i$ such that q_i moves to the vector $(d_1^i, ..., d_n^i)$. Intuitively, q_i sends d_j^i copies of q_j. Let D be the $\mathbb{N}_{n\times n}$ matrix whose entries are $D_{ij} = d_j^i$. It is easy to verify that when $\mathcal{A}$ reads b in configuration β, it moves to the vector $D\beta$ (where β is considered as a column vector). Furthermore, when $\mathcal{A}$ is in configuration β and reads b, every state q_i accumulates w_i cost in the transition. Thus, the total cost accumulated in the transition is $\sum_{i=1}^n \beta_i w_i = \langle \beta, w \rangle$ (where $w = (w_1, ..., w_n)$ and $\langle \cdot, \cdot \rangle$ denotes the standard inner product).

Let $\alpha_0, ..., \alpha_n, \beta_0, ..., \beta_n$ be the levels of τ. For all $0 < i \leq n$ it holds that $\beta_i = D\beta_{i-1}$. Observe that $\beta_0, ..., \beta_n$ are $n+1$ vectors in $\mathbb{Q}^n$. Thus, there exists an index $1 \leq k \leq n$ such that β_k is linearly dependent on $\beta_0, ..., \beta_{k-1}$, so we can write $\beta_k = \sum_{j=0}^{k-1} c_j \beta_j$ for some $c_0, ..., c_{k-1} \in \mathbb{Q}$. Consider what happens when $\mathcal{A}$ reads b from β_k. The new configuration is

$$D\beta_k = D\sum_{j=0}^{k-1} c_k \beta_j = \sum_{j=0}^{k-1} c_j D\beta_j = \sum_{j=0}^{k-1} c_j \beta_{j+1} = \sum_{j=1}^{k-1} c_{j-1}\beta_j + c_{k-1}\sum_{j=0}^{k-1} c_k \beta_j$$

In particular, β_{k+1} is also linearly dependent on $\beta_0, ..., \beta_{k-1}$. It is easy to prove by induction that for all $t \geq k$ it holds that β_t is linearly dependent on $\beta_0, ..., \beta_{k-1}$.

Next, observe that since $\mathcal{A}$ has a single run, then the run must accumulate the cost $n+1$ by configuration α_{n+1}, and must stay 0 through $\beta_0, ..., \beta_{n+1}$. Let $w = (w_0, ..., w_n)$ denote the cost vector accumulated at a b-transition, then for all $0 \leq i \leq n+1$ it holds that $\langle \beta_i, w \rangle = 0$. Let $\gamma = \sum_{j=0}^{k-1} e_j \beta_j$ for $e_1, ..., e_{k-1} \in \mathbb{Q}$, then the cost accumulated from γ when reading b is

$$\langle \gamma, w \rangle = \sum_{j=0}^{k-1} e_j \langle \beta_j, w \rangle = \sum_{j=0}^{k-1} e_j \cdot 0 = 0$$

We conclude that when $\mathcal{A}$ reads to $a^{n+1}b^{n+2}$ it accumulates cost 0 in all the transitions on the b-block. Hence, the weight assigned by $\mathcal{A}$ to $a^{n+1}b^{n+2}$ is $n+1$, which is a contradiction.

We proceed to SUM-AWA with non-negative costs. Let $\mathcal{A}$ be a SUM-AWA with n states and non-negative costs. Consider the cheapest run τ of $\mathcal{A}$ on $w = a^{2^n+1}b^{2^n+1}$. Since every prefix of this run is also a run of $\mathcal{A}$ on a prefix of w, we get that after reading a^{2^n+1}, the accumulated cost must be exactly $2^n + 1$. Indeed, a lower cost would imply that the value of a^{2^n+1} in $\mathcal{A}$ is less than $2^n + 1$, and, since accumulated costs are nonnegative, a higher cost implies that the weight of w is greater than $2^n + 1$. As we show in the full version, it follows that we can pump w to a word $w' = a^{2^n+1}b^k$, with $k > 2^n + 1$, such that $\mathcal{A}$ assigns to w' a weight smaller than k.

Since SUM-AWAs are closed under multiplication by a negative scalar, the fact they cannot recognize $\max\{L_a, L_b\}$ implies they also cannot recognize $\min\{L_a, L_b\}$, which can be recognized by an NWA.[4] Thus, keeping in mind the ability of SUM-AWA to assign super-linear weights, we can conclude with the following.

Corollary 3. SUM-UWA $\neq$ NWA, SUM-AWA $>$ *NWA, and* SUM-AWA $>$ SUM-UWA.

[4] A popular example for the added power of NWA with respect to DWA is the language that maps $a \cdot b^i \cdot c$ to i and $a \cdot b^i \cdot d$ to $2i$ [13]. Interestingly, this language can be recognized by a SUM-UWA.

4.3 SUM-UWA and polynomials

For an automaton $\mathcal{A}$ over a singleton alphabet $\{a\}$ and a function $f : \mathbb{N} \setminus \{0\} \to \mathbb{R}$, we say that $\mathcal{A}$ *represents* f if for all $n \in \mathbb{N} \setminus \{0\}$, we have that $L_{\mathcal{A}}(a^n) = f(n)$. In this section, we study the presentation of polynomials by SUM-UWAs. Note that this study is not interesting in the context of NWAs or MAX-AWAs, as the latter can only represent linear functions.

Since we restrict attention to the alphabet $\{a\}$, all the words are of the form a^n for some n, and we abbreviate $L_{\mathcal{A}}(a^n)$ by $L_{\mathcal{A}}(n)$. Observe that for every AWA $\mathcal{A}$ we have $L_{\mathcal{A}}(0) = 0$, which is why we consider the domain $\mathbb{N} \setminus \{0\}$. By adding ϵ-transitions, we can easily extend the results to functions $f : \mathbb{N} \to \mathbb{R}$.

We start by describing our basic "building blocks". Consider the sequence of SUM-UWAs $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots$ appearing in Fig. 3. It is easy to see that $\mathcal{A}_1$ defines the polynomial $f_1(n) = n$. As for $\mathcal{A}_d$, for $d \geq 2$, note that a run of $\mathcal{A}_d$ on a^n sends a copy of $\mathcal{A}_d$ and a copy of $\mathcal{A}_{d-1}$ to read a^{n-1}. Accordingly, a copy of $\mathcal{A}_{d-1}$ is sent to read a^i for all $0 \leq i \leq n-1$. Thus, $L_{A_d}(n) = \sum_{i=0}^{n} L_{\mathcal{A}_{d-1}}(n-i) = \sum_{i=0}^{n-1} L_{\mathcal{A}_{d-1}}(i)$. Let $f_d(n) = L_{\mathcal{A}_d}(n)$.

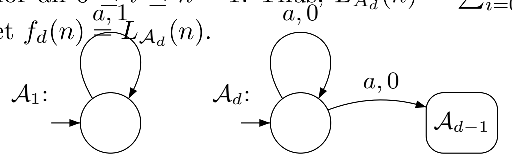

Fig. 3. The automata $\mathcal{A}_d$.

Lemma 3. *For all $n \geq 1$ and $d \geq 1$, we have that* $f_d(n) = \begin{cases} \binom{n}{d} & \text{if } d \leq n \\ 0 & \text{otherwise.} \end{cases}$

Since $\binom{n}{d}$ is a polynomial of degree d, the set $\{\mathcal{A}_1, ..., \mathcal{A}_d\}$ of SUM-UWAs represents a set of polynomials of degrees $\{1, ..., d\}$. It is easy to construct a DWA $\mathcal{A}_0$ such that $L_{\mathcal{A}_0}(n) = 1$. Thus, the set $\{\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_d\}$ represents a set of polynomials with degrees $\{0, 1, ..., d\}$. We claim that this set allows us to represent every polynomial. To prove this, we use the following lemma (the proof is a basic exercise in linear algebra).

Lemma 4. *Let $\{p_0, ..., p_d\}$ be polynomials over a commutative ring $\mathcal{R}$ such that p_i is of degree i. The set $\{p_0, ..., p_d\}$ forms a basis to the space of polynomials of degree at most d.*

By Lemma 4, and by the closure of SUM-UWAs under addition and scalar multiplication, we conclude that SUM-UWAs can span the entire space of polynomials of degree at most d. Note that according to Lemma 4, spanning the space requires coefficients from the ring $\mathcal{R}$. On the positive side, in order to span a polynomial with coefficients in $\mathbb{Z}$ or $\mathbb{Q}$, we only need costs in $\mathbb{Z}$ or $\mathbb{Q}$, respectively. On the negative side, this implies that for coefficients in $\mathbb{N}$ we may need costs from $\mathbb{Z}$ (which is the minimal ring containing $\mathbb{N}$). Note that while we only proved existence, it is easy to make the proof explicit and construct, given a polynomial p, a SUM-UWA that represents it. In order to do it, one simply multiplies $\mathcal{A}_d$ by the desired coefficient for n^d, then multiplies $\mathcal{A}_{d-1}$ by a proper scalar so as to fix the coefficient for n^{d-1}, and so on.

The fact that in order to represent a polynomial with coefficients in $\mathbb{N}$ we may need negative costs is disturbing. We now proceed to show that it is indeed not necessary: for every polynomial p with non-negative coefficients, we can construct a SUM-UWA $\mathcal{A}_p$ with non-negative costs such that $L_{\mathcal{A}_p}(n) = p(n)$. We also give an explicit construction of $\mathcal{A}_p$. As above, it is enough to show that we can construct a SUM-UWA for every monomial n^d. Recall that $f_d(n)$ is $\binom{n}{d}$ for $d \leq n$, and is 0 otherwise. The following lemma shows how to define n^d using $\{f_1(n), ..., f_d(n)\}$. It is a well known result and we give the proof in the full version. The lemma and the proof refer to $S(n,k)$ – the *Stirling number of the second kind*, which is the number of ways to partition a set of size n into k subsets.

Lemma 5. $n^d = \sum_{k=1}^{d} k!S(d,k)f_k(n)$.

By Lemma 5, we can define n^d using $\{f_1(n), ..., f_d(n)\}$, and the coefficients are non-negative. By adding $1 = n^0$ to the set $\{n^1, ..., n^d\}$ we conclude that we can span any polynomial with non-negative coefficients using SUM-UWAs with non-negative costs. Moreover, we observe that all our constructions are of SUM-UWAs with self loops only. This follows from the definition of the basic blocks and the constructions proving the closure of SUM-UWA to addition and multiplication by a scalar, which preserve this property.

We can thus conclude with the following (the result about non-positive coefficients follows from the closure under multiplication by a negative scalar).

Theorem 7. *For every polynomial p with coefficients in a ring $\mathcal{R} \in \{\mathbb{R}, \mathbb{Q}, \mathbb{Z}\}$, we can construct a* SUM*-UWA $\mathcal{A}$ that defines p (with cost in $\mathbb{R}$,$\mathbb{Q}$ or $\mathbb{Z}$, respectively). Moreover, if the coefficients of p are all non-negative (non-positive), then we can construct $\mathcal{A}$ with non-negative (non-positive, resp.) costs only.*

5 Discussion

We introduced and studied two semantics – max and sum, for universal transitions of alternating weighted automata on finite words. The two semantics correspond to different interpretations of conjunctions in a weighted setting, and are of interest in quantitative formal reasoning. We showed that in both semantics, alternation strictly increases the expressive power of the automata. Also, in the sum semantics, it enables the automaton to represent super-linear functions, making universal transitions more significant then their dual nondeterministic transitions.

We plan to continue our study in several directions. In the theoretical front, we find the ability to represent polynomial by automata very interesting. In particular, it is interesting to see which operations on polynomials can be performed on the SUM-AWAs that represent them. Our results here already include addition of two polynomials and their multiplication by a scalar. It is a nice exercise to see that given a SUM-AWA that represents a polynomial p, we can easily construct a SUM-AWA for the derivative of p, or its integration. More challenging are constructions that correspond to multiplication of polynomials, decision procedures about their roots, and so on. In the more practical front, we are developing a weighted version of LTL that can specify quality of satisfaction. Conjunctions in the new logic can be interpreted in both semantics, and the translation to AWAs is a basic procedure in reasoning about specifications in the logic. Problems like membership (that is, finding weights) are easily decidable, and we are studying fragments for which language containment (that is, its weighted variant) is decidable. We are also studying an extension to automata on infinite words.

References

1. B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.
2. 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.
3. K. Chatterjee, L. Doyen, and T. Henzinger. Quantative languages. In *Proc. 17th CSL*, pages 385–400, 2008.
4. K. Chatterjee, L. Doyen, and T. Henzinger. Alternating weighted automata. In *Proc. 17th FCS*, LNCS 5699, pages 3–13, 2009.
5. K. Culik and J. Kari. Digital images and formal languages. *Handbook of formal languages, vol. 3: beyond words*, pages 599–616, 1997.
6. M. Droste and P. Gastin. Weighted automata and weighted logics. In *Proc. 32nd ICALP*, pages 513–525, 2005.
7. E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd FOCS*, pages 368–377, 1991.
8. D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Int. J. of Algebra and Computation*, 4(3):405–425, 1994.
9. D. Kuperberg. Linear temporal logic for regular cost functions. In *Proc. 28th STACS*, pages 627–636, 2011.
10. O. Kupferman. Avoiding determinization. In *Proc. 21st LICS*, pages 243–254, 2006.
11. O. Kupferman and M.Y. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, 2000.
12. O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
13. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
14. 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.
15. M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.
16. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proc. 12th CAV*, LNCS 1855, pages 248–263, 2000.
17. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.
18. M.Y. Vardi. Nontraditional applications of automata theory. In *Proc. 11th STACS*, LNCS 789, pages 575–597, 1994.
19. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.