Nested Weighted Limit-Average Automata of Bounded Width*

Krishnendu Chatterjee[1], Thomas A. Henzinger[2], and Jan Otop[3]

1 **IST Austria**
krish.chat@ist.ac.at
2 **IST Austria**
tah@ist.ac.at
3 **University of Wrocław**
jotop@cs.uni.wroc.pl

Abstract

While weighted automata provide a natural framework to express quantitative properties, many basic properties like average response time cannot be expressed with weighted automata. Nested weighted automata extend weighted automata and consist of a master automaton and a set of slave automata that are invoked by the master automaton. Nested weighted automata are strictly more expressive than weighted automata (e.g., average response time can be expressed with nested weighted automata), but the basic decision questions have higher complexity (e.g., for deterministic automata, the emptiness question for nested weighted automata is PSpace-hard, whereas the corresponding complexity for weighted automata is PTime). We consider a natural subclass of nested weighted automata where at any point at most a bounded number k of slave automata can be active. We focus on automata whose master value function is the limit average. We show that these nested weighted automata with bounded width are strictly more expressive than weighted automata (e.g., average response time with no overlapping requests can be expressed with bound $k = 1$, but not with non-nested weighted automata). We show that the complexity of the basic decision problems (i.e., emptiness and universality) for the subclass with k constant matches the complexity for weighted automata. Moreover, when k is part of the input given in unary we establish PSpace-completeness.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases weighted automata; nested weighted automata; complexity; mean-payoff

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.24

1 Introduction

Traditional to quantitative verification. In contrast to the traditional view of formal verification that focuses on Boolean properties of systems, such as "every request is eventually granted", quantitative specifications consider properties like "the long-run average success rate of an operation is at least one half" or "the long-run average response time is below a threshold." Such properties are crucial for performance related properties, for resource-constrained systems, such as embedded systems, and significant attention has been devoted to them [21, 14, 13, 22, 2].

* This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), ERC Start grant (279307: Graph Games), Vienna Science and Technology Fund (WWTF) through project ICT15-003 and by the National Science Centre (NCN), Poland under grant 2014/15/D/ST6/04543.

licensed under Creative Commons License CC-BY

Weighted automata. A classical model to express quantitative properties is *weighted automata* that extends finite automata where every transition is assigned a rational number called a *weight*. Each run results in a sequence of weights, and a *value function* aggregates the sequence into a single value. For non-deterministic weighted automata, the value of a word is the infimum value of all runs over the word. Weighted automata provide a natural and flexible framework to express quantitative[1] properties [14]. Weighted automata have been studied over finite words with weights from a semiring [21], and extended to infinite words with limit averaging or supremum as a value function [14, 13, 12]. While weighted automata over semirings can express several quantitative properties [27], they cannot express long-run average properties that weighted automata with limit averaging can [14]. However, even weighted automata with limit averaging cannot express the basic quantitative property of average response time [16, Example 5].

Nested weighted automata. To express properties like average response time, weighted automata were extended to *nested weighted automata (NWA)* [16]. An NWA consists of a master automaton and a set of slave automata. The master automaton runs over infinite input words. At every transition the master automaton can invoke a slave automaton that runs over a finite subword of the infinite word, starting at the position where the slave automaton is invoked. Each slave automaton terminates after a finite number of steps and returns a value to the master automaton. Each slave automaton is equipped with a value function for finite words, and the master automaton aggregates the returned values from slave automata using a value function for infinite words. For Boolean finite automata, nested automata are as expressive as the non-nested counterpart, whereas NWA are strictly more expressive than non-nested weighted automata [16]. It has been shown in [16] that NWA provide a specification framework where many basic quantitative properties, which cannot be expressed by weighted automata, can be expressed easily, and they provide a natural framework to study quantitative run-time verification.

The basic decision questions. We consider the basic automata-theoretic decision questions of emptiness and universality. The importance of these basic questions in the weighted automata setting is as follows: (1) Consider a system modeled by a finite-automaton recognizing traces of the system and a quantitative property given as a weighted automaton or an NWA. Then whether the worst-case (resp., best-case) behavior has the value at least λ is the emptiness (resp., universality) question on the product. (2) Problems related to model measuring (that generalizes model checking) and model repair also reduce to the emptiness problem [25, 16].

Complexity gap. In this work we focus on the following classical value functions: LimAvg for infinite words, which is the long-run average property; and $\textsc{Sum}, \textsc{Sum}^+$ (where $\textsc{Sum}^+$ is the sum of absolute values) for finite words. While NWA are strictly more expressive than weighted automata, the complexity of the decision questions are either unknown or considerably higher. Table 1 (non-bold-faced results) summarizes the existing results for weighted automata [14] and NWA [16], for example, for NWA for $\textsc{Sum}^+$ the known bounds are ExpSpace and PSpace-hard, and for Sum even the decidability of the basic decision

[1] We use the term "quantitative" in a non-probabilistic sense, which assigns a quantitative value to each infinite run of a system, representing long-run average or maximal response time, or power consumption, or the like, rather than taking a probabilistic average over different runs.

Table 1 Decidability and complexity of emptiness and universality for weighted and nested weighted automata with LimAvg value function and Sum and Sum^+ value function for slave automata. Our results are bold faced. Moreover all PTime results become NLogSpace-complete when the weights are specified in unary.

	Deterministic (Emptiness/Universality)	Nondeterministic Emptiness	Nondeterministic Universality
Weighted aut.	PTime		Undecidable
NWA (LimAvg, Sum^+)	ExpSpace, PSpace-hard **PTime (width k is constant)** **PSpace-c. (bounded width)**		Undecidable
NWA (LimAvg, Sum)	Open **PTime (width k is constant)** **PSpace-c. (bounded width)**		Undecidable

questions is open (or undecidable). Thus, a fundamental question is whether there exist sub-classes of NWA that are strictly more expressive than weighted automata and yet have better complexity than general NWA. We address this question in this paper.

Nested weighted automata with bounded width. For NWA, let the maximum number of slave automata that can be active at any point be the *width* of the automaton. In this work we consider a natural special class of NWA, namely, NWA with bounded width, i.e., NWA where at any point at most k slave automata can be active. For example, the average response time with bounded number of requests pending at any point can be expressed by an NWA with bounded width, but not with a weighted automaton. Moreover, the class of NWA with bounded width is equivalent to automata with monitor counters [18], which are automata equipped with counters, where at each transition, a counter can be started, terminated, or the value of the counter can be increased or decreased. The transitions do not depend on the counter values, and hence they are referred to as monitor counters. The values of the counters when they are terminated give rise to the sequence of weights, which is aggregated into a single value with the LimAvg value function (see [18]). Automata with monitor counters are similar in spirit with the class of cost register automata of [2].

Our contributions. Our contributions are as follows (summarized as bold-faced results in Table 1):

1. *Constant width.* We show that the emptiness problem (resp., the emptiness and the universality problems) for non-deterministic (resp., deterministic) NWA with constant width (i.e., k is constant) can be solved in polynomial time and is NLogSpace-complete when the weights are specified in unary. Thus we achieve the same complexity as weighted automata for a much more expressive class of quantitative properties.
2. *Bounded width.* We show that the emptiness problem (resp., the emptiness and the universality problems) for non-deterministic (resp., deterministic) NWA with bounded width (i.e., k is a part of input given in unary) is PSpace-complete. Thus we establish precise complexity when k is a part of input given in unary.
3. *Deciding width.* We show that checking whether a given NWA has width k can be solved in polynomial time for constant k and in PSpace if k is given in the input (Theorem 6).

Technical contributions. Our main technical contributions for deterministic (LimAvg; Sum)-automata are as follows.

1. *Infinite infimum.* We first identify a necessary and sufficient condition for the infimum value over all words to be $-\infty$, and show that this condition can be checked efficiently.
2. *Lasso-approximation.* We show that if the above condition does not hold, then the infimum over all words can be approximated by lasso words, i.e., words of the form vu^ω. Moreover, we show that the infimum value is achieved with words where the slave automata run for short length relative to the point of the invocation, and hence the partial averages converge.
3. *Reduction to width* 1. Using the lasso-approximation we reduce the emptiness problem of width bounded by k to the corresponding problem of width 1. We show that the case of width 1 can be solved using standard techniques.

In the paper we present the key intuitions of the proofs, and due to space restrictions the technical details are in the full version [17].

Related works. Weighted automata over finite words have been extensively studied, the book [21] provides an excellent collection of results. Weighted automata on infinite words have been studied in [14, 13, 22]. The extension to weighted automata with monitor counters over finite words has been considered as cost register automata in [2]. A version of nested weighted automata over finite words has been studied in [6], and nested weighted automata over infinite words has been studied in [16]. Several quantitative logics have also been studied, such as [5, 7, 1]. In this work we consider a subclass of nested weighted automata which is strictly more expressive than weighted automata yet achieve the same complexity for the basic decision questions. Probabilistic models (such as Markov decision processes) with quantitative properties (such as limit-average or discounted-sum) have also been extensively studied for single objectives [23, 28], and for multiple objectives and their combinations [20, 10, 15, 8, 19, 9, 24, 11, 3, 4]. While NWA with bounded width have been studied under probabilistic semantics [18], the basic automata theoretic decision problems have not been studied for them.

2 Preliminaries

2.1 Words and automata

Words. We consider a finite *alphabet* of letters Σ. A *word* over Σ is a (finite or infinite) sequence of letters from Σ. We denote the i-th letter of a word w by $w[i]$, and for $i < j$ we have $w[i,j]$ is the word $w[i]w[i+1]\ldots w[j]$. The length of a finite word w is denoted by $|w|$; and the length of an infinite word w is $|w| = \infty$. For an infinite word w, thus $w[i,\infty]$ is the suffix of the word with first $i-1$ letters removed.

Labeled automata. For a set X, an X*-labeled automaton* $\mathcal{A}$ is a tuple $\langle \Sigma, Q, Q_0, \delta, F, C\rangle$, where (1) Σ is the alphabet, (2) Q is a finite set of states, (3) $Q_0 \subseteq Q$ is the set of initial states, (4) $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, (5) F is a set of accepting states, and (6) $C : \delta \mapsto X$ is a labeling function. A labeled automaton $\langle \Sigma, Q, \{q_0\}, \delta, F, C\rangle$ is *deterministic* if and only if δ is a function from $Q \times \Sigma$ into Q and Q_0 is a singleton. For deterministic labeled automata, we omit curly brackets for Q_0 and write $\langle \Sigma, Q, q_0, \delta, F, C\rangle$.

Semantics of (labeled) automata. A *run* π of a (labeled) automaton $\mathcal{A}$ on a word w is a sequence of states of $\mathcal{A}$ of length $|w|+1$ such that $\pi[0]$ belongs to the initial states of $\mathcal{A}$ and for every $0 \leq i \leq |w|-1$ we have $(\pi[i], w[i+1], \pi[i+1])$ is a transition of $\mathcal{A}$. A run π on a finite word w is *accepting* iff the last state $\pi[|w|]$ of the run is an accepting state of

$\mathcal{A}$. A run π on an infinite word w is *accepting* iff some accepting state of $\mathcal{A}$ occurs infinitely often in π. For an automaton $\mathcal{A}$ and a word w, we define $\mathsf{Acc}(w)$ as the set of accepting runs on w. Note that for deterministic automata, every word w has at most one accepting run ($|\mathsf{Acc}(w)| \leq 1$).

Weighted automata. A *weighted automaton* is a $\mathbb{Z}$-labeled automaton, where $\mathbb{Z}$ is the set of integers. The labels are called *weights*.

Semantics of weighted automata. We define the semantics of weighted automata in two steps. First, we define the value of a run. Second, we define the value of a word based on the values of its runs. To define values of runs, we will consider *value functions* f that assign real numbers to sequences of integers. Given a non-empty word w, every run π of $\mathcal{A}$ on w defines a sequence of weights of successive transitions of $\mathcal{A}$, i.e., $C(\pi) = (C(\pi[i-1], w[i], \pi[i]))_{1 \leq i \leq |w|}$; and the value $f(\pi)$ of the run π is defined as $f(C(\pi))$. We denote by $(C(\pi))[i]$ the weight of the i-th transition, i.e., $C(\pi[i-1], w[i], \pi[i])$. The value of a non-empty word w assigned by the automaton $\mathcal{A}$, denoted by $\mathcal{L}_{\mathcal{A}}(w)$, is the infimum of the set of values of all *accepting* runs; i.e., $\inf_{\pi \in \mathsf{Acc}(w)} f(\pi)$, and we have the usual semantics that infimum of an empty set is infinite, i.e., the value of a word that has no accepting run is infinite. Every run π on the empty word has length 1 and the sequence $C(\pi)$ is empty, hence we define the value $f(\pi)$ as an external (not a real number) value $\bot$. Thus, the value of the empty word is either $\bot$, if the empty word is accepted by $\mathcal{A}$, or ∞ otherwise. To indicate a particular value function f that defines the semantics, we will call a weighted automaton $\mathcal{A}$ an f-automaton.

Value functions. For finite runs we consider the following classical value functions: for runs of length $n+1$ we have

- *Sum, absolute sum:* the sum function $\textsc{Sum}(\pi) = \sum_{i=1}^{n} (C(\pi))[i]$, the absolute sum $\textsc{Sum}^{+}(\pi) = \sum_{i=1}^{n} \mathsf{Abs}((C(\pi))[i])$, where $\mathsf{Abs}(x)$ is the absolute value of x,

For infinite runs we consider:

- *Limit average:* $\textsc{LimAvg}(\pi) = \liminf\limits_{k \to \infty} \frac{1}{k} \cdot \sum_{i=1}^{k} (C(\pi))[i]$.

Silent moves. Consider a $(\mathbb{Z} \cup \{\bot\})$-labeled automaton. We can consider such an automaton as an extension of a weighted automaton in which transitions labeled by $\bot$ are *silent*, i.e., they do not contribute to the value of a run. Formally, for every function $f \in \mathsf{InfVal}$ we define $\mathsf{sil}(f)$ as the value function that applies f on sequences after removing $\bot$ symbols. The significance of silent moves is as follows: it allows to ignore transitions, and thus provide robustness where properties could be specified based on desired events rather than steps.

2.2 Nested weighted automata

In this section we describe nested weighted automata introduced in [16], and closely follow the description of [16]. For more details and illustration of such automata we refer the reader to [16]. We start with an informal description.

Informal description. A *nested weighted automaton* (NWA) consists of a labeled automaton over infinite words, called the *master automaton*, a value function f for infinite words, and a set of weighted automata over finite words, called *slave automata*. A nested weighted automaton can be viewed as follows: given a word, we consider the run of the master automaton on the word, but the weight of each transition is determined by dynamically running slave automata; and then the value of a run is obtained using the value function f. That is, the master automaton proceeds on an input word as an usual automaton, except that before it takes a transition, it starts a slave automaton corresponding to the label of the current transition. The slave automaton starts at the current position of the word of the master automaton and works on some finite part of the input word. Once a slave automaton finishes, it returns its value to the master automaton, which treats the returned value as the weight of the current transition that is being executed. The slave automaton might immediately accept and return value $\bot$, which corresponds to *silent* transitions. If one of slave automata rejects, the nested weighted automaton rejects. We first present an example and then the formal definition.

▶ **Example 1** (Average response time). Consider an alphabet Σ consisting of requests r, grants g, and null instructions #. The average response time (ART) property asks for the average number of instructions between any request and the following grant. This property cannot be expressed by a non-nested automaton: a quantitative property is a function from words to reals, and as a function the range of non-nested LimAvg-automata is bounded, whereas the ART can have unbounded values (for details see [16]).

Nested weighted automata. A *nested weighted automaton* (NWA) is a tuple $\langle \mathcal{A}_{mas}; f; \mathfrak{B}_1, \ldots, \mathfrak{B}_l \rangle$, where (1) $\mathcal{A}_{mas}$, called the *master automaton*, is a $\{1, \ldots, l\}$-labeled automaton over infinite words (the labels are the indexes of automata $\mathfrak{B}_1, \ldots, \mathfrak{B}_l$), (2) f is a value function on infinite words, called the *master value function*, and (3) $\mathfrak{B}_1, \ldots, \mathfrak{B}_l$ are weighted automata over finite words called *slave automata*. Intuitively, an NWA can be regarded as an f-automaton whose weights are dynamically computed at every step by the corresponding slave automaton. We define an $(f; g)$-*automaton* as an NWA where the master value function is f and all slave automata are g-automata.

Semantics: runs and values. A *run* of $\mathbb{A}$ on an infinite word w is an infinite sequence $(\Pi, \pi_1, \pi_2, \ldots)$ such that (1) Π is a run of $\mathcal{A}_{mas}$ on w; (2) for every $i > 0$ we have π_i is a run of the automaton $\mathfrak{B}_{C(\Pi[i-1], w[i], \Pi[i])}$, referenced by the label $C(\Pi[i-1], w[i], \Pi[i])$ of the master automaton, on some finite word of $w[i, j]$. The run $(\Pi, \pi_1, \pi_2, \ldots)$ is *accepting* if all runs $\Pi, \pi_1, \pi_2, \ldots$ are accepting (i.e., Π satisfies its acceptance condition and each $\pi_1, \pi_2, \ldots$ ends in an accepting state) and infinitely many runs of slave automata have length greater than 1 (the master automaton takes infinitely many non-silent transitions). The value of the run $(\Pi, \pi_1, \pi_2, \ldots)$ is defined as $\mathsf{sil}(f)(v(\pi_1)v(\pi_2)\ldots)$, where $v(\pi_i)$ is the value of the run π_i in the corresponding slave automaton. The value of a word w assigned by the automaton $\mathbb{A}$, denoted by $\mathcal{L}_{\mathbb{A}}(w)$, is the infimum of the set of values of all *accepting* runs. We require accepting runs to contain infinitely many non-silent transitions as f is a value function over infinite sequences, hence the sequence $v(\pi_1)v(\pi_2)\ldots$ with $\bot$ removed must be infinite.

Deterministic nested weighted automata. An NWA $\mathbb{A}$ is *deterministic* if (1) the master automaton and all slave automata are deterministic, and (2) slave automata recognize prefix-free languages, i.e., languages $\mathcal{L}$ such that if $w \in \mathcal{L}$, then no proper extension of w belongs to $\mathcal{L}$. Condition (2) implies that no accepting run of a slave automaton visits an accepting state twice. Intuitively, slave automata have to accept the first time they encounter an accepting state as they will not visit an accepting state again.

▶ **Definition 2** (Width of NWA). An NWA has *width* k if and only if in every run at every position at most k slave automata are active.

▶ **Example 3** (Non-overlapping ART). We consider a variant of the ART property, called the 1-ART property, where after a request till it is granted additional requests are not considered. Formally, we consider the ART property over the language $\mathcal{L}_1$ defined by $(r\#^*g\#^*)^\omega$ (equivalently, given a request, the automata can check if the slave automaton is not active, and only then invoke it). An NWA $\mathbb{A}_1$ computing the ART property over $\mathcal{L}_1$ is obtained from the NWA for the ART property (see [16]) by taking the product of the master automaton $\mathcal{A}_{mas}$ with an automaton recognizing the language $\mathcal{L}_1$. The automaton $\mathbb{A}_1$ is a deterministic ($\textsc{LimAvg}; \textsc{Sum}^+$)-automaton. Indeed, the master automaton and the slave automata are deterministic and the slave automata recognize prefix-free languages. Moreover, in any (infinite) run at most one slave automaton is active, i.e., $\mathbb{A}_1$ has width 1. The dummy slave automata do not increase the width as they immediately accept, and hence they are not considered as active even at the position they are invoked. Finally, observe that the 1-ART property can return unbounded values, which implies that there exists no (non-nested) LimAvg-automaton expressing it. Also see Example 3 of the full version [17].

Decision problems. The classical questions in automata theory are language *emptiness* and *universality*. These problems have their counterparts in the quantitative setting of weighted automata and NWA. The (quantitative) emptiness and universality problems are defined in the same way for weighted automata and NWA; in the following definition the automaton $\mathcal{A}$ can be either a weighted automaton or an NWA.

- **Emptiness and universality**: Given an automaton $\mathcal{A}$ and a threshold λ, the *emptiness* (resp. *universality*) problem asks whether there exists a word w with $\mathcal{L}_{\mathcal{A}}(w) \leq \lambda$ (resp., for every word w we have $\mathcal{L}_{\mathcal{A}}(w) \leq \lambda$).

▶ Remark. In this work we focus on value functions Sum and $\textsc{Sum}^+$ for finite words, and LimAvg for infinite words. There are other value functions for finite words, such as Max, Min and bounded sum. However, it was shown in [16] that for these value functions, there is a reduction to non-nested weighted automata. Also for infinite words, there are other value functions such as Sup, LimSup, where the complexity and decidability results have been established in [16]. Hence in this work we focus on the most conceptually interesting case of LimAvg function for master automaton and Sum and $\textsc{Sum}^+$ value functions for the slave automata.

3 Examples

We present several examples of properties that can be specified with NWA of bounded width.

▶ **Example 4** (Variants of ART). Recall the ART property (Example 1) and its variant 1-ART property (Example 3). We present two variants of the ART property.

First, we extend Example 3 and consider the k-ART property over languages L_k defined by $(\#^*r(\#^*r\#^*)^{\leq k-1}g\#^*)^\omega$, i.e., the language where there are at most k-pending requests before each grant. As Example 3, an NWA $\mathbb{A}_k$ computing the k-ART property can be constructed from the NWA from Example. 3 by taking the product of the master automaton $\mathcal{A}_{mas}$ of the ART property with an automaton recognizing $\mathcal{L}_k$. The NWA $\mathbb{A}_k$ has width k.

Second, we consider the 1-ART[k] property, where $\Sigma = \{r_i, g_i : i \in \{1, \ldots, k\}\} \cup \{\#\}$, i.e., there are k-different types of "request-grant" pairs. The 1-ART[k] property asks for the average number of instructions between any request and the following grant of the corresponding type. Moreover, we consider as for 1-ART property that for every i, between a request r_i and the following grant of the corresponding type g_i, there is no request r_i of the same type. The 1-ART[k] can be expressed with an $(\textsc{LimAvg}; \textsc{Sum}^+)$-automaton $\mathbb{A}_1^{[k]}$ of width bounded by k, which is similar to $\mathbb{A}_1$ from Example 3. Basically, the NWA $\mathbb{A}_1^{[k]}$ has k slave automata; for $i \in \{1, \ldots, k\}$ the slave automaton $\mathfrak{B}_i$ is invoked on letters r_i and it counts the number of steps to the following grant g_i. Additionally, the master automaton checks that for every i, between any two grants g_i, there is at most one request r_i.

In Examples 1, 3, and 4 we presented properties that can be expressed with $(\textsc{LimAvg}; \textsc{Sum}^+)$-automata. The following property of *average excess* can be expressed with slave automata with $\textsc{Sum}$ value functions that have both positive and negative weights, i.e., it can be expressed by an $(\textsc{LimAvg}; \textsc{Sum})$-automaton, but not $(\textsc{LimAvg}; \textsc{Sum}^+)$-automata.

▶ **Example 5** (Average excess). Consider the alphabet $\{r, g, \#\}$ from Example 1 with an additional letter \$. The *average excess* (AE) property asks for the average difference between requests and grants over blocks separated by \$. For example, for $\$(rr\#g\$)^\omega$ the average excess is 1. The AE property can be expressed by $(\textsc{LimAvg}; \textsc{Sum})$-automaton $\mathbb{A}_{\mathrm{AE}}$ of width 1 (presented below), but it cannot be expressed with $(\textsc{LimAvg}; \textsc{Sum}^+)$-automata; $(\textsc{LimAvg}; \textsc{Sum}^+)$-automata return values form the interval $[0, \infty)$, while AE ranges from $(-\infty, \infty)$. The automaton $\mathbb{A}_{\mathrm{AE}}$ invokes a slave automaton $\mathfrak{B}_1$ at positions of letter \$ and a dummy automaton $\mathfrak{B}_2$ on the remaining positions. The slave automaton $\mathfrak{B}_1$ runs until it sees \$ letter; it computes the difference between the number of r and g letters by taking transitions of weights $1, -1, 0$ respectively on letters $r, g, \#$. The master automaton as well as the slave automata of $\mathbb{A}_{\mathrm{AE}}$ are deterministic and the slave automata recognize prefix-free languages. Therefore, the NWA $\mathbb{A}_{\mathrm{AE}}$ is deterministic and has width 1.

4 Our Results

In this section we establish our main results. We first discuss complexity of checking whether a given NWA has width k. Next, we comment the results we need to prove.

Configurations. Let $\mathbb{A}$ be a non-deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton of width k. We define a *configuration* of $\mathbb{A}$ as a tuple $(q; q_1, \ldots, q_k)$ where q is a state of the master automaton and each $q_1, \ldots, q_k$ is either a state of a slave automaton of $\mathbb{A}$ or $\perp$. In the sequence $q_1, \ldots, q_k$ each state corresponds to one slave automaton, and the states are ordered w.r.t. the position when the corresponding slave automaton has been invoked, i.e., q_1 correspond to the least recently invoked slave automaton. If there are less than k slave automata active, then $\perp$ symbols follow the actual states (denoting there is no slave automata invoked). We define $\textsc{Conf}(\mathbb{A})$ as the number of configurations of $\mathbb{A}$.

Key ideas. NWA without weights are equivalent to Büchi automata [16]. The property of having width k is independent from weights. It can be decided with a construction of a (non-weighted) Büchi automaton, which tracks configurations $(q; q_1, \ldots, q_k)$ of a given NWA (assuming width k) and accepts only if the width-k condition is at some point violated.

▶ **Theorem 6.** *(1) Fix $k > 0$. We can check in polynomial time whether a given NWA has width k. (2) Given an NWA and a number k given in unary we can check in polynomial space whether the NWA has width k.*

Comment. We first note that for deterministic automata, emptiness and universality questions are similar. Hence we focus on the emptiness problem for non-deterministic automata (which subsumes the emptiness problem for deterministic automata) to establish the new results of Table 1. Moreover, the $\textsc{Sum}^+$ value function is a special case of the $\textsc{Sum}$ value function with only positive weights. Since our main results are algorithms to establish upper bounds, we will only present the result for the emptiness problem for non-deterministic ($\textsc{LimAvg}; \textsc{Sum}$)-automata. However, as a first step we show that without loss of generality, we can focus on the case of deterministic automata.

▶ **Lemma 7.** *Let $k > 0$. Given a non-deterministic* ($\textsc{LimAvg}; \textsc{Sum}$)*-automaton $\mathbb{A}$ over alphabet Σ of width k, a deterministic* ($\textsc{LimAvg}; \textsc{Sum}$)*-automaton $\mathbb{A}_d$ of width k over an alphabet $\Sigma \times \Gamma$ such that $\inf_{u \in \Sigma^\omega} \mathbb{A}(u) = \inf_{u' \in (\Sigma\times\Gamma)^\omega} \mathbb{A}_d(u')$ can be constructed in time exponential in k and polynomial in $|\mathbb{A}|$. Moreover, $\textsc{Conf}(\mathbb{A}_d)$ is polynomial in $\textsc{Conf}(\mathbb{A})$ and k and only the alphabet of $\mathbb{A}_d$ is exponential (in k) as compared to the alphabet of $\mathbb{A}$.*

Proof sketch. The main idea is that the part Γ of the alphabet encodes the possible non-deterministic choices, and the possible non-deterministic choices basically correspond to transitions between configurations.

Proof overview. We present our proof overview for the emptiness of deterministic ($\textsc{LimAvg}; \textsc{Sum}$)-automata. The proof consists of the following four key steps.

1. First, we identify a condition, and show in Lemma 9 that it is sufficient to ensure that the infimum value among all words is $-\infty$ (i.e., the least value possible). Moreover we show that the condition can be decided in PTIME if k is constant (even NLOGSPACE if additionally the weights are in unary) and in PSPACE if k is given in unary.
2. Second, we show that if the above condition does not hold, then there is a family of lasso words (i.e., a finite prefix followed by an infinite repetition of another finite word) that approximates the infimum value among all words. This shows that the above condition is both necessary and sufficient. Moreover, we consider *dense* words, in which an i-th invoked slave automaton runs for at most for $O(\log(i))$ steps. We show that the infimum is achieved by a dense word. These results are established in Lemma 11.
3. Third, we show using the above result, that the problem for bounded width can be reduced to the problem of width 1, and the reduction is polynomial in the size of the original automaton, and only exponential in k. Thus if k is constant, the reduction is polynomial. This is established in Lemma 12.
4. Finally, we show that for automata with width 1, the emptiness problem can be solved in NLOGSPACE if weights are in unary and otherwise in PTIME (Lemma 13).

The above four steps give our main result (Theorem 14). We start with the first item.

Intuition for the condition. We first illustrate with an example that for very similar automata, which just differ in order of invoking slave automata, the infima over the values are very different. For one automaton the infimum value is $-\infty$ and for the other it is 0. This example provides the intuition for the need of the condition to identify when the infimum value is $-\infty$.

▶ **Example 8.** Consider two deterministic ($\textsc{LimAvg}; \textsc{Sum}$)-automata $\mathbb{A}_1, \mathbb{A}_2$ defined as follows. The master automaton $\mathcal{A}_{mas}$ of $\mathbb{A}_1$ accepts the language $(12a^*\#)^\omega$. At letter 1 (resp., 2) it invokes an automaton $\mathfrak{B}_1$ (resp., $\mathfrak{B}_2$). The slave automaton $\mathfrak{B}_1$ increments its value at every a letter and it terminates once it reads $\#$. The slave automaton $\mathfrak{B}_2$ works as $\mathfrak{B}_1$ except that it decrements its value at a letters. NWA $\mathbb{A}_2$ is similar to $\mathbb{A}_1$ except that it accepts the language $(21a^*\#)^\omega$. It invokes the same slave automata as $\mathbb{A}_1$. Thus the two automata only differ in the order of invocation of the slave automata. Observe that the infimum over values of all words in $\mathbb{A}_1$ is 0. Basically, the values of slave automata are always the opposite, therefore the average of the values of slave automata is 0 infinitely often. However, the infimum over values of all words in $\mathbb{A}_2$ is $-\infty$. Indeed, consider a word $21a^1\# \dots 21a^{2^i} \dots$. At positions proceeding $1a^{2^i}$, the automaton $\mathfrak{B}_2$ returns the value -2^i and the average of all previous $2 \cdot i$ values is 0. Thus, the average at this position equals $-\frac{2^i}{2 \cdot i}$ (recall that the average is over the number of invocations of slave automata). Hence, the limit infimum of averages is $-\infty$.

Condition for infinite infimum. Let $k > 0$ and $\mathbb{A}$ be a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton of width k. Let C be the minimal weight of slave automata of $\mathbb{A}$. Condition (*):

(*) $C < 0$ and there exists a word w accepted by $\mathbb{A}$ and infinitely many positions b such that the sum of weights, which automata active at position b accumulate while running on $w[b, \infty]$, is less than $C \cdot k^2 \cdot \textsc{Conf}(\mathbb{A})$.

Intuitively, condition (*) implies that there is a subword u which can be repeated so that the values of slave automata invoked before position b can be decreased arbitrarily. Note that pumping that word may not decrease the total average of the word. However, with LimAvg value function, we need to ensure only the existence of a subsequence of positions at which the averages tend to $-\infty$, i.e., we only need to decrease the values of slave automata invoked before position b (for infinitely many positions).

Illustration of condition on example. Consider automata $\mathbb{A}_1, \mathbb{A}_2$ from Example 8. The automaton $\mathbb{A}_2$ satisfies condition (*), whereas $\mathbb{A}_1$ does not. In the word $21a^1\# \dots 21a^{2^i} \dots$, consider positions b, where $\mathfrak{B}_2$ is invoked by $\mathbb{A}_2$. The automaton $\mathfrak{B}_2$ works on the subword $21a^{2^i}$, where both automata $\mathfrak{B}_1, \mathfrak{B}_2$ are active and the sum of their values past any position is 0. However, the only slave automaton active at position b is $\mathfrak{B}_2$. These automaton accumulates the value -2^i past position b. Therefore, past some position N, all such positions b satisfy the statement from condition (*), and hence $\mathbb{A}_2$ satisfies condition (*). Now, for $\mathbb{A}_1$, at every position at which $\mathfrak{B}_2$ is active, $\mathfrak{B}_1$ is active as well, hence for any position b, the values accumulated by slave automaton active past this position is non-negative. Hence, $\mathbb{A}_1$ does not satisfy condition (*). We now present our lemma about the condition.

▶ **Lemma 9.** *Let $k > 0$ and $\mathbb{A}$ be a deterministic* $(\textsc{LimAvg}; \textsc{Sum})$*-automaton of width k.*

1. *If condition (*) holds for $\mathbb{A}$, then* $\inf_{u \in \Sigma^\omega} \mathbb{A}(u) = -\infty$.
2. *Condition (*) can be checked in* NLogSpace *for constant width and weights in unary,* PTime *for constant width, and in* PSpace *if the width is given in unary.*

Key intuitions. For (1), we show that the word from condition (*) can be pumped at some positions to achieve a word u' with $\mathbb{A}(u') = -\infty$. For (2), we show that condition (*) holds if and only if there exists a cycle in the graph of configurations of $\mathbb{A}$, which (a) can be visited infinitely often, and (b) for some $j \geq 1$, the sum of weights in this cycle of the j least recently invoked slave automata is negative. Recall that the order of invocation of slave automaton is encoded in the configuration, i.e., in $(q; q_1, \dots q_j, q_{j+1}, \dots, q_k)$ slave automata that correspond to the states $q_1, \dots, q_j$ are the j least recently invoked.

▶ **Definition 10.** Let $\mathbb{A}$ be a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton of width k. A word w is *dense* (w.r.t. $\mathbb{A}$) if in the run of $\mathbb{A}$ on w, for every $i > 0$, the i-th invoked slave automaton takes at most $O(\log(i))$ steps.

Intuitive explanation of dense words. In a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton, the average is over the number of invoked slave automata, but in general, the returned values of

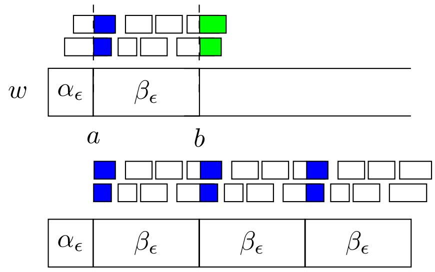

Figure 1 Explanation to Lemma 11; the blue part corresponds to H, while the green part corresponds to T.

the slave automata can be arbitrarily large as compared to the number of invocations, and hence the partial averages need not converge. Intuitively, in dense words, slave automata are invoked and terminated relatively densely, i.e., the length of their runs depends on the number of slave automata invoked till this position. In consequence, the value they can accumulate is small w.r.t. the average, i.e., their absolute contribution to the sum of the first n elements is $O(\log(n))$, and hence the contribution of the value a single slave automaton converges to 0 and the partial averages converge on dense words.

Illustration on example. Consider an automaton $\mathbb{A}_1$ from Example 8. We discuss the definition of density on an example of word $w = 12a^1\#12a^3\#\dots12a^{2\cdot i+1}\#\dots$, which is not dense (w.r.t. $\mathbb{A}_1$). Observe that at the position of subword $12a^{2\cdot i+1}$, the partial average is 0. Once $\mathfrak{B}_1$ is invoked it returns value $2\cdot i+1$ and it is $(2\cdot i+1)$-th invocation of a slave automaton. Hence, the average increases to 1 only to be decreased to 0 after invocation of $\mathfrak{B}_2$. Now, word $w' = 12a^1\#(12a^2\#)^3\dots(12a^{2\cdot i+1}\#)^{2^i}\dots$ is dense. Indeed, before the slave automaton invoked at subword $12a^{2\cdot i+1}\#$ there are at least $\sum_{j=1}^{i-1} 2^j = 2^i - 1$ invoked slave automata. Therefore, the value $2\cdot i+1$ returned by $\mathfrak{B}_1$ invoked on $12a^{2\cdot i+1}$ is logarithmic in the number of invoked slave automata 2^i-1 and it changes the average by at most $\frac{2\cdot i+1}{2^i}$; as previously invoking $\mathfrak{B}_2$ in the next step bring the average back to 0. Thus, the sequence of partial averages of values returned by slave automata converges to 0.

▶ **Lemma 11.** *Let $k > 0$ and $\mathbb{A}$ be a deterministic* (LimAvg; Sum)*-automaton of width k. Assume that condition (*) does not hold. Then the following assertions hold:*
1. *For every $\epsilon > 0$ there exist finite words $\alpha_\epsilon, \beta_\epsilon$ such that $|\inf_{u\in\Sigma^\omega}\mathbb{A}(u) - \mathbb{A}(\alpha_\epsilon(\beta_\epsilon)^\omega)| < \epsilon$.*
2. *The value $\inf_{u\in\Sigma^\omega}\mathbb{A}(u)$ is greater than $-\infty$.*
3. *There exists a dense word w_d such that $\inf_{u\in\Sigma^\omega}\mathbb{A}(u) = \mathbb{A}(w_d)$.*

Proof sketch: We present the key ideas for each item (detailed proof in [17]). Assume that condition (*) fails.
1. We consider $\epsilon > 0$ and a word w_ϵ, which is $\frac{\epsilon}{4}$-close to the infimum over all values of $\mathbb{A}$. We show that w_ϵ contains a subword β_ϵ on which (a) the automaton $\mathbb{A}$ starts and ends with the same configuration, and (b) the average of the values returned by the slave automata is at most $\mathbb{A}(w_\epsilon) + \frac{\epsilon}{4}$. The existence of such a word follows from the fact that the partial averages are infinitely often $\frac{\epsilon}{4}$-close to the value of w_ϵ. We then show that β_ϵ together with α_ϵ, the prefix preceding β_ϵ, satisfy $|\inf_{u\in\Sigma^\omega}\mathbb{A}(u) - \mathbb{A}(\alpha_\epsilon(\beta_\epsilon)^\omega)| < \epsilon$. If we consider the sequence of values returned by slave automata on the word β_ϵ^ω, then it differs from the sequence of values returned when we consider the corresponding suffix in w_ϵ: this is because the values of slave automata in β_ϵ as a subword of w_ϵ depend on the following letters. The difference of partial averages can be bounded with an estimate

of $H - T$ which is defined below. Consider the subword $\alpha_\epsilon \cdot \beta_\epsilon$. Let X be the set of slave automata that are active when β_ϵ is invoked (i.e., after α_ϵ). Let H be the sum of weights of the active slave automata in X accumulated in the part of their respective runs on β_ϵ. Let Y be the set of active slave automata after $\alpha_\epsilon \cdot \beta_\epsilon$. Let T be the sum of weights of the active slave automata in Y accumulated in the part of their respective runs on w_ϵ past $\alpha_\epsilon \cdot \beta_\epsilon$. See Fig 1 for an illustration. We establish an estimate on $H - T$ using the fact that (*) does not hold.

2. Almost all slave automata invoked in the run of $\mathbb{A}$ on a word of the form $\alpha(\beta)^\omega$ take at most $|\beta|$ steps. Only slave automata invoked at α can take more steps without looping. Thus, the value $\mathbb{A}(\alpha(\beta)^\omega)$ is finite. Therefore, (1) implies that $\inf_{u \in \Sigma^\omega} \mathbb{A}(u)$ is finite (given some words are accepted and condition(*) fails.)
3. We construct w_d from a word $\beta_1^{k[1]} \beta_{\frac{1}{2}}^{k[2]} \beta_{\frac{1}{3}}^{k[3]} \ldots$ by choosing the sequence $k[0], k[1], \ldots$ to increase sufficiently fast. By repeating $k[n]$ times word $\beta_{\frac{1}{n}}$, we increase the number of invoked slave automata at least by $k[n]$, so the number of steps of slave automaton invoked in $\beta_{\frac{1}{n+1}}$, which is bounded by $|\beta_{\frac{1}{n+1}}|$, can be made arbitrarily small w.r.t. $k[n]$.

▶ Remark. Lemma 9 together with (2) of Lemma 11 imply that for a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton $\mathbb{A}$ of width k condition (*) is both necessary and sufficient for the infimum over all values equal to $-\infty$. Moreover, this condition can be checked efficiently.

Lemma 12 reduces the emptiness problem for deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automata of width k to the same problem with automata of width 1.

▶ **Lemma 12.** *Let $k > 0$ and $\mathbb{A}$ be a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton of width k. Assume that condition (*) does not hold. Then, there exists a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton $\mathbb{A}_1$ of width 1 over an alphabet Δ such that $\inf_{u \in \Sigma^\omega} \mathbb{A}(u) = \inf_{u \in \Delta^\omega} \mathbb{A}_1(u)$. The size of $\mathbb{A}_1$ is $O(|\mathbb{A}|^k)$ and it can be constructed on-the-fly.*

Key intuitions. Consider a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton $\mathbb{A}$ of width k. We define the automaton $\mathbb{A}_1$, which uses a single slave automaton to keep track of all k automata of $\mathbb{A}$. This single slave automaton takes transitions whose weight is the sum of weights of transitions of tracked slave automata of $\mathbb{A}$. Therefore, $\mathbb{A}$ and $\mathbb{A}_1$ compute the averages of the same weights. Still, the way these weights are aggregated, i.e., their order in the sequence is different, and hence these automata may return different values on the same word. However, we show that on dense words the values of both automata coincide. This and Lemma 11 stating that there exists a dense word at which the automaton $\mathbb{A}_1$ realizes its infimum implies that the infimum over all values of $\mathbb{A}$ and $\mathbb{A}_1$ coincide.

▶ **Lemma 13.** *The emptiness problem for deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automata of width 1 is in* PTIME *and if the weights are in unary, then it is in* NLOGSPACE.

Key intuitions. We show that every transition of $\mathcal{A}_{mas}$, the master automaton of $\mathbb{A}$, at which a slave automaton is invoked, can be substituted by a transition whose weight is the minimal value the invoked slave automaton can achieve. More precisely, while a slave automaton is running on the input word, the master automaton $\mathcal{A}_{mas}$ is still active. Therefore, we substitute transitions (q, a, q', i) of $\mathcal{A}_{mas}$ by multiple transitions of the form $(q, (q, a, i, q''), q'')$, where (q, a, i, q'') is a new letter, q'' is a state of $\mathcal{A}_{mas}$ and the weight of this transition is the minimal value $\mathfrak{B}_i$ can achieve over words au such that $\mathcal{A}_{mas}$ moves from q to q'' upon reading au. Such a transformation preserves the infimum over all words and it transforms a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton of width 1 to a deterministic

LimAvg-automaton. The emptiness problem for LimAvg-automaton is decidable in PTime and even in NLogSpace provided that weights are given in unary.

We now present the algorithm and lower bound for our main result.

The algorithm. We present an algorithm, which, given a non-deterministic (LimAvg, Sum)-automaton $\mathbb{A}$ of width k and $\lambda \in \mathbb{Q}$, decides whether $\inf_{u \in \Sigma^\omega} \mathbb{A}(u) \leq \lambda$.

1. Transform $\mathbb{A}$ into a deterministic (LimAvg, Sum)-automaton $\mathbb{A}_d$ of the same width such that $\inf_{u \in \Sigma^\omega} \mathbb{A}(u) = \inf_{u \in (\Sigma \times \Gamma)^\omega} \mathbb{A}_d(u)$ (Lemma 7).
2. Check condition (*) for $\mathbb{A}_d$. If it holds, then $\inf_{u \in \Sigma^\omega} \mathbb{A}(u) = -\infty$ and return answer **YES**. Otherwise, continue the algorithm.
3. Transform $\mathbb{A}_d$ into a deterministic (LimAvg, Sum)-automaton $\mathbb{A}_1$ of width 1 such that $\inf_{u \in (\Sigma \times \Gamma)^\omega} \mathbb{A}_d(u) = \inf_{u \in \Delta^\omega} \mathbb{A}_1(u)$ (Lemma 12).
4. Compute $\inf_{u \in \Delta^\omega} \mathbb{A}_1(u)$ (Lemma 13), and return whether $\inf_{u \in \Delta^\omega} \mathbb{A}_1(u) \leq \lambda$.

Transformations in (1) and (3) are polynomial in the size of the automaton and exponential in k. Also, transformation from (1) does not increase k. Therefore, the size of $\mathbb{A}_1$ is polynomial in the size $\mathbb{A}$ and singly exponential in k. Moreover, these transformations can be done on-the-fly, i.e., there is not need to store the whole resulting automaton. Therefore, checks from (2) and (4), can be done in NLogSpace if k is constant and weights are in unary, PTime if k is constant, and PSpace if k is given in unary.

Hardness results. If k is constant, then the reachability problem on directed graphs, which is NLogSpace-complete, can be reduced to language emptiness of a finite automaton, which is a special case the emptiness problem for non-deterministic (LimAvg, Sum)-automata of width 1 with unary weights. If k is given in unary, consider the emptiness problem for the intersection of regular languages, which given k and regular languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$, asks whether $\mathcal{L}_1 \cap \ldots \cap \mathcal{L}_k = \emptyset$. This problem is PSpace-complete [26] and reduces to the emptiness problem for deterministic (LimAvg, Sum)-automata of width given in unary: the PSpace-hardness result for emptiness of NWA given in [16] uses NWA of width $|\mathbb{A}|$.

▶ **Theorem 14.** *The emptiness problem for non-deterministic* (LimAvg, Sum)*-automata is (a)* NLogSpace*-complete in the size of $\mathbb{A}$ for constant width k with weights in unary; (b)* PTime *in the size of $\mathbb{A}$ for constant width k; and (c)* PSpace*-complete when the bounded width k is given as input in unary.*

References

1. Shaull Almagor, Udi Boker, and Orna Kupferman. Discounting in LTL. In *TACAS, 2014*, pages 424–439, 2014.
2. Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *LICS 2013*, pages 13–22, 2013.
3. Christel Baier, Clemens Dubslaff, and Sascha Klüppelholz. Trade-off analysis meets probabilistic model checking. In *CSL-LICS 2014*, pages 1:1–1:10, 2014.
4. Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In *CSL-LICS 2014*, pages 11:1–11:10, 2014.
5. Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. *ACM TOCL*, 15(4):27:1–27:25, 2014.
6. Benedikt Bollig, Paul Gastin, Benjamin Monmege, and Marc Zeitoun. Pebble weighted automata and transitive closure logics. In *ICALP 2010, Part II*, pages 587–598. Springer, 2010.

7 Patricia Bouyer, Nicolas Markey, and Raj Mohan Matteplackel. Averaging in LTL. In *CONCUR 2014*, pages 266–280, 2014.

8 Tomás Brázdil, Václav Brozek, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Two views on multiple mean-payoff objectives in Markov decision processes. In *LICS 2011*, pages 33–42, 2011.

9 Tomás Brázdil, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Multigain: A controller synthesis tool for MDPs with multiple mean-payoff objectives. In *TACAS 2015*, pages 181–187, 2015.

10 Krishnendu Chatterjee. Markov decision processes with multiple long-run average objectives. In *FSTTCS*, pages 473–484, 2007.

11 Krishnendu Chatterjee and Laurent Doyen. Energy and mean-payoff parity Markov Decision Processes. In *MFCS 2011*, pages 206–218, 2011.

12 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In *FCT'09*, pages 3–13. Springer, 2009.

13 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. *LMCS*, 6(3), 2010.

14 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM TOCL*, 11(4):23, 2010.

15 Krishnendu Chatterjee, Vojtech Forejt, and Dominik Wojtczak. Multi-objective discounted reward verification in graphs and MDPs. In *LPAR*, pages 228–242, 2013.

16 Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted automata. In *LICS 2015*, pages 725–737, 2015.

17 Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted limit-average automata of bounded width. *CoRR*, abs/1606.03598, 2016. A conference version accepted to MFCS 2016. URL: `http://arxiv.org/abs/1606.03598`.

18 Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Quantitative automata under probabilistic semantics. *CoRR*, abs/1604.06764, 2016. A conference version accepted to LICS 2016. URL: `http://arxiv.org/abs/1604.06764`.

19 Krishnendu Chatterjee, Zuzana Komárková, and Jan Kretínský. Unifying two views on multiple mean-payoff objectives in Markov Decision Processes. In *LICS 2015*, pages 244–256, 2015.

20 Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov Decision Processes with multiple objectives. In *STACS 2006*, pages 325–336, 2006.

21 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 1st edition, 2009.

22 Manfred Droste and George Rahonis. Weighted automata and weighted logics on infinite words. In *DLT 2006*, pages 49–58, 2006.

23 Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer, 1996.

24 Vojtech Forejt, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Quantitative multi-objective verification for probabilistic systems. In *TACAS*, pages 112–127, 2011.

25 Thomas A. Henzinger and Jan Otop. From model checking to model measuring. In *CONCUR 2013*, pages 273–287, 2013.

26 Dexter Kozen. Lower bounds for natural proof systems. In *FOCS*, pages 254–266. IEEE Computer Society, 1977.

27 Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Aut. Lang. & Comb.*, 7(3):321–350, 2002.

28 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1st edition, 1994.