Quantitative Automata under Probabilistic Semantics

Krishnendu Chatterjee
Thomas A. Henzinger
IST Austria
{krishnendu.chatterjee,tah}@ist.ac.at

Jan Otop
University of Wrocaw
jotop@cs.uni.wroc.pl

Abstract

Automata with monitor counters, where the transitions do not depend on counter values, and nested weighted automata are two expressive automata-theoretic frameworks for quantitative properties. For a well-studied and wide class of quantitative functions, we establish that automata with monitor counters and nested weighted automata are equivalent. We study for the first time such quantitative automata under probabilistic semantics. We show that several problems that are undecidable for the classical questions of emptiness and universality become decidable under the probabilistic semantics. We present a complete picture of decidability for such automata, and even an almost-complete picture of computational complexity, for the probabilistic questions we consider.

Categories and Subject Descriptors F.1.1 [*Computation by Abstract Devices*]: Models of Computation—automata

Keywords weighted automata, nested weighted automata, probabilistic semantics, probability, expected value

1. Introduction

Traditional to quantitative verification. While traditional formal verification focused on Boolean properties of systems, such as “every request is eventually granted”, recently significant attention has been shifted to quantitative aspects such as expressing properties like “the long-run average success rate of an operation is at least one half” or “the long-run average (or the maximal, or the accumulated) resource consumption is below a threshold.” Quantitative properties are essential for performance related properties, for resource-constrained systems, such as embedded systems.

Overview. The first natural way to express quantitative properties is to consider automata with counters. However, computational analysis of such models quickly leads to undecidability, and a classical way to limit expressiveness for decidability is to consider *monitor counters*, i.e., the counter values do not influence the control. The second approach is to consider automata with weights (or weighted automata). However, weighted automata have limited expressiveness, and they have been extended as nested weighted automata [20] (nesting of weighted automata) for expressiveness. We establish that for a well-studied and wide class of quantitative functions, automata with monitor counters and nested weighted automata are equivalent, i.e., they represent a robust class of quantitative specifications. We study for the first time such quantitative automata under probabilistic semantics. Quite surprisingly we show that several problems that are undecidable for the classical questions of emptiness and universality become decidable under the probabilistic semantics. We present a complete picture of decidability for nested weighted automata and automata with monitor counters under probabilistic semantics.

Automata with monitor counters. A natural extension of automata is automata with monitor counters, which are automata equipped with counters. At each transition, a counter can be started, terminated, or the value of the counter can be increased or decreased. However, 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 gives rise to the sequence of weights. A value function aggregates the sequence into a single value. For example, for words over $\{a, \#\}$, such automata can express the maximal length of block of a’s that appear infinitely often. Automata with monitor counters are similar in spirit with the class of register automata of [2], and we consider them over infinite words.

Weighted automata. Weighted automata extend finite automata where every transition is assigned a rational number called a weight. Hence every run gives rise to a sequence of weights, which is aggregated into a single value by a value function. For non-deterministic weighted automata, the value of a word w is the infimum value of all runs over w. Weighted automata provide a natural and flexible framework for expressing quantitative[1] properties [16]. First, weighted automata were studied over finite words with weights from a semiring, and ring multiplication as a value function [24], and later extended to infinite words with limit averaging or supremum as value function [14, 16, 17]. While weighted automata over semirings can express several quantitative properties [29], they cannot express long-run average properties that weighted automata with limit averaging can [16]. However, even weighted automata with limit averaging cannot express the following basic quantitative property (the example is from [20]).

Example 1. *Consider infinite words over $\{r, g, i\}$, where r represents requests, g represents grants, and i represents idle. A basic and interesting property is the average number of i’s between a request and the corresponding grant, which represents the long-run average response time of the system.*

Nested weighted automata. To enrich expressiveness, weighted automata were extended to *nested weighted automata (NWA)* [20]. A

[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.

LICS ’16, July 05 - 08, 2016, New York, NY, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4391-6/16/07…$15.00.
http://dx.doi.org/10.1145/2933575.2933588

nested weighted automaton 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 invokes 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 equivalent to the non-nested counterpart, whereas nested weighted automata are strictly more expressive than non-nested weighted automata [20], for example, nested weighted automata can express the long-run average response time property (see [20, Example 5]). It has been shown in [20] that nested weighted automata 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.

Classical questions. The classical questions for automata are *emptiness* (resp., *universality*) that asks for the existence (resp., non-existence) of words that are accepted. Their natural extensions has been studied in the quantitative setting as well (such as for weighted automata, NWA, etc) [16, 20].

Motivation for probabilistic questions. One of the key reasons for quantitative specifications is to express performance related properties. While the classical emptiness and universality questions express the best/worst case scenarios (such as the best/worst-case trace of a system for average response time), they cannot express the average case average response time, where the average case corresponds to the expected value over all traces. Performance related properties are of prime interest for probabilistic systems, and quite surprisingly, quantitative automata have not been studied in a probabilistic setting, which we consider in this work.

Probabilistic questions. Weighted automata and their extensions as nested weighted automata, or automata with monitor counters are all measurable functions from infinite words to real numbers. We consider probability distribution over infinite words, and as a finite representation for probability spaces we consider the classical model of finite-state Markov chains. A stochastic environment is often modeled as a Markov chain [19]. Hence, the theoretical problems we consider correspond to measuring performance (expectation or cumulative distribution) under such stochastic environments, when the specification is a nested weighted automaton. Moreover, Markov chains are a canonical model for probabilistic systems [3, 28]. Given a measurable function (or equivalently a random variable), the classical quantities w.r.t. a probability distribution are: (a) the expected value; and (b) the cumulative distribution below a threshold. We consider the computation of the above quantities when the function is given by a nested weighted automaton or an automaton with monitor counters, and the probability distribution is given by a finite-state Markov chain. We also consider the approximate variants that ask to approximate the above quantities within a tolerance term $\epsilon > 0$. Moreover, for the cumulative distribution we consider the special case of *almost-sure* acceptance, which asks whether the probability is 1.

Our contributions. In this work we consider several classical value functions, namely, $\textsc{Sup}, \textsc{Inf}, \textsc{LimSup}, \textsc{LimInf}, \textsc{LimAvg}$ for infinite words, and $\textsc{Max}, \textsc{Min}, \textsc{Sum}, \textsc{Sum}^B, \textsc{Sum}^+$ (where $\textsc{Sum}^B$ is the sum bounded by B, and $\textsc{Sum}^+$ is the sum of absolute values) for finite words. First, we establish translations (in both directions) between automata with monitor counters and a special class of nested weighted automata, where at any point only a bounded number of slave automata can be active. However, in general, in nested weighted automata unbounded number of slave automata can be active. We describe our main results for nested weighted automata.

- $\textsc{LimSup}$ *and* $\textsc{LimInf}$ *functions.* We consider deterministic nested weighted automata with $\textsc{LimSup}$ and $\textsc{LimInf}$ functions for the master automaton, and show that for all value functions for finite words that we consider, all probabilistic questions can be answered in polynomial time. This is in contrast with the classical questions, where the problems are PSPACE-complete or undecidable (see Remark 16 for further details).
- $\textsc{Inf}$ *and* $\textsc{Sup}$ *functions.* We consider deterministic nested weighted automata with $\textsc{Sup}$ and $\textsc{Inf}$ functions for the master automaton, and show the following: the approximation problems for all value functions for finite words that we consider are $\#P$-hard and can be computed in EXPTIME; other than the $\textsc{Sum}$ function, the expected value, the distribution, and the almost-sure problems are PSPACE-hard and can be solved in EXPTIME; and for the $\textsc{Sum}$ function, the above problems are uncomputable. Again we establish a sharp contrast w.r.t. the classical questions as follows: for the classical questions, the complexity of $\textsc{LimSup}$ and $\textsc{Sup}$ functions always coincide, whereas we show a substantial complexity gap for probabilistic questions (see Remark 24 and Remark 25 for further details).
- $\textsc{LimAvg}$ *function.* We consider deterministic nested weighted automata with $\textsc{LimAvg}$ function for the master automaton, and show that for all value functions for finite words that we consider, all probabilistic questions can be answered in polynomial time. Again our results are in contrast to the classical questions (see Remark 29).
- *Non-deterministic automata.* For non-deterministic automata we show two results: first we present an example to illustrate the conceptual difficulty of evaluating a non-deterministic (even non-nested) weighted automata w.r.t. a Markov chain, and also show that for nested weighted automata with $\textsc{LimSup}$ value function for master automaton and $\textsc{Sum}$ value function for slave automata, all probabilistic questions are undecidable (in contrast to the deterministic case where we present polynomial-time algorithms).

Note that from above all decidability results we establish carry over to automata with monitor counters, and we show that all our undecidability (or uncomputability) results also hold for automata with monitor counters. Decidability results for nested weighted automata are more interesting as compared to automata with monitor counters as unbounded number of slaves can be active. Our results are summarized in Theorem 15 (in Section 6.2), Table 2 (in Section 6.3), and Theorem 28 (in Section 6.4). In summary, we present a complete picture of decidability of the basic probabilistic questions for nested weighted automata (and automata with monitor counters).

Technical contributions. We call a nested weighted automaton $\mathbb{A}$, an $(f; g)$*-automaton* if its master-automaton value function is f and the value function of all slave automata is g. We present the key details of our main technical contributions, and for sake of simplicity here explain for the case of the uniform distribution over infinite words. Our technical results are more general though (for distributions given by Markov chains).

- We show that in a deterministic $(\textsc{LimInf}; \textsc{Sum})$-automaton $\mathbb{A}$, whose master automaton is strongly connected as a graph, almost all words have the same value which, is the infimum over values of any slave automaton from $\mathbb{A}$ over all finite words.
- For a deterministic $(\textsc{Inf}; \textsc{Sum})$-automaton $\mathbb{A}$ and $C > 0$ we define $\mathbb{A}^C$ as the deterministic $(\textsc{Inf}; \textsc{Sum})$-automaton obtained from $\mathbb{A}$ by stopping every slave automaton if it exceeds C steps.

We show that for every deterministic $(\textsc{Inf}; \textsc{Sum})$-automaton $\mathbb{A}$ and $\epsilon > 0$, there exists C exponential in $|\mathbb{A}|$ and polynomial in ϵ such that the expected values of $\mathbb{A}$ and $\mathbb{A}^C$ differ by at most ϵ.

- We show that the expected value of a deterministic $(\textsc{LimAvg}; \textsc{Sum})$-automaton $\mathbb{A}$ coincides with the expected value of the following deterministic (non-nested) $\textsc{LimAvg}$-automaton $\mathcal{A}$. The automaton $\mathcal{A}$ is obtained from $\mathbb{A}$ by replacing in every transition an invocation of a slave automaton $\mathfrak{B}$ by the weight equal to the expected value of $\mathfrak{B}$.

Related works. Quantitative automata and logic have been extensively and intensively studied in recent years. The book [24] presents an excellent collection of results of weighted automata on finite words. Weighted automata on infinite words have been studied in [16, 17, 23]. The extension to weighted automata with monitor counters over finite words has been considered (under the name of cost register automata) in [2]. A version of nested weighted automata over finite words has been studied in [7], and nested weighted automata over infinite words have been studied in [20]. Several quantitative logics have also been studied, such as [1, 6, 8]. While a substantial work has been done for quantitative automata and logics, quite surprisingly none of the above works consider the automata (or the logic) under probabilistic semantics that we consider in this work. 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 [26, 31], and for multiple objectives and their combinations [4, 5, 9–13, 18, 21, 27]. However, these works do not consider properties that are expressible by nested weighted automata (such as average response time) or automata with monitor counters.

In the main paper, we present the key ideas and main intuitions of the proofs, and detailed proofs are relegated to the full version of this paper [22].

2. Preliminaries

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 $w[i]$. The length of a finite word w is denoted by $|w|$; and the length of an infinite word w is $|w| = \infty$.

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 the 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. In definitions of deterministic labeled automata we omit curly brackets in the description of 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 $(\pi[i], w[i], \pi[i+1])$ 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*. We assume that weights are given in the unary notation, and, hence, the values of weights are linearly bounded in the size of weighted automata.

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 an 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 $\perp$. Thus, the value of the empty word is either $\perp$, 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. We will consider the classical functions and their natural variants for value functions. For finite runs we consider the following value functions: for runs of length $n + 1$ we have

1. *Max and min:* $\textsc{Max}(\pi) = \max_{i=1}^{n}(C(\pi))[i]$ and $\textsc{Min}(\pi) = \min_{i=1}^{n}(C(\pi))[i]$.
2. *Sum, absolute sum and bounded 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, and the bounded sum value function returns the sum if all the partial absolute sums are below a bound B, otherwise it returns the exceeded bound $-B$ or B, i.e., formally, $\textsc{Sum}^{B}(\pi) = \textsc{Sum}(\pi)$, if for all prefixes π' of π we have $\mathsf{Abs}(\textsc{Sum}(\pi')) \leq B$, otherwise $\textsc{Sum}^{B}(\pi) = \mathrm{sgn} \cdot B$ where sgn is the sign of the shortest prefix whose sum is outside $[-B, B]$.

We denote the above class of value functions for finite words as $\mathsf{FinVal} = \{\textsc{Max}, \textsc{Min}, \textsc{Sum}^{B}, \textsc{Sum}\}$.

For infinite runs we consider:

1. *Supremum and Infimum, and Limit supremum and Limit infimum*: $\textsc{Sup}(\pi) = \sup\{(C(\pi))[i] : i > 0\}$, $\textsc{Inf}(\pi) = \inf\{(C(\pi))[i] : i > 0\}$, $\textsc{LimSup}(\pi) = \limsup\{(C(\pi))[i] : i > 0\}$, and $\textsc{LimInf}(\pi) = \liminf\{(C(\pi))[i] : i > 0\}$.
2. *Limit average:* $\textsc{LimAvg}(\pi) = \limsup\limits_{k \to \infty} \frac{1}{k} \cdot \sum_{i=1}^{k}(C(\pi))[i]$.

We denote the above class of value functions for infinite words as $\mathsf{InfVal} = \{\textsc{Sup}, \textsc{Inf}, \textsc{LimSup}, \textsc{LimInf}, \textsc{LimAvg}\}$.

Silent moves. Consider a $(\mathbb{Z} \cup \{\perp\})$-labeled automaton. We can consider such an automaton as an extension of a weighted automaton in which transitions labeled by $\perp$ 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 $\perp$ symbols. The significance of silent moves is as follows: they allow to ignore transitions, and thus provide robustness where properties could be specified based on desired events rather than steps.

3. Extensions of weighted automata

In this section we consider two extensions of weighted automata, namely, automata with monitor counters and nested weighted automata.

3.1 Automata with monitor counters

Intuitively, automata with monitor counters are an extension of weighted automata with counters, where the transitions do not depend on the counter value. We define them formally below.

Automata with monitor counters. An *automaton with n monitor counters* $\mathcal{A}^{\text{m-c}}$ is a tuple $\langle \Sigma, Q, Q_0, \delta, F\rangle$ where (1) Σ is the alphabet, (2) Q is a finite set of states, (3) $Q_0 \subseteq Q_0$ is the set of initial states, (4) δ is a finite subset of $Q \times \Sigma \times Q \times (\mathbb{Z} \cup \{s,t\})^n$ called a transition relation, (each component refers to one monitor counter, where letters s, t refer to starting and terminating the counter, respectively, and the value from $\mathbb{Z}$ is the value that is added to the counter), and (5) F is the set of accepting states. Moreover, we assume that for every $(q, a, q', \vec{u}) \in \delta$, at most one component in $\vec{u}$ contains s, i.e., at most one counter is started at each position. Intuitively, the automaton $\mathcal{A}^{\text{m-c}}$ is equipped with n counters. The transitions of $\mathcal{A}^{\text{m-c}}$ do not depend on the values of counters (hence, we call them monitor counters); and every transition is of the form $(q, a, q', \vec{v})$, which means that if $\mathcal{A}^{\text{m-c}}$ is in the state q and the current letter is a, then it can move to the state q' and update counters according to v. Each counter is initially inactive. It is started by the instruction s, and it changes its value at every step by adding the value of the corresponding component of v, until termination t. The value of the counter at the time it is terminated is then assigned to the position where it has been started. An automaton with monitor counters $\mathcal{A}^{\text{m-c}}$ is *deterministic* if and only if Q_0 is a singleton and δ is a function from $Q \times \Sigma$ into $Q \times (\mathbb{Z} \cup \{s,t\})^n$.

Semantics of automata with monitor counters. A sequence π of elements from $Q \times (\mathbb{Z} \times \{\bot\})^n$ is a *run* of $\mathcal{A}^{\text{m-c}}$ on a word w if (1) $\pi[0] = \langle q_0, \vec{\bot}\rangle$ and $q_0 \in Q_0$ and (2) for every $i > 0$, if $\pi[i-1] = \langle q, \vec{u}\rangle$ and $\pi[i] = \langle q', \vec{u}'\rangle$ then $\mathcal{A}^{\text{m-c}}$ has a transition $(q, w[i], q', \vec{v})$ and for every $j \in [1, n]$ we have (a) if $v[j] = s$, then $u[j] = \bot$ and $u'[j] = 0$, (b) if $v[j] = t$, then $u[j] \in \mathbb{Z}$ and $u'[j] = \bot$, and (c) if $v[j] \in \mathbb{Z}$, then $u'[j] = u[j] + v[j]$. A run π is *accepting* if some state from F occurs infinitely often on the first component of π, some counter is started infinitely often, and every started counter is finally terminated. An accepting run π defines a sequence π^W of integers and $\bot$ as follows: let the counter started at position i be j, and let the value of the counter j terminated at the earliest position after i be x_j, then $\pi^W[i]$ is x_j. The semantics of automata with monitor counters is given, similarly to weighted automata, by applying the value function to π^W.

Remark 2. *Automata with monitor counters are very similar in spirit to the register automata considered in the works of [2]. The key difference is that we consider infinite words and value functions associated with them, whereas previous works consider finite words. Another key difference is that in this work we will consider probabilistic semantics, and such semantics has not be considered for register automata before.*

Example 3 (Blocks difference)**.** *Consider an alphabet $\Sigma = \{a, \#\}$ and a language $\mathcal{L}$ of words $(\#^2 a^* \# a^* \#)^\omega$. On the words from $\mathcal{L}$ we consider a quantitative property "the maximal block-length difference between odd and even positions", i.e., the value of word $\#^2 a^{n[1]} \# a^{n[2]} \#^3 \ldots$ is $\sup_{0 \le i} |n[2*i+1] - n[2*i+2]|$. This property can be expressed by a* Sup*-automaton $\mathcal{A}_{\textit{diff}}$ with two monitor counters depicted in Figure 1.*

The automaton $\mathcal{A}_{\textit{diff}}$ has a single initial state q_0, which is also the only accepting state. It processes the word w in subwords $\#^2 a^k \# a^m \#$ in the following way. First, it reads $\#^2$ upon which it takes transitions from q_0 to q_1 and from q_1 to q_2, where it starts counters 1 *and* 2*. Next, it moves to the state q_2 where it counts letters a incrementing counter* 1 *and decrementing counter* 2*. Then, upon reading $\#$, it moves to q_3, where it counts letters a, but it decrements counter* 1 *and increments counter* 2*. After reading $\#^2 a^k \# a^m$ the value of counter* 1 *is $k - m$ and counter* 2 *is $m - k$. In the following transition from q_3 to q_0, the automaton terminates both counters. The aggregating function of $\mathcal{A}_{\textit{diff}}$ is* Sup*, thus the automaton discards the lower value, i.e., the value of $\#^2 a^k \# a^m \#$ is $|k - m|$ and the automaton computes the supremum over values of all blocks. It follows that the value of $\#^2 a^{n[1]} \# a^{n[2]} \#^3 \ldots$ is* $\sup_{0 \le i} |n[2*i+1] - n[2*i+2]|$.

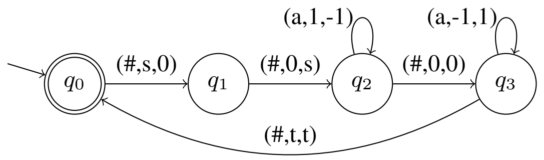

Figure 1: The automaton $\mathcal{A}_{\text{diff}}$ computing the maximal difference between the lengths of blocks of a's at odd and the following even positions.

3.2 Nested weighted automata

In this section we describe nested weighted automata introduced in [20], and closely follow the description of [20]. For more details and illustration of such automata we refer the reader to [20]. We start with an informal description.

Informal description. A *nested weighted automaton* 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 define this formally as follows.

Nested weighted automata. A *nested weighted automaton* (NWA) $\mathbb{A}$ is a tuple $\langle \mathcal{A}_{\text{mas}}; f; \mathfrak{B}_1, \ldots, \mathfrak{B}_k\rangle$, where (1) $\mathcal{A}_{\text{mas}}$, called the *master automaton*, is a $\{1, \ldots, k\}$-labeled automaton over infinite words (the labels are the indexes of automata $\mathfrak{B}_1, \ldots, \mathfrak{B}_k$), (2) f is a value function on infinite words, called the *master value function*, and (3) $\mathfrak{B}_1, \ldots, \mathfrak{B}_k$ 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 a 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 an NWA $\mathbb{A}$ on an infinite word w is an infinite sequence $(\Pi, \pi_1, \pi_2, \ldots)$ such that (1) Π is a run of $\mathcal{A}_{\text{mas}}$ 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 because f is a value function over infinite sequences, so we need the sequence $v(\pi_1)v(\pi_2)\ldots$ with $\bot$ symbols removed to 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 see an accepting state again.

Bounded width. An NWA has *bounded width* if and only if there exists a bound C such that in every run at every position at most C slave automata are active.

Example 4 (Average response time with bounded requests)**.** *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. It has been shown in [20] that NWA can express ART. However, the automaton from [20] does not have bounded width. To express the ART property with NWA of bounded width we consider only words such that between any two grants there are at most k requests.*

Average response time over words where between any two grants there are at most k requests can be expressed by an $(\text{LIMAVG}; \text{SUM})$*-automaton $\mathbb{A}$. Such an automaton $\mathbb{A} = (\mathcal{A}_{mas}; \text{LIMAVG}; \mathfrak{B}_1, \mathfrak{B}_2)$ is depicted in Fig. 2. The master automaton of $\mathbb{A}$ accepts only words with infinite number of requests and grants, where every grant is followed by a request and there are at most k requests between any two grants. On letters $\#$ and g, the master automaton invokes a dummy automaton $\mathfrak{B}_1$, which immediately accepts; the result of invoking such an automaton is equivalent to taking a silent transition as the automaton $\mathfrak{B}_1$ returns $\bot$, the empty value. On letters r, denoting requests, the master automaton invokes $\mathfrak{B}_2$, which counts the number of letters to the first occurrence of letter g, i.e., the automaton $\mathfrak{B}_2$ computes the response time for the request on the position it is invoked. The automaton $\mathbb{A}$ computes the limit average of all returned values, which is precisely ART (on the accepted words). Note that the width of $\mathbb{A}$ is k.*

3.3 Translation

We now present translations from NWA to automata with monitor counters and vice-versa.

Lemma 5. *[Translation Lemma] For every value function $f \in$* InfVal *on infinite words we have the following: (1) Every deterministic f-automaton with monitor counters $\mathcal{A}^{m\text{-}c}$ can be transformed in polynomial time into an equivalent deterministic* $(f; \text{SUM})$*-automaton of bounded width. (2) Every non-deterministic (resp., deterministic)* $(f; \text{SUM})$*-automaton of bounded width can be transformed in exponential time into an equivalent non-deterministic (resp., deterministic) f-automaton with monitor counters.*

We illustrate below the key ideas of the above translations of Lemma 5 to automata from Examples 3 and 4. The detailed technical proof is in the full version of this paper [22].

Example 6 (Translation of automata with monitor counters to nested weighted automata)**.** *Consider a deterministic automaton*

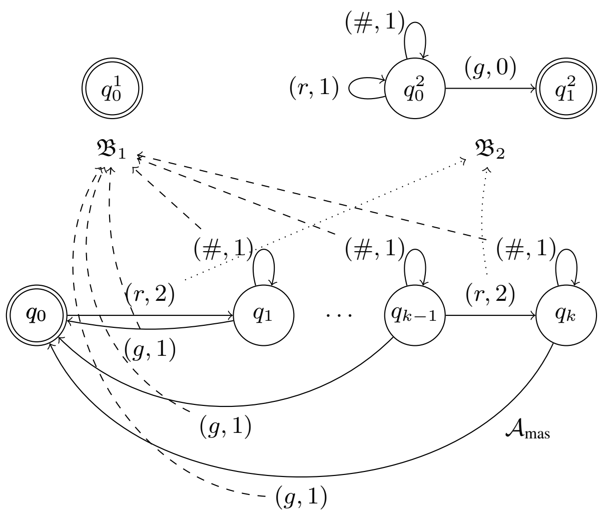

Figure 2: The $(\text{LIMAVG}; \text{SUM})$-automaton computing the average response time over words with infinite number of requests and grants such that between any two grants there are at most k requests.

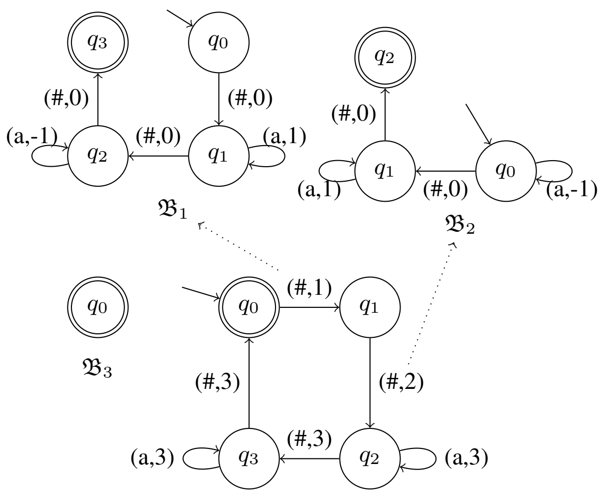

Figure 3: A nested weighted automaton resulting from translation of the automaton $\mathcal{A}_{\text{diff}}$ from Example 3.

$\mathcal{A}$ with k monitor counters. We construct an NWA $\mathbb{A}$ equivalent to $\mathcal{A}$. The automaton $\mathbb{A}$ uses k slave automata to track values of k monitor counters in the following way. The master automaton of $\mathbb{A}$ simulates $\mathcal{A}$; it invokes slave automata whenever $\mathcal{A}$ starts monitor counters. Slave automata simulate $\mathcal{A}$ as well. Each slave automaton is associated with some counter i; it starts in the state (of $\mathcal{A}$) the counter i is initialized, simulates the value of counter i, and terminates when counter i is terminated. Figure 3 presents the result of transition of the automaton $\mathcal{A}_{diff}$ from Example 3 to a $(\text{SUP}; \text{SUM})$*-automaton of width bounded by* 3.

Example 7. *(Translation of nested weighted automata of bounded width to automata with monitor counters) Consider an* $(f; \text{SUM})$*-automaton $\mathbb{A}$ of width bounded by k. The automaton $\mathbb{A}$ can be simulated by an automaton with monitor counters which simulates the master automaton and up to k slave automata running in*

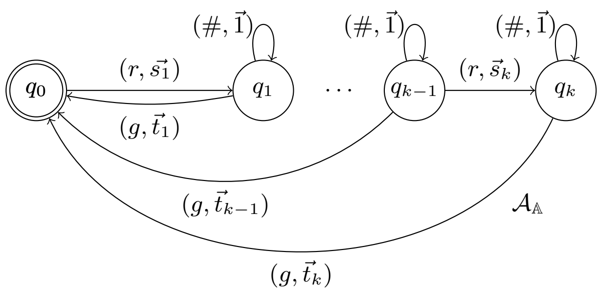

Figure 4: The (reduced) result of translation of the automaton $\mathbb{A}$ from Example 4 to an automaton with monitor counters. All vectors have dimension k. Vector $\vec{1}$ denotes the vector with all components equal 0. Vector $\vec{s_i}$ denotes the whose i-th component is s and other components are 1. Vector $\vec{t_i}$ denotes the vector whose components $1,\ldots,i$ are t and the remaining components are 0.

parallel. To simulate values of slave automata it uses monitor counters, each counter separately for each slave automaton.

Figure 4 shows the result of translation of the automaton $\mathbb{A}$ from Example 4 to the automaton with monitor counters $\mathcal{A}_{\mathbb{A}}$. The set of states of $\mathcal{A}_{\mathbb{A}}$ there is $\{q_0,\ldots,q_k\} \times (\{q_0^2,\perp\})^k$, i.e., the states of the master automaton and all non-accepting states of slave automata (in deterministic NWA accepting states are sink states, hence storing them is redundant). Now, observe that only reachable states of $\mathcal{A}_{\mathbb{A}}$ are $(q_0,\perp,\ldots,\perp),(q_1,q_0^2,\perp,\ldots,\perp),\ldots,(q_k,q_0^2,\ldots,q_0^2)$, i.e., the reachable part of $\mathcal{A}_{\mathbb{A}}$ is isomorphic (in the sense of graphs) to the master automaton of $\mathbb{A}$.

Remark 8 (Discussion). *Lemma 5 states that deterministic automata with monitor counters have the same expressive power as deterministic NWA of bounded width. However, the latter may be exponentially more succinct. In consequence, lower bounds on deterministic automata with monitor counters imply lower bounds on NWA of bounded width. Conversely, deterministic NWA can be considered as automata with infinite number of monitor counters, therefore upper bounds on deterministic NWA imply upper bounds on deterministic counter automata.*

4. Problems

4.1 Classical questions

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

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

The universality question asks for *non-existence* of a word w such that $\mathcal{L}_{\mathcal{A}}(w) > \lambda$.

4.2 Probabilistic questions

The classical questions ask for the existence (or non-existence) of words for input automata, whereas in the probabilistic setting, input automata are analyzed w.r.t. a probability distribution. We consider probability distributions over infinite words Σ^ω, and as a finite representation consider the classical model of Markov chains.

Labeled Markov chains. A *(labeled) Markov chain* is a tuple $\langle \Sigma, S, s_0, E\rangle$, where Σ is the alphabet of letters, S is a finite set of states, s_0 is an initial state, $E : S \times \Sigma \times S \mapsto [0,1]$ is the edge probability function, which for every $s \in S$ satisfies that $\sum_{a\in\Sigma, s'\in S} E(s,a,s') = 1$.

Distributions given by Markov chains. Consider a Markov chain $\mathcal{M}$. For every finite word u, the probability of u, denoted $\mathbb{P}_{\mathcal{M}}(u)$, w.r.t. the Markov chain $\mathcal{M}$ is the sum of probabilities of paths labeled by u, where the probability of a path is the product of probabilities of its edges. For basic open sets $u \cdot \Sigma^\omega = \{uw : w \in \Sigma^\omega\}$, we have $\mathbb{P}_{\mathcal{M}}(u\cdot\Sigma^\omega) = \mathbb{P}_{\mathcal{M}}(u)$, and then the probability measure over infinite words defined by $\mathcal{M}$ is the unique extension of the above measure (by Carathéodory's extension theorem [25]). We will denote the unique probability measure defined by $\mathcal{M}$ as $\mathbb{P}_{\mathcal{M}}$, and the associated expectation measure as $\mathbb{E}_{\mathcal{M}}$.

Automata as random variables. Note that weighted automata, NWA, or automata with monitor counters all define measurable functions, $f : \Sigma^\omega \mapsto \mathbb{R}$, and thus can be interpreted as random variables w.r.t. the probabilistic space we consider. Hence given an automaton $\mathcal{A}$ and a Markov chain $\mathcal{M}$, we consider the following fundamental quantities:

1. **Expected value**: $\mathbb{E}_{\mathcal{M}}(\mathcal{A})$ is the expected value of the random variable defined by the automaton $\mathcal{A}$ w.r.t. the probability measure defined by the Markov chain $\mathcal{M}$.
2. **(Cumulative) distribution**: $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda) = \mathbb{P}_{\mathcal{M}}(\{w : \mathcal{L}_{\mathcal{A}}(w) \leq \lambda\})$ is the cumulative distribution function of the random variable defined by the automaton $\mathcal{A}$ w.r.t. the probability measure defined by the Markov chain $\mathcal{M}$.

Computational questions. Given an automaton $\mathcal{A}$ and a Markov chain $\mathcal{M}$, we consider the following basic computational questions: (Q1) The *expected question* asks to compute $\mathbb{E}_{\mathcal{M}}(\mathcal{A})$. (Q2) The *distribution question* asks, given a threshold λ, to compute $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$. Questions (1) and (2) have their approximate variants, which, given an additional input $\epsilon > 0$, ask to compute values that are ϵ-close to $\mathbb{E}_{\mathcal{M}}(\mathcal{A})$ or $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$, i.e., given $\epsilon > 0$ (Q3) The *approximate expected question* asks to compute a value η such that $|\eta - \mathbb{E}_{\mathcal{M}}(\mathcal{A})| \leq \epsilon$, and (Q4) The *approximate distribution question* asks to compute a value η such that $|\eta - \mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)| \leq \epsilon$. Additionally, a special important case for the distribution question is (Q5) The *almost-sure distribution question* asks whether for a given λ the probability $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$ is exactly 1.

We refer to questions (Q1)-(Q5) as *probabilistic questions*. Note that an upper bound on the complexity of the expected and distribution questions imply the same upper bound on all probabilistic questions as approximate and almost-sure variants are special cases.

Example 9 (Expected average response time). *Consider an NWA $\mathbb{A}$ from Example 4. Recall that it computes ART on words it accepts (bounded number of requests between any two grants). Next, consider a Markov chain $\mathcal{M}$ which gives a distribution on words over $\{r, g, \#\}$. In such a case, the value $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ is the expected ART.*

5. Results on classical questions

Existing results. The complexity of the classical decision problems for NWA has been established in [20] which is presented in Table 1.

New results. Due to Lemma 5, decidability of deterministic $(f;\textsc{Sum})$-automata implies decidability of deterministic automata

		INF LIMINF	SUP LIMSUP	LIMAVG
MIN, MAX SUM^B	Empt.	PSP.-c		
	Univ.			
SUM	Empt.	PSP.-c	Undec.	Open
	Univ.	Undec.	PSP.-c	
SUM$^+$	Empt. Univ.	PSP.-c		EXPSP.

Table 1: Decidability and complexity of emptiness and universality for deterministic $(f;g)$-automata. Functions f are listed in the first row and functions g are in the first column. PSP. denotes PSPACE, EXPSP. denotes EXPSPACE, and Undec. denotes undecidability.

with monitor counters with the value function f. However, the undecidability result of NWA does not imply undecidability for automata with monitor counters. Our following result presents the decidability picture also for automata with monitor counters (i.e., the decidability result coincides with the SUM row of Table 1).

Theorem 10. *(1) The emptiness problem is undecidable for deterministic* SUP*-automata (resp.,* LIMSUP*-automata) with* 8 *monitor counters. (2) The universality problem is undecidable for deterministic* INF*-automata (resp.,* LIMINF*-automata) with* 8 *monitor counters.*

6. Results on probabilistic questions

In this section we present our results for the probabilistic questions and deterministic NWA. First we present some basic properties, then some basic facts about Markov chains, and then present our results organized by value functions of the master automaton.

Property about almost-sure acceptance. Observe that if the probability of the set of words rejected by an automaton $\mathcal{A}$ is strictly greater than 0, then the expected value of such an automaton is infinite or undefined. In the next lemma we show that given a deterministic NWA $\mathbb{A}$ and a Markov chain $\mathcal{M}$ we can decide in polynomial time whether the set of words accepted has probability 1. In the sequel when we consider all the computational problems we consider that the set of accepted words has probability 1. This assumption does not infuence the compexity of computatonal questions related to the expected value, but has an influence on the complexity of distribution questions, which we discuss in the full version [22].

Proposition 11. *Given a deterministic NWA $\mathbb{A}$ and a Markov chain $\mathcal{M}$, we can decide in polynomial time whether $\mathbb{P}_{\mathcal{M}}(\{w : \mathsf{Acc}(w) \neq \emptyset\}) = 1$?*

6.1 Basic facts about Markov chains

Labeled Markov chains with weights. A labeled Markov chain with weights is a (labeled) Markov chain $\mathcal{M}$ with a function r, which associates integers with edges of $\mathcal{M}$. Formally, a *(labeled) Markov chain with weights* is a tuple $\langle \Sigma, S, s_0, E, r\rangle$, where $\langle \Sigma, S, s_0, E\rangle$ is a labeled Markov chain and $r : S \times \Sigma \times S \mapsto \mathbb{Z}$.

Graph properties on Markov chains. Standard graph notions have their counterparts on Markov chains by considering edges with strictly positive probability as present and edges with probability 0 as absent. For examples, we consider the following graph notions:

- **(reachability)**: A state s is *reachable* from s' in a Markov chain if there exists a sequence of edges with positive probability starting in s' and ending in s.
- **(SCCs)**: A subset of states Q of a Markov chain is a *strongly connected component* (SCC) if and only if from any state of Q all states in Q are reachable.
- **(end SCCs)**: An SCC Q is an *end* SCC if and only if there are no edges leaving Q.

The product of an automaton and a Markov chain. Let $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F, C\rangle$ be a deterministic weighted automaton (accepting almost all words) and let $\mathcal{M} = \langle \Sigma, S, s_0, E, r\rangle$ be a Markov chain. We define the product of $\mathcal{A}$ and $\mathcal{M}$, denoted by $\mathcal{A} \times \mathcal{M}$, as a Markov chain $\langle \Sigma, Q \times S, (q_0, s_0), E', r'\rangle)$, where (1) $E'(\langle q_1, s_1\rangle, a, \langle q_2, s_2\rangle) = E(s_1, a, s_2)$ if $(q_1, a, q_2) \in \delta$ and $E'(\langle q_1, s_1\rangle, a, \langle q_2, s_2\rangle) = 0$ otherwise, and (2) $r'(\langle q_1, s_1\rangle, a, \langle q_2, s_2\rangle) = C(q_1, a, q_2) + r(s_1, a, s_2)$.

The expected value and distribution questions can be answered in polynomial time for deterministic weighted automata with value functions from InfVal [15].

Fact 12. *Let $f \in \mathsf{InfVal}$. Given a Markov chain $\mathcal{M}$, a deterministic f-automaton $\mathcal{A}$ and a value λ, the values $\mathbb{E}_{\mathcal{M}}(\mathcal{A})$ and $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$ can be computed in polynomial time.*

6.2 LIMINF and LIMSUP value functions

In this section we study NWA with LIMINF and LIMSUP value functions for the master automaton. We establish polynomial-time algorithms for all probabilistic questions. We start with a result for the special case when the master automaton is strongly connected w.r.t. the Markov chain.

An automaton strongly connected on a Markov chain. We say that a deterministic automaton $\mathcal{A}$ is *strongly connected on* a Markov chain $\mathcal{M}$ if and only if the states reachable (with positive probability) in $\mathcal{A} \times \mathcal{M}$ from the initial state form an SCC.

Lemma 13. *Let $g \in \mathsf{FinVal}$, $\mathcal{M}$ be a Markov chain, and $\mathbb{A}$ be a deterministic* (INF; g)*-automaton (resp.,* (LIMINF; g)*-automaton). If the master automaton of $\mathbb{A}$ is strongly connected on $\mathcal{M}$ then there exists a unique value λ such that $\mathbb{P}_{\mathcal{M}}(\{w : \mathbb{A}(w) = \lambda\}) = 1$. Moreover, we have (1) $|\lambda| \leq |\mathbb{A}|$ or $\lambda = -\infty$ (or $\lambda = -B$ for $g =$ SUM^B), and (2) given $\mathcal{M}$ and $\mathbb{A}$, the value λ can be computed in polynomial time in $|\mathcal{M}| + |\mathbb{A}|$.*

Intuitively, in the probabilistic setting, for the condition saying that a given infimal value appears infinitely often, we establish in Lemma 13 some sort of 0-1 law for SCCs which shows that a given infimum appears infinitely often either on almost all words or only on words of probability zero. Consider the product of $\mathcal{M}$ and the master automaton of $\mathbb{A}$, and consider a state (s, q) in the product. Consider a slave automaton $\mathfrak{B}_i$ that can be invoked in q, and let $\lambda_{s,q,i}$ be the minimal value that can be achieved over finite words with positive probability given that the Markov chain starts in state s. Then we establish that $\lambda = \min_{s,q,i} \lambda_{s,q,i}$, i.e., it is the minimal over all such triples. Lemma 13 implies the following main lemma of this section.

Lemma 14. *Let $g \in \mathsf{FinVal}$. For a deterministic* (LIMINF; g)*-automata (resp.,* (LIMSUP; g)*-automata) $\mathbb{A}$ and a Markov chain $\mathcal{M}$, given a threshold λ, both $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in polynomial time.*

The key ideas. Consider a (LIMINF; g)-automaton (resp., (LIMSUP; g)-automaton) $\mathbb{A}$. The value $\mathbb{A}(w)$ depends only on the infinite behavior of the (unique) run of $\mathbb{A}$ on w. Almost all runs of $\mathbb{A}$ end up in one of the end SCCs of $\mathcal{M} \times \mathcal{A}_{\text{mas}}$, where, by Lemma 13 almost all words have the same value which can be computed in polynomial time. Thus, to compute $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$, it suffices to compute probabilities of reaching all end SCCs and values of $\mathbb{A}$ in these components. In a similar way, we can compute $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$.

Theorem 15. *Let $g \in \mathsf{FinVal}$. All probabilistic questions for $(\textsc{LimInf}; g)$-automata (resp., $(\textsc{LimSup}; g)$-automata) can be solved in polynomial time.*

Remark 16 (Contrast with classical questions)**.** *Consider the results on classical questions shown in Table 1 and the results for probabilistic questions we establish in Theorem 15. While for classical questions the problems are* PSPACE*-complete or undecidable, we establish polynomial-time algorithms for all probabilistic questions.*

6.3 INF and SUP value functions

In contrast to LIMINF and LIMSUP value functions, for which all probabilistic questions can be answered in polynomial time (Lemma 15), we first present several hardness results for INF and SUP value functions for NWA.

Lemma 17. *[Hardness results] Let $g \in \mathsf{FinVal}$ be a value function, and $\mathcal{U}$ denote the uniform distribution over the infinite words.*

1. *The following problems are* PSPACE*-hard: Given a deterministic $(\textsc{Inf}; g)$-automaton (resp., $(\textsc{Sup}; g)$-automaton) $\mathbb{A}$, decide whether $\mathbb{E}_{\mathcal{U}}(\mathbb{A}) = 0$; and decide whether $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0) = 1$?*
2. *The following problems are #P-hard: Given $\epsilon > 0$ and a deterministic $(\textsc{Inf}; g)$-automaton (resp., $(\textsc{Sup}; g)$-automaton) $\mathbb{A}$, compute $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ up to precision ϵ; and compute $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0)$ up to precision ϵ.*

The key ideas. We present the key ideas:

PSPACE*-hardness.* NWA can invoke multiple slave automata which independently work over the same word. In particular, one can express that the intersection of languages of finite-word automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ is non-empty by turning these automata into slave automata that return 1 if the original automaton accepts and 0 otherwise. Then, the infimum over all values of slave automata is 1 if and only if the intersection is non-empty. Note however, that words of the minimal length in the intersection can have exponential length. The probability of such word can be doubly-exponentially small in the size of $\mathbb{A}$, and thus the PSPACE-hardness does not apply to the approximation problems (which we establish below).

$\#P$-hardness. We show $\#P$-hardness of the approximate variants by reduction from #SAT, which is $\#P$-complete [30, 32]. The #SAT problem asks, given a CNF formula φ, for the number of assignments satisfying φ. In the proof, the input word gives an assignment, which is processed by slave automata. Each slave automaton checks the satisfaction of one clause and returns 1 if it is satisfied and 0 otherwise. Thus, all slave automata return 1 if and only if all clauses are satisfied. In such case, one can compute from $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0)$ the number of satisfying assignments of φ.

Upper bounds for $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+, \textsc{Sum}\}$. We now present upper bounds for value functions $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+, \textsc{Sum}\}$ of the slave automata. First we show an exponential-time upper bound for general NWA with INF and SUP value functions (cf. with the PSPACE-hardness from Lemma 17).

Lemma 18. *Let $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+, \textsc{Sum}\}$ be a value function. Given a Markov chain $\mathcal{M}$, a deterministic $(\textsc{Inf}; g)$-automaton (resp., $(\textsc{Sup}; g)$-automaton) $\mathbb{A}$, and a threshold λ in binary, both $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time. Moreover, if $\mathbb{A}$ has bounded width, then the above quantities can be computed in polynomial time.*

Remark 19. *We show in Lemma 18 a polynomial-time upper bound for NWA with bounded width, which gives a polynomial-time upper bound for automata with monitor counters.*

Key ideas. For $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+, \textsc{Sum}\}$, it has been shown in [20] that $(\textsc{Inf}; g)$-automata (resp., $(\textsc{Sup}; g)$-automata) can be transformed to exponential-size INF-automata (resp., SUP-automata). We observe that the transformation preserves determinism. Then, using Fact 12, both $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time.

The $\textsc{Sum}^+$ and SUM value functions for slave automata. We now establish the result when $g = \textsc{Sum}, \textsc{Sum}^+$. First we establish decidability of the approximation problems, and then undecidability of the exact questions.

Lemma 20. *Let $g \in \{\textsc{Sum}^+, \textsc{Sum}\}$. Given $\epsilon > 0$, a Markov chain $\mathcal{M}$, a deterministic $(\textsc{Inf}; g)$-automaton (resp., $(\textsc{Sup}; g)$-automaton) $\mathbb{A}$, a threshold λ, both $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed up to precision ϵ in exponential time, and the dependency on ϵ is linear in the binary representation of ϵ.*

Key ideas. The main difference between INF and LIMINF value functions is that the latter discards all values encountered before the master automaton reaches an end SCC where the infimum of values of slave automata is easy to compute (Lemma 13). We show that for some B, exponential in $|\mathbb{A}|$ and polynomial in the binary representation of ϵ, the probability that any slave automaton returns value λ with $|\lambda| > B$ is smaller than ϵ. Therefore, to approximate $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ up to precision ϵ, we can regard a given $(\textsc{Inf}; \textsc{Sum})$-automaton (resp., $(\textsc{Sup}; \textsc{Sum})$-automaton) as $(\textsc{Inf}; \textsc{Sum}^B)$-automaton (resp., $(\textsc{Sup}; \textsc{Sum}^B)$-automaton) and use Lemma 18.

Lemma 21. *Let $\mathcal{U}$ be the uniform distribution over the infinite words. The following problems are undecidable: (1) Given a deterministic $(\textsc{Inf}; \textsc{Sum})$-automaton (resp., $(\textsc{Sup}; \textsc{Sum})$-automaton) $\mathbb{A}$ of width 8, decide whether $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$. (2) Given two deterministic $(\textsc{Inf}; \textsc{Sum})$-automata (resp., $(\textsc{Sup}; \textsc{Sum})$-automata) $\mathbb{A}_1, \mathbb{A}_2$ of width bounded by 8, decide whether $\mathbb{E}_{\mathcal{U}}(\mathbb{A}_1) = \mathbb{E}_{\mathcal{U}}(\mathbb{A}_2)$.*

Key ideas. On finite words, $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$ holds if and only if every word has the value not exceeding -1, i.e., the distribution question and the universality problem are equivalent. We observe that in automata from the proof of Theorem 10, which is given in the full version [22], such an equivalence holds as well, i.e., there exists a word with the value exceeding -1 if and only if $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) < 1$. To show (2), we consider the automaton $\mathbb{A}$ from (1) and its copy $\mathbb{A}'$ that at the first transition invokes a slave automaton that returns -1. On every word w, we have $\mathbb{A}'(w) = \min(-1, \mathbb{A}(w))$. Now, $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$ implies that $\mathbb{A}'$ and $\mathbb{A}$ are equal on almost every word. Conversely, if $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) < 1$, then the expected value of $\mathbb{A}$ is greater than the one of $\mathbb{A}'$. Thus, the expected values are equal if and only if $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$.

Finally, we have the following result for the absolute sum value function.

Lemma 22. *(1) Given a Markov chain $\mathcal{M}$, a deterministic $(\textsc{Inf}; \textsc{Sum}^+)$-automaton $\mathbb{A}$, and a threshold λ in binary, both $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time. (2) Given a Markov chain $\mathcal{M}$, a deterministic $(\textsc{Sup}; \textsc{Sum}^+)$-automaton $\mathbb{A}$, and a threshold λ in binary $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time.*

The problem, how to compute $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ for deterministic $(\textsc{Sup}; \textsc{Sum}^+)$-automata $\mathbb{A}$ remains open.

Theorem 23. *Let $g \in \mathsf{FinVal}$. The complexity results for the probabilistic questions for $(\textsc{Inf}; g)$-automata and $(\textsc{Sup}, g)$-automata are summarized in Table 2, with the exception of the expected question of $(\textsc{Sup}; \textsc{Sum}^+)$-automata.*

Open question. The decidability of the expected question of $(\textsc{Sup}; \textsc{Sum}^+)$-automata is open. This open problem is related to

	MIN, MAX, SUM^B, SUM$^+$	SUM
Expected value	EXPTIME (L. 18,20) PSPACE-hard (L. 17)	Uncomputable (L. 21)
Distribution		
Almost sure distribution		
Approximate: (a) expected value (b) distribution	EXPTIME (L. 18,20) #P-hard (L. 17)	

Table 2: The complexity results for various problems for deterministic NWA with INF and SUP value functions, with exception of expected question of $(\textsc{Sup}, \textsc{Sum}^+)$-automata which is open. Columns represent slave-automata value functions, rows represent probabilistic questions.

the language inclusion problem of deterministic $(\textsc{Sup}; \textsc{Sum}^+)$-automata which is also an open problem.

Remark 24 (Contrast with classical questions)**.** *Consider Table 1 for the classical questions and our results established in Table 2 for probabilistic questions. There are some contrasting results, such as, while for* $(\textsc{Sup}, \textsc{Sum})$*-automata the emptiness problem is undecidable, the approximation problems are decidable.*

Remark 25 (Contrast of LIMINF vs INF)**.** *We remark on the contrast of the* LIMINF *vs* INF *value functions. For the classical questions of emptiness and universality, the complexity and decidability always coincide for* LIMINF *and* INF *value functions for NWA (see Table 1). Surprisingly we establish that for probabilistic questions there is a substantial complexity gap: while the* LIMINF *problems can be solved in polynomial time, the* INF *problems are undecidable,* PSPACE*-hard, and even* $\#P$*-hard for approximation.*

6.4 LIMAVG value function

Lemma 26. *Let* $g \in \mathsf{FinVal}$*. Given a Markov chain* $\mathcal{M}$ *and a deterministic* $(\textsc{LimAvg}; g)$*-automaton* $\mathbb{A}$*, the value* $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ *can be computed in polynomial time.*

Proof sketch. We present the most interesting case when $g = \textsc{Sum}$. Let $\mathbb{A}$ be a $(\textsc{LimAvg}; \textsc{Sum})$-automaton and let $\mathcal{M}$ be a Markov chain. We define a weighted Markov chain $\mathcal{M}^{\mathbb{A}}$ as the product $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, where $\mathcal{A}_{\text{mas}}$ is the master automaton of $\mathbb{A}$. The weights of $\mathcal{M}^{\mathbb{A}}$ are the expected values of invoked slave automata, i.e., the weight of the transition $\langle (q,s), a, (q',s') \rangle$ is the expected value of $\mathfrak{B}_i$, the slave automaton started by $\mathcal{A}_{\text{mas}}$ in the state q upon reading a, w.r.t. the distribution given by $\mathcal{M}$ starting in s. One can show that the expected value of $\mathbb{A}$ w.r.t. $\mathcal{M}$, denoted by $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$, and the expected value of $\mathcal{M}^{\mathbb{A}}$ coincide. The Markov chain $\mathcal{M}^{\mathbb{A}}$ can be computed in polynomial time and has polynomial size in $|\mathbb{A}|+|\mathcal{M}|$. Thus, we can compute the expected values of $\mathcal{M}^{\mathbb{A}}$, and in turn $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$, in polynomial time in $|\mathbb{A}| + |\mathcal{M}|$. $\square$

Lemma 27. *Let* $g \in \mathsf{FinVal}$*. Given a Markov chain* $\mathcal{M}$*, a deterministic* $(\textsc{LimAvg}; g)$*-automaton* $\mathbb{A}$ *and a value* λ*, the value* $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ *can be computed in polynomial time.*

Key ideas. We show that the distribution is discrete. More precisely, let $\mathcal{A}$ be the product of the Markov chain $\mathcal{M}$ and the master automaton of $\mathbb{A}$. We show that almost all words, whose run end up in the same end SCC of $\mathcal{A}$, have the same value, which is equal to the expected value over words that end up in that SCC. Thus, to answer the distribution question, we have to compute for every end SCC C of $\mathcal{A}$, the expected value over words that end up in C

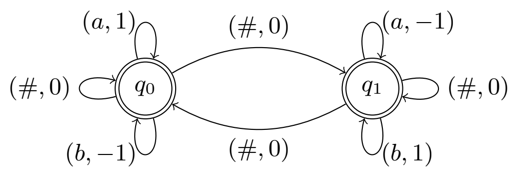

Figure 5: An example of non-deterministic automaton, in which non-deterministic choices has to "depend on the future" in order to obtain the infimum.

and the probability of reaching C. Both values can be computed in polynomial time (see Lemma 26).

Theorem 28. *Let* $g \in \mathsf{FinVal}$*. All probabilistic questions for* $(\textsc{LimAvg}; g)$*-automata can be solved in polynomial time.*

Remark 29 (Contrast with classical questions)**.** *Our results summarized in Theorem 28 contrast the results on classical questions shown in Table 1. While classical questions are* PSPACE*-complete, in* EXPSPACE *or open, we establish polynomial-time algorithms for all probabilistic questions.*

7. Results on non-deterministic automata

In this section, we briefly discuss non-deterministic NWA evaluated on Markov chains. We present two negative results.

Conceptual difficulty. The evaluation of non-deterministic (even non-nested) weighted automaton over a Markov chain is conceptually different as compared to the standard model of Markov decision processes (MDPs). Indeed, in an MDP, probabilistic transitions are interleaved with non-deterministic transitions, whereas in the case of an automaton, it runs over a word that has been already generated by the Markov chain. In MDPs, the strategy to resolve non-determinism can only rely on the past, whereas in the automaton model the whole future is available (i.e., there is a crucial distinction between online vs offline processing of the word). Below we present an example to illustrate this conceptual problem.

Example 30. *Consider a non-deterministic* LIMAVG*-automaton* $\mathcal{A}$*, depicted in Figure 5. Intuitively, the automaton processes a given word in blocks of letters* a, b *separated by letters* $\#$*. At the beginning of every block it decides whether the value of this block is the number of* a *letters* n_a *minus the number of* b *letters* n_b *divided by* $n_a + n_b$ *(i.e.,* $\frac{n_a - n_b}{n_a + n_b}$*) or the opposite (i.e.,* $\frac{n_b - n_a}{n_a + n_b}$*). Let* $\mathcal{U}$ *be the uniform distribution on infinite words over* Σ*. Suppose that the expected value of* $\mathcal{A}$ *w.r.t.* $\mathcal{U}$ *is evaluated as in MDPs case, i.e., non-deterministic choices depend only on the read part of the word. Then, since the distribution is uniform, any strategy results in the same expected value, which is equal to* 0*. Now, consider* $\mathbb{E}_{\mathcal{U}}(\mathcal{A})$*. The value of every block is at most* 0 *as the automaton works over fully generated word and at the beginning of each block can guess whether the number of* a*'s or* b*'s is greater. Also, the blocks* $a\#, b\#$ *with the average* $-\frac{1}{2}$ *appear with probability* $\frac{2}{9}$*, hence* $\mathbb{E}_{\mathcal{U}}(\mathcal{A}) < -\frac{1}{9}$*. Thus, the result of evaluating a non-deterministic weighted automaton over a Markov chain is different than evaluating it as an MDP.*

Computational difficulty. In contrast to our polynomial-time algorithms for the probabilistic questions for deterministic NWA with $(\textsc{LimSup}, \textsc{Sum})$ value function, we establish the following undecidability result for the non-deterministic automata with width 1. Lemma 31 implies Theorem 32.

Lemma 31. *The following problem is undecidable: given a non-deterministic* $(\textsc{LimSup}; \textsc{Sum})$*-automaton* $\mathbb{A}_M$ *of width* 1*, decide*

	Det.	Non-det.
Emptiness	Undec. (from [20])	
Probabilistic questions	PTIME (Th. 15)	Uncomputable (Th. 32)

Table 3: Decidability and complexity status of the classical and probabilistic questions for (LIMSUP; SUM)-automata. The negative results hold also for NWA of bounded width and automata with monitor counters.

whether $\mathbb{P}(\{w : \mathcal{A}_M(w) = 0\}) = 1$ *or* $\mathbb{P}(\{w : \mathcal{A}_M(w) = -1\}) = 1$ *w.r.t. the uniform distribution on infinite words.*

Theorem 32. *All probabilistic questions (Q1-Q5) are undecidable for non-deterministic* (LIMSUP, SUM)*-automata of width 1.*

8. Discussion

In this work we study the probabilistic questions related to NWA and automata with monitor counters. We establish the relationship between NWA and automata with monitor counters, and present a complete picture of decidability for all the probabilistic questions we consider. Our results establish a sharp contrast of the decidability and complexity of the classical questions (of emptiness and universality) and the probabilistic questions for deterministic automata (see Tables 1, 2 and Theorems 15, 28). In addition, there is also a sharp contrast for deterministic and non-deterministic automata. For example, for (LIMSUP, SUM)-automata, the classical questions are undecidable for deterministic and non-deterministic automata, while the probabilistic questions are decidable for deterministic automata, but remain undecidable for non-deterministic automata (see Table 3). We have some complexity gap (e.g., EXPTIME vs PSPACE) which is due to the fact that the computational questions we consider for Markov chains are in PTIME (as compared to NLOGSPACE for graphs), and we need to evaluate exponential-size Markov chains. Closing the complexity gap is an interesting open question.

Acknowledgments

This research was funded in part by the European Research Council (ERC) under grant agreement 267989 (QUAREM), by the Austrian Science Fund (FWF) projects S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award), FWF Grant No P23499- N23, FWF NFN Grant No S11407-N23 (RiSE/SHiNE), 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.

References

[1] S. Almagor, U. Boker, and O. Kupferman. Discounting in LTL. In *TACAS, 2014*, pages 424–439, 2014.

[2] R. Alur, L. D'Antoni, J. V. Deshmukh, M. Raghothaman, and Y. Yuan. Regular functions and cost register automata. In *LICS 2013*, pages 13–22, 2013.

[3] C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008. ISBN 978-0-262-02649-9.

[4] C. Baier, C. Dubslaff, and S. Klüppelholz. Trade-off analysis meets probabilistic model checking. In *CSL-LICS 2014*, pages 1:1–1:10, 2014.

[5] C. Baier, J. Klein, S. Klüppelholz, and S. Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In *CSL-LICS 2014*, pages 11:1–11:10, 2014.

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

[7] B. Bollig, P. Gastin, B. Monmege, and M. Zeitoun. Pebble weighted automata and transitive closure logics. In *ICALP 2010, Part II*, pages 587–598. Springer, 2010.

[8] P. Bouyer, N. Markey, and R. M. Matteplackel. Averaging in LTL. In *CONCUR 2014*, pages 266–280, 2014.

[9] T. Brázdil, V. Brozek, K. Chatterjee, V. Forejt, and A. Kucera. Two views on multiple mean-payoff objectives in Markov decision processes. In *LICS 2011*, pages 33–42, 2011.

[10] T. Brázdil, K. Chatterjee, V. Forejt, and A. Kucera. Multigain: A controller synthesis tool for MDPs with multiple mean-payoff objectives. In *TACAS 2015*, pages 181–187, 2015.

[11] K. Chatterjee. Markov decision processes with multiple long-run average objectives. In *FSTTCS*, pages 473–484, 2007.

[12] K. Chatterjee and L. Doyen. Energy and mean-payoff parity Markov Decision Processes. In *MFCS 2011*, pages 206–218, 2011.

[13] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Markov Decision Processes with multiple objectives. In *STACS 2006*, pages 325–336, 2006.

[14] K. Chatterjee, L. Doyen, and T. A. Henzinger. Alternating weighted automata. In *FCT'09*, pages 3–13. Springer, 2009.

[15] K. Chatterjee, L. Doyen, and T. A. Henzinger. A survey of stochastic games with limsup and liminf objectives. In *ICALP 2009, Part II*, pages 1–15, 2009.

[16] K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM TOCL*, 11(4):23, 2010.

[17] K. Chatterjee, L. Doyen, and T. A. Henzinger. Expressiveness and closure properties for quantitative languages. *LMCS*, 6(3), 2010.

[18] K. Chatterjee, V. Forejt, and D. Wojtczak. Multi-objective discounted reward verification in graphs and MDPs. In *LPAR*, pages 228–242, 2013.

[19] K. Chatterjee, T. A. Henzinger, B. Jobstmann, and R. Singh. Measuring and synthesizing systems in probabilistic environments. *J. ACM*, 62(1):9:1–9:34, 2015. .

[20] K. Chatterjee, T. A. Henzinger, and J. Otop. Nested weighted automata. In *LICS 2015*, pages 725–737, 2015.

[21] K. Chatterjee, Z. Komárková, and J. Kretínský. Unifying two views on multiple mean-payoff objectives in Markov Decision Processes. In *LICS 2015*, pages 244–256, 2015.

[22] K. Chatterjee, T. A. Henzinger, and J. Otop. Quantitative automata under probabilistic semantics. *CoRR*, abs/1604.06764, 2016. URL `http://arxiv.org/abs/1604.06764`.

[23] M. Droste and G. Rahonis. Weighted automata and weighted logics on infinite words. In *DLT 2006*, pages 49–58, 2006.

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

[25] W. Feller. *An introduction to probability theory and its applications*. Wiley, 1971. ISBN 9780471257097.

[26] J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer, 1996.

[27] V. Forejt, M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Quantitative multi-objective verification for probabilistic systems. In *TACAS*, pages 112–127, 2011.

[28] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS 2006*, pages 441–444, 2006.

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

[30] C. H. Papadimitriou. *Computational complexity*. Wiley, 2003.

[31] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1st edition, 1994.

[32] L. G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.