Logical Methods in Computer Science
Volume 15, Issue 3, 2019, pp. 16:1–16:45
https://lmcs.episciences.org/

Submitted May 16, 2018
Published Aug. 13, 2019

QUANTITATIVE AUTOMATA UNDER PROBABILISTIC SEMANTICS

KRISHNENDU CHATTERJEE [a], THOMAS A. HENZINGER [a], AND JAN OTOP [b]

[a] IST Austria
e-mail address: krish.chat@gmail.com, tah@ist.ac.at

[b] University of Wrocław
e-mail address: jan.otop@uwr.edu.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.

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 [CHO17] (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

Key words and phrases: weighted automata, Markov chains, nested weighted automata, expected value, distribution.

This is a combined version of [CHO16b] and [CHO16c].

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-15(3:16)2019

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. Automata with monitor counters are natural extension of weighted automata, where automata are equipped with integer-valued 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 give rise to a 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 *cost register automata* [ADD$^+$13], and we consider them over infinite words.

Weighted automata. Weighted automata extend finite automata where every transition is assigned an integer 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 [CDH10b]. First, weighted automata were studied over finite words with weights from a semiring, and ring multiplication as a value function [DKV09], and later extended to infinite words with limit averaging or supremum as value functions [CDH10b, CDH10a, CDH09a]. While weighted automata over semirings can express several quantitative properties [Moh02], they cannot express long-run average properties that weighted automata with limit averaging can [CDH10b]. However, even weighted automata with limit averaging cannot express the following basic quantitative property (the example is from [CHO17]).

Example 1.1. 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 and r'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)* [CHO17]. A 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. Slave automata are equipped with finite-word value functions to compute the returned values, which are then aggregated by the master automaton using an infinite-word value function. 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 [CHO17], for example, nested weighted automata can

[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.

express the long-run average response time property (see [CHO17, Example 5]). It has been shown in [CHO17] 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. Classical questions for automata are *emptiness* and *universality* that ask for the existence and respectively non-existence of words that are accepted. Their natural extensions have been studied in the quantitative setting as well (such as for weighted automata and NWA) [CDH10b, CHO17].

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-case/worst-case scenarios (such as the best-case/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 represent 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 [CHJS15]. 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 [HKNP06, BK08]. 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* distribution, which asks whether the probability in the distribution question is exactly 1.

Our contributions. In this work we consider several classical value functions, namely, Sup, Inf, LimSup, LimInf, LimAvg for infinite words, and Max, Min, Sum, Sum^B, Sum^+ (where Sum^B is the sum bounded by B, and Sum^+ is the sum of absolute values) for finite words. First, we establish translations (in both directions) between automata with monitor counters and a subclass of nested weighted automata, called bounded-width nested weighted automata [CHO16a], 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.

- LimSup *and* LimInf *functions.* We consider deterministic nested weighted automata with LimSup and 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 7.4 for further details).

- LimAvg *function.* We consider deterministic nested weighted automata with 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 7.13).
- Inf *and* Sup *functions.* We consider deterministic nested weighted automata with Sup and 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 exponential time; other than the Sum function, the expected value, the distribution, and the almost-sure problems are PSpace-hard and can be solved in ExpTime; and for the 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 LimSup and Sup functions always coincide, whereas we show a substantial complexity gap for probabilistic questions (see Remark 8.10 and Remark 8.11 for further details).
- *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 automaton with respect to a Markov chain, and also show that for nested weighted automata with LimSup value function for the master automaton and 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 because in NWA unbounded number of slaves can be active. Our results are summarized in Theorem 7.3 (in Section 7.1), Table 2 (in Section 8), and Theorem 7.12 (in Section 7.2). 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 for a deterministic (LimInf; 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.
- We show that the expected value of a deterministic (LimAvg; Sum)-automaton $\mathbb{A}$ coincides with the expected value of the following deterministic (non-nested) 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 $\mathfrak{B}$.
- For a deterministic (Inf; Sum)-automaton $\mathbb{A}$ and $C > 0$ we define $\mathbb{A}^C$ as the deterministic (Inf; Sum)-automaton obtained from $\mathbb{A}$ by stopping every slave automaton if it exceeds C steps. We show that for every deterministic (Inf; 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.

This paper is an extended and corrected version of [CHO16b, CHO16c]. We present detailed proofs, which could not be published in [CHO16b] due to space constrains. The main corrections over [CHO16b] are: Table 1 and Theorem 10 (Theorem 5.1 in this paper). These flaws in [CHO16b] are consequences of a false claim about duality between deterministic (INF; g)-automata (resp., (LIMINF; g)-automata) and deterministic (SUP; $-g$)-automata (resp. (LIMSUP; $-g$)-automata). This duality indeed holds for non-deterministic NWA or deterministic NWA that accept all words (or almost all words for probabilistic questions). However, it does not extend to all deterministic NWA (see [CHO17] and Remark 6.4).

Moreover, we discuss extensions of our main results in Section 10, which is a new contribution. We consider there (1) the case of NWA that do not accept almost all words, (2) the probabilistic variant of the quantitative inclusion problem for NWA, and (3) the parametric complexity of the probabilistic questions, in which we fix the NWA and ask for the complexity w.r.t. the Markov chain. The parametric complexity corresponds to evaluation of a fixed specification (for example average response time from Example 1.1) represented by an NWA on a system represented by a Markov chain. Finally, we elaborate on translations between NWA and automata with monitor counters discussed in [CHO16b, CHO16c].

Related works. Quantitative automata and logic have been extensively and intensively studied in recent years. The book [DKV09] presents an excellent collection of results of weighted automata on finite words. Weighted automata on infinite words have been studied in [CDH10b, CDH10a, DR06]. The extension to weighted automata with monitor counters over finite words has been considered (under the name of cost register automata) in [ADD+13]. A version of nested weighted automata over finite words has been studied in [BGMZ10], and nested weighted automata over infinite words have been studied in [CHO17]. Several quantitative logics have also been studied, such as [BCHK14, BMM14, ABK14]. 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 [FV96, Put94], and for multiple objectives and their combinations [CMH06, Cha07, CFW13, BBC+11, CKK15, BCFK15, FKN+11, CD11, BDK14, BKKW14]. However, these works do not consider properties that are expressible by nested weighted automata (such as average response time) or automata with monitor counters.

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 $Q_0 = \{q_0\}$ 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* if and only if 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* if and only if 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 and their semantics. 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.

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 runs 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) *Min and max*: $\textsc{Min}(\pi) = \min_{i=1}^{n}(C(\pi))[i]$ and $\textsc{Max}(\pi) = \max_{i=1}^{n}(C(\pi))[i]$,
(2) *Sum*: $\textsc{Sum}(\pi) = \sum_{i=1}^{n}(C(\pi))[i]$,
(3) *Absolute sum*: $\textsc{Sum}^{+}(\pi) = \sum_{i=1}^{n} \mathsf{Abs}((C(\pi))[i])$ is the sum of the absolute values of the weights (Abs denotes the absolute value of a number), and
(4) *Bounded sum*: $\textsc{Sum}^{B}(\pi) = \textsc{Sum}(\pi)$, if for all prefixes π' of π we have $\mathsf{Abs}(\textsc{Sum}(\pi')) \leq B$, otherwise $\textsc{Sum}^{B}(\pi)$ is equal to first crossed bound $-B$ or B, i.e., the bounded sum value function returns the sum if all the partial absolute sums are below a bound B, otherwise it returns the first crossed bound. Weighted automata with the bounded-sum value function can model bounded quantities such as energy with the lower and the upper bound [BMR+18].

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

We denote the above class of infinite-word value functions as

$$\mathsf{InfVal} = \{\textsc{Sup}, \textsc{Inf}, \textsc{LimSup}, \textsc{LimInf}, \textsc{LimAvg}\}.$$

Silent moves. Consider a $(\mathbb{Z} \cup \{\bot\})$-labeled automaton. We regard 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: 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 values of counters. 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, and $Q_0 \subseteq Q$ is the set of initial states,
(3) 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
(4) 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 terminates 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 $\pi = \pi[0]\pi[1] \ldots$ of elements from $Q \times (\mathbb{Z} \times \{\bot\})^n$ is a *run* of $\mathcal{A}^{\text{m-c}}$ on a word $w = w[1]w[2] \ldots$ if

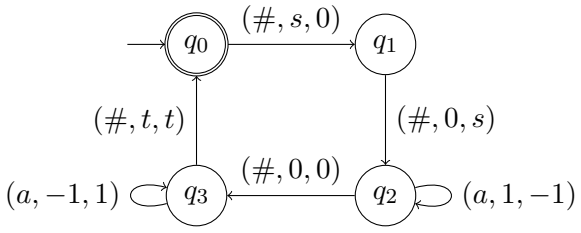

FIGURE 1. The automaton $\mathcal{A}_{\text{diff}}$ computing the maximal difference between the lengths of blocks of a's at odd and the following even positions.

(1) $\pi[0] = \langle q_0, \vec{\perp} \rangle$ 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 $u[j] = \perp$ and $u'[j] = 0$,
   (b) if $v[j] = t$, then $u[j] \in \mathbb{Z}$ and $u'[j] = \perp$, 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 $\perp$ 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. Otherwise, if no counter has been started at position i, we define $\pi^W[i] = \perp$, Observe that for an accepting π, the sequence π^W contains infinitely positions with integer values. The semantics of automata with monitor counters is given, similarly to weighted automata, by applying the value function to the sequence π^W with $\perp$ elements removed.

Remark 3.1. Automata with monitor counters are very similar in spirit to *cost register automata* considered in [ADD+13]. 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 cost register automata before.

Example 3.2 (Blocks difference)**.** Consider an alphabet $\Sigma = \{a, \#\}$ and the language $\mathcal{L}$ defined as $(\#^2 a^* \# a^* \#)^\omega$. We consider a quantitative property "the maximal block-length difference between odd and even positions" on the words from the language $\mathcal{L}$, i.e., the value of word $\#^2 a^{n[1]} \# a^{n[2]} \#^3 \ldots$ is $\sup_{0 \leq i} |n[2 \cdot i + 1] - n[2 \cdot i + 2]|$. This property can be expressed by a SUP-automaton $\mathcal{A}_{\text{diff}}$ with two monitor counters depicted in Figure 1.

The automaton $\mathcal{A}_{\text{diff}}$ 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 $\mathcal{A}_{\text{diff}}$ 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 $\sup_{0 \leq i} |n[2 \cdot i + 1] - n[2 \cdot i + 2]|$.

3.2. **Nested weighted automata.** In this section we describe nested weighted automata introduced in [CHO17], and closely follow the description of [CHO17]. For more details and illustrations of such automata we refer the reader to [CHO17]. 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 in the word of the master automaton and works on some finite part of the input word. Once the 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

$$\langle \mathcal{A}_{\text{mas}}; f; \mathfrak{B}_1, \ldots, \mathfrak{B}_k \rangle$$

such that

(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 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 an NWA $\mathbb{A}$ on an infinite word w is an infinite sequence $(\Pi, \pi_1, \pi_2, \ldots)$ such that (1) $\Pi = \Pi[0]\Pi[1] \ldots$ is a run of $\mathcal{A}_{\text{mas}}$ on $w = w[1]w[2] \ldots$; (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 subword $w[i, j]$ of the input word w. 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 $v(\pi_1)v(\pi_2)\dots$ 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) in all slave automata accepting states have no outgoing transitions. 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 reach an accepting state again.

Bounded width. An NWA has *width* k if and only if k is the minimal number such that in every accepting run at every position at most k slave automata are active.

Example 3.3 (Average response time with bounded requests)**.** Consider an alphabet Σ consisting of requests r, grants g and idle instructions i. 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 [CHO17] that NWA can express ART. However, the automaton from [CHO17] 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 a $(\textsc{LimAvg}; \textsc{Sum})$-automaton $\mathbb{A}$. Such an automaton $\mathbb{A} = (\mathcal{A}_{\text{mas}}; \textsc{LimAvg}; \mathfrak{B}_1, \mathfrak{B}_2)$ 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 i 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. To state correctness of translation, we first define equivalence.

Equivalence of quantitative automata. We say that $\mathcal{A}_1, \mathcal{A}_2$, each being a weighted automaton, an automaton with monitor counters or an NWA over infinite words from Σ, are *equivalent* if and only if for all words $w \in \Sigma^\omega$ we have $\mathcal{A}_1(w) = \mathcal{A}_2(w)$.

Now, we state the main translation lemma:

Lemma 3.4 (Translation Lemma)**.** *For every value function $f \in \mathsf{InfVal}$ on infinite words we have the following:*

(1) *Every deterministic f-automaton with monitor counters $\mathcal{A}^{\text{m-c}}$ can be transformed in polynomial time into an equivalent deterministic $(f; \textsc{Sum})$-automaton of bounded width.*
(2) *Every non-deterministic (resp., deterministic) $(f; \textsc{Sum})$-automaton of bounded width can be transformed in exponential time into an equivalent non-deterministic (resp., deterministic) f-automaton with monitor counters.*

Before the formal proof, we illustrate below the key ideas of the above translations of Lemma 3.4 to automata from Examples 3.2 and 3.3.

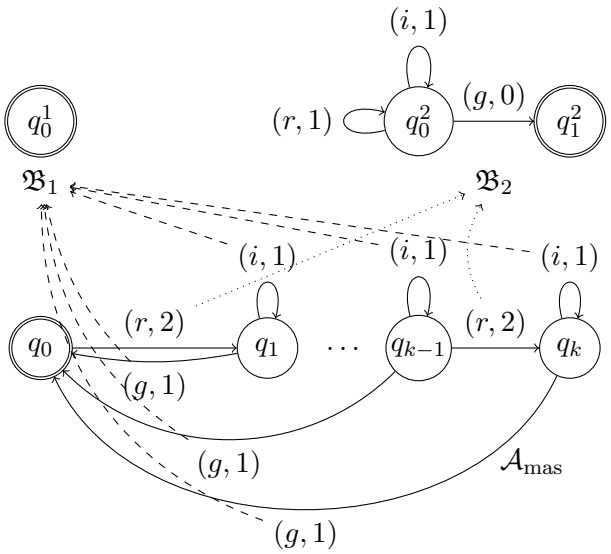

FIGURE 2. The (LIMAVG; SUM)-automaton computing the average response time over words with infinite number of requests and grants such that between any two grants there are at most k requests.

Example 3.5 (Translation of automata with monitor counters to nested weighted automata)**.** Consider a deterministic automaton $\mathcal{A}$ 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 $\mathcal{A}_{\text{diff}}$ from Example 3.2 to a (SUP; SUM)-automaton of width bounded by 3.

Example 3.6 (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. We construct an automaton with monitor counters $\mathcal{A}_{\mathbb{A}}$, which simulates the master automaton and up to k slave automata running in 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 3.3 to the automaton with monitor counters $\mathcal{A}_{\mathbb{A}}$. The set of states of $\mathcal{A}_{\mathbb{A}}$ there is $\{q_0,\ldots,q_k\}\times(\{q_0^2,\bot\})^k$, 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 $(q_0,\bot,\ldots,\bot),(q_1,q_0^2,\bot,\ldots,\bot),\ldots,(q_k,q_0^2,\ldots,q_0^2)$, i.e., the reachable part of $\mathcal{A}_{\mathbb{A}}$ is isomorphic (in the sense of graphs) to the master automaton of $\mathbb{A}$.

Proof of Lemma 3.4. **(Translation of automata with monitor counters to NWA)**: Consider a deterministic f-automaton $\mathcal{A}^{\text{m-c}}$ with k monitor counters and the set of states Q^{m-c}. We define an $(f;\text{SUM})$-automaton $\mathbb{A}$, which consists of a master automaton $\mathcal{A}_{\text{mas}}$ and slave automata $\{\mathfrak{B}_{i,q}\mid i\in\{1,\ldots,k\},q\in Q^{m-c}\}\cup\{\mathfrak{B}_\bot\}$. The slave automaton $\mathfrak{B}_\bot$ is a

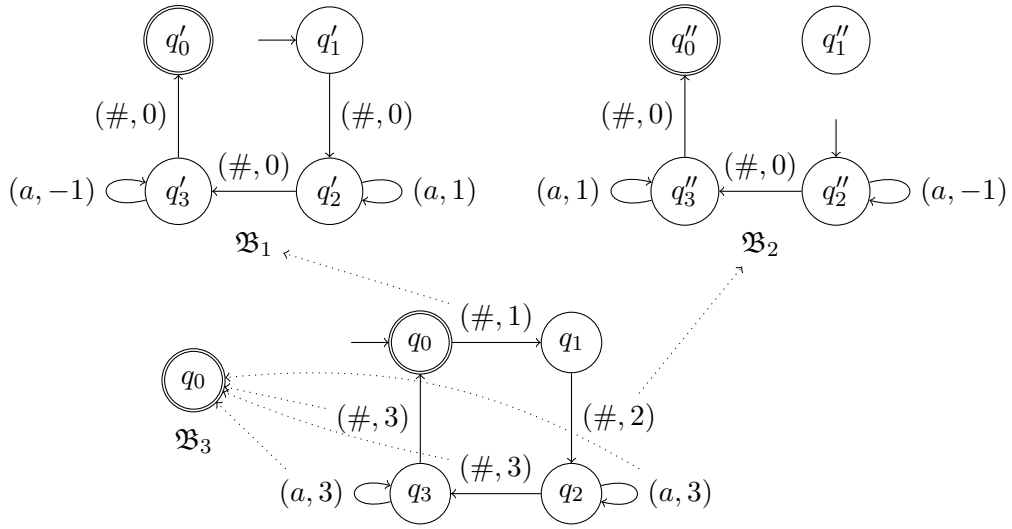

FIGURE 3. A nested weighted automaton resulting from translation of the automaton $\mathcal{A}_{\text{diff}}$ from Example 3.2. The master automaton is obtained from $\mathcal{A}_{\text{diff}}$ (see Figure 1) by changing the labels of transitions. All slave automata are defined based on $\mathcal{A}_{\text{diff}}$; each slave automaton corresponds to $\mathcal{A}_{\text{diff}}$ starting in a state q', in which a counter i is initialized, and contains all the transitions that can be taken before the counter i is terminated.

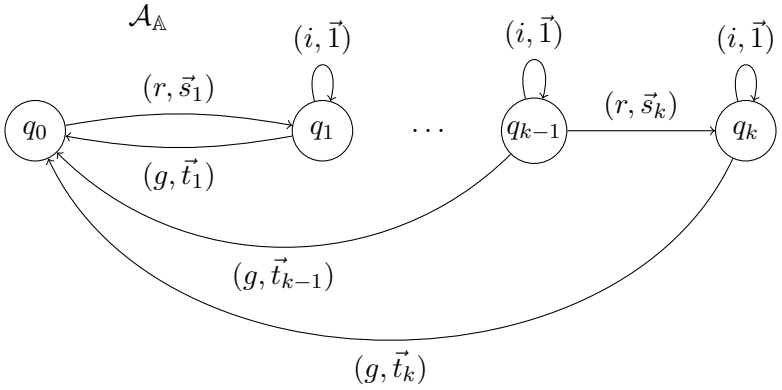

FIGURE 4. The (reduced) result of translation of the automaton $\mathbb{A}$ from Example 3.3 to an automaton with monitor counters. All vectors have dimension k. Vector $\vec{1}$ denotes the vector with all components equal 0. Vector $\vec{s_i}$ denotes the whose i-th component is s and other components are 1. Vector $\vec{t_i}$ denotes the vector whose components $1, \ldots, i$ are t and the remaining components are 0.

dummy automaton, i.e., it has only a single state which is both the initial and the accepting state. Invoking such an automaton is equivalent to taking a silent transition (with no weight). Next, the master automaton $\mathcal{A}_{\text{mas}}$ and slave automata $\{\mathfrak{B}_{i,q} \mid i \in \{1, \ldots, k\}, q \in Q^{m-c}\}$ are variants of $\mathcal{A}^{\text{m-c}}$, i.e., they share the underlying transition structure. The automaton

$\mathcal{A}_{\text{mas}}$ simulates $\mathcal{A}^{\text{m-c}}$, i.e., it has the same states and the transitions among these states as $\mathcal{A}^{\text{m-c}}$. However, whenever $\mathcal{A}^{\text{m-c}}$ activates counter i, the master automaton invokes the slave automaton $\mathfrak{B}_{i,q}$, where q is their current state (both $\mathcal{A}_{\text{mas}}$ and the simulated $\mathcal{A}^{\text{m-c}}$). The accepting condition of $\mathcal{A}_{\text{mas}}$ is the same as of $\mathcal{A}^{\text{m-c}}$. We can construct $\mathcal{A}_{\text{mas}}$ in polynomial time in $|\mathcal{A}^{\text{m-c}}|$. For every $i \in \{1, \ldots, k\}$, the slave automaton $\mathfrak{B}_{i,q}$ keeps track of counter i, i.e., it simulates $\mathcal{A}^{\text{m-c}}$ and applies instructions of $\mathcal{A}^{\text{m-c}}$ for counter i to its value. That is, whenever $\mathcal{A}^{\text{m-c}}$ changes the value of counter i by m, the automaton $\mathfrak{B}_{i,q}$ takes a transition of the weight m. Finally, $\mathfrak{B}_{i,q}$ terminates precisely when $\mathcal{A}^{\text{m-c}}$ terminates counter i. The automaton $\mathfrak{B}_{i,q}$ can be constructed in polynomial time in $|\mathcal{A}^{\text{m-c}}|$. There are at most $k \cdot |\mathcal{A}^{\text{m-c}}|$ such slave automata and each of them has the size bounded by $|\mathcal{A}^{\text{m-c}}|$. Therefore, $|\mathbb{A}|$ is polynomial in $|\mathcal{A}^{\text{m-c}}|$ and can be constructed in polynomial time in $|\mathcal{A}^{\text{m-c}}|$.

The semantics of automata with monitor counters implies that $\mathbb{A}$ accepts if and only if $\mathcal{A}^{\text{m-c}}$ accepts and, for every word, the sequences of weights produced by the runs of $\mathbb{A}$ and $\mathcal{A}^{\text{m-c}}$ on that word coincide. Therefore, the values of $\mathbb{A}$ and $\mathcal{A}^{\text{m-c}}$ coincide on every word.

(Translation of NWA of bounded width to automata with monitor counters): We show that non-deterministic (resp., deterministic) f-automata with monitor counters subsume non-deterministic (resp., deterministic) $(f; \textsc{Sum})$-automata of bounded width. Consider a non-deterministic $(f; \textsc{Sum})$-automaton $\mathbb{A}$ with width bounded by k. We define an f-automaton $\mathcal{A}^{\text{m-c}}$ with k monitor counters that works as follows. Let Q_{mas} be the set of states of the master automaton of $\mathbb{A}$ and Q_s be the union of the sets of states of the slave automata of $\mathbb{A}$. The set of states of $\mathcal{A}^{\text{m-c}}$ is $Q_{mas} \times (Q_s \cup \{\bot\}) \times \cdots \times (Q_s \cup \{\bot\}) = Q_{mas} \times (Q_s \cup \{\bot\})^k$. The automaton $\mathcal{A}^{\text{m-c}}$ simulates runs of the master automaton and slave automata by keeping track of the state of the master automaton and states of up to k active slave automata. If there are less than k active slave automata, $\mathcal{A}^{\text{m-c}}$ uses $\bot$ to mark slots that can be used in the future to simulate slave automata. Moreover, it uses counters to simulate the values of slave automata, i.e., whenever a slave automaton is activated, $\mathcal{A}^{\text{m-c}}$ simulates the execution of this automaton and assigns some counter i to that automaton. Next, when the simulated slave automaton takes a transition of the weight m the automaton $\mathcal{A}^{\text{m-c}}$ changes the value of counter i by m. Finally, $\mathcal{A}^{\text{m-c}}$ terminates counter i when the corresponding slave automaton terminates. The size of $|\mathcal{A}^{\text{m-c}}|$ is bounded by $|\mathbb{A}|^k$ and it can be constructed in time $O(|\mathbb{A}|^k)$.

Since $\mathbb{A}$ has width bounded by k, the simulating automaton $\mathcal{A}^{\text{m-c}}$ never runs out of counters to simulate slave automata. Moreover, as it simulates runs of the master automaton and slave automata of $\mathbb{A}$, there is a one-to-one correspondence between runs of $\mathcal{A}^{\text{m-c}}$ and runs of $\mathbb{A}$ and accepting runs of $\mathbb{A}$ correspond to accepting runs of $\mathcal{A}^{\text{m-c}}$. Finally, the sequence of weights for the master automaton determined by a given run of $\mathbb{A}$ coincides with the sequence of weights of $\mathcal{A}^{\text{m-c}}$ on the corresponding run. Therefore, the values of $\mathbb{A}$ and $\mathcal{A}^{\text{m-c}}$ coincide on every word. Thus, non-deterministic f-automata with monitor counters subsume non-deterministic $(f; \textsc{Sum})$-automata of bounded width.

Now, assume that $\mathbb{A}$ is deterministic. Therefore, the master automaton and all slave automata are deterministic and accepting states of slave automata have no outgoing transitions. We claim that $\mathcal{A}^{\text{m-c}}$ is deterministic as well. Consider a state $(s_1, q_1, \ldots, q_k)$ from $Q_{mas} \times (Q_s \cup \{\bot\})^k$ and every letter $a \in \Sigma$. The successor over a of s_1 is uniquely determined as the master automaton is deterministic. For all q_i, which are not accepting, the successor states of deterministic slave automata are uniquely determined. If some state q_i is accepting, then the slave automaton has no outgoing transition and the successor state is $\bot$. Finally, for q_i equal $\bot$, the one with the least index becomes the initial state of the newly

invoked slave automaton and the other states remain $\bot$. Therefore, the automaton $\mathcal{A}^{\text{m-c}}$ is deterministic. □

A direct consequence of Lemma 3.4 is the following theorem:

Theorem 3.7. *For every $f \in \mathsf{InfVal}$, deterministic bounded-width $(f, \textsc{Sum})$-automata and deterministic f-automata with monitor counters are expressively equivalent.*

Remark 3.8 (Discussion)**.** Theorem 3.7 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 automata with monitor counters

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 $\lambda \in \mathbb{Q}$, decide whether there is a word w with $\mathcal{L}_{\mathcal{A}}(w) \leq \lambda$?
- **Universality**: Given an automaton $\mathcal{A}$ and $\lambda \in \mathbb{Q}$, decide whether for all words 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 (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, $E \colon S \times \Sigma \times S \mapsto [0,1]$ 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 [Fel71]). We will denote the unique probability measure defined by $\mathcal{M}$ as $\mathbb{P}_{\mathcal{M}}$, and the associated expectation measure as $\mathbb{E}_{\mathcal{M}}$.

We define *the uniform probability measure* $\mathcal{U}$ such that for every $u \in \Sigma^*$ we have $\mathbb{P}_{\mathcal{U}}(u \cdot \Sigma^\omega) = |\Sigma|^{-|u|}$. It can be defined by a single-state Markov chain, in which all transitions are self-loops labled with the same probability $\frac{1}{|\Sigma|}$.

Automata as random variables. Note that deterministic weighted automata, NWA or automata with monitor counters all define functions $h\colon \Sigma^\omega \mapsto \mathbb{R}$, which are measurable with respect to probability measures given by Markov chains, and hence these functions can be interpreted as random variables. Therefore, 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**: $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda) = \mathbb{P}_{\mathcal{M}}(\{w \mid \mathcal{L}_{\mathcal{A}}(w) \leq \lambda\})$ 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 $\lambda \in \mathbb{Q}$, to compute $\mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$.

Questions (Q1) and (Q2) 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 $\lambda \in \mathbb{Q}$ 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 4.1 (Expected average response time)**.** Consider an NWA $\mathbb{A}$ from Example 3.3. 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 $\{r, g, i\}$. 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 [CHO17] which is presented in Table 1.

New results. Due to Lemma 3.4, decidability of deterministic $(f; \textsc{Sum})$-automata implies decidability of deterministic automata with monitor counters with the value function f. However, the undecidability result of NWA does not imply undecidability for automata with monitor counters as the NWA in the reduction may have unbounded width. We present the undecidability result for NWA of bounded width, which implies undecidability of the

		Inf LimInf	Sup LimSup	LimAvg
Min, Max Sum^B	Empt.	PSpace-comp.		
	Univ.		PTime	
Sum$^+$	Empt.	PSpace-comp.		ExpSpace
	Univ.		PTime	
Sum	Empt.	PSpace-comp.	Undecidable	Open
	Univ.		PTime	

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.

emptiness problem for automata with monitor counters. Thus, our following result completes the decidability picture also for automata with monitor counters (i.e., the decidability results coincide with the Sum row of Table 1).

Theorem 5.1. *The emptiness problem is undecidable for deterministic* Sup*-automata (resp.,* LimSup*-automata) with* 8 *monitor counters.*

Proof. We show undecidability of the emptiness problem for deterministic (LimSup, Sum)-automata of width 8. The proof for deterministic (Sup, Sum)-automata is virtually the same. Then, the theorem follows from the translation lemma (Lemma 3.4).

We show a reduction from the halting problem for deterministic two-counter machines, which is undecidable [Min61]. Let $\mathbf{M}$ be a deterministic two-counter machine and let Q be the set of states of $\mathbf{M}$. We define a deterministic (LimSup, Sum)-automaton $\mathbb{A}$ of width 8 such that $\mathbb{A}$ has a run of the value not exceeding 0 if and only if $\mathbf{M}$ has an accepting computation.

Consider the alphabet $\Sigma = Q \cup \{1, 2, \#, \$\}$. We encode computations of $\mathbf{M}$ as a sequence of configurations separated by #. A single configuration of $\mathbf{M}$, where the machine is in the state q, the first counter has the value x and the second y is encoded by the word $q1^x2^y$. Finally, computations of $\mathbf{M}$ are separated by \$. We define the automaton $\mathcal{A}$ that for a word $w \in \Sigma^*$ returns the value 0 if (some infinite suffix of) w encodes a sequence valid accepting computations of $\mathbf{M}$. Otherwise, $\mathbb{A}$ returns the value at least 1.

The automaton $\mathbb{A}$ works as follows. On a single computation, i.e., between symbols \$, $\mathbb{A}$ checks consistency of the transitions by checking two conditions: (C1) Boolean consistency, and (C2) counter consistency. The condition (C1) states that encoded subsequence configurations, which are represented by subwords $q1^x2^y\#q'1^{x'}2^{y'}$, are consistent with the transition function of $\mathbf{M}$ modulo counter values, i.e., under counter abstraction to values 0 and "strictly positive". Observe that a finite automaton can check that. The conditions that need to be checked are as follows: (C1-1) Boolean parts of transitions are consistent; the automaton checks only emptiness/nonemptiness of counters and based on that verifies whether a subword $q1^x2^y\#q'$ is valid w.r.t. transitions of $\mathbf{M}$. For example, consider transition $(q, \bot, +, q', +1, -1)$ of $\mathbf{M}$ stating that "if $\mathbf{M}$ is in state q, the first counter is 0 and the second counter is positive, then change the state to q' increment the first counter and decrement the second one". This transition corresponds to the regular expression $q2^+\#q'$. (C1-2) The initial and final configurations in each computation (between \$ symbols) are respectively $q_I1^02^0$ and $q_f1^02^0$. (C1-3) The word encodes infinitely many computations, i.e.,

the word contains infinitely many \$ symbols. The last conditions rejects words encoding non-terminating computations.

To check the condition (C2), $\mathbb{A}$ uses slave automata. It uses 4 slave automata to check transitions between even and odd positions and the other 4 slave automata to check validity of the remaining transitions. Then, between even and odd positions it uses 2 slave automata for each counter of $\mathbf{M}$. These slave automata encode the absolute values between the intended values of counters (i.e., assuming that counter values are consistent with the instructions) and the actual values. For example, for a subword $q1^x2^y\#q'1^{x'}2^{y'}$, the automaton $\mathbb{A}$ checks whether the value of counter 1 is consistent with transition $(q, \bot, +, q', +1, -1)$ in the following way. The first slave automaton ignores letters 2 and initially decrements its value at every letter 1 until it reads letter # (where its value is $-x$). Next, it switches its mode and increments its value at letters 1 while ignoring letters 2. In that way its value upon reading $q1^x2^y\#q'1^{x'}2^{y'}$ equals $-x + x'$. Finally, it increments its value by 1. Thus, the value of the slave automaton is $-x + x' + 1$. The second slave automaton works in a similar way, but it decrements whenever the first counter increments and vice versa. At the end, the value of the second slave automaton is $x - x' - 1$. Observe that the maximum of these value is $|x - (x' + 1)|$, which is 0 if and only if the value of counter 1 is consistent with the transition $(q, \bot, +, q', +1, -1)$. It follows that the supremum over the values returned by all slave automata is 0 only if all counter values are consistent with the transitions. Therefore, the value LimSup of the whole word is 0 if and only if starting at some point all computations are valid and accepting. The latter is possible only if $\mathbf{M}$ has at least one such a computation. Otherwise, the value of LimSup is at least 1.

Observe that this construction works for Sup as well. □

6. Basic results on probabilistic questions

In this section we discuss basic properties of the probabilistic questions and present some basic facts about Markov chains. Next, in the following Section 7 and Section 8 we study the probabilistic questions for NWA. We consider there separately NWA with LimInf, LimSup, LimAvg value functions for the master automaton (Section 7) and NWA with Inf, Sup value functions for the master automaton (Section 8).

We begin with the discussion on the acceptance by an NWA.

6.1. 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 NWA is *almost-surely accepting*, i.e., the set of words whose runs are accepting has probability 1. In Section 7 we consider all the computational problems for NWA which are almost-surely accepting. This assumption does not influence the complexity of computational questions related to the expected value, but has an influence on the complexity of distribution questions, which we discuss in Section 10.1.

Proposition 6.1. *Given a deterministic NWA $\mathbb{A}$ and a Markov chain $\mathcal{M}$, we can decide in polynomial time whether $\mathbb{P}_{\mathcal{M}}(\{w \mid \mathsf{Acc}(w) \neq \emptyset\}) = 1$?*

Proof. First, the master automaton has to accept almost all words. We can check this in polynomial time by considering the master automaton as a Büchi automaton and applying the classical methods [BK08].

For all pairs (q, s), where q is the initial state of some slave automaton $\mathfrak{B}_i$ and s is a state of the Markov chain $\mathcal{M}$, we check that either $\mathfrak{B}_i$ is never invoked while $\mathcal{M}$ is in the state s or $\mathfrak{B}_i$ almost-surely accepts (w.r.t. the distribution given by $\mathcal{M}$ started in s). Observe that $\mathfrak{B}_i$ almost-surely accepts if for every finite word u generated by $\mathcal{M}$ starting in the state s there exists its finite extension u' (generated by $\mathcal{M}$), which is accepted by $\mathfrak{B}_i$ (i.e., $\mathfrak{B}_i$ terminates in an accepting state and returns a finite value). One can easily check that this condition is necessary and sufficient, and it can be checked in polynomial time [BK08]. □

Remark 6.2 (Almost-sure acceptance)**.** The answer to the expected value problem does not change even without the assumption. We show next that without the almost-sure acceptance condition, the distribution questions become similar to Inf and Sup value functions. Hence, in Section 7 we consider the almost-sure acceptance property, and presented the conceptually interesting results. Moreover, classically weighted automata have been considered without any acceptance conditions (i.e., all words are accepted), and then the almost-sure acceptance is trivially ensured.

6.2. **Duality property between infimum and supremum.** In Section 7 and Section 8, when we consider the expected value and the distribution, in most cases we consider only Inf and LimInf value functions, and by duality, we obtain results for Sup and LimSup value functions, respectively. The only exception are $(\textsc{Inf}, \textsc{Sum}^+)$-automata and $(\textsc{Sup}, \textsc{Sum}^+)$-automata, which have to be considered separately. For every value function $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+\}$ we define $-g$ as follows: $-\textsc{Min} = \textsc{Max}$, $-\textsc{Max} = \textsc{Min}$ and $-g = g$ for $g \in \{\textsc{Sum}^B, \textsc{Sum}\}$.

Lemma 6.3. *For every $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+\}$, every deterministic $(\textsc{Sup}; g)$-automaton (resp. $(\textsc{LimSup}; g)$-automaton) $\mathbb{A}_1$ accepting almost-all words can be transformed to a deterministic $(\textsc{Inf}; -g)$-automaton (resp. $(\textsc{LimInf}; -g)$-automaton) $\mathbb{A}_2$ of the same size such that for almost all words w we have $\mathcal{L}_{\mathbb{A}_1}(w) = -\mathcal{L}_{\mathbb{A}_2}(w)$.*

Proof. The automaton $\mathbb{A}_2$ is obtained from $\mathbb{A}_1$ by multiplying all the weights by -1. □

Remark 6.4 (Limited duality)**.** As we work under the almost-sure acceptance assumption, the above duality result implies that all complexity results for NWA with Inf (resp,. LimInf) master value function transfer to NWA with Sup (resp., LimSup) master value function. However, this duality does not extend to the classical problems of emptiness and universality. Indeed, if $\mathbb{A}_1$ does not have an accepting run over w then $\mathcal{L}_{\mathbb{A}_1}(w) = \mathcal{L}_{\mathbb{A}_2}(w) = \infty$ (where $\mathbb{A}_1, \mathbb{A}_2$ are from Lemma 6.3). This cannot be fixed with a different construction as the master automaton in NWA has Büchi acceptance conditions and deterministic Büchi automata are not closed under complementation. This leads to different complexity results for these automata. In particular, the emptiness problem for deterministic $(\textsc{Sum}, \textsc{Sum})$-automata is undecidable, while the universality problem for deterministic $(\textsc{Inf}, \textsc{Sum})$-automata is PSpace-complete (Table 1).

6.3. **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 rationals 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 $r\colon S \times \Sigma \times S \mapsto \mathbb{Q}$.

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 example, 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.
- **(bottom SCCs)**: An SCC Q is a *bottom* 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 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 $\langle \Sigma, Q \times S, \langle q_0, s_0\rangle, E', r'\rangle$, 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 [CDH09b].

Fact 6.5. 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.

7. Results on limit value functions

In this section we study NWA with LimInf, LimSup and LimAvg value functions for the master automaton. All these value functions are prefix independent and hence in a (deterministic) strongly-connected almost-surely accepting NWA, returning a value λ is a tail event, which has probability either 0 or 1. It follows that almost all words have the same value. We use this property to establish polynomial-time algorithms for all probabilistic questions.

Throughout this section we assume that all NWA that are almost-surely accepting, i.e., for almost all words w, the run on w is accepting. In the classical setting of weighted automata, which have no accepting condition, the almost-sure acceptance is trivially satisfied. This is a conceptually interesting case as we are mainly interested in the quantitative aspect of (nested) weighted automata. Moreover, we can check whether a given deterministic NWA is almost-surely accepting in polynomial time (Proposition 6.1). If it is not, the expected value is either infinite or undefined and hence the complexity of the expected question does not change. However, the complexity of the distribution question changes and we discuss it in Section 10.1.

7.1. **LimInf and LimSup value functions.** In this section we study NWA with LimInf and LimSup value functions for the master automaton. 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.

The key ideas. The value functions LimInf and LimSup return values that occur infinitely often. Therefore, in a strongly connected Markov chain, for every finite word u, the set of infinite words that contain u infinitely many times has probability 0 or 1. We extend this property to establish some sort of 0-1 law for NWA with LimInf or LimSup master value function (Lemma 7.1), which states that if the product of the Markov chain and the master automaton of $\mathbb{A}$ form an SCC, then almost all words have the same value which is the infimum returned by slave automata of $\mathbb{A}$.

In the following result we do not assume that a given NWA accepts almost surely.

Lemma 7.1. *Let $g \in \mathsf{FinVal}$, $\mathcal{M}$ be a Markov chain, and $\mathbb{A}$ be a deterministic* (Inf; g)*-automaton (resp.,* (LimInf; g)*-automaton). Assume that the master automaton of $\mathbb{A}$ is strongly connected on $\mathcal{M}$. Then, the following conditions hold:*

1. *either runs on almost all words are accepting or runs on almost all words are rejecting,*
2. *there exists a unique value λ such that $\mathbb{P}_{\mathcal{M}}(\{w \mid \mathcal{L}_{\mathbb{A}}(w) = \lambda\}) = 1$,*
3. *$|\lambda| \leq |\mathbb{A}| \cdot |\mathcal{M}|$ or λ is extreme, i.e., $\lambda \in \{-\infty, \infty\}$ for $g \in \{\text{Sum}, \text{Sum}^+\}$ or $\lambda \in \{-B, B, \infty\}$ for $g = \text{Sum}^B$, and*
4. *given $\mathcal{M}$ and $\mathbb{A}$, the value λ can be computed in polynomial time in $|\mathcal{M}| + |\mathbb{A}|$.*

Proof. We first show duality (1) and then consider the case of almost-surely accepting NWA.

Duality. Since $\mathbb{A}$ is deterministic, all runs of $\mathbb{A}$ on the distribution given by $\mathcal{M}$ correspond to the paths in $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, where $\mathcal{A}_{\text{mas}}$ is the master automaton of $\mathbb{A}$. Since $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ is strongly connected, any finite path occurs infinitely often with the probability either 0 or 1 [BK08]. Therefore, either almost all runs satisfy the Büchi condition of $\mathcal{A}_{\text{mas}}$ or almost all runs violate it. We can check in polynomial time which of these two cases holds.

Now, either in almost all words all slave automata accept or there is a state (q, s) of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, where the master automaton invokes some slave automaton $\mathfrak{B}_i$, which does not accept with positive probability. In the latter case, almost all words are rejected by $\mathbb{A}$. Observe that there exists a finite word u generated by $\mathcal{M}$ in state s such that no finite extension of u (generated by $\mathcal{M}$ in state s) is accepted by $\mathfrak{B}_i$. Again, since $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ is strongly connected, the sequence of states (q, s) followed by states forming the word u occurs infinitely often in almost all words. It follows that on almost all words $\mathbb{A}$ does not have an accepting run. We can check existence of (q, s), which is visited infinitely often in $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ and some invoked slave automaton rejects with positive probability, in polynomial time in $|\mathbb{A}| + |\mathcal{M}|$.

Recall that for a word w, the run of $\mathbb{A}$ on w is accepting if and only if the run of $\mathcal{A}_{\text{mas}}$ on w is accepting (it satisfies its Büchi condition), and the runs of all invoked slave automata are accepting. In summary, one of the following holds: (i) on almost all words w, the run of $\mathbb{A}$ on w is accepting, or (ii) on almost all words w, the run of $\mathbb{A}$ on w is rejecting, and hence $\mathcal{L}_{\mathbb{A}}(w) = \infty$. We can decide in polynomial time which of these cases holds. Moreover, if (ii) holds, then $\mathbb{P}_{\mathcal{M}}(\{w \mid \mathcal{L}_{\mathbb{A}}(w) = \infty\}) = 1$, i.e., $\lambda = \infty$. We next assume that (i) holds.

Almost-surely accepting NWA. Assume that $\mathbb{A}$ is almost-surely accepting. Consider a state (q, s) of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, where the master automaton invokes some slave automaton $\mathfrak{B}_i$, and the slave automaton $\mathfrak{B}_i$ attains its minimal value on the following letters. (If $\mathfrak{B}_i$ does not attain its minimal value, we consider a sequence that tends to $-\infty$.) Therefore, almost all runs

contain the considered sequence infinitely often. It follows that the value of almost all runs is the minimum over reachable states (q, s) from $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ and transitions (s, a, s') of $\mathcal{M}$ of the minimal value the slave automaton invoked in (q, a, q') can achieve on all words generated by $\mathcal{M}$ starting with the transition (s, a, s'). This value can be computed in polynomial time in $|\mathcal{M}| + |\mathbb{A}|$.

Observe that either some invoked slave automaton can reach a cycle with the sum of weights being negative and iterate over it (i.e., a cycle in $\mathfrak{B}_i \times \mathcal{M}$), and hence the minimum is $-\infty$ (the minimum is bounded by $-B$ for $g = \textsc{Sum}^B$). Otherwise, the minimum is attained over some word which does not form a cycle in $\mathfrak{B}_i \times \mathcal{M}$, i.e., the length of this word is bounded by the number of states of $\mathfrak{B}_i$ times the size of $\mathcal{M}$. Since we consider weights to be given in the unary notation, the sum of weights over such a word is bounded by $|\mathfrak{B}_i|$ times $|\mathcal{M}|$. Thus, $|\lambda| \leq |\mathbb{A}| \cdot |\mathcal{M}|$ or $\lambda = -\infty$ (resp., $-B$ for $g = \textsc{Sum}^B$). If $g = \textsc{Sum}^B$ and $B < |\mathbb{A}| \cdot |\mathcal{M}|$, then $\lambda = B$. □

Lemma 7.1 implies the following main lemma of this section.

The key ideas. Consider a (LimInf; g)-automaton (resp., (LimSup; g)-automaton) $\mathbb{A}$ that accepts almost all words. The value $\mathcal{L}_{\mathbb{A}}(w)$ depends only on the infinite behavior of the (unique) run of $\mathbb{A}$ on w, which ends up in some bottom SCC (for almost all words w). In a bottom SSC, almost all words have the same value, which can be computed in polynomial time (Lemma 7.1). Thus, to compute $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$, we compute probabilities of reaching each of the bottom SCCs and values of $\mathbb{A}$ in these SSCs. In a similar way, we can compute $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$.

Lemma 7.2. *Let $g \in$ FinVal. For a deterministic almost-surely accepting* (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.*

Proof. First, we discuss how to compute the expected and the distribution questions of a deterministic (LimInf; Sum)-automaton $\mathbb{A}$.

The value of (LimInf; Sum)-automaton $\mathbb{A}$ on a word depends on weights that appear infinitely often. Since $\mathbb{A}$ reaches some bottom SCC with probability 1, we can neglect values of slave automata returned before the master automaton $\mathcal{A}_{\text{mas}}$ (of $\mathbb{A}$) reaches a bottom SCC of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$. Thus, the expected value of (LimInf; Sum)-automaton $\mathbb{A}$ w.r.t. a Markov chain $\mathcal{M}$ can be computed in the following way. Let $S_1, \ldots, S_l$ be all bottom SCCs of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$. We compute probabilities $p_1, \ldots, p_l$ of reaching the components $S_1, \ldots, S_l$ respectively. These probabilities can be computed in polynomial time [BK08]. Next, for every component S_i we compute in polynomial time the unique value m_i, which $\mathbb{A}$ returns on almost every word whose run ends up in S_i (Lemma 7.1). The expected value $\mathbb{E}_{\mathcal{M},\mathbb{A}}$ is equal to $p_1 \cdot m_1 + \cdots + p_l \cdot m_l$. Observe that, given a value λ, the distribution $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ is equal to the sum the probabilities p_i over such i that $m_i \leq \lambda$. Hence, the expected and the distribution questions can be computed in polynomial time.

Due to Lemma 6.3, the case of LimSup reduces to the case of LimInf. All value functions from FinVal are special cases of Sum. This concludes the proof. □

Lemma 7.2 states the the expected question and the distribution question can be computed in polynomial time. The remaining probabilistic questions are their special cases and hence they can be computed in polynomial time as well. The following theorem summarizes results of this section.

Theorem 7.3. *Let $g \in \mathsf{FinVal}$. All probabilistic questions for deterministic almost-surely accepting* (LimInf; g)*-automata (resp.,* (LimSup; g)*-automata) can be solved in polynomial time.*

Remark 7.4 (Contrast with classical questions)**.** Consider the results on classical questions shown in Table 1 and the results for probabilistic questions we establish in Theorem 7.3. While for classical questions the problems are PSpace-complete or undecidable, we establish polynomial-time algorithms for all probabilistic questions.

7.2. **The expected question for the LimAvg value function.** In this section we study NWA with the LimAvg value function for the master automaton. We essentially show that to compute the expected value of a given (LimAvg; g)-automaton, it suffices to substitute in each transition invoking a slave automaton $\mathfrak{B}_i$ by the expected value of $\mathfrak{B}_i$.

We assume that considered (LimAvg; g)-automata are deterministic and accept almost all words. We discuss the case of almost-surely accepting NWA. This assumption does not change the complexity of the expected question (Remark 6.2).

Lemma 7.5. *Let $g \in \mathsf{FinVal}$. Given a Markov chain $\mathcal{M}$ and a deterministic almost-surely accepting* (LimAvg; g)*-automaton $\mathbb{A}$, the value $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ can be computed in polynomial time.*

Overview. We present the most interesting case when $g = \textsc{Sum}$. Let $\mathbb{A}$ be a (LimAvg; 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.

In the remaining part of this section we show that the expected value of $\mathbb{A}$ w.r.t. $\mathcal{M}$ and the expected value of $\mathcal{M}^{\mathbb{A}}$ coincide (Lemma 7.6). 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 $|\mathbb{A}| + |\mathcal{M}|$.

Lemma 7.6. *Let $\mathbb{A}$ be a deterministic almost-surely accepting* (LimAvg; Sum)*-automaton. The values $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{E}(\mathcal{M}^{\mathbb{A}})$ coincide.*

In the following we prove Lemma 7.6. First, we show that lemma for (LimAvg; Sum)-automata in which duration of runs of slave automata is bounded by some $N \in \mathbb{N}$. Next, we show how to solve the general case of all (LimAvg; Sum)-automata by the reduction to this special case.

Note that $\mathcal{M}^{\mathbb{A}}$ can have silent moves labeled by $\perp$. Indeed, an automaton that starts in the accepting state always returns value $\perp$, which is its expected value. Before we continue, we discuss computing the expected values of Markov chains with silent moves.

Expected limit averages of Markov chains with silent moves. Let $\mathcal{M}_{\text{sil}}$ be a Markov chain labeled by $\mathbb{Q} \cup \{\perp\}$, where $\perp$ corresponds to a silent transition. We consider the limit average value function with silent moves sil(LimAvg), which applied to a sequence $a_1 a_2 \dots$ of elements of $\mathbb{Q} \cup \{\perp\}$ removes all $\perp$ symbols are applies the standard LimAvg function (defined on the sequences of rational numbers in the similar way as for integers) to the sequence consisting of the remaining elements. The expected value of the limit average of a path in $\mathcal{M}_{\text{sil}}$ can be computed by a slight modification of the standard method for Markov chains without silent transitions [FV96].

Without loss of generality we can assume that $\mathcal{M}_{\text{sil}}$ is strongly connected. It if is not, we can compute bottom strongly connected components $B_1, \ldots, B_k$ of $\mathcal{M}_{\text{sil}}$, then compute probabilities $p_1, \ldots, p_k$ of reaching these components and $\mathbb{E}(\mathcal{M}_{\text{sil}}) = \sum_{i=1}^{k} p_i \mathbb{E}(\mathcal{M}_{\text{sil}}[B_i])$, where $\mathcal{M}_{\text{sil}}[B_i]$ is $\mathcal{M}_{\text{sil}}$ with the initial state being some state from B_i.

Assume that $\mathcal{M}_{\text{sil}}$ is strongly connected and contains non-silent transitions. We associate with each transition $e = (s, a, s')$ of $\mathcal{M}_{\text{sil}}$ a real-valued variable $x[e]$, which is the frequency of transition e. Formally, given an infinite path ρ in $\mathcal{M}_{\text{sil}}$ we define $|\rho[1, n]|_e$ as the number of transitions e among first n transitions of ρ. Let $e_1, \ldots, e_k$ be all non-silent transitions in $\mathcal{M}_{\text{sil}}$. We state a system of equations and inequalities such that for almost all infinite paths ρ in $\mathcal{M}_{\text{sil}}$ and all $i \in \{1, \ldots, k\}$ we have

$$\lim_{n\to\infty} \frac{|\rho[1, n]|_{e_i}}{|\rho[1, n]|_{e_1} + \cdots + |\rho[1, n]|_{e_k}} = x[e_i]. \tag{7.1}$$

These equations and inequalities are as follows:

(E1) for every transition $e = (s, a, s')$ we put

$$x[(s, a, s')] = E(s, a', s') \cdot \sum_{s'' \in S_{\mathcal{M}_{\text{sil}}}, a' \in \Sigma} x[(s'', a', s)],$$

the frequency of (s, a, s') is the probability of taking (s, a, s') from s multiplied by the sum of frequencies of all transitions leading to s,

(E2) $x[e_1] + \cdots + x[e_k] = 1$, where the sum of frequencies of all non-silent transitions is 1,

(E3) $0 \leq x[e]$ for every transition e.

Following the argument for Markov chains without silent moves [FV96, BK08], we can show that the above system of equations has the unique solution and it satisfies (7.1). Then, the expected limit average of $\mathcal{M}_{\text{sil}}$ is given as $c(e_1) \cdot x[e_1] + \cdots + c(e_k) \cdot x[e_k]$, where $c(e_i)$ is the cost of transition e_i.

7.2.1. *The expected value in the bounded-duration case.* First, we show that Lemma 7.6 holds if we assume that for some $N > 0$ all slave automata take at most N transitions.

Lemma 7.7. *Let $\mathbb{A}$ be an almost-surely accepting deterministic* (LimAvg; Sum)*-automaton in which duration of runs of slave automata is bounded by N and let $\mathcal{M}^{\mathbb{A}}$ be the weighted Markov chain corresponding to $\mathbb{A}$. The values $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{E}(\mathcal{M}^{\mathbb{A}})$ coincide.*

Before we proceed with the proof of Lemma 7.7, we present an example.

Example 7.8. Recall the average response time property (ART) presented in Example 1.1. We consider a variant of ART called N-bounded ART. We define the *N-bounded response time* of a request as the minimum of N and the number of steps to the following grant. The N-bounded ART is the limit average of N-bounded response times over all requests. The N-bounded ART property can be computed by a (LimAvg; Sum$^+$)-automaton $\mathbb{A}_N$ that at each request invokes a slave automaton that takes at most N steps and computes the N-bounded response time. The NWA $\mathbb{A}_N$ invokes a dummy slave automaton on the remaining transitions, which corresponds to taking a silent transition. For simplicity, we restrict the events to requests and grants only (no idle events).

We consider the uniform probability measure over $\{r, g\}^\omega$ (without events i), which can be given by a single-state Markov chain $\mathcal{M}_U$. First, the expected N-bounded response time

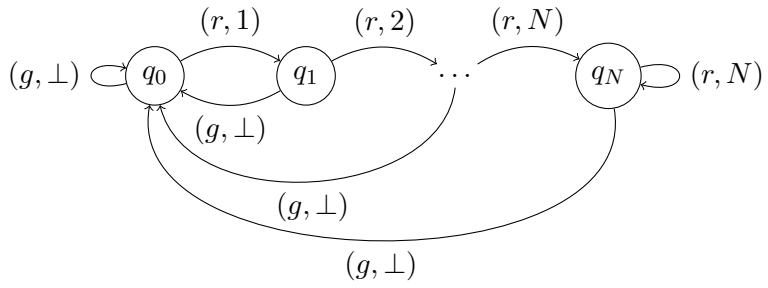

FIGURE 5. The automaton $\mathcal{A}_N$

equals

$$\sum_{i=1}^{N-1} \left(\frac{1}{2}\right)^i \cdot i + N \sum_{i=N}^{\infty} \left(\frac{1}{2}\right)^i = 2 - (N-1) \cdot \left(\frac{1}{2}\right)^{N-1} + N \cdot \left(\frac{1}{2}\right)^{N-1} = 2 - \left(\frac{1}{2}\right)^{N-1}.$$

The Markov chain $\mathcal{M}^{\mathbb{A}_N}$ has a single state with a self-loop labeled by a grant of the empty weight $\perp$ and a self-loop labeled by a request of weight $2 - (\frac{1}{2})^{N-1}$. Therefore, $\mathbb{E}(\mathcal{M}^{\mathbb{A}_N}) = 2 - (\frac{1}{2})^{N-1}$.

Now, to compute $\mathbb{E}_{\mathcal{M}_U}(\mathbb{A}_N)$ we construct a $\mathsf{sil}(\textsc{LimAvg})$-automata $\mathcal{A}_N$ that works as follows. In each block r^*g, for the first N requests, the automaton $\mathcal{A}_N$ assigns weights $1, 2, \ldots, N$, and then for the following requests it assigns weight N. It takes silent transitions over grants g. The automaton $\mathcal{A}_N$ is depicted in Figure 5.

Observe that in each block $r^k g$ the N-bounded response times are

$$\min(N, k), \min(N, k-1), \ldots, 2, 1,$$

while the weights returned by the automaton $\mathcal{A}$ are

$$1, 2, \ldots, \min(N, k-1), \min(N, k).$$

On all words w with infinitely many grants the values $\mathcal{L}_{\mathbb{A}_N}(w)$ and $\mathcal{L}_{\mathcal{A}_N}(w)$ are equal. Therefore, $\mathbb{E}_{\mathcal{M}_U}(\mathbb{A}_N) = \mathbb{E}_{\mathcal{M}_U}(\mathcal{A}_N)$.

We compute the $\mathbb{E}_{\mathcal{M}_U}(\mathcal{A}_N)$ in the standard way. Let x_i be the density of visiting state q_i in $\mathcal{A}_i$. Clearly, for $i = 0, \ldots, N-1$, we have $x_i = (\frac{1}{2})^i$ and $x_N = (\frac{1}{2})^{N-1}$. Since transitions labeled with g are silent, $\mathbb{E}_{\mathcal{M}_U}(\mathcal{A}_N) = (\sum_{i=0}^{N-1} x_i \cdot (i+1)) + N \cdot x_N = 2 - (\frac{1}{2})^{N-1}$. Thus, the values $\mathbb{E}_{\mathcal{M}}(\mathbb{A}_N)$ and $\mathbb{E}(\mathcal{M}^{\mathbb{A}_N})$ coincide.

The plan of the proof. We define a $\mathsf{sil}(\textsc{LimAvg})$-automaton $\mathcal{A}$ that simulates runs of $\mathbb{A}$; the value on $\mathcal{A}$ on every word coincides with $\mathbb{A}$. Then, we transform the Markov chain $\mathcal{A} \times \mathcal{M}$ into a Markov chain $\mathcal{M}_E$ by adjusting its weights only. We change all weights to the empty weight $\perp$ except for the transitions corresponding to the invocation of slave automata, where the weight is the expected value of the invoked slave automaton w.r.t. the distribution given by $\mathcal{M}$ in the current state. In the proof we argue that the expected values of limit average of $\mathcal{A} \times \mathcal{M}$ and $\mathcal{M}_E$ coincide. We show that by looking at the linear equations corresponding to computing the expected limit average of each of the Markov chains. Basically, the frequency of each transition is the same in both Markov chains and changing the value of the slave automaton from its actual value to the expected value does not affect the solution to the set of equations. Next, we observe that runs of slave automata

past the first transition do not matter. Indeed, all runs of slave automata are accepting and all weights past the first transition are 0. Thus, we can reduce $\mathcal{M}_E$ to a Markov chain $\mathcal{M}_R$ by projecting out information about the runs of slave automata past the first transition. Finally, we observe that the Markov chain $\mathcal{M}_R$ is in fact $\mathcal{M}^{\mathbb{A}}$. Hence, we have shown that

$$\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = \mathbb{E}_{\mathcal{M}}(\mathcal{A}) = \mathbb{E}(\mathcal{M}_E) = \mathbb{E}(\mathcal{M}_R) = \mathbb{E}(\mathcal{M}^{\mathbb{A}})$$

Proof. Every slave automaton of $\mathbb{A}$ takes at most N steps. Therefore, $\mathbb{A}$ has width bounded by N. Moreover, without loss of generality, we assume that each slave automaton takes transitions of weight 0 except for the last transition, which may have a non-zero weight, and all slave automata are either trivial, i.e., they start in the accepting state and take no transitions, or they take precisely N transitions. Basically, slave automata may keep track of the accumulated values and the number of steps in their states.

The automaton $\mathcal{A}$. Let Q_{mas} be the set of states of the master automaton of $\mathbb{A}$ and let Q_s be the union of the set of states of the slave automata of $\mathbb{A}$. We define $\mathcal{A}$ as a $\mathsf{sil}(\textsc{LimAvg})$ automaton over the set of states $Q_{mas} \times (Q_s \cup \{\bot\})^N$. The component Q_{mas} is used to keep track of the run of the master automaton while the component $(Q_s \cup \{\bot\})^N$ is used to keep track of up to N slave automata running concurrently. The symbol $\bot$ corresponds to an empty slot that can be used to simulate another slave automaton. Since $\mathbb{A}$ has width bounded by N, the automaton $\mathcal{A}$ can simulate the Boolean part of the run of $\mathbb{A}$. The weight of a transition of $\mathcal{A}$ is either $\bot$ if no automaton terminates or it is the value of a terminating slave automaton (non-trivial slave automata take precisely N steps, so at most one can terminate at each position). Transitions at which no slave automaton terminates are silent transitions. The automata $\mathbb{A}$ and $\mathcal{A}$ encounter the same weights but differ in their aggregation. The value of a slave automaton is associated to the position at which it is invoked, while in $\mathcal{A}$ it is associated with the position at which the slave automaton terminates. However, these positions differ by N, therefore the limit averages of both sequences coincide. Hence, for every word w, the values $\mathcal{L}_{\mathbb{A}}(w)$ and $\mathcal{L}_{\mathcal{A}}(w)$ coincide. It follows that $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = \mathbb{E}_{\mathcal{M}}(\mathcal{A})$.

The Markov chain $\mathcal{M}_E$. We define $\mathcal{M}_E$ as $\mathcal{A} \times \mathcal{M}$ with altered weights defined as follows. All transitions which correspond to the invocation of a slave automaton $\mathfrak{B}_i$ with the state of the Markov chain $\mathcal{M}$ being s have weight equal to the expected value of $\mathfrak{B}_i$ w.r.t. the distribution given by $\mathcal{M}$ starting in the state s. Other transitions are silent.

Expected values of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ and $\mathcal{M}_E$ coincide. Assume that $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ and $\mathcal{M}_E$ are strongly connected. If they are not, we can apply the following reasoning for all bottom strongly connected components of both Markov chains as they have the same underlying structure.

Recall that the expected limit average of a Markov chain with silent moves is given by $c(e_1) \cdot x[e_1] + \cdots + c(e_k) \cdot x[e_k]$ where variables $x[e]$, over all transitions e, form a solution to the system of equations and inequalities (E1), (E2) and (E3), and $e_1, \ldots, e_k$ are all non-silent transitions. Now, observe that the equations (E1) and inequalities (E3) are the same for both Markov chains $\mathcal{A} \times \mathcal{M}$ and $\mathcal{M}_E$ as they have the same structure with the same probabilities. The equation (E2) is, in general, different for $\mathcal{A} \times \mathcal{M}$ and for $\mathcal{M}_E$. However, non-silent transitions of $\mathcal{A} \times \mathcal{M}$, denoted by $e_1, \ldots, e_k$, are all states at which at least one slave automaton terminates, while non-silent transitions of $\mathcal{M}_E$, denoted by $e'_1, \ldots, e'_l$ are all states at which some (non-trivial) slave automaton is invoked. Observe that every terminating slave automaton has been invoked, and, in $\mathcal{A}$, every invoked slave

automaton terminates. Therefore, the sum of frequencies of invocations and terminations of slave automata are equal, i.e., equations (E1) imply

$$x[e_1] + \cdots + x[e_k] = x[e'_1] + \cdots + x[e'_l].$$

It follows that the unique solution to equations and inequalities (E1), (E2) and (E3) corresponding to $\mathcal{A} \times \mathcal{M}$ and to $\mathcal{M}_E$ are the same. It remains to show that

$$c(e_1) \cdot x[e_1] + \cdots + c(e_k) \cdot x[e_k] = c'(e'_1) \cdot x[e'_1] + \cdots + c'(e'_l) \cdot x[e'_l],$$

where c (resp. c') are weights in $\mathcal{A} \times \mathcal{M}$ (resp., $\mathcal{M}_E$).

Since $c'(e')$ is the expected value of the slave automaton started at e', the expected value $c'(e')$ is given by $c'(e') = \sum_{e'' \in T} p(e', e'') \cdot c(e'')$, where T is the set of transitions that correspond the the final transitions of the slave automaton started at the transition e', and $p(e', e'')$ is the probability of reaching the transition e'' from e' omitting the set T. Indeed, each (non-trivial) slave automaton takes precisely N transitions, hence at each position at most one non-trivial slave automaton terminates and $c(e'')$ is the value of the slave automaton terminating at e''. Therefore, $c'(e') = \sum_{e'' \in T} p(e', e'') \cdot c(e'')$.

Now, we take $c'(e'_1) \cdot x[e'_1] + \cdots + c'(e'_l) \cdot x[e'_l]$ and substitute each $c(e'_i)$ by the corresponding $c'(e'_i) = \sum_{e'' \in T_i} p(e', e'') \cdot c(e'')$. Then, we now group in all the terms by e'' and we get

$$c'(e'_1) \cdot x[e'_1] + \cdots + c'(e'_l) \cdot x[e'_l] = \sum_{i=1}^{k} c(e_i) \cdot \big(x[e'_1] \cdot p(e'_1, e_i) + \cdots + x[e'_l] \cdot p(e'_l, e_i)\big)$$

Observe that the frequency of taking the transition e_i at which some slave automaton $\mathfrak{B}$ terminates is equal to the sum of frequencies on transitions at which this slave automaton $\mathfrak{B}$ has been invoked, in which each frequency is multiplied by the probability of reaching the terminating transition e_1 from a given invoking transition. Therefore, we have

$$x[e'_1] \cdot p(e'_1, e_i) + \cdots + x[e'_l] \cdot p(e'_l, e_i) = x[e_i].$$

It follows that

$$c(e_1) \cdot x[e_1] + \cdots + c(e_k) \cdot x[e_k] = c'(e'_1) \cdot x[e'_1] + \cdots + c'(e'_l) \cdot x[e'_l]$$

and $\mathbb{E}_{\mathcal{M}}(\mathcal{A}) = \mathbb{E}(\mathcal{M}_E)$.

The Markov chain $\mathcal{M}_R$. We construct $\mathcal{M}_R$ from $\mathcal{M}_E$ by projecting out the component $(Q_s \cup \{\bot\})^N$. We claim that this step preserves the expected value. First, observe that the distribution is given by an unaffected component $\mathcal{M}$ and the weights depend only on the state of the Markov chain $\mathcal{M}$ and the state of the master automaton $\mathcal{A}_{\text{mas}}$. Thus, projecting out the component $(Q_s \cup \{\bot\})^N$ does not affect the expected value, i.e., $\mathbb{E}_{\mathcal{M}}(\mathcal{M}_E) = \mathbb{E}(\mathcal{M}_R)$. Now, observe that the set of states of $\mathcal{M}_R$ is $Q_{mas} \times Q_{\mathcal{M}}$. Observe that the probability and the weights of the transitions of $\mathcal{M}_R$ match the conditions of the definition of $\mathcal{M}^{\mathbb{A}}$. Therefore, $\mathcal{M}_R = \mathcal{M}^{\mathbb{A}}$. □

7.2.2. *Reduction to the bounded-duration case.* Let $\mathbb{A}$ be a (LimAvg; Sum)-automaton. For every N, we define $\mathbb{A}^N$ as $\mathbb{A}$ with the bound N imposed on slaves, i.e., each slave automaton terminates either by reaching an accepting state or when it takes the N-th step. Let $\mathcal{M}_N^{\mathbb{A}}$ be the Markov chain that corresponds to $\mathbb{A}^N$. Observe that as N tends to infinity, weights in $\mathcal{M}_N^{\mathbb{A}}$ converge to the weights in $\mathcal{M}^{\mathbb{A}}$. It remains to be shown that, as N tends to infinity, the expected values of $\mathbb{A}^N$ converge to the expected value of $\mathbb{A}$. We show in the following

Lemma 7.9 that random variables generated by $\mathbb{A}^N$ converge in probability to the random variable generated by $\mathbb{A}$, i.e., for every $\epsilon > 0$ we have

$$\lim_{N\to\infty} \mathbb{P}_{\mathcal{M}}(\{w \mid |\mathcal{L}_{\mathbb{A}}(w) - \mathcal{L}_{\mathbb{A}^N}(w)| \geq \epsilon\}) = 0$$

Convergence in probability implies convergence of the expected values. It follows that the expected values of $\mathbb{A}$ and $\mathcal{M}^{\mathbb{A}}$ coincide.

Lemma 7.9. *The random variables defined by $\{\mathcal{L}_{\mathbb{A}^N}\}_{N\geq 0}$ converge in probability to the random variable defined by $\mathbb{A}$.*

Example 7.10. Recall Example 7.8. Lemma 7.9 implies that with N tending to infinity, the limit of the expected N-bounded ARTs converges to the expected ART. For $N > 0$, the expected N-bounded ART is $\mathbb{E}_{\mathcal{M}_U}(\mathcal{A}_N) = 2 - (\frac{1}{2})^{N-1}$. Therefore, the expected ART is $\lim_{N\to\infty} \mathbb{E}_{\mathcal{M}_U}(\mathcal{A}_N) = 2$.

Proof. We define an (LimAvgSup; Sum)-automaton $\mathbb{A}^{\geq N}$ as the automaton obtained from $\mathbb{A}$ in the following way. First, each slave automaton take transitions of weight 0 for the first (up to) N steps, past which it takes transitions of weight 1 until it terminates. Second, the value function of the master automaton is LimAvgSup defined on $a_1, a_2, \ldots$ as $\text{LimAvgSup}(a_1 \ldots) = \limsup_n \frac{1}{n} \sum_{i=1}^{n} a_i$. Intuitively, the automaton $\mathbb{A}^{\geq N}$ computes the limit average (supremum) of the steps slave automata take above the threshold N. Let C be the maximal absolute weight in slave automata of $\mathbb{A}$. Then, for every word w we have

$$\mathcal{L}_{\mathbb{A}^N}(w) - C \cdot \mathcal{L}_{\mathbb{A}^{\geq N}}(w) \leq \mathcal{L}_{\mathbb{A}}(w) \leq \mathcal{L}_{\mathbb{A}^N}(w) + C \cdot \mathcal{L}_{\mathbb{A}^{\geq N}}(w).$$

It follows that

$$\mathbb{P}_{\mathcal{M}}(\{w \mid |\mathcal{L}_{\mathbb{A}}(w) - \mathcal{L}_{\mathbb{A}^N}(w)| \geq \epsilon\}) = \mathbb{P}_{\mathcal{M}}(\{w \mid |\mathcal{L}_{\mathbb{A}^{\geq N}}(w)| \geq \frac{\epsilon}{C}\})$$

We show that with N increasing to infinity, probabilities $\mathbb{P}_{\mathcal{M}}(\{w \mid |\mathcal{L}_{\mathbb{A}^{\geq N}}(w)| \geq \frac{\epsilon}{C}\})$ converge to 0. From that we conclude that random variables $\mathcal{L}_{\mathbb{A}^N}$ converge in probability to $\mathcal{L}_{\mathbb{A}}$ as N tends to infinity.

Observe that for every word w and every N we have $0 \leq \mathcal{L}_{\mathbb{A}^{\geq N}}(w)$ and $\mathcal{L}_{\mathbb{A}^{\geq N}}(w) \geq \mathcal{L}_{\mathbb{A}^{\geq N+1}}(w)$. Therefore, we only need to show that for every $\epsilon > 0$ there for N large enough $\mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\geq N}) \leq \epsilon$. Then, by Markov inequality, $\mathbb{P}_{\mathcal{M}}(\{w \mid |\mathcal{L}_{\mathbb{A}^{\geq N}}(w)| \geq \sqrt{\epsilon}) < \sqrt{\epsilon}$.

To estimate the value of $\mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\geq N})$ we consider $\mathsf{sil}(\text{LimAvgSup})$-automata $\mathbb{A}[K, i]$ defined as follows. The automaton $\mathbb{A}[K, i]$ simulates the master automaton $\mathbb{A}$ and slaves that are invoked at positions $\{K \cdot l + i \mid l \in \mathbb{N}\}$. For every $l > 0$, the transition at the position $K \cdot (l + 1) + i$ has the weight 1 if the slave invoked at the position $K \cdot l + i$ works for at least K steps. Otherwise, this transition has weight 0. On the remaining positions, transitions have weight 0. Observe that due to distributivity of the limit supremum, the limit average supremum of the number of slave automata that take at least K steps at a given word w is bounded by $\sum_{i=0}^{K-1} \mathcal{L}_{\mathbb{A}[K,i]}(w)$. It follows that for every word w we have $\mathcal{L}_{\mathbb{A}^{\geq N}}(w) \leq \sum_{K\geq N} \sum_{i=0}^{K-1} \mathcal{L}_{\mathbb{A}[K,i]}(w)$. Therefore,

$$(*) \quad \mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\geq N}) \leq \sum_{K\geq N} \sum_{i=0}^{K-1} \mathbb{E}_{\mathcal{M}}(\mathbb{A}[K, i]).$$

Now, we estimate $\mathbb{E}_{\mathcal{M}}(\mathbb{A}[K, i])$. Let n be the maximal size of a slave automaton in $\mathbb{A}$ and let k be the number of slave automata. We assume, without loss of generality, that every state of slave automata is reached along some run on words generated by $\mathcal{M}$.

Now, observe that from every state of slave automata some accepting state is reachable. Otherwise, there would be a set of strictly positive probability at which $\mathbb{A}$ does not accept. Moreover, as it is reachable, it is reachable within n steps. Therefore, the probability p such that any slave automaton in any state terminates after next n steps is greater than 0. It follows that $\mathbb{E}_{\mathcal{M}}(\mathbb{A}[K,i]) \leq \frac{1}{K} p^{\lfloor \frac{K}{n} \rfloor}$. We apply this estimate to (*) and obtain $\mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\geq N}) \leq \sum_{K \geq N} p^{\lfloor \frac{K}{n} \rfloor} \leq n \cdot \frac{p^{\lfloor \frac{N}{n} \rfloor}}{1-p}$. Therefore, $\mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\geq N})$ converges to 0 as N increases to infinity. Finally, this implies that $\mathbb{A}^N$ converges in probability to $\mathbb{A}$ as N tends to infinity. □

7.3. The distribution question for the LimAvg value function.

Lemma 7.11. *Let $g \in$ FinVal. Given a Markov chain $\mathcal{M}$, a deterministic almost-surely accepting* (LimAvg; g)*-automaton $\mathbb{A}$ and a value λ, the value $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in polynomial time.*

The 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 bottom 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 bottom SCC C of $\mathcal{A}$, the expected value over words that end up in C and the probability of reaching C. Both values can be computed in polynomial time (see Lemma 7.5).

Proof. Let $\mathbb{A}$ be a deterministic (LimAvg; Sum)-automaton with the master automaton $\mathcal{A}_{\text{mas}}$ and let $\mathcal{M}$ be a Markov chain. Moreover, let $\mathcal{M}^{\mathbb{A}}$ be the Markov chain obtained from $\mathcal{M}$ and $\mathbb{A}$. We show that the distribution $\mathbb{D}_{\mathcal{M},\mathbb{A}}$ and the distribution defined by $\mathcal{M}^{\mathbb{A}}$ coincide.

The single-SCC case. Assume that $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ is an SCC. Observe that the event "the value of $\mathbb{A}$ equals λ" is a tail event w.r.t. the Markov chain $\mathcal{M}$, i.e., it does not depend on finite prefixes. Therefore, its probability is either 0 or 1 [Fel71]. It follows that the value of almost all words is equal to the expected value of $\mathbb{A}$. Now, $\mathcal{M}^{\mathbb{A}}$ is structurally the same as $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, hence it is also an SCC. Therefore, also in $\mathcal{M}^{\mathbb{A}}$ almost all words have the same value, which is equal to $\mathbb{E}(\mathcal{M}^{\mathbb{A}})$. As $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = \mathbb{E}(\mathcal{M}^{\mathbb{A}})$ (Lemma 7.6) we have $\mathbb{D}_{\mathcal{M},\mathbb{A}}$ and the distribution defined by $\mathcal{M}^{\mathbb{A}}$ coincide.

The general case. Consider the case where $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ consists multiple bottom SCCs $S_1, \ldots, S_k$. Using conditional probability, we can repeat the single-SCC-case argument to show that in all bottom SCCs $S_1, \ldots, S_k$ the values of $\mathbb{A}$ are the same and equal to the expected values in these SCCs. Similarly, in each bottom SCC of $\mathcal{M}^{\mathbb{A}}$, all words have the same value, which is equal to the expected value in that SCC. Since $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ is structurally the same as $\mathcal{M}^{\mathbb{A}}$, each SCC $S_1, \ldots, S_k$ corresponds to an SCC in $\mathcal{M}^{\mathbb{A}}$. Lemma 7.6 states that $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = \mathbb{E}(\mathcal{M}^{\mathbb{A}})$. By applying Lemma 7.6 to $\mathcal{M}$ and $\mathbb{A}$ with different initial states of $\mathcal{M}$ and $\mathcal{A}_{\text{mas}}$ (in each $S_1, \ldots, S_k$), we infer that in every SCC $S_1, \ldots, S_k$ the expected values of $\mathbb{A}$ and $\mathcal{M}^{\mathbb{A}}$ coincide. Therefore, the distribution $\mathbb{D}_{\mathcal{M},\mathbb{A}}$ and the distribution defined by $\mathcal{M}^{\mathbb{A}}$ coincide. □

Theorem 7.12. *Let $g \in$ FinVal. All probabilistic questions for deterministic almost-surely accepting* (LimAvg; g)*-automata can be solved in polynomial time.*

Remark 7.13 (Contrast with classical questions)**.** Our results summarized in Theorem 7.12 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.

8. Results on non-limit value functions

In this section we study NWA with INF, SUP value functions for the master automaton. In contrast to LIMINF and LIMSUP value functions, for which all probabilistic questions can be answered in polynomial time (Theorem 7.3), we show #P-hardness, PSPACE-hardness and uncomputability results for INF, SUP value functions as well as exponential-time upper bounds in some cases.

In contrast to LIMINF and LIMSUP value functions the almost-sure acceptance condition does not change the complexity of the probabilistic questions. We show the lower bounds under the almost-sure acceptance condition. However, for the upper bounds we do not assume almost-sure acceptance of NWA. We first present several hardness results for INF and SUP value functions.

8.1. Lower bounds for all value functions for slave automata.

We present the key ideas of the hardness results.

PSPACE*-hardness.* NWA can invoke multiple slave automata working independently over the same finite subword, which we use to express the problem: is the intersection of given regular languages empty, which is PSPACE-complete. We transform given DFA $\mathcal{A}_1, \ldots, \mathcal{A}_k$ into slave automata that return 1 if the original automaton accepts and 0 otherwise. The resulting NWA $\mathbb{A}$ is deterministic, accepts almost all words and $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0) = 1$ if the intersection is empty. Note however, that words of the minimal length in the intersection can have exponential length and their probability can be doubly-exponentially small. Therefore, even if $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0) \neq 1$, the difference between $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0)$ and 1 can be small and hence PSPACE-hardness does not apply to the approximation problems (which we establish below).

#P-hardness. We show #*P*-hardness of the approximate probabilistic questions by reduction from #SAT, which is #*P*-complete [Val79, Pap03]. The #SAT problem asks for the number of variable assignments satisfying a given CNF formula φ. In the proof, the input word gives an assignment, and each slave automaton checks the satisfaction of one clause and returns 1 if it is satisfied and 0 otherwise. Therefore, all slave automata return 1 if and only if all clauses are satisfied, and hence we can compute the number of satisfying assignments of φ from $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0)$, where $\mathcal{U}$ is the uniform distribution over infinite words. The lower bounds hold even under the additional almost-sure acceptance condition.

Lemma 8.1 (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 almost-surely accepting* (INF; g)*-automaton (resp.,* (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 almost-surely accepting* (INF; g)*-automaton (resp.,* (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.*

Proof. We present the following argument for $g = \textsc{Min}$. The same proof works for $g = \textsc{Max}$. Lemma 6.3 implies that problems in (1) and (2) for nested weighted automaton with Sup value function reduce to the corresponding problems for nested weighted automaton with Inf value function. Since Min can be regarded as a special case of $\textsc{Sum}^B, \textsc{Sum}^+$ or Sum, the result holds for these functions as well. Hence, we consider only the case of (Inf, Min)-automata.

PSpace-hardness: We show PSpace-hardness by reduction from the emptiness problem for the intersection of regular languages. Let $\mathcal{L}_1, \ldots, \mathcal{L}_n \subseteq \{a, b\}^*$ be regular languages recognized by deterministic finite automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$. We define a deterministic (Inf; Min)-automaton $\mathbb{A}$ that at the first n steps starts slave automata $\mathfrak{B}_1, \ldots, \mathfrak{B}_n$ and then it invokes only a dummy slave automaton that returns 1 after a single step. For every i, the slave automaton $\mathfrak{B}_i$ first reads $n - i$ letters which it ignores, then it simulates $\mathcal{A}_i$ until the first # when it terminates. It returns 1 if the simulated automaton $\mathcal{A}_i$ accepts and 0 otherwise. More precisely, $\mathfrak{B}_i$ works on subwords $uv\#$, where $u \in \{a, b, \#\}^{n-i}, v \in \{a, b\}^*$ and returns 1 if $v \in \mathcal{L}_i$ and 0 otherwise. Observe that on a word $w = uv\#w'$ where $u \in \{a, b, \#\}^n, v \in \{a, b\}^*$ and $w' \in \{a, b, \#\}^\omega$, the automaton $\mathbb{A}$ returns 1 if and only if all automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accept v. Otherwise, $\mathbb{A}$ assigns value 0 to w. In consequence, the following conditions are equivalent: (1) the intersection $\mathcal{L}_1 \cap \ldots \cap \mathcal{L}_n$ is empty, (2) the expected value $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ is 0, and (3) the distribution $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0) = 1$. Note that the almost-sure distribution question in PSpace-hard as well.

Observe that if the intersection $\mathcal{L}_1 \cap \ldots \cap \mathcal{L}_n$ is non-empty it might be the case that the word of the minimal length in the intersection consists of a single word of exponential length. In such a case, the values $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ and $|1 - \mathbb{D}_{\mathcal{U},\mathbb{A}}(0)|$ are non-zero, but doubly-exponentially small. Therefore, we cannot use this reduction to show hardness of the approximate versions of the probabilistic problems.

#P-hardness: We show #P-hardness by reduction from the #SAT problem, which, given a propositional formula φ in conjunctive normal form asks for the number of valuations that satisfy φ. Let n be the number of variables of φ and let $C_1, \ldots, C_m$ be the clauses of φ. For every $i \in [1, m]$, we define a slave automaton $\mathfrak{B}_i$ (associated with C_i) that ignores the first $m - i$ letters, next considers the following n letters $0, 1$ as the valuation of the successive variables and checks whether this valuation satisfies the clause C_i. If it does, the slave automaton returns 1, otherwise it returns 0. The master automaton first invokes slave automata $\mathfrak{B}_1, \ldots, \mathfrak{B}_m$ and then it invokes a dummy slave automaton that returns 1 after a single step. Observe that for $w = uvw'$, where $u \in \{0, 1\}^m, v \in \{0, 1\}^n$ and $w' \in \{0, 1\}^\omega$, the automaton $\mathbb{A}$ returns 1 on w if and only if the valuation given by v satisfies all clauses $C_1, \ldots, C_m$, i.e., it satisfies φ. Otherwise, $\mathbb{A}$ returns 0 on w. Therefore, the values $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ and $1 - \mathbb{D}_{\mathcal{U},\mathbb{A}}(0)$ are equal, and multiplied by 2^n give the number of valuations satisfying φ. In follows that all approximate probabilistic questions are #P-hard. □

8.2. **Upper bounds for value functions $g \in \mathsf{FinVal} \setminus \{\mathbf{Sum}^+, \mathbf{Sum}\}$.** We now present upper bounds for value functions for (Inf; g)-automata and (Sup; g)-automata, where $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+, \textsc{Sum}\}$.

Overview. We begin with the discussion on the bounded-sum value function. We show the translation lemma (Lemma 8.2), which states that deterministic (Inf; $\textsc{Sum}^B$)-automata can be translated to deterministic Inf-automata with exponential blow-up. Moreover, this blow-up can be avoided by considering NWA of bounded width and the bound in the sum B given

in unary. Since the probabilistic questions can be solved for INF-automata in polynomial time, Lemma 8.2 implies that all probabilistic questions can be solved in exponential time for deterministic $(\text{INF}; \text{SUM}^B)$-automata. Next, we use Lemma 8.2 to show the upper bounds on the probabilistic questions for all slave value functions $g \in \mathsf{FinVal} \setminus \{\text{SUM}^+, \text{SUM}\}$.

The key ideas. It has been shown in [CHO17] that for $f \in \mathsf{InfVal}$ and $g \in \mathsf{FinVal} \setminus \{\text{SUM}^+, \text{SUM}\}$, $(f; g)$-automata can be transformed to exponential-size f-automata. In the original transformation the threshold B is fixed. We slightly modify the construction from [CHO17], to show the case of variable threshold for $(\text{INF}; \text{SUM}^B)$-automata.

Lemma 8.2. *(1) Given $B > 0$ in the binary notation and a deterministic $(\text{INF}; \text{SUM}^B)$-automaton $\mathbb{A}$, one can construct in exponential time an exponential-size deterministic* INF*-automaton $\mathcal{A}$ such that for every word w we have $\mathcal{L}_{\mathbb{A}}(w) = \mathcal{L}_{\mathcal{A}}(w)$. (2) Let $k > 0$. Given $B > 0$ in the unary notation and a deterministic $(\text{INF}; \text{SUM}^B)$-automaton $\mathbb{A}$ of width bounded by k, one can construct in polynomial time a polynomial-size deterministic* INF*-automaton $\mathcal{A}$ such that for every word w we have $\mathcal{L}_{\mathbb{A}}(w) = \mathcal{L}_{\mathcal{A}}(w)$.*

Proof. **(1)**: Let Q_m be the set of states of the master automaton and Q_s be the union of the sets of states of slave automata of $\mathbb{A}$. We define an INF-automaton $\mathcal{A}$ over the set of states $Q_m \times (Q_s \times [-B, B] \cup \{\bot\})^{|Q_s|}$. Intuitively, $\mathcal{A}$ simulates runs of $\mathbb{A}$ by simulating (a) the run of the master automaton using the component Q_m and (b) selected runs of up to $|Q_s|$ slave automata using the component $(Q_s \times [-B, B])^{|Q_s|}$. Slave automata are simulated along with their values, which are stored in the state, i.e., the state (q, l) encodes that a given slave automaton is in the state q and its current value is l. Then, the value of a given transition of $\mathcal{A}$ is the minimum over the values of simulated slave automata that terminate at the current step. Finally, the symbol $\bot$ denotes "free" components in the product $(Q_s \times [-B, B] \cup \{\bot\})^{|Q_s|}$, which can be used to simulate newly invoked slave automata. We need to convince ourselves that we need to simulate at most $|Q_s|$ slave automata. Therefore, every time a new slave automaton is invoked, we have a free component to simulate it.

Observe that if at some position two slave automata $\mathfrak{B}_1, \mathfrak{B}_2$ are in the same state q and they have computed partial sums of weights $l_1 \leq l_2$, then we can discard the simulation of the automaton $\mathfrak{B}_2$, which computed the value l_2. Indeed, since slave automata are deterministic and recognize prefix-free languages, the remaining runs of both slave automata $\mathfrak{B}_1, \mathfrak{B}_2$ are the same, i.e., they either both reject or both return values, respectively, $l_1 + v$ and $l_2 + v$ for some common v. Thus, the run of $\mathfrak{B}_2$ does not change the value of the infimum and we can stop simulating it, i.e., we can substitute (q, l_2) by $\bot$. Therefore, at every position at most $|Q_s|$ components are necessary. It follows from the construction that the values of $\mathbb{A}$ and $\mathcal{A}$ coincide on every word.

(2): If B is given in the unary notation and the width is bounded by k, we can basically repeat the construction as above for the automaton with the set of states $Q_m \times (Q_s \times [-B, B] \cup \{\bot\})^k$, which is polynomial in $\mathbb{A}$. Thus, the resulting automaton has polynomial size in $\mathbb{A}$ and can be constructed in polynomial time (in $\mathbb{A}$). □

Key ideas. We consider almost surely accepting automata and hence by Lemma 6.3, the results of Lemma 8.2 apply to $(\text{SUP}; \text{SUM}^B)$-automata. The bounded-sum value function SUM^B subsumes all value functions from $g \in \mathsf{FinVal} \setminus \{\text{SUM}^+, \text{SUM}\}$, and hence Lemma 8.2 implies that a deterministic $(\text{INF}; g)$-automaton (resp., $(\text{SUP}; g)$-automaton) can be transformed to an equivalent exponential-size INF-automaton (resp., SUP-automaton). Therefore, using Fact 6.5, both $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time.

Lemma 8.3. *Let $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}^+, \textsc{Sum}\}$ be a value function. (1) 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. (2) If $\mathbb{A}$ has bounded width, then the above quantities can be computed in polynomial time.*

Remark 8.4. Observe the following:

(1) We show in Lemma 8.3 a polynomial-time upper bound for NWA with bounded width, which gives a polynomial-time upper bound for automata with monitor counters.
(2) For $g = \textsc{Sum}^B$, the value B can be given in binary in input, and the complexity in (1) from Lemma 8.3 does not change.

We first prove Lemma 8.3 for $g \in \mathsf{FinVal} \setminus \{\textsc{Sum}, \textsc{Sum}^+\}$. Next, in Lemma 8.7 we show that Lemma 8.3 holds for deterministic $(\textsc{Inf}; \textsc{Sum}^+)$-automata. The statement of Lemma 8.7 is more general though.

Proof. Observe that deterministic Min-automata and Max-automata can be transformed in polynomial time to equivalent deterministic $\textsc{Sum}^B$-automata. Basically, a deterministic $\textsc{Sum}^B$-automaton simulating a Min-automaton (resp., Max-automaton) uses its bounded sum to track the currently minimal (resp., maximal) weight taken by the automaton. Therefore, we focus on $g = \textsc{Sum}^B$. Consider a deterministic $(\textsc{Inf}; \textsc{Sum}^B)$-automaton $\mathbb{A}$. By Lemma 8.2, $\mathbb{A}$ can be transformed in exponential time into an equivalent exponential-size deterministic Inf-automaton $\mathcal{A}$ and hence $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = \mathbb{E}_{\mathcal{M}}(\mathcal{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda) = \mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$ (for all λ). The values $\mathbb{E}_{\mathcal{M}}(\mathcal{A}), \mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$ can be computed in polynomial time in $\mathcal{A}$ (Fact 6.5), which amounts to exponential time in $\mathbb{A}$. Observe, however, that for $\mathbb{A}$ of bounded width the automaton $\mathcal{A}$ has polynomial size (assuming that the bound on the width is constant), and the values $\mathbb{E}_{\mathcal{M}}(\mathcal{A}), \mathbb{D}_{\mathcal{M},\mathcal{A}}(\lambda)$ can be computed in polynomial time in $\mathbb{A}$. □

Now, we turn to deterministic $(\textsc{Inf}; \textsc{Sum})$-automata.

8.3. **The $\textsc{Sum}^+$ and Sum value functions for slave automata.** We now establish the result when $g \in \{\textsc{Sum}, \textsc{Sum}^+\}$. First we establish decidability of the approximation problems, and then undecidability of the exact questions. Finally, we show that for deterministic $(\textsc{Inf}; \textsc{Sum}^+)$-automata all probabilistic questions are decidable.

The key ideas. The main difference between Inf and LimInf value functions is that the latter discards all values encountered before the master automaton reaches a bottom SCC where the infimum of values returned by slave automata coincides with the limit infimum and hence it can be computed in polynomial time (Lemma 7.1). We show that we can bound the values returned by slave automata and the expected values and the distributions do not change much. More precisely, given $\mathbb{A}$ and ϵ, we show that for some B, exponential in $|\mathbb{A}|$ and polynomial in the binary representation of ϵ, the probability that any slave automaton collects a (partial) sum outside the interval $[-B, B]$ 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 8.3.

Lemma 8.5. *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 in $\mathbb{A}$, polynomial time in $\mathcal{M}$ and the binary representation of ϵ.*

Proof. Consider a deterministic (Inf; Sum)-automaton $\mathbb{A}$. Let $\mathbb{A}^{\lim}$ be $\mathbb{A}$ considered as a (LimInf; Sum)-automaton. First, we assume that $\mathbb{A}^{\lim}$ has finite expected value (in particular it accepts almost all words). We can check whether this assumption holds in polynomial time by computing $\mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\lim})$ (Lemma 7.2). Then, we show the following **claim**: for every $\epsilon > 0$, there exists $B > 0$, exponential in $|\mathbb{A}|$ and linear in $|\log(\epsilon)|$ such that for $\mathbb{A}_B$ defined as $\mathbb{A}$ considered as an (Inf; Sum^B)-automaton we have $|\mathbb{E}_{\mathcal{M}}(\mathbb{A}) - \mathbb{E}_{\mathcal{M}}(\mathbb{A}_B)| \leq \epsilon$.

The claim implies the lemma. Observe that due to Lemma 8.3 with the following remark on B, the expected value $\mathbb{E}_{\mathcal{M}}(\mathbb{A}_B)$ can be computed in polynomial time in $\mathbb{A}_B$, hence exponential time in $\mathbb{A}$ and polynomial time in $\mathcal{M}$. Therefore, we can approximate $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ up to ϵ in exponential time in $\mathbb{A}$ and polynomial time in $\mathcal{M}$. Due to Markov inequality, for every λ we have $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda + \epsilon) - \mathbb{D}_{\mathcal{M},\mathbb{A}_B}(\lambda - \epsilon) < \epsilon$. However, the values of $\mathbb{A}$ are integers, therefore for $\epsilon < 0.5$ we get $|\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda) - \mathbb{D}_{\mathcal{M},\mathbb{A}_B}(\lambda)| < \epsilon$. Therefore, again by Lemma 8.3, we can approximate $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ in exponential time in $\mathbb{A}$, polynomial time in $\mathcal{M}$ and $|\log(\epsilon)|$.

The proof of the claim. First, we observe that every run ends up in some some SCC of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, and, hence, Lemma 7.1 implies that values of all words are bounded from above by $|\mathbb{A}| \cdot |\mathcal{M}|$. Next, the values of all slave automata invoked in bottom SCCs of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ are bounded from below. Otherwise, the expected value of $\mathbb{A}$ as a (LimInf; Sum)-automaton is $-\infty$. Assume that the values of all slave automata invoked in bottom SCCs of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ are bounded from below, which implies that they are bounded by $-|\mathbb{A}|$. Then, we need to estimate the influence on the expected value of the slave automata invoked before the master automaton reaches a bottom SCC of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$.

Let Y_B be a random variable on finite words such that $Y_B(u)$ is the maximum of 0 and the number of steps of any slave automaton on $\mathbb{A}$ takes on u minus B. Let E_2 be the expected number of steps of the master automaton before it reaches a bottom SCC. It follows that for $B > |\mathbb{A}|$ and C being the maximal absolute weight in slave automata of $\mathbb{A}$, we have $|\mathbb{E}_{\mathcal{M}}(\mathbb{A}) - \mathbb{E}_{\mathcal{M}}(\mathbb{A}_B)| < C\mathbb{E}(Y_B) \cdot E_2$.

Let p be the minimal positive probability that occurs in $\mathcal{M}$ and let $n = |\mathbb{A}|$. We show that for $B > \frac{n}{p^n} |\log \frac{n^2}{p^n}\epsilon|$, we have $\mathbb{E}(Y_B) \cdot E_2 < \epsilon$. We first estimate E_2. Observe that starting from every state, there exists at least one word of length at most $|\mathcal{A}_{\text{mas}}|$ upon which the master automaton reaches a bottom SCC of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$. Therefore, the master automaton reaches a bottom SCC in $|\mathcal{A}_{\text{mas}}|$ steps with probability at least $p^{|\mathcal{A}_{\text{mas}}|}$, and, hence, the number of steps before $\mathcal{A}_{\text{mas}}$ reaches a bottom SCC is estimated from above by $|\mathcal{A}_{\text{mas}}|$ multiplied by the geometric distribution with the parameter $p^{|\mathcal{A}_{\text{mas}}|}$. Hence, E_2 is bounded by $\frac{n}{p^n}$.

Now, we estimate $\mathbb{E}(Y_B)$. Observe that for every reachable state q of any slave automaton $\mathfrak{B}$, there exists a word of the length at most $|\mathfrak{B}|$ such that $\mathfrak{B}$, starting in q terminates upon reading that word. Therefore, the probability $q_l(\mathfrak{B})$ that $\mathfrak{B}$ works at least l steps is bounded by $(1 - p^{|\mathfrak{B}|})^{\frac{l}{|\mathfrak{B}|}}$. Now, $\mathbb{E}(Y_B)$ is bounded by the maximum over slave automata $\mathfrak{B}$ of $\sum_{l \geq B} q_l(\mathfrak{B})$. We have $\sum_{l \geq B} q_l(\mathfrak{B}) \leq \frac{n}{p^n} \cdot (1 - p^n)^{\frac{B}{n}}$. Hence, $\mathbb{E}(Y_B) \leq \frac{n}{p^n} \cdot (1 - p^n)^{\frac{B}{n}}$ and $\mathbb{E}(Y_B) \cdot E_2 \leq \frac{n^2}{p^n} \cdot (1 - p^n)^{\frac{B}{n}}$. Observe that for $B > \frac{n}{p^n} s$, where $s = |\log \frac{n^2}{p^n}\epsilon|$, we have $\frac{n^2}{p^n} \cdot (1 - p^n)^{\frac{B}{n}} \leq \frac{n^2}{p^n} \cdot (\frac{1}{2})^s$ and $\mathbb{E}(Y_B) \cdot E_2 \leq \epsilon$. Observe that $\frac{n}{p^n} \cdot |\log \frac{n^2}{p^n}\epsilon|$ is exponential in $|\mathbb{A}|$ and linear in $|\log(\epsilon)|$.

Lifting the assumption. Now, we discuss how to remove the assumption that $\mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\lim})$ is finite. First, we check in polynomial time whether $\mathbb{A}$ accepts almost all words (Proposition 6.1). For the expected question, observe that $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) \leq \mathbb{E}_{\mathcal{M}}(\mathbb{A}^{\lim})$, hence if the latter is $-\infty$, we can return the answer $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = -\infty$ if $\mathbb{A}$ accepts almost all words, and otherwise the expected value is undefined. For the distribution question, consider threshold λ. Observe that for every w, we have $\mathcal{L}_{\mathbb{A}}(w) \leq \mathcal{L}_{\mathbb{A}_B}(w)$. Moreover, $\mathcal{L}_{\mathbb{A}}(w) < \lambda$ while $\mathcal{L}_{\mathbb{A}_B}(w) \geq \lambda$ holds only if there is a slave automaton run before $\mathcal{A}_{\text{mas}}$ reaches a bottom SCC which runs more than B steps. Therefore, the probability $\mathbb{P}_{\mathcal{M}}(\{w \mid \mathcal{L}_{\mathbb{A}}(w) \leq \lambda \wedge \mathcal{L}_{\mathbb{A}_B}(w) \geq \lambda\})$ is bounded from above by $\mathbb{E}(Y_B) \cdot E_2$. Thus, by the previous estimate on $\mathbb{E}(Y_B) \cdot E_2$, for $B > \max(\lambda + 1, \frac{n}{p^n} \log \frac{n^2}{p^n} \epsilon|)$ we have $\mathbb{P}_{\mathcal{M}}(\{w \mid \mathcal{L}_{\mathbb{A}}(w) \leq \lambda \wedge \mathcal{L}_{\mathbb{A}_B}(w) \geq \lambda\}) < \epsilon$ and $|\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda) - \mathbb{D}_{\mathcal{M},\mathbb{A}_B}(\lambda)| < \epsilon$. Again, $\mathbb{D}_{\mathcal{M},\mathbb{A}_B}(\lambda)$ can be computed in exponential time in $|\mathbb{A}|$ and polynomial in $\mathcal{M}$ and $|\log(\epsilon)|$. □

We now show that the exact values in probabilistic questions are uncomputable for deterministic almost-surely accepting (Inf; Sum)-automata (resp., (Sup; Sum)-automata).

The key ideas. To show uncomputability of the distribution question, we show undecidability of the almost-sure distribution problem. Given a two-counter machine $\mathbf{M}$, we modify the construction from the proof of Theorem 5.1 and construct a deterministic (Inf; Sum)-automaton $\mathbb{A}_{\mathbf{M}}$ such that the following conditions are equivalent: (a) $\mathbf{M}$ has an accepting computation, (b) there exists a finite word u such that for all $w = uw'$ we have $\mathcal{L}_{\mathbb{A}_{\mathbf{M}}}(w) \geq 0$, and (c) $\mathbb{P}_{\mathcal{M}}(\{w \mid \mathcal{L}_{\mathbb{A}_{\mathbf{M}}}(w) \geq 0\}) > 0$. Condition (c) is equivalent to $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) < 1$. We show uncomputability of the expected question via reduction from the almost-sure distribution problem to deciding equality of expected values of given to automata. Given an (Inf; Sum)-automaton $\mathbb{A}_{\mathbf{M}}$, we construct $\mathbb{A}'_{\mathbf{M}}$ such that for all words w we have $\mathcal{L}_{\mathbb{A}'_{\mathbf{M}}}(w) = \min(-1, \mathcal{L}_{\mathbb{A}_{\mathbf{M}}}(w))$. Observe that the following conditions are equivalent: (i) the expected values of $\mathbb{A}'_{\mathbf{M}}$ and $\mathbb{A}_{\mathbf{M}}$ are equal, (ii) $\mathbb{A}'_{\mathbf{M}}$ and $\mathbb{A}_{\mathbf{M}}$ are equal on almost every word, and (iii) $\mathbb{D}_{\mathcal{U},\mathbb{A}_{\mathbf{M}}}(-1) = 1$.

Lemma 8.6. *Let $\mathcal{U}$ be the uniform distribution over the infinite words. The following problems are undecidable: (1) Given a deterministic almost-surely accepting* (Inf; Sum)*-automaton (resp.,* (Sup; Sum)*-automaton) $\mathbb{A}$ of width 8, decide whether $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$. (2) Given two deterministic almost-surely accepting* (Inf; Sum)*-automata (resp.,* (Sup; Sum)*-automata) $\mathbb{A}_1, \mathbb{A}_2$ of width bounded by 8, decide whether $\mathbb{E}_{\mathcal{U}}(\mathbb{A}_1) = \mathbb{E}_{\mathcal{U}}(\mathbb{A}_2)$.*

Proof. **(1)**: In the following, we discuss how to adapt the proof of Theorem 5.1 to prove this lemma.

Given a deterministic two-counter machine $\mathbf{M}$, we construct a deterministic (Inf; Sum)-automaton $\mathbb{A}_M$ such that for a word w of the form $\$u\$w'$ it returns 0 if u is a valid accepting computation of $\mathbf{M}$ and a negative value otherwise. We use $\Sigma = Q \cup \{1, 2, \#, \$\}$ for convenience; one can encode letters from Σ using two-letter alphabet $\{0, 1\}$. On words that are not of the form $\$u\$w'$, the automaton $\mathbb{A}_M$ returns values less or equal to -1. Basically, the automaton $\mathbb{A}_M$ simulates on u the execution of $\mathbb{A}$ (as defined in the proof of Theorem 5.1) with the opposite values of slave automata, i.e., all weights of slave automata are multiplied by -1. Recall, that the supremum of the values returned by slave automata on a subword $\$u\$$ is 0 if and only if u encodes a valid and accepting computation of $\mathbf{M}$. Otherwise, the supremum is at least 1. Thus, in our case, the infimum over the values of slave automata is 0 if and only if u encodes a valid and accepting computation of $\mathbf{M}$. Otherwise, the value of

INF is at most -1. Therefore, $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$ if and only if $\mathbf{M}$ does not have an accepting computation.

(2): We show that knowing how to compute the expected value of deterministic (INF; SUM)-automata, we can decide equality in the distribution question. Let $\mathbb{A}$ be an automaton and we ask whether $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1) = 1$. We construct another automaton $\mathbb{A}'$ that simulates $\mathbb{A}$, but at the first transition it invokes a slave automaton that returns the value -1. The values of automata $\mathbb{A}$ and $\mathbb{A}'$ differ precisely on words which have values (assigned by $\mathbb{A}$) greater than -1. Thus, their expected values $\mathbb{E}_{\mathcal{U}}(\mathbb{A})$ and $\mathbb{E}_{\mathcal{U}}(\mathbb{A}')$ differ if and only if $\mathbb{D}_{\mathcal{U},\mathbb{A}}(-1)$ is different than 1. Due to undecidability of the latter problem, there exists no Turing machine that computes the expected value of (INF; SUM)-automata over the uniform distribution. □

Finally, we have the following result for the absolute sum value function, which guarantees that the return values are at least 0. We present a slightly more general result. Recall that we assume that weights in slave automata are given in unary.

Lemma 8.7. *Given a Markov chain $\mathcal{M}$, a value $\lambda \in \mathbb{Q}$ and a deterministic* (INF; SUM)*-automaton such that the value of every slave automaton is bounded from below, the values $\mathbb{E}_{\mathcal{M}}(\mathbb{A})$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time in $|\mathbb{A}|$ and polynomial time in $|\mathcal{M}|$.*

Proof. Consider a deterministic (INF; SUM)-automaton $\mathbb{A}$ such that the value of every slave automaton is bounded from below. Let $B = |\mathbb{A}| \cdot |\mathcal{M}|$ and let $\mathbb{A}'$ be $\mathbb{A}$ considered as a deterministic (INF; SUM^B)-automaton. We show that on almost all words w we have $\mathcal{L}_{\mathbb{A}}(w) = \mathcal{L}_{\mathbb{A}'}(w)$. Then, $\mathbb{E}_{\mathcal{M}}(\mathbb{A}) = \mathbb{E}_{\mathcal{M}}(\mathbb{A}')$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda) = \mathbb{D}_{\mathcal{M},\mathbb{A}'}(\lambda)$ and the values $\mathbb{E}_{\mathcal{M}}(\mathbb{A}')$ and $\mathbb{D}_{\mathcal{M},\mathbb{A}'}(\lambda)$ can be computed in exponential time by Lemma 8.3 taking into account the remark about B being input.

Since the value of every slave automaton $\mathfrak{B}_i$ is bounded from below, the (reachable part of) the weighted Markov chain $\mathfrak{B}_i \times \mathcal{M}$ considered as a weighted graph does not have negative cycles. Therefore, the minimal value $\mathfrak{B}_i$ can achieve is greater than $-|\mathfrak{B}_i| \cdot |\mathcal{M}| > -|\mathbb{A}| \cdot |\mathcal{M}|$. Moreover, every accepting run of $\mathbb{A}$ ends up in some SCC of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, where almost all words have the same value (Lemma 7.1), which is either ∞ (if almost all words are rejected) or bounded from above by $|\mathbb{A}| \cdot |\mathcal{M}|$. This value can be computed in polynomial time. Therefore, the value of almost all words belong to the interval $[-|\mathbb{A}| \cdot |\mathcal{M}|, |\mathbb{A}| \cdot |\mathcal{M}|]$ or it is ∞ if the run on $\mathbb{A}$ on this word is rejecting. Finally, the sets of words with accepting runs in $\mathbb{A}$ and $\mathbb{A}'$ coincide. □

The above lemma implies that the probabilistic questions for deterministic (INF; SUM$^+$)-automata can be answered in exponential time in $|\mathbb{A}|$ and polynomial time in $|\mathcal{M}|$. Note that (INF; SUM$^+$)-automata and (SUP; SUM$^+$)-automata are not dual. Indeed, in Lemma 6.3 we multiply weights by -1, which turns SUM$^+$-automata into SUM-automata with negative weights. Thus, we consider separately the distribution question for (SUP; SUM$^+$)-automata. We show that the distribution question for deterministic (SUP; SUM$^+$)-automata is decidable in EXPTIME.

Lemma 8.8. *The distribution question for deterministic* (SUP; SUM$^+$)*-automata can be computed in exponential time in $|\mathbb{A}|$ and polynomial time in $|\mathcal{M}|$.*

Proof. Let $\mathbb{A}$ be a deterministic (SUP; SUM$^+$)-automaton, $\mathcal{M}$ be a Markov chain and let λ be a threshold in the distribution question. Consider $\mathbb{A}'$ defined as $\mathbb{A}$ considered as a (SUP; SUM^B)-automaton with $B = \lambda + 1$. Observe that for every word w we have $\mathcal{L}_{\mathbb{A}}(w) \leq \lambda$

if and only if $\mathcal{L}_{\mathbb{A}'}(w) \leq \lambda$. Therefore, $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda) = \mathbb{D}_{\mathcal{M},\mathbb{A}'}(\lambda)$. The latter value can be computed in exponential time in $\mathbb{A}$ and polynomial time in $|\mathcal{M}|$ (Lemma 8.3). □

Theorem 8.9. *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.*

	Min, Max, $\textsc{Sum}^B, \textsc{Sum}^+$	Sum
Expected value	ExpTime (Lemma 8.3, 8.5, 8.8) PSpace-hard (Lemma 8.1)	Uncomputable (Lemma 8.6)
Distribution		
Almost sure distribution		
Approximate: (a) expected value (b) distribution	ExpTime (Lemma 8.3, 8.5) #P-hard (Lemma 8.1)	

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

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

Remark 8.10 (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 8.11 (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.

9. Results on non-deterministic automata

In this section, we briefly discuss non-deterministic NWA evaluated on Markov chains. First, we discuss the definition of random variables defined by non-deterministic NWA. Next, we present two negative results.

Non-deterministic NWA as random variables. A non-deterministic NWA $\mathbb{A}$ defines a function $h\colon \Sigma^\omega \to \mathbb{R}$ as in the deterministic case via $h(w) = \mathcal{L}_{\mathbb{A}}(w)$. Recall that $\mathcal{L}_{\mathbb{A}}(w) = \inf_{\pi \in \mathsf{Acc}(w)} f(\pi)$, where $\mathsf{Acc}(w)$ is the set of accepting runs of $\mathbb{A}$ on w. We show that h is measurable w.r.t. any probability measure given by a Markov chain. It suffices to show

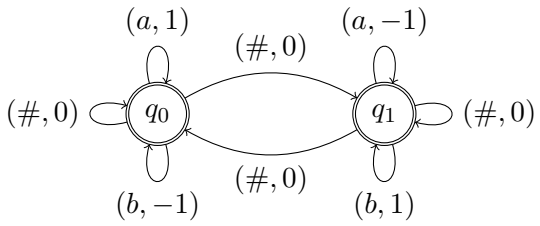

FIGURE 6. An example of non-deterministic automaton, in which non-deterministic choices has to "depend on the future" in order to obtain the infimum.

that for every interval $(-\infty, x]$, its preimage $h^{-1}[(-\infty, x]]$ is measurable. Consider a set $A_x \subseteq \Sigma^\omega \times Q_{\text{mas}}^\omega \times (\mathbb{Z} \cup \{\bot\})^\omega$ of words with runs of $\mathbb{A}$ of the value at most x. More precisely, let f be the the master value function of $\mathbb{A}$. Then, $(w, \Pi, \alpha) \in A_x$ if and only if (1) Π is a run of the master automaton of $\mathbb{A}$ on w, (2) for every $i \in \mathbb{N}$, the slave automaton invoked by the master automaton at position i has a finite run on $w[i, \infty]$ of the value $\alpha[i]$, (3) $f(\alpha) \leq x$. Observe that A_x is a Borel set in the product topology. The preimage $h^{-1}[(-\infty, x]]$ is the projection of the Borel set A_x, and hence it is an *analytic set*, which is measurable [Kec12].

Conceptual difficulty. The evaluation of a 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 9.1. Consider a non-deterministic LIMAVG-automaton $\mathcal{A}$, depicted in Figure 6. 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 words 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.

Non-deterministic LIMAVG*-automata under probabilistic semantics.* Non-deterministic LIMAVG-automata evaluated over Markov chains have been studied in [MO18]. It has been established that the expected value or the distribution of such automata (over the uniform distribution) can be irrational and these values are not computable. This is in stark contrast to deterministic LIMAVG-automata, where the answers are rational and can be computed in polynomial time [BK08].

	Deterministic	Non-deterministic
Emptiness	Undecidable (from [CHO17])	
Probabilistic questions	Polynomial time (Theorem 7.3)	Uncomputable (Theorem 9.3)

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.

Computational difficulty. In contrast to our polynomial-time algorithms for the probabilistic questions for deterministic (LimSup, Sum)-automata, we establish the following undecidability result for the non-deterministic automata with width 1. Lemma 9.2 implies Theorem 9.3.

Lemma 9.2. *The following problem is undecidable: given a non-deterministic* (LimSup; Sum)*-automaton* $\mathbb{A}_M$ *of width* 1*, decide whether* $\mathbb{P}(\{w \mid \mathcal{L}_{\mathcal{A}_M}(w) = 0\}) = 1$ *or* $\mathbb{P}(\{w \mid \mathcal{L}_{\mathcal{A}_M}(w) = -1\}) = 1$ *w.r.t. the uniform distribution on infinite words.*

Proof. In the following, we discuss how to adapt the proof of Theorem 5.1 to prove this lemma. First, observe that we can encode any alphabet Σ using a two-letter alphabet $\{0, 1\}$, therefore we will present our argument for multiple-letters alphabet as it is more convenient. Given a two-counter machine $\mathbf{M}$ we construct a non-deterministic (LimSup; Sum)-automaton $\mathbb{A}_M$ such that $\mathcal{L}_{\mathcal{A}_M}(w) = 0$ if and only if w contains infinitely many subsequences that correspond to valid accepting computations of $\mathbf{M}$. As in the proof of Theorem 5.1, for every subsequence $\$u\$$, where u does not contain \$, we check whether u is an encoding of a valid accepting computation of $\mathbf{M}$. To do that, we check conditions (C1) and (C2) as in the proof of Theorem 5.1. At the letter \$, the master automaton non-deterministically decides whether u violates (C1) or (C2) and either starts a slave automaton checking (C1) or (C2). The slave automaton checking (C1) works as in the proof of Theorem 5.1. It returns -1 if (C1) is violated and 0 otherwise. The slave automaton checking (C2) non-deterministically picks the position of invocation of a slave automaton from the proof of Theorem 5.1 that returns a negative value. Finally, at the letter \$ following u, the master automaton starts the slave automaton that returns the value -1. It follows that the supremum of all values of slave automata started at $u\$$ is either -1 or 0. By the construction, there is a (sub) run on $u\$$ such that the supremum of the values of all slave automata is -1 if and only if u does not encode valid accepting computation of $\mathbf{M}$. Otherwise, this supremum is 0. Therefore, the value of the word w is 0 if and only if w contains infinitely many subsequences that correspond to valid accepting computations of $\mathbf{M}$. Now, if $\mathbf{M}$ has at least one valid accepting computation u, then almost all words contain infinitely many occurrences of u and almost all words have value 0. Otherwise, all words have value -1. □

Lemma 9.2 implies that there is no terminating Turing machine that computes any of probabilistic questions.

Theorem 9.3. *All probabilistic questions (Q1-Q5) are undecidable for non-deterministic* (LimSup, Sum)*-automata of width 1.*

10. Discussion

In this section we discuss extensions of our main results. We begin with the distribution question for NWA without the almost-sure acceptance condition. Next, we show how to apply the obtained results to compute the probabilistic variant of the quantitative inclusion problem [CDH10b, CHO17]. Finally, we discuss the parametric complexity of the questions considered in the paper.

10.1. **Non almost-sure acceptance.** In Section 7 we consider NWA that accept almost all words, i.e., they assign finite values to almost all words. Dropping almost-sure acceptance condition does not change the complexity of the expected question, but it influences the complexity of the distribution question, which we discuss in this section.

We begin with the lower bounds for the distribution question for NWA without almost-sure acceptance condition.

The key ideas. We show that removing the restriction that almost all words are accepted changes the complexity of distribution questions. The intuition behind this is that the condition "all slave automata accept" allows us to simulate (in a restricted way) the Sup master value function.

Lemma 10.1. *For all $f \in \mathsf{InfVal}$ and $g \in \mathsf{FinVal}$, we have*

(1) *The distribution question for deterministic $(f;g)$-automata is* PSpace*-hard.*
(2) *The approximate distribution question for deterministic $(f;g)$-automata is #P-hard.*

Proof. We say that an NWA is *simple* if it is deterministic, accepts almost all words and its slave automata have only weights 0 and 1. First, we show that the almost-sure distribution and the approximate distribution questions for simple (Sup; Max)-automata are respectively PSpace-hard and #P-hard. Second, we reduce the almost-sure distribution and the approximate distribution problems for simple (Sup; Max)-automata to the corresponding problems for deterministic $(f;g)$-automata (which are not almost-surely accepting). These two steps show (1) and (2).

We observe that (Inf; Min)-automata considered in the proof of Lemma 8.1 are simple; they are deterministic, accept almost all words and all slave automata have only weights $0, 1$. Given an (Inf; Min)-automaton $\mathbb{A}$, let $\mathbb{A}'$ be a (Sup; Max)-automaton obtained from $\mathbb{A}$ by swapping weights 0 and 1 in all slave automata. Note that for all words w, $\mathbb{A}(w)$ returns 0 (resp. 1) if and only if $\mathbb{A}'$ returns 1 (resp. 0). Therefore, $\mathbb{D}_{\mathcal{U},\mathbb{A}}(0) = 1 - \mathbb{D}_{\mathcal{U},\mathbb{A}'}(0)$. It follows that the almost-sure distribution and the approximate distribution questions for simple (Sup; Max)-automata are respectively PSpace-hard and #P-hard.

Now, we show reduction from simple (Sup; Max)-automata to deterministic $(f;g)$-automata (which are not almost-surely accepting). For a deterministic Max-automaton $\mathcal{A}$ with only weights 0 and 1, we define $\mathcal{A}^g$ as a deterministic g-automaton obtained from $\mathcal{A}$ by deletion of transitions of weight 1. Observe that $\mathcal{A}^g$ returns 0 whenever $\mathcal{A}$ returns 0, and it rejects whenever $\mathcal{A}$ rejects or returns 1. This construction works for all g, which return 0 on any sequence of 0's. All value functions g from FinVal have this property. Now, consider any $f \in \mathsf{InfVal}$. Given a simple (Sup; Max)-automaton $\mathbb{A}$, we apply to all slave automata $\mathfrak{B}$ of $\mathbb{A}$ the transformation $\mathfrak{B} \to \mathfrak{B}^g$. Let $\mathbb{A}^f$ be the resulting $(f;g)$-automaton. This NWA is deterministic and its slave automata return only value 0 or reject (return ∞). Therefore, for every $f \in \mathsf{InfVal}$ and every word w, we have (a) $\mathcal{L}_{\mathbb{A}^f}(w) = 0$ if and only if $\mathcal{L}_{\mathbb{A}}(w) = 0$, and (b) $\mathcal{L}_{\mathbb{A}^f}(w) = \infty$ if and only if $\mathcal{L}_{\mathbb{A}}(w) \in \{1, \infty\}$. It follows that $\mathbb{D}_{\mathcal{U},\mathbb{A}^f}(0) = \mathbb{D}_{\mathcal{U},\mathbb{A}}(0)$.

Therefore, the almost-sure distribution (resp., the approximate distribution) problem for simple (Sup; Max)-automata reduces to the almost-sure distribution (resp., the approximate distribution) problem for deterministic $(f; g)$-automata. □

We now show the upper bound for the distribution question for NWA considered in Section 7.

Lemma 10.2. *Let $f \in \{\text{LimInf}, \text{LimSup}, \text{LimAvg}\}$ and let $g \in \mathsf{FinVal}$ be a value function. Given a Markov chain $\mathcal{M}$, a deterministic $(f; g)$-automaton $\mathbb{A}$, and a threshold λ in binary, the value $\mathbb{D}_{\mathcal{M},\mathbb{A}}(\lambda)$ can be computed in exponential time in $\mathbb{A}$ and polynomial time in $\mathcal{M}$. Moreover, if $\mathbb{A}$ has bounded width, then the above quantities can be computed in polynomial time.*

Proof. Consider $\mathcal{A}_{\text{mas}} \times \mathcal{M}$, where $\mathcal{A}_{\text{mas}}$ is the master automaton of $\mathbb{A}$. For all bottom SCCs of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$ either almost all words have an accepting run or all words are rejected (Lemma 7.1 — acceptance is independent of the value function and hence it works for LimAvg as well). Moreover, in a bottom SCC in which almost all words are accepted, almost all words have the same value (Lemma 7.1 and Lemma 7.11). We compute these values in all almost-accepting bottom SCCs. Next, we need to compute the probability of the set of words, which are accepted by $\mathbb{A}$ and reach any almost-accepting bottom SCC, in which almost all words have the value not exceeding λ.

To do that, we consider $\mathbb{A}$ as an $(\text{Inf}; \text{Sum}^B)$-automaton, transform it into an Inf-automaton $\mathcal{A}$ (Lemma 8.2). Then, we compute the probability of reaching the the corresponding SCCs in $\mathcal{A} \times \mathcal{M}$ with the standard reachability analysis [BK08]. Recall that each state of $\mathcal{A}$ contains a state of $\mathcal{A}_{\text{mas}}$ as a component and hence we can identify states of $\mathcal{A} \times \mathcal{M}$ that corresponds to the selected SCC of $\mathcal{A}_{\text{mas}} \times \mathcal{M}$. The size of $\mathcal{A} \times \mathcal{M}$ is exponential in $\mathbb{A}$ and polynomial in $\mathcal{M}$. Moreover, if $\mathbb{A}$ has bounded width, then $\mathcal{A} \times \mathcal{M}$ is polynomial in $|\mathbb{A}|$. □

10.2. **Quantitative inclusion.** In this section, we discuss *probabilistic inclusion question*, which is the probabilistic variant of the quantitative inclusion problem for weighted automata. We show how to adapt the results from Section 7 and Section 8 to establish decidability and complexity of the probabilistic inclusion question.

The following definition is common for weighted automata, automata with monitor counters and NWA, and hence we refer to them collectively as automata. The *probabilistic inclusion question* asks, given two automata $\mathcal{A}_1, \mathcal{A}_2$ and a Markov chain $\mathcal{M}$, to compute the probability of the set $\{w \mid \mathcal{L}_{\mathcal{A}_1}(w) \leq \mathcal{L}_{\mathcal{A}_2}(w)\}$ w.r.t. the probability measure given by $\mathcal{M}$.

The key ideas. Let $\mathcal{M}$ be a Markov chain, $f \in \{\text{LimInf}, \text{LimSup}, \text{LimAvg}\}$ and let $\mathbb{A}_1, \mathbb{A}_2$ be deterministic $(f; \text{Sum})$-automata accepting almost all words. For all such NWA, almost all words reaching the same bottom SSC (of the product $\mathcal{A}_{\text{mas}} \times \mathcal{M}$) have the same value (Section 7), and hence to decide probabilistic inclusion we examine all pairs of bottom SSCs B_1, B_2 from $\mathbb{A}_1$ and $\mathbb{A}_2$ respectively, in which the value of $\mathbb{A}_1$ does not exceed the value of $\mathbb{A}_2$ and compute the probability of words that lead to B_1 in $\mathbb{A}_1$ and B_2 in $\mathbb{A}_2$. The sum of such probabilities is the answer to the quantitative inclusion problem. These probabilities can be computed in polynomial time [BK08].

For $f \in \{\text{Inf}, \text{Sup}\}$ and $g \in \{\text{Min}, \text{Max}, \text{Sum}^B\}$, deterministic $(f; g)$-automata $\mathbb{A}_1, \mathbb{A}_2$ are equivalent to deterministic f-automata $\mathcal{A}_1, \mathcal{A}_2$ respectively. The automata $\mathcal{A}_1, \mathcal{A}_2$

have exponential size in $\mathbb{A}_1$ and $\mathbb{A}_2$ respectively. The probabilistic inclusion question for deterministic Inf-automata (resp., Sup-automata) can be computed in polynomial time, which gives us the exponential-time upper bound in $\mathbb{A}_1$ and $\mathbb{A}_2$. Finally, the probabilistic inclusion question subsumes the distribution question, which gives us lower bounds.

Theorem 10.3. *The following conditions hold:*

(1) *For all $f \in \{\text{LimInf}, \text{LimSup}, \text{LimAvg}\}$ and $g \in \mathsf{FinVal}$, the probabilistic inclusion question for deterministic almost-surely accepting $(f; g)$-automata can be solved in polynomial time.*
(2) *For all $f \in \{\text{Inf}, \text{Sup}\}$ and $g \in \{\text{Min}, \text{Max}, \text{Sum}^B\}$, the probabilistic inclusion question for deterministic $(f; g)$-automata is* PSpace*-hard and can be solved in exponential time.*
(3) *For all $f \in \{\text{Inf}, \text{Sup}, \}$, the probabilistic inclusion question for deterministic $(f; \text{Sum})$-automata is uncomputable.*

Proof. **LimInf and LimSup and LimAvg value functions**. We discuss the case of $g = \text{Sum}$ as it subsumes other value functions from FinVal. Let $\mathcal{M}$ be a Markov chain, $f \in \{\text{LimInf}, \text{LimSup}, \text{LimAvg}\}$ and let $\mathbb{A}_1, \mathbb{A}_2$ be deterministic $(f; \text{Sum})$-automata accepting almost all words. Let $\mathcal{A}^1_{\text{mas}}, \mathcal{A}^2_{\text{mas}}$ be the master automata of respectively $\mathbb{A}_1, \mathbb{A}_2$. We construct the product Markov chain $(\mathcal{A}^1_{\text{mas}} \times \mathcal{A}^2_{\text{mas}}) \times \mathcal{M}$ and compute all its bottom SCCs $S_1, \ldots, S_k$. For almost all words w, both runs of respectively $\mathbb{A}_1, \mathbb{A}_2$ on w finally reach one of these SCCs. Note that each S_i projected to $\mathcal{A}^1_{\text{mas}} \times \mathcal{M}$ or $\mathcal{A}^2_{\text{mas}} \times \mathcal{M}$ is still an SCC and hence almost all words w, whose run end up in S_i have the same value in $\mathbb{A}_1$ (resp., in $\mathbb{A}_2$). We compute these value for all SCCs $S_1, \ldots, S_k$ and select these components, in which the value in $\mathbb{A}_1$ does not exceed the value in $\mathbb{A}_2$. This can be done in polynomial time for $f \in \{\text{LimInf}, \text{LimSup}\}$ (Lemma 7.1) and for $f = \text{LimAvg}$ (Lemma 7.11). Finally, we compute the probability of reaching the selected SCCs in $(\mathcal{A}^1_{\text{mas}} \times \mathcal{A}^2_{\text{mas}}) \times \mathcal{M}$, which can be done in polynomial time [BK08].

Inf and Sup value functions. We consider the case of $f = \text{Inf}$ as the case of $f = \text{Sup}$ is symmetric. First, consider the case of $g \in \{\text{Min}, \text{Max}, \text{Sum}^B\}$ and consider two deterministic $(\text{Inf}; g)$-automata $\mathbb{A}_1, \mathbb{A}_2$. These NWA can be transformed to equivalent deterministic Inf-automata $\mathcal{A}_1, \mathcal{A}_2$ respectively, which have exponential size in $\mathbb{A}_1$ and $\mathbb{A}_2$ respectively (Lemma 8.2). Now, let $x_1, \ldots x_n$ be weights of $\mathcal{A}_1$. For each of these weights x_i, we compute the probability p_i of the set of words such that the value of $\mathcal{A}_1$ is x_i and the value of $\mathcal{A}_2$ is at least x_i. To compute p_i, we construct the product Markov chain $\mathcal{M} \times \mathcal{A}^1_{\text{mas}} \times \mathcal{A}^2_{\text{mas}}$, remove from it all transitions, which correspond to transition of $\mathcal{A}_1$ or $\mathcal{A}_2$ of weight less than x_i, and compute the probability of the set of paths which take at least once a transition of $\mathcal{A}_1$ of weight x_i. Finally, the answer to the probabilistic inclusion question for $\mathcal{A}_1$ and $\mathcal{A}_2$, and hence $\mathbb{A}_1$ and $\mathbb{A}_2$, is the sum of p_i.

Observe that for $\mathbb{A}_2$ that returns λ for every word, the probabilistic inclusion problem becomes the distribution question for $\mathbb{A}_1$. □

10.3. **Parametric complexity.** The problems we consider correspond to measuring performance (expectation or cumulative distribution) under stochastic environments, when the specification is an NWA and the system is modeled by a Markov chain. In this section we discuss the parametric complexity of the probabilistic problems, where the specification, represented by an NWA, is fixed. We discuss the parametric complexity for different value functions for the master automaton:

- For LimInf, LimSup, LimAvg value functions for the master automaton the expected question solvable in polynomial time in $\mathbb{A}$ and $\mathcal{M}$ (Theorem 7.3 and Theorem 7.12). The distribution question is solvable in polynomial time (in $\mathbb{A}$ and $\mathcal{M}$) as well, but only for NWA that are almost-surely accepting (Theorem 7.3 and Theorem 7.12). For NWA that are not almost-surely accepting Lemma 10.2 states that the distribution question can be solved in polynomial time in $|\mathcal{M}|$.
- For $f \in \{\text{Inf}, \text{Sup}\}$ and $g \in \{\text{Min}, \text{Max}, \text{Sum}^B\}$, a deterministic $(f;g)$-automaton $\mathbb{A}$ can be transformed to an equivalent deterministic f-automaton $\mathcal{A}$ (Lemma 8.2). The size of $\mathcal{A}$ is exponential in $\mathbb{A}$. However, if $\mathbb{A}$ is fixed, the complexity of all probabilistic questions is the same as for weighted automata; they can be solved in polynomial time (Lemma 6.5).
- For $f \in \{\text{Inf}, \text{Sup}\}$ and $g \in \{\text{Sum}, \text{Sum}^+\}$, the approximate expected question and the approximate distribution questions for $(f;g)$-automata can be solved in polynomial time in $\mathcal{M}$ and the length of the binary representation of ϵ (Lemma 8.5).
- For $f \in \{\text{Inf}, \text{Sup}\}$, the distribution question for $(f;\text{Sum}^+)$-automata can be solved in polynomial time in $|\mathcal{M}|$ (Lemma 8.7 and Lemma 8.8).
- Finally, for deterministic (Inf; Sum)-automata and (Sup; Sum)-automata, surprisingly the expected question, the distribution question and the almost-sure distribution question remain uncomputable. We sketch this in the following Lemma 10.4.

In summary, in all computable cases fixing the NWA makes the complexity of all probabilistic problems drop to polynomial time. Interestingly, the uncomputable cases remain uncomputable. This is similar to the parametric complexity analysis from [CHO17], where fixing the size of slave automata reduces the complexity of the classic decision questions for NWA, but the undecidable cases remain undecidable.

The key ideas. We modify the construction from the proof of Lemma 8.6, where given a Minsky machine $\mathbf{M}$, we construct an (Inf; Sum)-automaton $\mathbb{A}$ that returns 0 if the input word w is of the form $u\$w'$ and u encodes an accepting computation of $\mathbf{M}$. Otherwise it returns negative values. The constructed NWA checks two types of conditions: (C1) Boolean conditions stating that the sequence of configurations is consistent with instructions of $\mathbf{M}$, and (C2) quantitative conditions, which imply that if the counters values are inconsistent with increments and decrements, the NWA returns negative values. We observe that (C2) are independent of $\mathbf{M}$, while conditions (C1) are Boolean and can be checked with a finite automaton, or they can be enforced by the Markov chain. Thus, given a Minsky machine $\mathbf{M}$, we construct a Markov chain $\mathcal{M}_{\mathbf{M}}$ checking (C1), while the NWA $\mathbb{A}'$ that checks (C2) is independent of $\mathbf{M}$, and hence can be fixed.

Lemma 10.4. *The following conditions hold:*

(1) *There exists a deterministic* (Inf; Sum)*-automaton* $\mathbb{A}$ *such that the problem: given a Markov chain* $\mathcal{M}$*, decide whether* $\mathbb{D}_{\mathbb{A},\mathcal{M}}(-1) = 1$ *is undecidable.*
(2) *There exist deterministic* (Inf; Sum)*-automata* $\mathbb{A}_1, \mathbb{A}_2$ *such that the problem: given a Markov chain* $\mathcal{M}$*, decide whether* $\mathbb{E}_{\mathcal{M}}(\mathbb{A}_1) = \mathbb{E}_{\mathcal{M}}(\mathbb{A}_2)$ *is undecidable.*

Proof. Let $\Sigma = \{0, 1, 2, \#, \$\}$. We construct an NWA working over words of the form $u\$w'$, where $u \in (0^*1^*2^*\#)^*$. Each block $0^n1^k2^j$ represents a configuration of some Minsky machine such that the machine is in the state q_n the value of the first counter is k and the value of the second counter is j. We define (Inf; Sum)-automaton $\mathbb{A}$ that for each counter invokes two slave automata, which respectively compute the difference between two consecutive values of the same counter plus 1 and its inverse. Moreover, $\mathbb{A}$ invokes one slave automaton returning value 0 to ensure that the value of each word is at most 0. Thus, $\mathbb{A}$ returns 0

on an input word $u\$w'$ if the counter values in all blocks differ by at most 1, i.e., for any infix $\#0^{n_1}1^{k_1}2^{l_1}\#0^{n_2}1^{k_2}2^{l_2}\#$ we have $|k_1 - k_2| \leq 1$ and $|l_1 - l_2| \leq 1$. On all other words it returns values less or equal to -1.

Now, given a Minsky machine $\mathbf{M}$ we define a Markov chain $\mathcal{M}$ that generates sequences of the form $\#0^{n[0]}1^{k[0]}2^{j[0]}\#0^{n[1]}1^{k[1]}2^{j[1]}\ldots\$\Sigma^\omega$ consistent with the instructions from $\mathbf{M}$. More precisely, using Boolean conditions we specify that (i) states encoded by 0^n change according to the instructions of $\mathbf{M}$ (we can verify zero and non-zero tests), (ii) the first configuration and the configuration before \$ are respectively initial and the final configuration of $\mathbf{M}$, and (iii) values of counters modulo 3 change according to the instructions of $\mathbf{M}$. Observe that if there is a word $u\$w'$ generated by $\mathcal{M}$ has value 0 assigned by $\mathbb{A}$, then the values of counters in u change by at most 1 and the change modulo 3 is verified in condition (iii). Both conditions imply that the counters change according to instructions of $\mathbf{M}$. Therefore, the word u encodes an accepting computation of $\mathbf{M}$. It follows that for the NWA $\mathbb{A}$, the problem given a Markov chain $\mathcal{M}$ decide whether $\mathbb{D}_{\mathbb{A},\mathcal{M}}(-1) = 1$ is undecidable. In consequence, the expected and the distribution questions are undecidable (see Lemma 8.6).

For the expected value, we reduce the distribution question to the equality of the expected values as in the proof of (2) from Lemma 8.6. □

11. Conclusions

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 7.3, 7.12). 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.

12. 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

[ABK14] Shaull Almagor, Udi Boker, and Orna Kupferman. Discounting in LTL. In *TACAS, 2014*, pages 424–439, 2014.

[ADD+13] 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.

[BBC+11] 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.

[BCFK15] 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.

[BCHK14] Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. *ACM TOCL*, 15(4):27:1–27:25, 2014.

[BDK14] Christel Baier, Clemens Dubslaff, and Sascha Klüppelholz. Trade-off analysis meets probabilistic model checking. In *CSL-LICS 2014*, pages 1:1–1:10, 2014.

[BGMZ10] 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.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BKKW14] 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.

[BMM14] Patricia Bouyer, Nicolas Markey, and Raj Mohan Matteplackel. Averaging in LTL. In *CONCUR 2014*, pages 266–280, 2014.

[BMR+18] Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim G. Larsen, and Simon Laursen. Average-energy games. *Acta Inf.*, 55(2):91–127, 2018.

[CD11] Krishnendu Chatterjee and Laurent Doyen. Energy and mean-payoff parity Markov Decision Processes. In *MFCS 2011*, pages 206–218, 2011.

[CDH09a] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In *FCT'09*, pages 3–13. Springer, 2009.

[CDH09b] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. A survey of stochastic games with limsup and liminf objectives. In *ICALP 2009, Part II*, pages 1–15, 2009.

[CDH10a] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. *LMCS*, 6(3), 2010.

[CDH10b] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM TOCL*, 11(4):23, 2010.

[CFW13] Krishnendu Chatterjee, Vojtech Forejt, and Dominik Wojtczak. Multi-objective discounted reward verification in graphs and MDPs. In *LPAR*, pages 228–242, 2013.

[Cha07] Krishnendu Chatterjee. Markov decision processes with multiple long-run average objectives. In *FSTTCS*, pages 473–484, 2007.

[CHJS15] Krishnendu Chatterjee, Thomas A. Henzinger, Barbara Jobstmann, and Rohit Singh. Measuring and synthesizing systems in probabilistic environments. *J. ACM*, 62(1):9:1–9:34, 2015.

[CHO16a] Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted limit-average automata of bounded width. In *MFCS 2016*, pages 24:1–24:14, 2016.

[CHO16b] Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Quantitative automata under probabilistic semantics. In *LICS 2016*, pages 76–85, 2016.

[CHO16c] Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Quantitative monitor automata. In *SAS 2016*, pages 23–38, 2016.

[CHO17] Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted automata. *ACM Trans. Comput. Log.*, 18(4):31:1–31:44, 2017.

[CKK15] 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.

[CMH06] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov Decision Processes with multiple objectives. In *STACS 2006*, pages 325–336, 2006.

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 1st edition, 2009.

[DR06] Manfred Droste and George Rahonis. Weighted automata and weighted logics on infinite words. In *DLT 2006*, pages 49–58, 2006.

[Fel71] W. Feller. *An introduction to probability theory and its applications*. Wiley, 1971.

[FKN+11] 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.

[FV96] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer, 1996.

[HKNP06] Andrew Hinton, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS 2006*, pages 441–444, 2006.

[Kec12] Alexander Kechris. *Classical descriptive set theory*, volume 156. Springer Science & Business Media, 2012.

[Min61] Marvin Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, pages 437–455, 1961.

[MO18] Jakub Michaliszyn and Jan Otop. Non-deterministic weighted automata on random words. In *CONCUR 2018*, pages 10:1–10:16, 2018.

[Moh02] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Aut. Lang. & Comb.*, 7(3):321–350, 2002.

[Pap03] Christos H Papadimitriou. *Computational complexity*. Wiley, 2003.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1st edition, 1994.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.