Quantitative Language Automata

Thomas A. Henzinger
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Pavol Kebis
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Nicolas Mazzocchi
Slovak University of Technology in Bratislava, Slovakia

N. Ege Saraç
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

A *quantitative word automaton* (QWA) defines a function from infinite words to values. For example, every infinite run of a limit-average QWA $\mathcal{A}$ obtains a mean payoff, and every word $w \in \Sigma^\omega$ is assigned the maximal mean payoff obtained by nondeterministic runs of $\mathcal{A}$ over w. We introduce *quantitative language automata* (QLAs) that define functions from language generators (i.e., implementations) to values, where a language generator can be nonprobabilistic, defining a set of infinite words, or probabilistic, defining a probability measure over infinite words. A QLA consists of a QWA and an aggregator function. For example, given a QWA $\mathcal{A}$, the infimum aggregator maps each language $L \subseteq \Sigma^\omega$ to the greatest lower bound assigned by $\mathcal{A}$ to any word in L. For boolean value sets, QWAs define boolean properties of traces, and QLAs define boolean properties of sets of traces, i.e., hyperproperties. For more general value sets, QLAs serve as a specification language for a generalization of hyperproperties, called *quantitative hyperproperties*. A nonprobabilistic (resp. probabilistic) quantitative hyperproperty assigns a value to each set (resp. distribution) G of traces, e.g., the minimal (resp. expected) average response time exhibited by the traces in G. We give several examples of quantitative hyperproperties and investigate three paradigmatic problems for QLAs: evaluation, nonemptiness, and universality. In the *evaluation* problem, given a QLA $\mathbb{A}$ and an implementation G, we ask for the value that $\mathbb{A}$ assigns to G. In the *nonemptiness* (resp. *universality*) problem, given a QLA $\mathbb{A}$ and a value k, we ask whether $\mathbb{A}$ assigns at least k to some (resp. every) language. We provide a comprehensive picture of decidability for these problems for QLAs with common aggregators as well as their restrictions to ω-regular languages and trace distributions generated by finite-state Markov chains.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Quantitative automata

Keywords and phrases Quantitative hyperproperties, quantitative automata, automata-based verification

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2025.21

Related Version *Full Version*: `https://arxiv.org/abs/2506.05158`

Funding This work was supported in part by the ERC-2020-AdG 101020093.

1 Introduction

The specification and verification of system properties traditionally take a boolean view. While this view is appropriate for correctness properties, it lacks the ability to reason about quantitative aspects of system behaviors, such as performance or robustness. Quantitative trace properties and quantitative word automata [12] address this gap: instead of partitioning the set of traces into correct and incorrect traces (as boolean properties do), they define functions from system executions to richer value domains, e.g., the real numbers. Using such a formalism, we can specify the maximal or average response time of a server's execution, or how badly a system behavior violates a desired boolean property [19].

Many interesting system properties lie beyond the trace setting; especially security properties often refer to multiple traces. In the boolean case, they can be specified by hyperproperties [13], which are interpreted over sets of traces rather than over individual traces. For example, a prominent hyperproperty is observational determinism, which requires every pair of executions with matching observable inputs to have matching observable outputs. In general, while trace properties specify which system behaviors are correct, hyperproperties specify which system implementations are correct. In a similar vein, while quantitative trace properties and quantitative word automata describe system properties on the level of individual executions, *quantitative hyperproperties* are needed to express quantitative aspects of system-wide properties. For example, a quantitative hyperproperty may measure the maximal or average response time of an implementation instead of a single execution, or how badly an implementation violates a desired boolean property. In this paper, we introduce *quantitative language automata* (QLAs), an automaton model for the specification and verification of quantitative hyperproperties. In contrast to quantitative word automata, quantitative language automata can measure system-wide properties.

Quantitative word automata (QWAs) extend boolean ω-automata with weighted transitions. An infinite run yields an infinite sequence of weights, which are accumulated by a *run aggregator* (a.k.a. value function). Common run aggregators are Sup (the maximal weight along an infinite run), LimSup (the largest weight that occurs infinitely often), or $\mathsf{LimInfAvg}$ (the long-term average of an infinite sequence of weights). When a given infinite word yields more than one run, as is generally the case for nondeterministic automaton specifications, the possible run values are accumulated by a *word aggregator*. The most common word aggregator is Sup (the least upper bound of all values that can be achieved by resolving the nondeterministic choices), which generalizes the standard view that a single accepting run suffices to accept a word, but other word aggregators are possible. For example, the word aggregator LimSup assigns to each infinite word w the l.u.b. of the values realized by infinitely many runs over w. When the specification is probabilistic (rather than nondeterministic), the word aggregator $\mathbb{E}$ assigns to w an expected value for the automaton reading w.

Quantitative language automata extend quantitative word automata with a third kind of aggregator function, called *language aggregator*, which summarizes the values of all infinite words which are obtained from an implementation and defined by a so-called "language generator." A language generator can be nonprobabilistic, defining a set of infinite words, or probabilistic, defining a probability measure over infinite words. A QLA $\mathbb{A} = (h, \mathcal{A})$ consists of a language aggregator h and a QWA $\mathcal{A}$, and maps each language generator G to the value $h_{w \sim G}\mathcal{A}(w)$, where $\mathcal{A}(w)$ is the value assigned by the QWA $\mathcal{A}$ to the infinite word w, and $h_{w \sim G}$ denotes the accumulation over all words generated by G. When G is nonprobabilistic, we interpret the aggregator h over the words that belong to the language defined by G; when probabilistic, over the words generated with respect to the probability measure defined by G. Using the language aggregators Inf and Sup, language automata can measure the best-case and the worst-case values that can be obtained by all executions of an implementation. With language aggregators LimInf and LimSup, we can express the "almost" best- and worst-cases, considering only the values achieved by infinitely many words in the language. Finally, when G is probabilistic, the language aggregator $\mathbb{E}$ captures the average-case with respect to the its probability measure. Using all three aggregator functions, we can specify, for example, the expected value (taken over all possible implementation traces) of the best case (realizable by a nondeterministic specification) of the average weight along a run of the specification.

Table 1 Complexity of evaluation, nonemptiness, and universality for QLAs with language aggregator h, word aggregator g, and run aggregator f. **(a)** QLAs with $h, g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$ (Sections 5 and 6). *Open-hard* means the problem is at least as hard as the universality problem for nondeterministic discounted-sum QWA [6,12]. Universality of QLAs is dual to their nonemptiness, e.g., it is in PSPACE when $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$ and $g = \mathsf{Sup}$ (see Remark 6.1). **(b)** QLAs with $h, g, f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ with g or h in $\{\mathsf{LimInf}, \mathsf{LimSup}\}$ (Section 7).

(a)	Evaluation					Nonemptiness		
	$h = g$	$h \neq g$	$h = \mathbb{E}$	$h \in \{\mathsf{Inf}, \mathsf{Sup}\}$	$h = \mathbb{E}$	$h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$		
	$\in \{\mathsf{Inf}, \mathsf{Sup}\}$	$\in \{\mathsf{Inf}, \mathsf{Sup}\}$	$g \in \{\mathsf{Inf}, \mathsf{Sup}\}$	$g = \mathbb{E}$	$g = \mathbb{E}$	$g = \mathsf{Sup}$	$g = \mathsf{Inf}$	$g = \mathbb{E}$
$f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$	PTIME	PSPACE	EXPTIME	Undecidable	PTIME	PTIME	PSPACE	Undecidable
$f \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$	PTIME	Undecidable	Undecidable	Undecidable	PTIME	PTIME	Undecidable	Undecidable
$f = \mathsf{DSum}$	PTIME	Open-hard	Open-hard	Undecidable	PTIME	PTIME	Open-hard	Undecidable

(b)	Evaluation	Nonemptiness	Universality
$h, g, f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ and at least one of h or g is in $\{\mathsf{LimInf}, \mathsf{LimSup}\}$	PSPACE	PSPACE	PSPACE

The standard decision problems for automata extend naturally to our framework. Consider a language automaton $\mathbb{A}$ and a rational threshold k. Given a finite-state language generator G (i.e., an ω-regular automaton or a finite Markov chain), the evaluation problem asks to compute the value to which $\mathbb{A}$ maps G. The nonemptiness (resp. universality) problem asks whether $\mathbb{A}$ maps some (resp. every) language generator to a value of at least k, or strictly above k. We investigate these problems for QLAs over unrestricted language generators, as well as with regard to natural subclasses of language generators such as finite-state ones.

Contribution and Overview. Our main contribution is the definition and systematic study of QLAs within a three-dimensional framework of aggregator functions (run, word, and language aggregators), which supports the specification and verification of quantitative hyperproperties over nonprobabilistic and probabilistic language generators.

Section 2 presents our definitional framework and Section 3 gives several examples of QLAs for specifying quantitative hyperproperties. Section 4 introduces the QLA evaluation, nonemptiness, and universality problems that we study. Section 5 focuses on the evaluation problem of QLAs with word and language aggregators Inf, Sup, or $\mathbb{E}$, and Section 6 on the nonemptiness and universality problems of these. Section 7 switches the focus to QLAs with word and language aggregators LimInf and LimSup. Section 8 concludes with potential research directions. We summarize our decidability and complexity results in Table 1. We provide proof sketches in the main text; the details are available in the full version.

To present a comprehensive picture, we overcome several technical challenges: First, for the evaluation problem (Section 5), we demonstrate that the value of an ω-regular language L may not be realized (i) by a lasso word in L for limit-average (a.k.a. mean-payoff) QLAs and (ii) by any word in L for discounted-sum QLAs. Although the evaluation problem for these automata is not always solvable, for its solvable cases we resolve these issues by proving that (i) in the limit-average case, the value of L can still be computed by analyzing strongly connected components even if it is not realized by a lasso word, and (ii) in the discounted-sum case, the value of L matches the value of its safety closure L' and is realized by some word in L'. These results yield PTIME algorithms for the evaluation of these QLAs. We complement these results with hardness proofs via reductions from universality problems of the underlying QWAs.

Second, for nonemptiness and universality (Section 6), we examine the behavior of common QWA classes regarding the greatest lower bound of their values – their so-called bottom values [4]. We show (i) discounted-sum QWAs always have a word achieving their bottom value, and (ii) the bottom value of limit-superior-average QWAs can be approximated by lasso words, contrasting sharply with limit-inferior-average QWAs [12]. This enables us to establish the hardness of nonemptiness and universality for certain classes of QLAs. Moreover, we prove that for most classes of QLAs the unrestricted versions of nonemptiness and universality coincide with their finite-state restrictions, indicating a finite-model property.

Third, for QLAs using LimInf and LimSup as word and language aggregators (Section 7), we study values realized by infinitely many runs of a word and infinitely many words of a given language. Using combinatorial arguments, we characterize structural patterns in ω-automata that precisely capture when a value is infinitely realizable with the run aggregators Inf, Sup, LimInf, and LimSup.

Related Work. Our work builds on quantitative languages and quantitative word automata [10, 12], as well as on games with quantitative objectives [14]. There have been other definitions of quantitative hyperproperties [16, 28, 31].

While our motivations align, QLAs in their current form and the formalisms of [16, 28] are orthogonal. In [16, 28], a quantitative hyperproperty is based on counting the number of executions that appear in a given relation. Therefore, these formalisms cannot express, e.g., the worst-case server uptime, which QLAs can (see Section 3). Similarly, QLAs with the aggregators we considered cannot express properties that can only be defined using counting or cardinality constraints.

In [31], the authors provide a predicate-transformer theory for reasoning on finite executions of imperative programs. Their formalism can define quantitative hyperproperties that can be expressed as functions from a set or distribution of values at a program's final state to a single value, which is closer to our view compared to [16, 28]. For example, it can express the maximal value of a variable x given the value of variable y upon termination (akin to QLAs) or the variance of a random variable. However, it cannot express long-run aggregates such as limit superior or limit averages. Conversely, QLAs cannot express variances with the current aggregators, since it requires combining two expectations at the language-aggregator level. To the best of our knowledge, our work is the first to define quantitative hyperproperties as functions from sets or distributions of infinite words to quantitative values, and therefore the first to study the specification and verification of such properties through automata.

2 Definitional Framework

Let Σ be a finite *alphabet* of letters. A *word* (or *trace*) over Σ is a finite or infinite sequence of letters from Σ. We denote by $|w|$ the length of a finite word w. We denote by Σ^* (resp. Σ^ω) the set of all finite (resp. infinite) words over Σ. An infinite word w is *ultimately periodic* (a.k.a. *lasso*) iff $w = uv^\omega$ for some $u, v \in \Sigma^*$ such that $|v| \geq 1$. A *language* is a set of infinite words. Given $u \in \Sigma^*$ and $w \in \Sigma^\omega$, we write $u \prec w$ when u is a prefix of w. We denote by $\mathbb{N}$ the set of natural numbers (including 0), $\mathbb{Q}$ the set of rational numbers, $\mathbb{R}$ the set of real numbers. We further let $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$ and $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. Consider a set S. An S-multiset is a function $M : S \to \overline{\mathbb{N}}$ that maps each element of S to a value denoting its multiplicity. The support of a multiset M is the set $supp(M) = \{x \in S \mid M(x) \geq 1\}$ of distinct elements in M. A *value domain* $\mathbb{D}$ is a nontrivial complete lattice. A *quantitative property* is a total function $\Phi : \Sigma^\omega \to \mathbb{D}$.

Quantitative Word Automata. A *weighted labeled transition system* is a tuple $\mathcal{T} = (\Sigma, Q, s, \delta, d)$, where: Σ is a finite alphabet, Q is a finite nonempty set of states, $s \in Q$ is the initial state, $\delta \colon Q \times \Sigma \to 2^{\mathbb{Q} \times Q}$ is a finite transition function over weight-state pairs, and $d : Q \times \Sigma \times \mathbb{Q} \times Q \to [0,1]$ is a probability distribution such that for all $q \in Q$ and $a \in \Sigma$, (i) $d(q,a,x,q') > 0$ iff $(x,q') \in \delta(q,a)$, and (ii) $\sum_{(x,q') \in \delta(q,a)} d(q,a,x,q') = 1$. Given a transition system $\mathcal{T}$, the *dual* of $\mathcal{T}$ is a copy of $\mathcal{T}$ with the weights multiplied by -1.

A *transition* is a tuple $(q, \sigma, x, q') \in Q \times \Sigma \times \mathbb{Q} \times Q$ such that $(x, q') \in \delta(q, \sigma)$, denoted $q \xrightarrow{\sigma:x} q'$. Given a transition $t = q \xrightarrow{\sigma:x} q'$, we denote its weight by $\gamma(t) = x$. We say that $\mathcal{T}$ is *complete* (a.k.a. *total*) iff $|\delta(q,a)| \geq 1$ for every $q \in Q$ and $a \in \Sigma$, and *deterministic* iff $|\delta(q,a)| = 1$ for every $q \in Q$ and $a \in \Sigma$. Throughout the paper, we assume that weighted labeled transition systems are complete.

Although weighted labeled transition systems are probabilistic by definition, they can be viewed as *nondeterministic* by considering only the support of d, i.e., treating all transitions with positive probability as nondeterministic transitions.

A *run* of $\mathcal{T}$ is an infinite sequence $\rho = q_0 \xrightarrow{\sigma_0:x_0} q_1 \xrightarrow{\sigma_1:x_1} \ldots$ of transitions such that $q_0 = s$ and $(x_i, q_{i+1}) \in \delta(q_i, \sigma_i)$ for all $i \geq 0$. We write $\gamma(\rho)$ for the corresponding infinite sequence $x_0 x_1 \ldots$ of rational weights. Given a word $w \in \Sigma^\omega$, we denote by $R(\mathcal{T}, w)$ the set of runs of w over $\mathcal{T}$. Let $\rho = t_0 t_1 \ldots$ be a run and $r = t_0 t_1 \ldots t_n$ be a finite prefix of ρ. The probability of r is $\prod_{i=0}^{n} d(t_i)$. For each infinite word w, we define $\mu_{\mathcal{T},w}$ as the unique probability measure over Borel sets of infinite runs of $\mathcal{T}$ over w, induced by the transition probabilities d.

A *run aggregator* (a.k.a. value function) is a function $f : \mathbb{Q}^\omega \to \overline{\mathbb{R}}$ that accumulates an infinite sequence of weights into a single value. We consider the run aggregators below over an infinite sequence $x = x_0 x_1 \ldots$ of rational weights and a discount factor $\lambda \in \mathbb{Q} \cap (0,1)$. We write DSum when the discount factor $\lambda \in \mathbb{Q} \cap (0,1)$ is unspecified. The run aggregators Inf and Sup (resp., LimInf and LimSup, $\mathsf{LimInfAvg}$ and $\mathsf{LimSupAvg}$) are duals. The dual of DSum_λ is itself.

- $\mathsf{Inf}(x) = \inf\{x_n \mid n \in \mathbb{N}\}$
- $\mathsf{Sup}(x) = \sup\{x_n \mid n \in \mathbb{N}\}$
- $\mathsf{LimInf}(x) = \liminf_{n \to \infty} x_n$
- $\mathsf{LimSup}(x) = \limsup_{n \to \infty} x_n$
- $\mathsf{LimInfAvg}(x) = \liminf_{n \to \infty} \left(\frac{1}{n} \sum_{i=0}^{n-1} x_i\right)$
- $\mathsf{LimSupAvg}(x) = \limsup_{n \to \infty} \left(\frac{1}{n} \sum_{i=0}^{n-1} x_i\right)$
- $\mathsf{DSum}_\lambda(x) = \sum_{i \geq 0} \lambda^i x_i$

A *word aggregator* is a function $g : \overline{\mathbb{N}}^{\overline{\mathbb{R}}} \to \overline{\mathbb{R}}$ that accumulates a multiset of values obtained from the runs of a word into a single value. We consider the word aggregators defined below where M is an $\overline{\mathbb{R}}$-multiset and μ is a probability measure over $supp(M)$. The word aggregators Inf and Sup (resp., LimInf and LimSup) are duals. The dual of $\mathbb{E}_\mu$ is itself.

- $\mathsf{Inf}(M) = \inf\{x \mid M(x) \geq 1\}$
- $\mathsf{Sup}(M) = \sup\{x \mid M(x) \geq 1\}$
- $\mathbb{E}_\mu(M) = \int_{x \in supp(M)} x d\mu(x)$
- $\mathsf{LimInf}(M) = \inf\{x \mid M(x) = \infty\}$
- $\mathsf{LimSup}(M) = \sup\{x \mid M(x) = \infty\}$

A *quantitative word automaton* (QWA) [12] is a tuple $\mathcal{A} = (g, f, \mathcal{T})$ where $\mathcal{T}$ is a complete weighted labeled transition system, f is a run aggregator, and g is a word aggregator. Given a word w, a transition system $\mathcal{T}$, and a run aggregator f, we define an $\overline{\mathbb{R}}$-multiset $M_{w,\mathcal{T},f}$ such that for every $x \in \overline{\mathbb{R}}$ the value $M_{w,\mathcal{T},f}(x)$ equals the number of runs ρ of $\mathcal{T}$ over w such that

$f(\gamma(\rho)) = x$. For all $x \in \mathit{supp}(M_{w,\mathcal{T},f})$ we define $\mu(x) = \mu_{\mathcal{T},w}(\{\rho \in R(\mathcal{T}, w) \mid f(\gamma(\rho)) = x\})$. We let $\mathcal{A}(w) = g(M_{w,\mathcal{T},f})$ for all $w \in \Sigma^\omega$ and write $\mathbb{E}$ for $\mathbb{E}_\mu$ when μ is clear from the context.

A QWA is *nondeterministic* when its word aggregator is Sup, and *universal* when Inf. The *top value* of a word automaton $\mathcal{A}$ is $\top_\mathcal{A} = \sup_{w \in \Sigma^\omega} \mathcal{A}(w)$, and its *bottom value* is $\bot_\mathcal{A} = \inf_{w \in \Sigma^\omega} \mathcal{A}(w)$. We say that $\top_\mathcal{A}$ (resp. $\bot_\mathcal{A}$) is *achievable* iff there exists a word w with $\mathcal{A}(w) = \top_\mathcal{A}$ (resp. $\mathcal{A}(w) = \bot_\mathcal{A}$). Given a word automaton $\mathcal{A} = (g, f, \mathcal{T})$ its *dual* is $\hat{\mathcal{A}} = (\hat{g}, \hat{f}, \hat{\mathcal{T}})$ where $\hat{g}$, $\hat{f}$, and $\hat{\mathcal{T}}$ are the duals of g, f, and $\mathcal{T}$, respectively. For a QWA $\mathcal{A} = (g, f, \mathcal{T})$ and a word w, if $M_{w,\mathcal{T},f}(x)$ is finite for all $x \in \overline{\mathbb{R}}$, then we let $\mathcal{A}(w) = \top_\mathcal{A}$ when $g = \mathsf{LimInf}$ and $\mathcal{A}(w) = \bot_\mathcal{A}$ when $g = \mathsf{LimSup}$.

Boolean word automata are a special case of QWAs with weights in $\{0, 1\}$ and $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$. In particular, a nondeterministic Büchi automaton is a boolean word automaton with $g = \mathsf{Sup}$ and $f = \mathsf{LimSup}$. Given a boolean word automaton $\mathcal{B}$, we write $L(\mathcal{B}) = \{w \in \Sigma^\omega \mid \mathcal{B}(w) = 1\}$ for its language.

Quantitative Language Automata. A *language generator* is a function $G : 2^{\Sigma^\omega} \to [0, 1]$. A language generator G is *nonprobabilistic* iff there is a language $L_G \subseteq \Sigma^\omega$ such that $G(L_G) = 1$ and $G(L) = 0$ for every other language $L \neq L_G$. A nonprobabilistic language generator G is *nonempty* iff $L_G \neq \emptyset$. A language generator G is *probabilistic* iff it defines a probability measure μ_G over Σ^ω. A *nonprobabilistic quantitative hyperproperty* (resp. *probabilistic quantitative hyperproperty*) is a total function from the set of all nonprobabilistic (resp. probabilistic) language generators to a value domain $\mathbb{D}$.

A *language aggregator* is a function $h : \overline{\mathbb{N}}^{\overline{\mathbb{R}}} \to \overline{\mathbb{R}}$ that accumulates a multiset of values obtained from a set of words into a single value. We consider the language aggregators Inf, Sup, LimInf, LimSup, and $\mathbb{E}_\mu$, which are defined the same as for the word aggregators above.

A *quantitative language automaton* (QLA) is a pair $\mathbb{A} = (h, \mathcal{A})$ where $\mathcal{A} = (g, f, \mathcal{T})$ is a QWA and h is a language aggregator. Consider a language generator G and a QWA $\mathcal{A}$. If G is nonprobabilistic (resp. probabilistic), we define $M_{G,\mathcal{A}}$ as an $\overline{\mathbb{R}}$-multiset such that for every $x \in \overline{\mathbb{R}}$ the value $M_{G,\mathcal{A}}(x)$ equals the number of words w in $L_G \subseteq \Sigma^\omega$ (resp. in Σ^ω) such that $\mathcal{A}(w) = x$. Moreover, if G is probabilistic, we additionally let $\mu(x) = \mu_G(\{w \in \Sigma^\omega \mid \mathcal{A}(w) = x\})$. Then, we let $\mathbb{A}(G) = h(M_{G,\mathcal{A}})$ for every language generator G where μ is as above. We write $\mathbb{E}$ for $\mathbb{E}_\mu$ when μ is clear from the context.

Below, we assume the input to a QLA is a nonprobabilistic (resp. probabilistic) language generator when $h \neq \mathbb{E}$ (resp. $h = \mathbb{E}$). We write L to denote a nonprobabilistic language generator, and μ to denote a probabilistic one.

The *top value* of a QLA $\mathbb{A}$, denoted $\top_\mathbb{A}$, is the supremum of the values $\mathbb{A}(G)$ over all inputs G, and its *bottom value*, denoted $\bot_\mathbb{A}$, is the infimum. For a QLA $\mathbb{A} = (h, \mathcal{A})$ and a nonprobabilistic language generator L, if $M_{L,\mathcal{A}}(x) = 0$ for all $x \in \overline{\mathbb{R}}$ (i.e., when $L = \emptyset$), then we let $\mathbb{A}(L) = \top_\mathbb{A}$ when $h = \mathsf{Inf}$ and $\mathbb{A}(L) = \bot_\mathbb{A}$ when $h = \mathsf{Sup}$. Similarly, if $M_{L,\mathcal{A}}(x)$ is finite for all $x \in \overline{\mathbb{R}}$, then we let $\mathbb{A}(L) = \top_\mathbb{A}$ when $h = \mathsf{LimInf}$ and $\mathbb{A}(L) = \bot_\mathbb{A}$ when $h = \mathsf{LimSup}$.

▶ **Proposition 2.1.** *Consider a QLA $\mathbb{A} = (h, \mathcal{A})$ with $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$. Then, $\top_\mathbb{A} = \top_\mathcal{A}$ and $\bot_\mathbb{A} = \bot_\mathcal{A}$.*

Given a QLA $\mathbb{A} = (h, \mathcal{A})$ its *dual* is $\hat{\mathbb{A}} = (\hat{h}, \hat{\mathcal{A}})$ where $\hat{h}$ and $\hat{\mathcal{A}}$ are the duals of h and $\mathcal{A}$.

▶ **Proposition 2.2.** *Consider a QLA $\mathbb{A}$ and its dual $\hat{\mathbb{A}}$. Then, $\mathbb{A}(G) = -\hat{\mathbb{A}}(G)$ for every language generator G.*

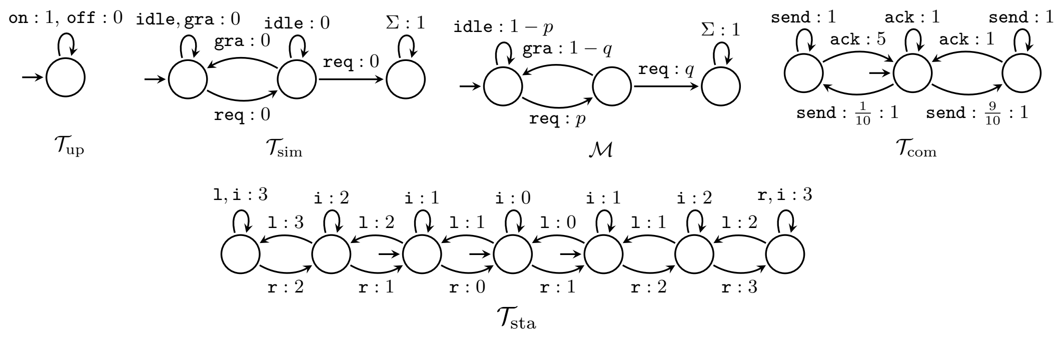

Figure 1 Weighted labeled transition systems $\mathcal{T}_{\text{up}}$, $\mathcal{T}_{\text{sim}}$, $\mathcal{T}_{\text{com}}$, and $\mathcal{T}_{\text{sta}}$ can be used to specify QWAs and QLAs with various aggregators, as demonstrated in Section 3. Transitions are marked with $\sigma : p : x$ where σ denotes a letter, p the transition probability (dropped when $p = 1$), and x the transition weight. The Markov chain $\mathcal{M}$ is a language generator where $0 \leq p, q \leq 1$ (its transitions are not weighted, and those with probability 0 are not shown).

3 Applications of Language Automata

While QWAs describe system specifications *per execution*, QLAs describe *system-wide* specifications. We show several applications of QLAs for specifying quantitative hyperproperties. Notice that QLAs are an *operational* specification language – their underlying word automata serve as part of the specification itself.

Server Uptime. QWAs can model performance metrics for individual executions. Let $\Sigma = \{\texttt{on}, \texttt{off}\}$ be a finite alphabet of observations modeling a server's activity. Consider the weighted labeled transition system $\mathcal{T}_{\text{up}}$ given in Figure 1, and let $\mathcal{A}_{\text{up}} = (g, f, \mathcal{T}_{\text{up}})$ be a QWA where $g = \mathsf{Sup}$ and $f = \mathsf{LimInfAvg}$. The automaton $\mathcal{A}_{\text{up}}$ maps each infinite word w to the long-run ratio of $\texttt{on}$, i.e., the average uptime of the execution modeled by w. For example, if $w = \texttt{on} \cdot (\texttt{on} \cdot \texttt{off})^\omega$, we have $\mathcal{A}(w) = 0.5$.

QLAs can model *system-wide* performance metrics. Let $\mathbb{A}_{\text{up}} = (h, \mathcal{A}_{\text{up}})$. Setting $h = \mathsf{Inf}$ tunes $\mathbb{A}_{\text{up}}$ to map any given language L, capturing the implementation of a server, to the worst-case uptime over all the executions allowed by the implementation, i.e., $\mathbb{A}_{\text{up}}(L) = \inf_{w \in L} \mathcal{A}_{\text{up}}(w)$. For example, let $U = \{\texttt{on}^n\texttt{off}^k \mid n > k \geq 0\}$ and consider the language $L = \{u_1u_2 \ldots \mid \forall i : u_i \in U\}$ of server executions where each block of $\texttt{on}$'s is followed by a strictly shorter block of $\texttt{off}$'s. Although for every $u \in U$ there are strictly more $\texttt{on}$'s than $\texttt{off}$'s, we have $w = s_1s_2 \ldots \in L$ where $s_i = \texttt{on}^i\texttt{off}^{i-1} \in U$ for each $i \geq 1$, thus $\mathbb{A}_{\text{up}}(L) = 0.5$. In particular, notice that the value of $\mathbb{A}_{\text{up}}(L)$ cannot be achieved by any ultimately periodic word in L; we will later show that this may happen even for ω-regular languages (Proposition 5.2). As another example of using QLAs to specify quantitative hyperproperties, setting $h = \mathsf{LimInf}$ tunes $\mathbb{A}_{\text{up}}$ to express the "almost" worst-case uptime by considering only the uptime values realized by infinitely many executions. For example, let $L = \Sigma^*\texttt{on}^\omega \cup \{(\texttt{on}^n\texttt{off})^\omega \mid n \geq 0\}$. The part $\Sigma^*\texttt{on}^\omega$ yields infinitely many executions with uptime 1, while each $(\texttt{on}^n\texttt{off})^\omega$ with uptime $\frac{n}{n+1}$ appears only once. Thus, even though the infimum over L is 0, only uptime 1 occurs infinitely often, so $\mathbb{A}(L) = 1$.

Implementation Distance. QWAs can specify the distance of individual executions to a desired boolean trace property. Let $\Sigma = \{\texttt{req}, \texttt{gra}, \texttt{idle}\}$ be a finite alphabet of observations modeling a server receiving requests and issuing grants, and consider the boolean safety property P requiring that no two requests are simultaneously open (i.e., there is a $\mathtt{gra}$ between every two $\mathtt{req}$'s). Consider the transition system $\mathcal{T}_{\text{sim}}$ given in Figure 1, and let $\mathcal{A}_{\text{sim}} = (g, f, \mathcal{T}_{\text{sim}})$ be a QWA where $f = \mathsf{DSum}_{0.5}$ and $g \in \{\mathsf{Inf}, \mathsf{Sup}\}$ (the choice of g does not matter since $\mathcal{T}_{\text{sim}}$ is deterministic). The automaton $\mathcal{A}_{\text{sim}}$ maps each infinite word w to the smallest Cantor distance between w and some word $w' \in P$, i.e., $\mathcal{A}_{\text{sim}}(w) = \inf_{w' \in P} d(w, w')$. For example, if $w = \mathtt{req} \cdot \mathtt{idle} \cdot \mathtt{req}^\omega$, we have $\mathcal{A}(w) = 0.25$.

QLAs can specify the distance of *systems* to a desired boolean trace property. Let $\mathbb{A}_{\text{sim}} = (h, \mathcal{A}_{\text{sim}})$. Setting $h = \mathsf{Sup}$ tunes $\mathbb{A}_{\text{sim}}$ to map any given language L, representing a server implementation, to the worst-case distance from L to P. In other words, $\mathbb{A}_{\text{sim}}(L) = \sup_{w \in L} \inf_{w' \in P} d(w, w')$ where d denotes the Cantor distance. For example, if $L = \Sigma^* \mathtt{req}^2 \Sigma^\omega$, then $\mathbb{A}_{\text{sim}}(L) = 0.5$. Setting $h = \mathsf{Inf}$ tunes $\mathbb{A}_{\text{sim}}$ to map each language to the best-case distance from L to P. For example, if L is as above, the value of L would be 0 although all the words in L violate P. This is because for every $i \geq 0$ the word $w_i = \mathtt{idle}^i \mathtt{req}^2 \mathtt{idle}^\omega$ belongs to L and $\lim_{i \to \infty} \mathbb{A}_{\text{sim}}(w_i) = 0$, highlight a challenge for the evaluation of discounted-sum QLAs: the value of L may not be realized by any word in L, even if L is omega-regular. Setting $h = \mathbb{E}$ tunes $\mathbb{A}_{\text{sim}}$ to map each language to the average-case distance to P. For example, consider the Markov chain $\mathcal{M}$ from Figure 1. The expected value can be computed by solving the corresponding system of linear equations, resulting in $\mathbb{A}_{\text{sim}}(\mu_{\mathcal{M}}) = \frac{2pq}{2+p(1+q)}$.

Robot Stability. QWAs can express stability constraints of individual executions. Let $\Sigma = \{\mathtt{l}, \mathtt{r}, \mathtt{i}\}$ be a finite alphabet representing robot movements – *left*, *right*, and *idle* – on a one-dimensional finite grid. An execution is ε-stable, for some $\varepsilon \geq 0$, if there exists $0 \leq \delta \leq \varepsilon$ such that whenever the system starts within a δ-ball around the origin, it remains indefinitely within an ε-ball around the origin. For a fixed $\delta \geq 0$, QWAs can express the most permissive ε associated with each execution. Consider the weighted labeled transition system $\mathcal{T}_{\text{sta}}$ given in Figure 1, and let $\mathcal{A}_{\text{sta}} = (\mathsf{Sup}, \mathsf{Sup}, \mathcal{T}_{\text{sta}})$. The automaton captures the scenario where $\delta = 1$, meaning the robot starts at most one step away from the origin. Transition weights indicate distances from the origin, and a run's value is the maximal distance reached. Thus, the automaton's value on a word is the worst-case distance (or most permissive ε) over all initial positions. For instance, the word $w = (\mathtt{l} \cdot \mathtt{r})^\omega$ has three runs with values 1, 1, and 2, so $\mathcal{A}_{\text{sta}}(w) = 2$.

QLAs can express stability constraints of *systems.* Let $\mathbb{A}_{\text{sta}} = (h, \mathcal{A}_{\text{sta}})$. If $h = \mathsf{Sup}$, then given a language L modeling the robot's behavior, the automaton $\mathbb{A}_{\text{sta}}$ maps L to least upper bound of the set of per-execution ε values obtained from L and $\mathcal{A}_{\text{sta}}$. In other words, $\mathbb{A}_{\text{sta}}(L) = \mathsf{Sup}_{w \in L} \mathbb{A}_{\text{sta}}(w)$ is the smallest ε ensuring all executions in L are ε-stable with $\delta = 1$. Alternatively, if $h = \mathsf{LimSup}$, then $\mathbb{A}_{\text{sta}}(L)$ captures the smallest ε such that infinitely many executions are ε-stable for $\delta = 1$, allowing to discard "outlier" values achieved only by finitely many executions.

Communication Channel Cost. Probabilistic QWAs can specify the expected cost of individual executions. Let $\Sigma = \{\mathtt{send}, \mathtt{ack}\}$ be an alphabet modeling a communication channel. Consider the transition system $\mathcal{T}_{\text{com}}$ given in Figure 1, and let $\mathcal{A}_{\text{com}} = (\mathbb{E}, \mathsf{LimSup}, \mathcal{T}_{\text{com}})$ be a probabilistic QWA. Each run of the automaton is mapped to a long-term maximal cost, i.e., LimSup of the corresponding weight sequence. Then, each infinite word is mapped to the expected value over the distribution of its runs. For example, considering $w = (\mathtt{send} \cdot \mathtt{ack})^\omega$ gives us $\mathcal{A}_{\text{com}}(w) = 5$ because the set of runs with a LimSup value 5 (i.e., those in which the high-cost cycle occurs infinitely often) has probability 1.

QLAs can specify the aggregate cost of a communication channel. Let $\mathbb{A}_{\text{com}} = (h, \mathcal{A}_{\text{com}})$. If $h = \mathbb{E}$, the QLA $\mathbb{A}_{\text{com}}$ specifies the expected cost of the underlying probabilistic model of a communication channel. Consider a Markov chain defining a uniform probability measure μ over the alphabet $\Sigma = \{\texttt{send}, \texttt{ack}\}$. Then, $\mathbb{A}(\mu) = 1.05$ which can be computed by analyzing the product of the Markov chain defining μ and the underlying word automaton $\mathcal{A}_{\text{com}}$.

4 Problems on Language Automata

Let $\mathbb{A} = (h, \mathcal{A})$ be a QLA. We study the following problems for $\triangleright \in \{>, \geq\}$.

Nonprobabilistic Evaluation ($h \neq \mathbb{E}$). Given a Büchi automaton $\mathcal{B}$, compute $\mathbb{A}(\mathcal{B})$.
Probabilistic Evaluation ($h = \mathbb{E}$). Given a finite-state Markov chain $\mathcal{M}$, compute $\mathbb{A}(\mathcal{M})$.
$\triangleright$-Nonemptiness. Given $k \in \mathbb{Q}$, is $\mathbb{A}(G) \triangleright k$ for some (nonempty) language generator G?
$\triangleright$-Universality. Given $k \in \mathbb{Q}$, is $\mathbb{A}(G) \triangleright k$ for every (nonempty) language generator G?

For nonemptiness and universality, we also consider the following variants with restricted quantification over language generators.

Borel. Nonprobabilistic generators G with L_G Borel in Σ^ω under the Cantor topology.
Markov. Probabilistic generators G with μ_G Markovian.
Finite-state. Nonprobabilistic generators G with L_G ω-regular; probabilistic generators G with μ_G finite-state Markovian.

We establish the relations between the questions about QLAs and the corresponding questions about their underlying QWA. For completeness, we provide these problem definitions here. Consider a QWA $\mathcal{A}$, a rational $k \in \mathbb{Q}$, and $\triangleright \in \{>, \geq\}$. Then, $\mathcal{A}$ is $\triangleright$-nonempty (resp -universal) for k iff $\mathcal{A}(w) \triangleright k$ for some (resp. all) $w \in \Sigma^\omega$. Similarly as for the finite-state restriction for QLAs problems, the *lasso-word restriction* for these problems requires quantifying over lasso words instead of all words. Moreover, $\mathcal{A}$ is *approximate-nonempty* for k iff $\top_{\mathcal{A}} \geq k$, i.e., if there exists $w \in \Sigma^\omega$ such that $\mathcal{A}(w) \geq k$ or if for every $\varepsilon > 0$ there exists w such that $\mathcal{A}(w) > k - \varepsilon$. Hence, the achievability of $\top_{\mathcal{A}}$ implies that approximate-nonemptiness and $\geq$-nonemptiness are equivalent problems. Note that $\top_{\mathcal{A}} > k$ iff $\mathcal{A}(w) > k$ for some word w – it holds independently of the achievability of $\top_{\mathcal{A}}$. Dually, $\mathcal{A}$ is *approximate-universal* for k iff $\bot_{\mathcal{A}} > k$.

For the classes of QLAs we consider, the Borel (resp. Markov) restrictions of the decision problems coincide with the unrestricted cases. Thus, for their decision problems we only focus on the unrestricted and finite-state cases.

▶ **Proposition 4.1.** *Consider a QLA $\mathbb{A} = (h, \mathcal{A})$ with $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimSup}, \mathsf{LimInf}\}$ (resp. $h = \mathbb{E}$), a rational $k \in \mathbb{Q}$, and $\triangleright \in \{>, \geq\}$. Then, $\mathbb{A}$ is $\triangleright$-nonempty for k iff $\mathbb{A}$ is Borel $\triangleright$-nonempty (resp. Markov $\triangleright$-nonempty) for k. The same holds for universality.*

For QLAs with language aggregators Inf, Sup, and $\mathbb{E}$, we demonstrate that their decision problems often reduce to those of their underlying QWAs. Moreover, we identify several cases in which QLAs enjoy a finite-model property.

▶ **Proposition 4.2.** *Consider a QLA $\mathbb{A} = (h, \mathcal{A})$ with $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$, a rational $k \in \mathbb{Q}$, and $\triangleright \in \{>, \geq\}$.*
(a) *Unrestricted variant*
  (i) *If $h \neq \mathsf{Sup}$ or $\triangleright = >$, then $\mathbb{A}$ is $\triangleright$-nonempty for k iff $\mathcal{A}$ is $\triangleright$-nonempty for k.*
  (ii) *If $h = \mathsf{Sup}$, then $\mathbb{A}$ is $\geq$-nonempty for k iff $\mathcal{A}$ is approximate-nonempty for k.*
(b) *Finite-state variant*
  (i) *If $h = \mathsf{Sup}$, then $\mathbb{A}$ is finite-state $\triangleright$-nonempty for k iff $\mathbb{A}$ is $\triangleright$-nonempty for k.*

(ii) *If $h = \mathsf{Inf}$, then $\mathbb{A}$ is finite-state $\triangleright$-nonempty for k iff $\mathcal{A}$ is lasso-word $\triangleright$-nonempty for k.*

(iii) *If $\top_{\mathcal{A}}$ is achievable by a lasso word, then $\mathbb{A}$ is finite-state $\triangleright$-nonempty for k iff $\mathbb{A}$ is $\triangleright$-nonempty for k iff $\mathcal{A}$ is lasso-word $\triangleright$-nonempty for k.*

For universality, we have the duals where we exchange Inf with Sup, $>$ with $\geq$, nonempty with universal, approximate-nonempty with approximate-universal, and $\top_{\mathcal{A}}$ with $\bot_{\mathcal{A}}$.

We show that approximating top or bottom values via lasso words lets us reduce certain unrestricted variants to their finite-state counterparts, establishing a finite-model property.

▶ **Proposition 4.3.** *Consider a QLA $\mathbb{A} = (h, \mathcal{A})$ with $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$ and $k \in \mathbb{Q}$. If for every $\varepsilon > 0$ there is a lasso word w such that $\mathcal{A}(w) \geq \top_{\mathcal{A}} - \varepsilon$, then $\mathbb{A}$ is $>$-nonempty for k iff $\mathbb{A}$ is finite-state $>$-nonempty for k. Dually, if for every $\varepsilon > 0$ there is a lasso word w such that $\mathcal{A}(w) \leq \bot_{\mathcal{A}} + \varepsilon$, then $\mathbb{A}$ is $\geq$-universal for k iff $\mathbb{A}$ is finite-state $\geq$-universal for k.*

5 Solving Evaluation

In this section, we investigate the evaluation problem for QLAs with language and word aggregators $h, g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$, for which we provide a full picture of complexity results. First, in Section 5.1, we focus on the nonprobabilistic evaluation problem (where $h \in \{\mathsf{Inf}, \mathsf{Sup}\}$) and then, in Section 5.2, on the probabilistic one (where $h = \mathbb{E}$).

5.1 Nonprobabilistic Evaluation

First, we consider the evaluation problem for QLAs with $h \in \{\mathsf{Inf}, \mathsf{Sup}\}$. We start with QLAs whose underlying word automata are universal or nondeterministic (i.e., $g \in \{\mathsf{Inf}, \mathsf{Sup}\}$). We study various run aggregators f separately and show that the problem is in PTime when the word aggregator g and the language aggregator h coincide. When they differ, the problem becomes harder: while it remains algorithmically solvable in PSpace for the "standard" run aggregators (i.e., $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$), we show that it is not computable for limit-average and at least as hard as a long-standing open problem for discounted-sum. Finally, for QLAs whose underlying word automata are probabilistic (i.e., $g = \mathbb{E}$), we show that the problem is not computable. At their core, the easiness results rely on analyzing the extreme values of the underlying word automata. Similarly, we establish the hardness results by reductions from word automata problems.

QLAs with Standard QWAs. For QLAs with run aggregators $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, the nonprobabilistic evaluation problem can be solved by reasoning on lasso words since both top and bottom values of the underlying word automata are realized by lasso words.

▶ **Theorem 5.1.** *Consider a QLA $\mathbb{A} = (h, (g, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ and $g, h \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let $S \subseteq \Sigma^\omega$ be an ω-regular language given by a Büchi automaton. The value $\mathbb{A}(S)$ is computable in PTime when $g = h$ and in PSpace when $g \neq h$.*

QLAs with Limit-Average QWAs. QLAs with run aggregators $f \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$ differ from the previous case in the sense that it is not sufficient to consider only the lasso words in a given ω-regular language, even when the underlying word automaton is deterministic. To witness this, consider the best-case average uptime QLA $\mathbb{A} = (\mathsf{Sup}, (\mathsf{Sup}, \mathsf{LimInfAvg}, \mathcal{T}_{\text{up}}))$ as in Section 3 and Figure 1. Having $L = (\Sigma^* \texttt{off})^\omega$, we get $\mathbb{A}(L) = 1$ as there is a word in L with infinitely many `off`'s but longer and longer blocks of `on`'s, but no lasso word in L has an average uptime of 1.

▶ **Proposition 5.2.** *There is a QLA $\mathbb{A}$ with a run aggregator $f \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$ and an ω-regular language S such that no lasso word in S achieves the value $\mathbb{A}(S)$.*

Nonetheless, for QLAs with matching word and language aggregators, we show that the value of an ω-regular language given by a Büchi automaton $\mathcal{B}$ is computable by analyzing the strongly connected components (SCCs) of the underlying word automaton's product with $\mathcal{B}$ as follows: Among all SCCs that are reachable from the initial state, we find the ones that contain at least one state whose Büchi component is accepting. Then, in each such SCC, we compute the maximum mean weight of its cycles by Karp's algorithm [22]. The largest among these mean values is exactly the value of the given language. Even though such a cycle may not involve an accepting state of $\mathcal{B}$, we can construct a run of the product that visits an accepting state infinitely often while going over this cycle with increasing frequency (hence the long-run average converges to the cycle's mean). When these aggregators differ, the problem is undecidable by reduction from the universality of limit-average QWAs [9,14,20].

▶ **Theorem 5.3.** *Consider a QLA $\mathbb{A} = (h, (g, f, \mathcal{T}))$ with $f \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$ and $g, h \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let $S \subseteq \Sigma^\omega$ be an ω-regular language given by a Büchi automaton. The value $\mathbb{A}(S)$ is computable in PTIME when $g = h$ and not computable when $g \neq h$.*

QLAs with Discounted-Sum QWAs. QLAs with the run aggregator $f = \mathsf{DSum}$ have the particular behavior that the value assigned to an ω-regular language L may be not achievable by any word in L, even when the underlying word automaton is deterministic. To witness this, consider $\mathbb{A} = (\mathsf{Sup}, \mathcal{A})$ with $\mathcal{A} = (\mathsf{Sup}, \mathsf{DSum}_{0.5}, \mathcal{T}_{\text{up}})$ as in Figure 1. We have $\mathbb{A}((\Sigma^* \texttt{off})^\omega) = 2$ since $\lim_{n\to\infty} \mathcal{A}(\texttt{on}^n \texttt{off}^\omega) = 2$. However, only for $w = \texttt{on}^\omega \notin (\Sigma^* \texttt{off})^\omega$ we have $\mathcal{A}(w) = 2$.

▶ **Proposition 5.4.** *There is a QLA $\mathbb{A}$ with the run aggregator $f = \mathsf{DSum}$ and an ω-regular language S such that no word in S achieves the value $\mathbb{A}(S)$.*

We establish that such a behavior is not possible when the input language includes all its limit points, i.e., it is safe in the boolean sense [2,24]: Consider a sequence of words in the safety language whose values approach the supremum. We build by a diagonalization argument an infinite word w whose every finite prefix already appears in the language, so w is in the safety language. Applying the same construction to the corresponding optimal runs yields an infinite run on w. This run's value equals the supremum since the contribution of the remaining tail is bounded by a vanishing geometric series due to discounting.

▶ **Proposition 5.5.** *Consider a QLA $\mathbb{A} = (\mathsf{Sup}, (\mathsf{Sup}, \mathsf{DSum}, \mathcal{T}))$. For every nonempty safety language $S \subseteq \Sigma^\omega$, the value $\mathbb{A}(S)$ is achievable by some run of a word in S.*

The value of a language S matches that of its safety closure S' (i.e., the smallest safety language containing it) because every word in the safety closure can be approximated arbitrarily closely by words from the original language: If S' achieves a value on a word w, we can isolate a prefix of w whose contribution is close enough to the value of w. By construction, the same prefix also occurs in a word of S, and completing the run along this word in S can change the total value by at most an arbitrarily small amount due to discounting. Hence the maximal value in S' can be approximated arbitrarily closely by words in S, and the two suprema coincide.

▶ **Proposition 5.6.** *Consider a QLA $\mathbb{A} = (\mathsf{Sup}, (\mathsf{Sup}, \mathsf{DSum}, \mathcal{T}))$. For every language $S \subseteq \Sigma^\omega$ we have $\mathbb{A}(S) = \mathbb{A}(S')$ where S' is the safety closure of S.*

The observation above helps us provide a PTIME algorithm when the word and language aggregators match: We first construct the Büchi automaton's safety closure, so the optimal value is achieved by a run that never reaches the rejecting sink. Then, we compute the product of the underlying word automaton and the safety closure automaton. Computing the best (or worst) discounted sum over all sink-avoiding paths in the product can be done by solving a one-player discounted-payoff game [3]. When the two aggregators differ, the evaluation problem is at least as hard as the universality problem for nondeterministic discounted-sum automata, which is a long-standing open problem [6, 12].

▶ **Theorem 5.7.** *Consider a QLA $\mathbb{A} = (h, (g, \mathsf{DSum}, \mathcal{T}))$ with $g, h \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let $S \subseteq \Sigma^\omega$ be an ω-regular language given by a Büchi automaton. The value $\mathbb{A}(S)$ is computable in PTIME when $g = h$. If $\mathbb{A}(S)$ is computable when $g \neq h$, then the $\geq$-universality of nondeterministic discounted-sum word automata is decidable.*

QLAs with Probabilistic QWAs. When the underlying word automaton is probabilistic, i.e., has the word aggregator $g = \mathbb{E}$, the nonprobabilistic evaluation problem has no algorithmic solution due to inapproximability of their top values [26].

▶ **Theorem 5.8.** *Consider a QLA $\mathbb{A} = (h, (\mathbb{E}, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}, \mathsf{LimInfAvg}, \mathsf{LimSupAvg}, \mathsf{DSum}\}$ and $h \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let μ be a probability measure given by a finite-state Markov chain. The value $\mathbb{A}(\mu)$ is not computable.*

5.2 Probabilistic Evaluation

Now, we consider the evaluation problem for QLAs with $h = \mathbb{E}$ and follow the same structure as in Section 5.1: we start with the cases of $g \in \{\mathsf{Inf}, \mathsf{Sup}\}$ and study various run aggregators f separately, and then look at the case of $g = \mathbb{E}$.

QLAs with Standard QWAs. First, we provide an EXPTIME algorithm for QLAs with the run aggregators $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$. Our proof builds on that of [27, Thm. 8], which considers only $f \in \{\mathsf{Inf}, \mathsf{Sup}\}$. The idea is to determinize the underlying word automaton, which leads to an exponential blow-up, and evaluate its product with the given Markov chain.

▶ **Theorem 5.9.** *Consider a QLA $\mathbb{A} = (\mathbb{E}, (g, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ and $g \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let μ be a probability measure given by a finite-state Markov chain. The value $\mathbb{A}(\mu)$ is computable in EXPTIME.*

QLAs with Limit-Average QWAs. Undecidability of probabilistic evaluation for limit-average QLAs was shown in [27] by a reduction from the universality problem of quantitative automata on finite words with the summation run aggregator and weights in $\{-1, 0, 1\}$, a.k.a. weighted automata over the tropical semiring of integers [1, 23].

▶ **Theorem 5.10** ([27, Thm. 7]). *Consider a QLA $\mathbb{A} = (\mathbb{E}, (g, f, \mathcal{T}))$ with $f \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$ and $g \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let μ be a probability measure given by a finite-state Markov chain. The value $\mathbb{A}(\mu)$ is not computable.*

QLAs with Discounted-Sum QWAs. Next, we show the hardness of probabilistic evaluation for DSum QLAs. As in the nonprobabilistic case, we provide a reduction from the universality problem of nondeterministic discounted-sum QWAs.

▶ **Theorem 5.11.** *Consider a QLA $\mathbb{A} = (\mathbb{E}, (g, \mathsf{DSum}, \mathcal{T}))$ with $g \in \{\mathsf{Inf}, \mathsf{Sup}\}$. Let μ be a probability measure given by a finite-state Markov chain. If $\mathbb{A}(\mu)$ is computable, then the $\geq$-universality of nondeterministic discounted-sum word automata is decidable.*

QLAs with Probabilistic QWAs. Finally, for QLAs with $h = g = \mathbb{E}$, the evaluation problem reduces to evaluating a Markov chain with rewards on infinite words, which is solved using linear programming [21].

▶ **Theorem 5.12** ([21])**.** *Consider a QLA $\mathbb{A} = (\mathbb{E}, (\mathbb{E}, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}, \mathsf{LimInfAvg}, \mathsf{LimSupAvg}, \mathsf{DSum}\}$. Let μ be a probability measure given by a finite-state Markov chain. The value $\mathbb{A}(\mu)$ is computable in* PTIME*.*

6 Deciding Nonemptiness and Universality

In this section, we investigate the nonemptiness and universality problems for QLAs with language and word aggregators $h, g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$. We give a complete picture of decidability results for the unrestricted cases. When the unrestricted cases are decidable, our algorithms provide a solution for the finite-state cases in the same complexity class. When they are undecidable or not known to be decidable, some finite-state cases remain open, which we make explicit in the corresponding statements.

Thanks to the duality between nondeterministic and universal automata, solving the nonemptiness problem solves the universality problem as well.

▶ Remark 6.1. Consider a QLA $\mathbb{A} = (h, \mathcal{A})$ and its dual $\hat{\mathbb{A}}$. Let $k \in \mathbb{Q}$. Thanks to Proposition 2.2, the QLA $\mathbb{A}$ is $\geq$-nonempty (resp. $\geq$-universal) for k iff $\hat{\mathbb{A}}$ is not $>$-universal (resp. $>$-nonempty) for $-k$. The statement holds also for the finite-state restriction.

The rest of the section focuses on the nonemptiness problem and is organized by the type of the underlying QWAs. We first consider the case of nondeterministic automata (i.e., $g = \mathsf{Sup}$) and show that the problem is decidable in PTIME for these. Then, we consider universal automata (i.e., $g = \mathsf{Inf}$) with various run aggregators f separately. Similarly as for the evaluation problem, nonemptiness is in PSPACE for the "standard" run aggregators, undecidable for limit-average, and at least as hard as a long-standing open problem for discounted sum. Finally, we consider probabilistic automata (i.e., $g = \mathbb{E}$) and show that the problem is undecidable.

QLAs with Nondeterministic QWAs. Let us first consider QLAs whose underlying word automata are nondeterministic, i.e., those where the word aggregator is $g = \mathsf{Sup}$. We show that the nonemptiness problems for such QLAs can be solved efficiently, independently of the choice of the remaining aggregators or additional restrictions on the problem. Intuitively, this is because their top values coincide with those of the underlying QWAs (Proposition 2.1), which are achievable by lasso words and can be computed efficiently.

▶ **Theorem 6.2.** *Consider a QLA $\mathbb{A} = (h, (\mathsf{Sup}, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}, \mathsf{LimInfAvg}, \mathsf{LimSupAvg}, \mathsf{DSum}\}$ and $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$. Let $\triangleright \in \{>, \geq\}$. The $\triangleright$-nonemptiness of $\mathbb{A}$ is in* PTIME*. The statement holds also for the finite-state restriction.*

QLAs with Universal Standard QWAs. We turn our attention to QLAs whose underlying word automata are universal, i.e., those where the word aggregator is $g = \mathsf{Inf}$. For run aggregators $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, we show that computing the automaton's top value suffices, which can be done in PSPACE when $g = \mathsf{Inf}$.

▶ **Theorem 6.3.** *Consider a QLA $\mathbb{A} = (h, (\mathsf{Inf}, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ and $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$. Let $\triangleright \in \{>, \geq\}$. The $\triangleright$-nonemptiness of $\mathbb{A}$ is in PSpace. The statement holds also for the finite-state restriction.*

QLAs with Universal Limit-Average QWAs. Next, we focus on QLAs with the limit-average run aggregators. We first show that the bottom value of nondeterministic $\mathsf{LimSupAvg}$ QWAs (dually, top value of universal $\mathsf{LimInfAvg}$ QWAs) can be approximated arbitrarily closely by lasso words. This is in stark contrast with nondeterministic $\mathsf{LimInfAvg}$ QWAs, where an automaton may be lasso-word universal for some threshold k while there exist non-lasso words with values below k [12, Lem. 4]. The intuition behind this is that for $\mathsf{LimSupAvg}$, if $\mathcal{A}(w) = k$ for some word w and value k, then for any $\varepsilon > 0$, there exists a length ℓ such that the average value of any run prefix on w of length at least ℓ is below $k + \varepsilon$, which does not hold in general for $\mathsf{LimInfAvg}$. Consequently, we can find two distant-enough occurrences of the same state and pump the low-average segment between them to obtain a lasso whose overall value stays under the threshold.

▶ **Proposition 6.4.** *Consider a QWA $\mathcal{A} = (\mathsf{Sup}, \mathsf{LimSupAvg}, \mathcal{T})$. For every $\varepsilon > 0$ there is a lasso word w such that $\mathcal{A}(w) < \bot_{\mathcal{A}} + \varepsilon$.*

In general, all variants of the nonemptiness problem for universal limit-average QWAs (dually, universality for the nondeterministic ones) are undecidable [9, 14, 20]. Combining these with our approximability result above, we obtain the following.

▶ **Theorem 6.5.** *Consider a QLA $\mathbb{A} = (h, (\mathsf{Inf}, f, \mathcal{T}))$ with $f \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$ and $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$. Let $\triangleright \in \{>, \geq\}$. The $\triangleright$-nonemptiness of $\mathbb{A}$ is undecidable. The statement holds also for the finite-state restriction when (i) $h \neq \mathbb{E}$ or (ii) $f = \mathsf{LimInfAvg}$ and $\triangleright = >$.*

QLAs with Universal Discounted-Sum QWAs. Now, we consider QLAs with the discounted-sum run aggregators. We show that the bottom value of nondeterministic DSum QWAs (dually, top value of universal DSum QWAs) is achievable. At a high level, we repeatedly extend a finite prefix with a letter that does not increase the current infimum over continuation values. In the limit, this process results in an infinite word whose sequence of prefix infima converges to the automaton's bottom value. Since discounted-sum automata are co-safe in the quantitative sense [4, 5, 18], i.e., the value of every infinite word equals the limit of its prefix infima, this constructed word attains exactly the bottom value.

▶ **Proposition 6.6.** *Consider a QWA $\mathcal{A} = (\mathsf{Sup}, \mathsf{DSum}, \mathcal{T})$. There is a word w such that $\mathcal{A}(w) = \bot_{\mathcal{A}}$.*

By combining the achievability result above with Proposition 4.2, we obtain a concise proof to establish that the nonemptiness problem for QLAs whose underlying QWAs are universal DSum QWAs is at least as hard as the universality problem for nondeterministic DSum QWAs, which is a long-standing open problem [6, 12].

▶ **Theorem 6.7.** *Consider a QLA $\mathbb{A} = (h, (\mathsf{Inf}, \mathsf{DSum}, \mathcal{T}))$ with $h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathbb{E}\}$. Let $\triangleright \in \{>, \geq\}$. If the $\triangleright$-nonemptiness of $\mathbb{A}$ is decidable, then the universality of nondeterministic discounted-sum word automata is decidable. The statement holds also for the finite-state restriction when (i) $h = \mathsf{Sup}$ or (ii) $\triangleright = >$.*

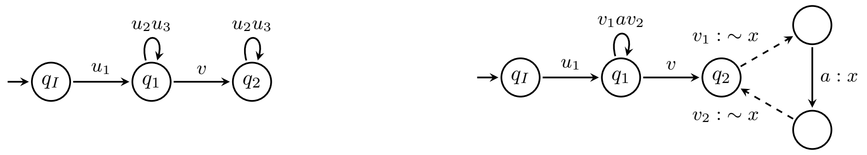

Figure 2 (a) Pattern parameterized by q_2 and n where $u_1, u_2, u_3, v \in \Sigma^+$ are such that $|u_2| = |v|$, $u_2 \neq v$, and $m|v| = |u_2u_3|$ for some $m \in \{1, \ldots, 4n^4\}$. **(b)** Pattern parameterized by $x \in \mathbb{Q}$, $n \in \mathbb{N}$, $\sim \in \{\leq, \geq\}$, and $w \in \Sigma^\omega$, where $u_1, v, v_1, v_2 \in \Sigma^*$ and $a \in \Sigma$ are such that $q_1 \neq q_2$, $w = u(v_1av_2)^\omega$, and $(v)^m = v_1av_2$ for some $m \in \{1, \ldots, 4n^4\}$. Dashed runs visit weights $y \in \mathbb{Q}$ satisfying $y \sim x$.

QLAs with Probabilistic QWAs. Let us now focus on QLAs whose underlying QWA is probabilistic, i.e., where $g = \mathbb{E}$. We show that several variants of the nonemptiness problem for probabilistic QWAs are undecidable by building on the undecidability results of probabilistic word automata [26].

▶ **Theorem 6.8.** *Consider a QWA $\mathcal{A} = (\mathbb{E}, f, \mathcal{T})$ with $f \in \{$*Inf*, *Sup*, *LimInf*, *LimSup*, *LimInfAvg*, *LimSupAvg*, *DSum*$\}$. Let $\triangleright \in \{>, \geq\}$. The $\triangleright$-nonemptiness of $\mathcal{A}$ is undecidable, whether we consider all words from Σ^ω or only lasso words. Moreover, its approximate-nonemptiness is also undecidable.*

Finally, putting together the above undecidability result with Propositions 4.2 and 4.3, we show the undecidability of the nonemptiness problem for QLAs whose underlying QWA is probabilistic.

▶ **Theorem 6.9.** *Consider a QLA $\mathbb{A} = (h, (\mathbb{E}, f, \mathcal{T}))$ with $f \in \{$*Inf*, *Sup*, *LimInf*, *LimSup*, *LimInfAvg*, *LimSupAvg*, *DSum*$\}$ and $h \in \{$*Inf*, *Sup*, $\mathbb{E}\}$. Let $\triangleright \in \{>, \geq\}$. The $\triangleright$-nonemptiness of $\mathbb{A}$ is undecidable. The statement holds also for the finite-state restriction when (i) $h \neq \mathbb{E}$ or (ii) $f =$ *DSum* and $\triangleright = >$.*

7 Language Automata with Limit Aggregators

We finally focus on QLAs with the aggregators $h, g, f \in \{$Inf, Sup, LimInf, LimSup$\}$ where at least one of the language aggregator h or the word aggregator g belongs to $\{$LimInf, LimSup$\}$.

7.1 Deciding Infinite Achievability

To work with QLAs that use LimInf or LimSup as word or language aggregators, we need to reason about the values that appear infinitely often in a multiset. In this subsection, we develop the necessary tools for this purpose, focusing on QLAs with run aggregators $f \in \{$Inf, Sup, LimInf, LimSup$\}$.

We begin by characterizing the conditions under which a state in an automaton can be visited infinitely often by infinitely many words. The following lemma uses proof techniques similar to those used in the characterization of ambiguity in finite-state automata [25, 30].

▶ **Lemma 7.1.** *Let $\mathcal{T}$ be a (weighted) labeled transition system with n states and q_I its initial state. For every state q_2 of $\mathcal{T}$, there exist infinitely many words with a run from q_I visiting q_2 infinitely often iff $\mathcal{T}$ complies with the pattern parameterized by q_2 and n given in Figure 2a.*

It is well known that Büchi automata can be effectively complemented in ExpTime [17,29]. Moreover, checking whether a finite-state automaton satisfies a given syntactic pattern, such as the one in Figure 2a, is in NLogSpace [15]. As a direct consequence of Lemma 7.1, keeping a symbolic representation of the complement provides the following PSpace procedure.

▶ **Corollary 7.2.** *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two Büchi automata. We can decide in* PSPACE *whether $L(\mathcal{B}_1) \setminus L(\mathcal{B}_2)$ is infinite.*

By interpreting the transitions of a run as input letters, we can use Figure 2a to reason about runs instead of words. Building on Lemma 7.1, we present a similar pattern that incorporates weights, enabling us to identify the words for which an automaton admits infinitely many runs of a given value.

▶ **Lemma 7.3.** *Consider a QWA $\mathcal{A} = (g, f, \mathcal{T})$ with n states, $f \in \{\mathsf{LimInf}, \mathsf{LimSup}\}$, $g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, and initial state q_I. Let $\sim = \leq$ if $f = \mathsf{LimSup}$ and $\sim = \geq$ if $f = \mathsf{LimInf}$. For every $x \in \mathbb{Q}$ and every lasso word $w \in \Sigma^\omega$, we have that w admits infinitely many runs of value x in $\mathcal{A}$ iff $\mathcal{A}$ complies with the pattern in Figure 2b parameterized by x, n, $\sim$, and w.*

As a corollary of Lemma 7.3, we can construct a boolean automaton accepting every lasso word with infinitely many runs of value x in $\mathcal{A}$. To achieve this, we construct a Büchi or co-Büchi automaton that initially guesses a fixed pair of states q_1 and q_2 with the same properties as in Figure 2b. The automaton then guesses on-the-fly the segments u_1 and v from Figure 2b. During the processing of each period v, it simulates $3 \times m$ runs, with $m \in \{1, \ldots, 4n^4\}$, simultaneously verifying weights and connectivity. At the end of each period, the automaton resets (except for the fixed q_1 and q_2) to guess another period and visit an accepting state. Consequently, any accepted lasso word resets infinitely often and ultimately repeats a finite periodic segment v, complying with Figure 2b. Conversely, every word conforming to Figure 2b is accepted by appropriately selecting $v = v_1 a v_2$.

▶ **Corollary 7.4.** *Let $\mathcal{A} = (\mathsf{Sup}, f, \mathcal{T})$ be QWA with $f = \mathsf{LimSup}$ (resp. $f = \mathsf{LimInf}$). Given $x \in \mathbb{Q}$, we can construct in* PTIME *a Büchi automaton (resp. a co-Büchi automaton) recognizing all lasso words admitting infinitely many runs of value x in $\mathcal{A}$.*

7.2 Expressive Power of Limit Aggregators

This subsection investigates the expressive power of word and language aggregators $g, h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$. The following proposition generalizes the results of [4, 12] to QLAs.

▶ **Proposition 7.5.** *Consider a QLA $\mathbb{A} = (h, (g, f, \mathcal{T}))$ with $f \in \{\mathsf{Inf}, \mathsf{Sup}\}$, $g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, and h any language aggregator function. We can construct in* PTIME *a QLA $\mathbb{B} = (h, (g, f', \mathcal{T}'))$ with $f' \in \{\mathsf{LimInf}, \mathsf{LimSup}\}$ such that $\mathbb{A}(S) = \mathbb{B}(S)$ for all $S \subseteq \Sigma^\omega$.*

▶ Remark 7.6. The construction of [12] allowing to convert a nondeterministic QWA with run aggregator LimInf to one with LimSup can be extended to QLAs with word aggregator Sup but not with LimSup. By duality, the conversion of a universal LimSup QWA to a LimInf one can be extended to QLAs with word aggregators Inf but not LimInf. This is because deterministic LimInf QWA and deterministic LimSup QWA are expressively incomparable [12]. Hence, there is no conversion that preserves, for every word w, the number runs with the same value over w.

We now focus on word aggregators, first showing that Inf and Sup are at least as expressive as LimInf and LimSup when they are combined with the run aggregators $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$. The construction relies on Corollary 7.4, the closure of QWAs under max and min operation [11], and an argument that equivalence of QWAs with run and word aggregators in $\{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ can be determined by considering only lasso words.

▶ **Lemma 7.7.** *Let $\mathbb{A} = (h, (g, f, \mathcal{T}))$ be a QLA with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, $g = \mathsf{LimSup}$ (resp. $g = \mathsf{LimInf}$), and h any language aggregator function. We can construct in* PSPACE *a QLA $\mathbb{B} = (h, (g', f', \mathcal{T}'))$ with $f' = \mathsf{LimSup}$ and $g' = \mathsf{Sup}$ (resp. $f' = \mathsf{LimInf}$ and $g' = \mathsf{Inf}$) such that $\mathbb{A}(S) = \mathbb{B}(S)$ for all $S \subseteq \Sigma^\omega$.*

Now, we prove the other direction: the word aggregators LimInf and LimSup are at least as expressive as Inf and Sup for any run aggregator and any language aggregator. The idea is to duplicate the transition system so that the automaton can switch between the two copies at any step, ensuring that every word yields infinitely many runs with the same value.

▶ **Lemma 7.8.** *Let $\mathbb{A} = (h, (g, f, \mathcal{T}))$ be a QLA with f any run aggregator function, $g =$ Sup (resp. $g =$ Inf), and h any language aggregator function. We can construct in PTime a QLA $\mathbb{B} = (h, (g', f, \mathcal{T}'))$ where $g' =$ LimSup (resp. $g' =$ LimInf) such that $\mathbb{A}(S) = \mathbb{B}(S)$ for all $S \subseteq \Sigma^\omega$.*

The following comes as a consequence of Lemmas 7.7 and 7.8.

▶ **Theorem 7.9.** *For QLAs with run aggregators $f \in \{$Inf, Sup, LimInf, LimSup$\}$, the word aggregators Inf and LimInf (resp. Sup and LimSup) are equally expressive.*

Finally, we turn our attention to language aggregators. In contrast to the case of word aggregators, even when combined with run aggregators $f \in \{$Inf, Sup, LimInf, LimSup$\}$, the language aggregators Inf and Sup are expressively incomparable with LimInf and LimSup. Intuitively, when the top or the bottom value of the underlying QWA is achievable by a single word, a QLA with a limit language aggregator cannot achieve this value, while one with a non-limit language aggregator can. Conversely, if the extreme value of a QLA emerges only as the value of finite languages, a limit language aggregator can capture this behavior, while a non-limit aggregator cannot.

▶ **Proposition 7.10.** *QLAs with the language aggregators Inf and LimInf (resp. Sup and LimSup) are expressively incomparable.*

7.3 Decision Problems with Limit Aggregators

Finally, we consider the evaluation, nonemptiness, and universality problems for these classes of QLAs. We first provide a PSpace procedure for evaluation. Intuitively, each weight appearing on the underlying transition system are treated individually since the value of an input language must be one of them. For $h \in \{$LimSup, LimInf$\}$, we leverage Corollary 7.2 to identify the weights achievable by infinitely many words from the input languages in PSpace. The problem reduces to checking the nonemptiness of a Büchi automaton when $h \in \{$LimSup, Sup$\}$ and to checking their inclusion when $h \in \{$LimInf, Inf$\}$. We note that the proof of Theorem 7.11 also establishes the PTime cases presented in Theorem 5.1.

▶ **Theorem 7.11.** *Consider a QLA $\mathbb{A} = (h, (g, f, \mathcal{T}))$ with $f, g, h \in \{$Inf, Sup, LimInf, LimSup$\}$ and at least one g and h belong to $\{$LimInf, LimSup$\}$. Let $S \subseteq \Sigma^\omega$ be an ω-regular language given by a Büchi automaton. The value $\mathbb{A}(S)$ is computable in PSpace.*

Next, we prove that QLAs with $f, g, h \in \{$Inf, Sup, LimInf, LimSup$\}$, checking whether a given QLA is an upper bound on another (namely, their inclusion problem) is decidable in PSpace. The proof starts by showing that this problem can be solved while reasoning exclusively on ω-regular languages and then generalizes our algorithm for the evaluation problem (Theorem 5.1) to handle two QLAs.

▶ **Theorem 7.12.** *Consider two QLAs $\mathbb{A} = (h, (g, f, \mathcal{T}))$ and $\mathbb{B} = (h', (g', f', \mathcal{T}'))$ with $f, f', g, g', h, h' \in \{$Inf, Sup, LimInf, LimSup$\}$. Let $\triangleright \in \{>, \geq\}$. Deciding whether $\mathbb{A}(S) \triangleright \mathbb{B}(S)$ for every language $S \subseteq \Sigma^\omega$ is in PSpace. The same holds when S ranges over ω-regular languages.*

Using Theorem 7.12, we obtain a matching solution to the corresponding nonemptiness and universality problems. Note that Theorems 6.2 and 6.3 capture the cases of non-limit aggregators, including the PTime fragments.

▶ **Corollary 7.13.** *Consider a QLA $\mathbb{A} = (h, (g, f, \mathcal{T}))$ with $f, g, h \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$ and at least one g and h belong to $\{\mathsf{LimInf}, \mathsf{LimSup}\}$. Let $\triangleright \in \{>, \geq\}$. The $\triangleright$-nonemptiness (resp. $\triangleright$-universality) of $\mathbb{A}$ is in PSpace. The statement holds also for the finite-state restriction.*

8 Conclusion

We introduced quantitative language automata (QLAs) as a uniform framework for specifying and verifying quantitative hyperproperties. Our framework extends beyond both the traditional boolean perspective of system properties and the "one-trace limitation" of traditional quantitative properties as system specifications, enabling reasoning about quantitative aspects of system-wide behaviors such as performance and robustness. We established a thorough foundation for QLAs by investigating the evaluation, nonemptiness, and universality problems, for which we provided a extensive picture of decidability results. In addition to closing the finite-state cases we left open, future research directions include exploring aggregators capable of specifying richer relational system properties, investigating decidable expressive fragments of QLAs, studying equivalent logical formalisms, and augmenting the software tool Quantitative Automata Kit (QuAK) [7,8] with a support for QLAs.

References

1 Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? *Inf. Comput.*, 282:104651, 2022. `doi:10.1016/J.IC.2020.104651`.

2 Bowen Alpern and Fred B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985. `doi:10.1016/0020-0190(85)90056-0`.

3 Daniel Andersson. An improved algorithm for discounted payoff games. In Janneke Huitink and Sophia Katrenko, editors, *Proceedings of the 11th ESSLLI Student Session*, pages 91–98, August 2006.

4 Udi Boker, Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Safety and liveness of quantitative automata. In Guillermo A. Pérez and Jean-François Raskin, editors, *34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium*, volume 279 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPICS.CONCUR.2023.17`.

5 Udi Boker, Thomas A Henzinger, Nicolas Mazzocchi, and N Ege Saraç. Safety and liveness of quantitative properties and automata. *Logical Methods in Computer Science*, 21, 2025.

6 Udi Boker, Thomas A. Henzinger, and Jan Otop. The target discounted-sum problem. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 750–761. IEEE Computer Society, 2015. `doi:10.1109/LICS.2015.74`.

7 Marek Chalupa, Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Quak: Quantitative automata kit. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Software Engineering Methodologies - 12th International Symposium, ISoLA 2024, Crete, Greece, October 27-31, 2024, Proceedings, Part IV*, volume 15222 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2024. `doi:10.1007/978-3-031-75387-9_1`.

8 Marek Chalupa, Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Automating the analysis of quantitative automata with quak. In Arie Gurfinkel and Marijn Heule, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 31st International*

Conference, TACAS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings, Part I, volume 15696 of *Lecture Notes in Computer Science*, pages 303–312. Springer, 2025. doi:10.1007/978-3-031-90643-5_16.

9 Krishnendu Chatterjee, Laurent Doyen, Herbert Edelsbrunner, Thomas A. Henzinger, and Philippe Rannou. Mean-payoff automaton expressions. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 269–283. Springer, 2010. doi:10.1007/978-3-642-15375-4_19.

10 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In Miroslaw Kutylowski, Witold Charatonik, and Maciej Gebala, editors, *Fundamentals of Computation Theory, 17th International Symposium, FCT 2009, Wroclaw, Poland, September 2-4, 2009. Proceedings*, volume 5699 of *Lecture Notes in Computer Science*, pages 3–13. Springer, 2009. doi:10.1007/978-3-642-03409-1_2.

11 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. *Log. Methods Comput. Sci.*, 6(3), 2010. URL: http://arxiv.org/abs/1007.4018.

12 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4):23:1–23:38, 2010. doi:10.1145/1805950.1805953.

13 Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.

14 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Torunczyk. Energy and mean-payoff games with imperfect information. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2010. doi:10.1007/978-3-642-15205-4_22.

15 Emmanuel Filiot, Nicolas Mazzocchi, and Jean-François Raskin. A pattern logic for automata with outputs. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, volume 11088 of *Lecture Notes in Computer Science*, pages 304–317. Springer, 2018. doi:10.1007/978-3-319-98654-8_25.

16 Bernd Finkbeiner, Christopher Hahn, and Hazem Torfah. Model checking quantitative hyperproperties. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, volume 10981 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2018. doi:10.1007/978-3-319-96145-3_8.

17 Vojtech Havlena, Ondrej Lengál, and Barbora Smahlíková. Complementing Büchi automata with ranker. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 188–201. Springer, 2022. doi:10.1007/978-3-031-13188-2_10.

18 Thomas A. Henzinger, Nicolas Mazzocchi, and N. Ege Saraç. Quantitative safety and liveness. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*, volume 13992 of *Lecture Notes in Computer Science*, pages 349–370. Springer, 2023. doi:10.1007/978-3-031-30829-1_17.

19 Thomas A. Henzinger and Jan Otop. From model checking to model measuring. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013.*

Proceedings, volume 8052 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2013. doi:10.1007/978-3-642-40184-8_20.

20 Paul Hunter, Arno Pauly, Guillermo A. Pérez, and Jean-François Raskin. Mean-payoff games with partial observation. *Theor. Comput. Sci.*, 735:82–110, 2018. doi:10.1016/J.TCS.2017.03.038.

21 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4):373–396, 1984. doi:10.1007/BF02579150.

22 Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discret. Math.*, 23(3):309–311, 1978. doi:10.1016/0012-365X(78)90011-0.

23 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Int. J. Algebra Comput.*, 4(3):405–426, 1994. doi:10.1142/S0218196794000063.

24 Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Trans. Software Eng.*, 3(2):125–143, 1977. doi:10.1109/TSE.1977.229904.

25 Christof Löding and Anton Pirogov. On finitely ambiguous Büchi automata. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, volume 11088 of *Lecture Notes in Computer Science*, pages 503–515. Springer, 2018. doi:10.1007/978-3-319-98654-8_41.

26 Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34, 2003. doi:10.1016/S0004-3702(02)00378-8.

27 Jakub Michaliszyn and Jan Otop. Non-deterministic weighted automata evaluated over Markov chains. *J. Comput. Syst. Sci.*, 108:118–136, 2020. doi:10.1016/J.JCSS.2019.10.001.

28 Shubham Sahai, Pramod Subramanyan, and Rohit Sinha. Verification of quantitative hyperproperties using trace enumeration relations. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, pages 201–224. Springer, 2020. doi:10.1007/978-3-030-53288-8_11.

29 Sven Schewe. Büchi complementation made tight. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 661–672. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPICS.STACS.2009.1854.

30 Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991. doi:10.1016/0304-3975(91)90381-B.

31 Linpeng Zhang, Noam Zilberstein, Benjamin Lucien Kaminski, and Alexandra Silva. Quantitative weakest hyper pre: Unifying correctness and incorrectness hyperproperties via predicate transformers. *CoRR*, abs/2404.05097, 2024. doi:10.48550/arXiv.2404.05097.

A Proofs of Section 7

▶ **Lemma 7.1.** Let $\mathcal{T}$ be a (weighted) labeled transition system with n states and q_I its initial state. For every state q_2 of $\mathcal{T}$, there exist infinitely many words with a run from q_I visiting q_2 infinitely often iff $\mathcal{T}$ complies with the pattern parameterized by q_2 and n given in Figure 2a.

Proof. We start with the backward implication. Given the state q_2, we assume that $\mathcal{T}$ complies with the pattern in Figure 2a exhibiting four finite runs of the form $\rho_1 : q_I \xrightarrow{u_1} q_1$, $\ell_1 : q_1 \xrightarrow{u_2 u_3} q_1$, $\rho_2 : q_1 \xrightarrow{v} q_2$, $\ell_2 : q_2 \xrightarrow{u_2 u_3} q_2$. Additionally we have that $|u_2| = |v|$ and $u_2 \neq v$. For all $i \in \mathbb{N}$, we define the run $\pi_i = \rho_1(\ell_1)^i \rho_2(\ell_2)^\omega$ from q_I over $u_1(u_2u_3)^i v(u_2u_3)^\omega$.

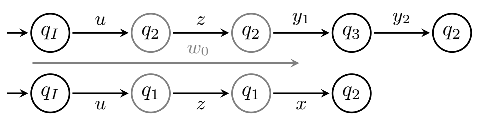

Figure 3 Run synchronization in proof of Lemma 7.1, where $u, x, y_1, y_2, z \in \Sigma^+$ are such that $|x| = |y_1|$, $x \neq y_1$, $|z| = |y_1 y_2|$, and $m|x| = |z|$ for some $m \in \{1, \dots, 4n^4\}$.

Since $|u_2| = |v|$ and $u_2 \neq v$, we have that π_i and π_j read distinct words for all $i \neq j$. In particular, the set $\{u_1(u_2u_3)^i v(u_2u_3)^\omega \mid i \in \mathbb{N}\}$ of ultimately periodic words with a run visiting q_2 infinitely often is infinite.

Next, we prove the forward implication having an infinite set $S \subseteq \Sigma^\omega$ of words admitting some run from the initial state q_I that visits the state q_2 infinitely often. Consider a word $w_1 \in S$ together with one of its runs π_1 that visits q_2 infinitely. Let $w_0 \in \Sigma^*$ be a finite prefix of w_1 for which π_1 visits at least n^2 times the state q_2 while reading w_0, where n refers to the number of states of $\mathcal{T}$. Assuming that w_1 mismatches with all the other words of S exclusively over its $|w_0|$ first letters contradicts $|S| = \infty$. Hence, there exists another word $w_2 \in S$, distinct from w_1, but starting with the finite prefix w_0. Let π_2 be a run over w_2 that visits q_2 infinitely often. Since π_1 visits n^2 times the state q_2 while reading w_0, by the pigeon hole principle, there exists a state q_1 such that the pair (q_1, q_2) is visited twice by (π_1, π_2) while reading w_0. Figure 3 displays the two runs π_1 and π_2. The finite word z refers the factor of w_0 corresponding to the synchronized loop, and the finite word u stands for the prefix of w_0 before z. We ensure that $z \leq n^2$ by removing inner synchronized loops. We denote by x and y_1 the synchronized section where π_1 and π_2 mismatch. In particular, $|x| = |y_1|$ and $x \neq y_1$. Since both π_1 and π_2 visit q_2 infinitely often, we can take x and y_1 long enough so that x leads to q_2. Then, we denote by $\hat{y}_2$ the continuation of y_1 ending in q_2. Additionally, we ensure that $|y_1\hat{y}_2| \leq 3n^2 + 1$. Indeed, any synchronized loop in the factors x and y_1 appearing before or after the mismatching letter can be removed without changing the reachability of q_2. Moreover, any loop in the factor $\hat{y}_2$ can be removed as well. Note that z, x, y_1 are nonempty by construction. By unfolding the synchronized loop, we can assume without loss of generality that all u and $\hat{y}_2$ are nonempty. Hence, all words appearing in Figure 3 are nonempty. From $\hat{y}_2$, we define y_2 as $\hat{y}_2(y_1\hat{y}_2)^{|x|-1}$. This implies that $|y_1y_2| = m|x|$ for some $m \in \{1, \dots, 3n^2 + 1\}$. Furthermore, we can assume that $|z| = |y_1y_2|$, otherwise we lengthen y_2 and z by pumping the synchronized loop over z. In particular $|z| = |y_1y_2| = m|x|$ for some $m \in \{1, \dots, 4n^4\}$.

Now we distinguish the following two cases. First, if $z \neq y_1y_2$, then we define $u_1 = u$, $u_2 = z$, $u_3 = \varepsilon$, and $v = y_1y_2$. Second, if $z = y_1y_2$, then we define $u_1 = u$, $u_2 = y_1$, $u_3 = y_2$, and $v = x$. Either way, u_1, u_2, u_3, v allow $\mathcal{T}$ to comply with the pattern in Figure 2a. ◀

▶ **Lemma A.1.** *Consider a QWA $\mathcal{A} = (g, f, \mathcal{T})$ with n states, $f \in \{\mathsf{LimInf}, \mathsf{LimSup}\}$, $g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$. Given $x \in \mathbb{Q}$, we can construct in polynomial time a Büchi automaton $\mathcal{B}$ such that (i) $L(\mathcal{B}) = \{w \in \Sigma^\omega \mid \mathcal{A}(w) = x\}$ and (ii) for every word w, the automaton $\mathcal{B}$ admits infinitely many accepting runs over w iff $\mathcal{A}$ admits infinitely many runs of value x over w.*

Proof. We start by constructing the Büchi automaton $\mathcal{B}_1$ that captures the runs visiting infinitely a transition of weight x. To do so, we consider two copies of $\mathcal{T}$ (the transition system of $\mathcal{A}$) such that: (1.i) the states of the first copy are all accepting, (1.ii) firing any transition from the first copy leads to the second copy, (2.i) the states of the second copy are all non-initial and non-accepting, (2.ii) firing a transition from the second copy leads to the first copy if and only if its weight is x. Essentially, $\mathcal{B}_1$ visits an accepting state only when a transition weighted x is fired.

Then, from $\mathcal{B}_1$, we construct the Büchi automaton $\mathcal{B}_2$ that captures the runs of value exactly x. To do so, we consider three copies of $\mathcal{B}_1$ such that: (1.i) the states of the first copy are all non-accepting, (1.ii) firing any transition from the first copy leads nondeterministically to the second or the third copy, (2.i) the states of the second copy are all non-initial and non-accepting, (2.ii) firing a transition from the second copy leads to the first copy if and only if its weight is respectively larger or lower than x when $f = \mathsf{LimSup}$ or $f = \mathsf{LimInf}$ (otherwise it stays in the second copy), (3.i) the states of the third copy are all accepting and non-initial, (3.ii) all transitions weighted more than x are removed from the third copy, and the others are unchanged. Essentially, $\mathcal{B}_2$ reaches the copy with accepting states (the third one) only once no more transitions dismissing the value x can be fired.

Observe that there is a bijection between the accepting runs of $\mathcal{B}_2$ and the runs of value x in $\mathcal{A}$. It follows from the fact that $\mathcal{B}_1$ jumps between the copies of $\mathcal{A}$ deterministically and the nondeterministic choices of $\mathcal{B}_2$ in its first copy are unambiguous (in the following sense for $f = \mathsf{LimSup}$: jumping to the second copy and not seeing a weight larger than x in $\mathcal{A}$ and jumping to the third copy and seeing a weight larger than x both result in a rejecting run). Note that this does not hold for rejecting runs due to the nondeterminism of $\mathcal{B}_2$. ◀

▶ **Lemma 7.3.** Consider a QWA $\mathcal{A} = (g, f, \mathcal{T})$ with n states, $f \in \{\mathsf{LimInf}, \mathsf{LimSup}\}$, $g \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, and initial state q_I. Let $\sim = \leq$ if $f = \mathsf{LimSup}$ and $\sim = \geq$ if $f = \mathsf{LimInf}$. For every $x \in \mathbb{Q}$ and every lasso word $w \in \Sigma^\omega$, we have that w admits infinitely many runs of value x in $\mathcal{A}$ iff $\mathcal{A}$ complies with the pattern in Figure 2b parameterized by x, n, $\sim$, and w.

Proof. We first prove the backward implication. Given a weight x and a lasso word $w \in \Sigma^\omega$, we assume that $\mathcal{A}$ complies with the pattern given in Figure 2b exhibiting four runs of the form $\rho_1 : q_I \xrightarrow{u_1} q_1$, $\ell_1 : q_1 \xrightarrow{v_1 a v_2} q_1$, $\rho_2 : q_1 \xrightarrow{u_2} q_2$ and $\ell_2 : q_2 \xrightarrow{v_1 a v_2} q_2$ where $q_1 \neq q_2$, $w = u(v_1 a v_2)^\omega$, and $(u_2)^m = v_1 a v_2$ for some $m \in \{1, \dots, 4n^4\}$. For all $i \in \mathbb{N}$, we define the run $\pi_i = \rho_1(\ell_1)^i \rho_2(\ell_2)^\omega$ in $\mathcal{A}$. Since $(u_2)^m = v_1 a v_2$, the run π_i is over w and achieves the value x thanks to the definition of $\sim$. As a direct consequence of $q_1 \neq q_2$, we have that $\pi_i \neq \pi_j$ for all $i \neq j$. In particular, w have infinitely many runs of value x in $\mathcal{A}$.

Next, we prove the forward implication. Given a weight x and a lasso word $w \in \Sigma^\omega$, we assume that w has infinitely many runs of value x in $\mathcal{A}$. Let $\tilde{u} \in \Sigma^*$ and $\tilde{v} \in \Sigma^* \setminus \{\varepsilon\}$ be such that $w = \tilde{u}(\tilde{v})^\omega$.

In the first step, we construct a Büchi automaton $\mathcal{B}$ that captures the runs over w of value x in $\mathcal{A}$. Thanks to Lemma A.1, we construct the Büchi automaton $\mathcal{B}'$ such that (i) $L(\mathcal{B}') = \{w \in \Sigma^\omega \mid \mathcal{A}(w) = x\}$ and (ii) for every word w, the automaton $\mathcal{B}'$ admits infinitely many accepting runs over w iff $\mathcal{A}$ admits infinitely many runs of value x over w. Finally, from $\mathcal{B}'$, we construction the Büchi automaton $\mathcal{B}$ that captures the runs over w of value x in $\mathcal{A}$. To do so, we consider a deterministic Büchi automaton $\mathcal{B}''$ accepting only w and we construct $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{B}') \cap L(\mathcal{B}'')$. Observe that there is a bijection between the accepting runs of $\mathcal{B}$ and the runs over w of value x in $\mathcal{A}$. This does not hold for non-accepting runs due to the nondeterminism of $\mathcal{B}'$.

The second step consists of lifting runs to words and leveraging Lemma 7.1 to get the desired pattern. We thus construct a Büchi automaton $\mathcal{C}$ from $\mathcal{B}$ by re-labeling each transition $p_1 \xrightarrow{\sigma} p_2$ by $p_1 \xrightarrow{(p_1,\sigma,p_2)} p_2$. Since w has infinitely many runs in $\mathcal{A}$ and $w \in L(\mathcal{B})$, it has infinitely many accepting runs in $\mathcal{C}$. By Lemma 7.1, some accepting state of $\mathcal{C}$ allows $\mathcal{A}$ to comply with the pattern given in Figure 2a. Consequently there exist $u_1, u_2, u_3, v \in \Sigma^+$ such that $|u_2| = |v|$, $u_2 \neq v$, and $m|v| = |u_2u_3|$ for some $m \in \{1, \dots, 4n^4\}$, together with four runs of the form $\rho_1 : q_I \xrightarrow{u_1} q_1$, $\ell_1 : q_1 \xrightarrow{u_2u_3} q_1$, $\rho_2 = q_1 \xrightarrow{v} q_2$, $\ell_2 : q_2 \xrightarrow{u_2u_3} q_2$ in $\mathcal{C}$. Observe that u_1, u_2, u_3, v are not over Σ^* but over the alphabet consisting of the transitions of $\mathcal{B}$, called Γ here after. For all finite words $w_0 \in \Gamma^*$ we denote $\Sigma(w_0) \in \Sigma^*$ the projection of all letters of Γ to Σ, e.g., $\Sigma((p_1, \sigma_1, p_2)(p_2, \sigma_2, p_3)) = \sigma_1\sigma_2$. We established that $\mathcal{C}$ admits infinitely many ultimately periodic words of Γ^ω visiting q_2 infinitely often, and thus $\mathcal{B}$ admits infinitely many ultimately periodic runs (over its singleton language $\{w\}$). In particular, the set $\{u_1(u_2u_3)^i v(u_2u_3)^\omega \mid i \in \mathbb{N}\}$ of ultimately periodic words with a run visiting q_2 infinitely often is infinite. However, $\{\Sigma(u_1(u_2u_3)^i v)(\Sigma(u_2u_3))^\omega \mid i \in \mathbb{N}\} = \{w\}$. It is worth emphasizing that despite $u_2 \neq v$ we have $\Sigma(u_2) = \Sigma(v)$ because $L(\mathcal{B}) = \{w\}$.

In the third step, we prove how to ensure $q_1 \neq q_2$. If in the above construction we obtain $q_1 = q_2$, called q here after, then we can construct $u_1', u_2', u_2'', u_3', u_3'', v'$ to get the desired property as follows. We identified $\rho_1 : q_I \xrightarrow{u_1} q$, $\rho_2 : q \xrightarrow{v} q$ and $\ell : q \xrightarrow{u_2u_3} q$ in $\mathcal{C}$. We recall that Lemma 7.1 ensures $|u_2| = |v|$ and $u_2 \neq v$. Since $\Sigma(u_2) = \Sigma(v)$, the mismatch is caused by the states of some letter of Γ. Thus, there are two distinct states $q_1' \neq q_2'$ of $\mathcal{C}$ such that $q \xrightarrow{y_1} q_1'$, $q_1' \xrightarrow{y_2} q$, $q \xrightarrow{z_1} q_2'$, $q_2' \xrightarrow{z_2} q$ in $\mathcal{C}$, where $v = y_1y_2$, $u_2 = z_1z_2$, $|y_1| = |z_1|$, and $|y_2| = |z_2|$. Let $u_1' = u_1y_1$, $u_2' = y_2u_2u_3$, $u_2'' = z_2u_3v$, $u_3' = y_1$, $u_3'' = z_1$, and $v' = y_2vz_1$ together with the four runs of the form $\rho_1' : q_I \xrightarrow{u_1'} q_1'$, $\ell_1' : q_1' \xrightarrow{u_2'u_3'} q_1'$, $\rho_2' : q_1' \xrightarrow{v'} q_2'$, $\ell_2' : q_2' \xrightarrow{u_2''u_3''} q_2'$. For all $i \in \mathbb{N}$, we define the run $\pi_i = \rho_1'(\ell_1')^i\rho_2'(\ell_2')^\omega$ over w in $\mathcal{C}$. Observe that $\pi_i = \rho_1(\rho_2\ell)^i\rho_2\rho_2(\ell\rho_2)^\omega$ because $q_1 = q_2 = q$. In particular, the set $\{u_1'(u_2'u_3')^i v'(u_2''u_3'')^\omega \mid i \in \mathbb{N}\}$ of ultimately periodic words with a run visiting q_2' infinitely often is infinite. We now prove that new pattern complies with Figure 2b. Since $\Sigma(u_2) = \Sigma(v)$, $|y_1| = |z_1|$, and $|y_2| = |z_2|$, we have that $\Sigma(y_1) = \Sigma(z_1)$ and $\Sigma(y_2) = \Sigma(z_2)$. Hence, $\Sigma(u_2'u_3') = \Sigma(u_2''u_3'')$. Additionally, $\Sigma(v)^m = \Sigma(u_2'u_3')$ for some $m \in \{1, \dots, 4n^4\}$ comes as a consequence of $m|v| = |u_2u_3|$ for some $m \in \{1, \dots, 4n^4\}$.

Finally, we decompose u_2u_3 into v_1av_2. We identified $\rho_1 : q_I \xrightarrow{u_1} q_1$, $\ell_1 : q_1 \xrightarrow{u_2u_3} q_1$, $\rho_2 : q_1 \xrightarrow{v} q_2$, and $\ell_2 : q_2 \xrightarrow{u_2u_3} q_2$ where $q_1 \neq q_2$ in $\mathcal{B}$. Recall that there is a bijection between the accepting runs of $\mathcal{B}$ and the runs over w of value x in $\mathcal{A}$. By construction of $\mathcal{B}$, the accepting run $u_1v(u_2u_3)^\omega$ of $\mathcal{B}$ (which is a word accepted by $\mathcal{C}$) corresponds to a run over w of value x in $\mathcal{A}$. Therefore, the loop $\ell_2 : q_2 \xrightarrow{u_2u_3} q_2$ corresponds to a loop visiting the weight x and other weights of value at most (resp. at least) x when $f = \mathsf{LimSup}$ (resp. $f = \mathsf{LimInf}$). To conclude, there exist $v_1, v_2 \in \Sigma^*$ and $a \in \Sigma$ such that $u_2u_3 = v_1av_2$ together with the runs $q_2 \xrightarrow{v_1:\sim x} q_3$, $q_4 \xrightarrow{v_2:\sim x} q_2$ and the transition $q_3 \xrightarrow{a:x} q_4$. ◀

▶ **Lemma 7.7.** Let $\mathbb{A} = (h, (g, f, \mathcal{T}))$ be a QLA with $f \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$, $g = \mathsf{LimSup}$ (resp. $g = \mathsf{LimInf}$), and h any language aggregator function. We can construct in PSpace a QLA $\mathbb{B} = (h, (g', f', \mathcal{T}'))$ with $f' = \mathsf{LimSup}$ and $g' = \mathsf{Sup}$ (resp. $f' = \mathsf{LimInf}$ and $g' = \mathsf{Inf}$) such that $\mathbb{A}(S) = \mathbb{B}(S)$ for all $S \subseteq \Sigma^\omega$.

Proof. Consider a language automaton $\mathbb{A} = (h, \mathcal{A})$ where $\mathcal{A} = (g, f, \mathcal{T})$ is as in the statement. We show the case $g = \mathsf{LimSup}$, as the case $g = \mathsf{LimInf}$ can be solved by duality (Proposition 2.2). Thanks to Proposition 7.5, we can assume that $f \in \{\mathsf{LimInf}, \mathsf{LimSup}\}$. The proof describes the construction of a QWA $\mathcal{B} = (\mathsf{Sup}, f, \mathcal{T}')$ such that $\mathcal{A}(w) = \mathcal{B}(w)$ for every ultimately periodic word $w \in \Sigma^\omega$, and argue why this equivalence generalizes to all words.

Recall that, by definition (when $g = \mathsf{LimSup}$), $\bot_{\mathcal{A}}$ is the smallest weight x in $\mathcal{T}$ such that there is a word w with infinitely many runs of value x in $\mathcal{T}$ and at most finitely many runs of value larger than x in $\mathcal{T}$. Therefore, we can compute $\bot_{\mathcal{A}}$ as follows: Take a weight x in $\mathcal{T}$ and construct in PTime a Büchi automaton $\mathcal{D}_x$ from the transition system $\mathcal{T}$, recognizing exactly the set of words with some run of value x in $\mathcal{T}$. Then, using [15,25], we can decide in

PTime if $\mathcal{D}_x$ has an infinite degree of ambiguity, i.e., some word in its language has infinitely many accepting runs (namely, runs of value x in $\mathcal{T}$). Let Y be the set of weights x such that $\mathcal{D}_x$ has an infinite degree of ambiguity. Then, $\bot_{\mathcal{A}}$ is the largest $y \in Y$ such that the automaton $\mathcal{D} = (\mathsf{Sup}, f, \mathcal{T})$ is $\geq$-universal for y, which can be computed in PSpace. We can compute $\bot_{\mathcal{A}}$ similarly when $g = \mathsf{LimInf}$. Now, let X be the set of all weights in $\mathcal{T}$. By Corollary 7.4, for all $x \in X$, we can construct in PTime a Büchi (if $f = \mathsf{LimSup}$) or a co-Büchi (if $f = \mathsf{LimInf}$) automaton $\mathcal{C}_x$ recognizing all ultimately periodic words admitting infinitely many runs of value x in $\mathcal{A}$. From $\mathcal{C}_x$ we construct in PTime the QWA $\mathcal{A}'_x = (\mathsf{Sup}, f, \mathcal{T}_x)$ such that $\mathcal{A}'_x(w) = x$ when $w \in L(\mathcal{C}_x)$ and $\mathcal{A}'_x(w) = \bot_{\mathcal{A}'}$ when $w \notin L(\mathcal{C}_x)$. The construction consists in changing the transition weight 1 to x and 0 to $\bot_{\mathcal{A}}$. The QWA $\mathcal{B} = (\mathsf{Sup}, f, \mathcal{T}')$ is defined by $\mathcal{B} = \max_{x\in X} \mathcal{A}'_x$, effectively computable in PTime [11].

For every ultimately periodic word $w \in \Sigma^\omega$, the value $\mathcal{A}(w)$ corresponds to the maximal value $x \in X$ such that there exist infinitely many runs of value x for w in $\mathcal{T}$. Let $X_w \subseteq X$ denote the set of values x for which w admits infinitely many runs in $\mathcal{T}$; equivalently, $x \in X_w$ iff $w \in L(C_x)$. By construction, we have $\mathcal{A}'_x(w) = x$ for each $x \in X_w$ and $\mathcal{A}'_x(w) = \bot_{\mathcal{A}'}$ for each $x \in X \setminus X_w$. If $X_w \neq \emptyset$, it follows that $\mathcal{A}(w) = \max X' = \max_{x\in X'} \mathcal{A}'_x(w) = \mathcal{B}(w)$. Otherwise, if $X_w = \emptyset$, we have $\mathcal{A}(w) = \bot_{\mathcal{A}'} = \mathcal{B}(w)$.

Next, we argue that if $\mathcal{A}(w) = \mathcal{B}(w)$ for every ultimately periodic word $w \in \Sigma^\omega$, then $\mathcal{A}(w) = \mathcal{B}(w)$ for every word $w \in \Sigma^\omega$. We prove the contrapositive. Suppose the existence of a word $\hat{w}$ such that $\mathcal{A}(\hat{w}) \neq \mathcal{B}(\hat{w})$. Clearly, we have $\mathcal{A}(\hat{w}) \in X$ and $\mathcal{B}(\hat{w}) \in X$. Using Lemma A.1, we construct a Büchi automaton $\mathcal{A}'_{\hat{w}}$ such that (i) $L(\mathcal{A}'_{\hat{w}}) = \{w \in \Sigma^\omega \mid \mathcal{A}'(w) = \mathcal{A}'(\hat{w})\}$ and (ii) for every word w, the automaton $\mathcal{A}'_{\hat{w}}$ admits infinitely many accepting runs over w iff $\mathcal{A}'$ admits infinitely many runs of value $\mathcal{A}'(\hat{w})$ over w. Similarly, we construct a Büchi automaton $\mathcal{B}_{\hat{w}}$ such that (i) $L(\mathcal{B}_{\hat{w}}) = \{w \in \Sigma^\omega \mid \mathcal{B}(w) = \mathcal{B}(\hat{w})\}$ and (ii) for every word w, the automaton $\mathcal{B}_{\hat{w}}$ admits infinitely many accepting runs over w iff $\mathcal{B}$ admits infinitely many runs of value $\mathcal{B}(\hat{w})$ over w. Thanks to [25], we can construct from $\mathcal{B}_{\hat{w}}$ an unambiguous Büchi automaton $\mathcal{B}'_{\hat{w}}$ recognizing the same language. The cross product between $\mathcal{A}'_{\hat{w}}$ and $\mathcal{B}'_{\hat{w}}$ yields a Büchi automaton $\mathcal{C}$ such that (i) $L(C) = L(\mathcal{B}_{\hat{w}}) \cap L(\mathcal{A}'_{\hat{w}})$ and (ii) for every word w, the automaton $\mathcal{C}$ admits infinitely many accepting runs over w iff $\mathcal{A}'$ admits infinitely many runs of value $\mathcal{A}'(\hat{w})$ over w. Observe that $\hat{w} \in L(\mathcal{C})$ and for all $w \in L(\mathcal{C})$ we have $\mathcal{A}'(w) = \mathcal{A}'(\hat{w}) \neq \mathcal{B}(\hat{w}) = \mathcal{B}(w)$. By [25, 30], we can decide whether $\mathcal{C}$ admits infinitely many accepting runs on some word. Furthermore, it follows from [25, 30] that if $\mathcal{C}$ admits infinitely many accepting runs on a word, then it recognizes an ultimately periodic word $\tilde{w}$ with infinitely many accepting runs. In this case, $\mathcal{A}'$ admits infinitely many runs of value $\mathcal{A}'(\hat{w})$ over $\tilde{w}$, implying that $\mathcal{A}(\tilde{w}) = \mathcal{A}'(\tilde{w}) = \mathcal{A}'(\hat{w})$. Otherwise, we have $\mathcal{A}(\hat{w}) = \bot_{\mathcal{A}'} = \mathcal{A}'(\hat{w})$, and since $L(\mathcal{C}) \neq \emptyset$, there exists an ultimately periodic word $\tilde{w}$ for which $\mathcal{A}'(\tilde{w}) = \bot_{\mathcal{A}'}$. In either scenario, since $\tilde{w} \in L(\mathcal{C})$, we have $\mathcal{B}(\tilde{w}) = \mathcal{B}(\hat{w})$. Therefore, we conclude that there is an ultimately periodic word $\tilde{w}$ satisfying $\mathcal{A}'(\tilde{w}) \neq \mathcal{B}(\tilde{w})$. ◀