Discounting in LTL

Shaull Almagor* Udi Boker† Orna Kupferman*

May 3, 2014

Abstract

In recent years, there is growing need and interest in formalizing and reasoning about the quality of software and hardware systems. As opposed to traditional verification, where one handles the question of whether a system satisfies, or not, a given specification, reasoning about quality addresses the question of *how well* the system satisfies the specification. One direction in this effort is to refine the "eventually" operators of temporal logic to *discounting operators*: the satisfaction value of a specification is a value in $[0, 1]$, where the longer it takes to fulfill eventuality requirements, the smaller the satisfaction value is.

In this paper we introduce an augmentation by discounting of Linear Temporal Logic (LTL), and study it, as well as its combination with propositional quality operators. We show that one can augment LTL with an arbitrary set of discounting functions, while preserving the decidability of the model-checking problem. Further augmenting the logic with unary propositional quality operators preserves decidability, whereas adding an average-operator makes some problems undecidable. We also discuss the complexity of the problem, as well as various extensions.

1 Introduction

One of the main obstacles to the development of complex hardware and software systems lies in ensuring their correctness. A successful paradigm addressing this obstacle is *temporal-logic model checking* – given a mathematical model of the system and a temporal-logic formula that specifies a desired behavior of it, decide whether the model satisfies the formula [5]. Correctness is Boolean: a system can either satisfy its specification or not satisfy it. The richness of today's systems, however, justifies specification formalisms that are *multi-valued*. The multi-valued setting arises directly in systems with quantitative aspects (multi-valued / probabilistic / fuzzy) [9, 10, 11, 16, 23], but is applied also with respect to Boolean systems, where it origins from the semantics of the specification formalism itself [1, 7].

When considering the *quality* of a system, satisfying a specification should no longer be a yes/no matter. Different ways of satisfying a specification should induce different levels of quality, which should be reflected in the output of the verification procedure. Consider for example the specification $\mathsf{G}(request \rightarrow \mathsf{F}(response\_grant \vee response\_deny))$ ("every request is eventually responded, with either a grant or a denial"). There should be a difference between a computation that satisfies it with responses generated soon after requests and one

*The Hebrew University, Jerusalem, Israel.

†The Interdisciplinary Center, Herzliya, Israel.

that satisfies it with long waits. Moreover, there may be a difference between grant and deny responses, or cases in which no request is issued. The issue of generating high-quality hardware and software systems attracts a lot of attention [13, 26]. Quality, however, is traditionally viewed as an art, or as an amorphic ideal. In [1], we introduced an approach for formalizing quality. Using it, a user can specify quality formally, according to the importance he gives to components such as security, maintainability, runtime, and more, and then can formally reason about the quality of software.

As the example above demonstrates, we can distinguish between two aspects of the quality of satisfaction. The first, to which we refer as "temporal quality" concerns the waiting time to satisfaction of eventualities. The second, to which we refer as "propositional quality" concerns prioritizing related components of the specification. Propositional quality was studied in [1]. In this paper we study temporal quality as well as the combinations of both aspects. One may try to reduce temporal quality to propositional quality by a repeated use of the X ("next") operator or by a use of bounded (prompt) eventualities [2, 3]. Both approaches, however, partitions the future into finitely many zones and are limited: correctness of LTL is Boolean, and thus has inherent dichotomy between satisfaction and dissatisfaction. On the other hand, the distinction between "near" and "far" is not dichotomous. This suggests that in order to formalize temporal quality, one must extend LTL to an unbounded setting. Realizing this, researchers have suggested to augment temporal logics with *future discounting* [8]. In the discounted setting, the satisfaction value of specifications is a numerical value, and it depends, according to some discounting function, on the time waited for eventualities to get satisfied.

In this paper we add discounting to Linear Temporal Logic (LTL), and study it, as well as its combination with propositional quality operators. We introduce $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ – an augmentation by discounting of LTL. The logic $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ is actually a family of logics, each parameterized by a set $\mathcal{D}$ of discounting functions – strictly decreasing functions from $\mathbb{N}$ to $[0, 1]$ that tend to 0 (e.g., linear decaying, exponential decaying, etc.). $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ includes a discounting-"until" (U_η) operator, parameterized by a function $\eta \in \mathcal{D}$. We solve the model-checking threshold problem for $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$: given a Kripke structure $\mathcal{K}$, an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ and a threshold $t \in [0, 1]$, the algorithm decides whether the satisfaction value of φ in $\mathcal{K}$ is at least t.

In the Boolean setting, the automata-theoretic approach has proven to be very useful in reasoning about LTL specifications. The approach is based on translating LTL formulas to nondeterministic Büchi automata on infinite words [28]. Applying this approach to the discounted setting, which gives rise to infinitely many satisfaction values, poses a big algorithmic challenge: model-checking algorithms, and in particular those that follow the automata-theoretic approach, are based on an exhaustive search, which cannot be simply applied when the domain becomes infinite. A natural relevant extension to the automata-theoretic approach is to translate formulas to *weighted automata* [22]. Unfortunately, these extensively-studied models are complicated and many problems become undecidable for them [15]. We show that for threshold problems, we can translate $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas into (Boolean) nondeterministic Büchi automata, with the property that the automaton accepts a lasso computation iff the formula attains a value above the threshold on that computation. Our algorithm relies on the fact that the language of an automaton is non-empty iff there is a lasso witness for the non-emptiness. We cope with the infinitely many possible satisfaction values by using the discounting behavior of the eventualities and the given threshold in order to partition the state space into a finite number of classes. The complexity of our algorithm

depends on the discounting functions used in the formula. We show that for standard discounting functions, such as exponential decaying, the problem is PSPACE-complete – not more complex than standard LTL. The fact our algorithm uses Boolean automata also enables us to suggest a solution for threshold satisfiability, and to give a partial solution to threshold synthesis. In addition, it allows to adapt the heuristics and tools that exist for Boolean automata.

Before we continue to describe our contribution, let us review existing work on discounting. The notion of discounting has been studied in several fields, such as economy, game-theory, and Markov decision processes [25]. In the area of formal verification, it was suggested in [8] to augment the μ-calculus with discounting operators. The discounting suggested there is exponential; that is, with each iteration, the satisfaction value of the formula decreases by a multiplicative factor in $(0, 1]$. Algorithmically, [8] shows how to evaluate discounted μ-calculus formulas with arbitrary precision. Formulas of LTL can be translated to the μ-calculus, thus [8] can be used in order to approximately model-check discounted-LTL formulas. However, the translation from LTL to the μ-calculus involves an exponential blowup [6] (and is complicated), making this approach inefficient. Moreover, our approach allows for arbitrary discounting functions, and the algorithm returns an exact solution to the threshold model-checking problem, which is more difficult than the approximation problem.

Closer to our work is [7], where CTL is augmented with discounting and weighted-average operators. The motivation in [7] is to introduce a logic whose semantics is not too sensitive to small perturbations in the model. Accordingly, formulas are evaluated on weighted-systems or on Markov-chains. Adding discounting and weighted-average operators to CTL preserves its appealing complexity, and the model-checking problem for the augmented logic can be solved in polynomial time. As is the case in the traditional, Boolean, semantics, the expressive power of discounted CTL is limited. The fact the same combination, of discounting and weighted-average operators, leads to undecidability in the context of LTL witnesses the technical challenges of the $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ setting.

Perhaps closest to our approach is [19], where a version of discounted-LTL was introduced. Semantically, there are two main differences between the logics. The first is that [19] uses discounted sum, while we interpret discounting without accumulation, and the second is that the discounting there replaces the standard temporal operators, so all eventualities are discounted. As discounting functions tend to 0, this strictly restricts the expressive power of the logic, and one cannot specify traditional eventualities in it. On the positive side, it enables a clean algebraic characterization of the semantics, and indeed the contribution in [19] is a comprehensive study of the mathematical properties of the logic. Yet, [19] does not study algorithmic questions about to the logic. We, on the other hand, focus on the algorithmic properties of the logic, and specifically on the model-checking problem.

Let us now return to our contribution. After introducing $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ and studying its model-checking problem, we augment $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ with propositional quality operators. Beyond the operators $\min$, $\max$, and $\neg$, which are already present, two basic propositional quality operators are the multiplication of an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula by a constant in $[0, 1]$, and the averaging between the satisfaction values of two $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas [1]. We show that while the first extension does not increase the expressive power of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ or its complexity, the latter causes the validity and model-checking problems to become undecidable. In fact, things become undecidable even if we allow averaging in combination with a single discounting function. Recall that this is in contrast with the extension of

discounted CTL with an average operator, where the complexity of the model-checking problem stays polynomial [7].

We consider additional extensions of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$. First, we study a variant of the discounting-eventually operators in which we allow the discounting to tend to arbitrary values in $[0,1]$ (rather than to 0). This captures the intuition that we are not always pessimistic about the future, but can be, for example, ambivalent about it, by tending to $\frac{1}{2}$. We show that all our results hold under this extension. Second, we add to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ *past* operators and their discounting versions (specifically, we allow a discounting-"since" operator, and its dual). In the traditional semantics, past operators enable clean specifications of many interesting properties, make the logic exponentially more succinct, and can still be handled within the same complexity bounds [17, 18]. We show that the same holds for the discounted setting. Finally, we show how $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ and algorithms for it can be used also for reasoning about weighted systems.

Due to lack of space, the full proofs appear in the appendix.

2 The Logic $\mathrm{LTL}^{\mathbf{disc}}[\mathcal{D}]$

The linear temporal logic $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ generalizes LTL by adding discounting temporal operators. The logic is actually a family of logics, each parameterized by a set $\mathcal{D}$ of discounting functions.

Let $\mathbb{N} = \{0, 1, ...\}$. A function $\eta : \mathbb{N} \to [0,1]$ is a *discounting function* if $\lim_{i\to\infty} \eta(i) = 0$, and η is strictly monotonic-decreasing. Examples for natural discounting functions are $\eta(i) = \lambda^i$, for some $\lambda \in (0,1)$, and $\eta(i) = \frac{1}{i+1}$.

Given a set of discounting functions $\mathcal{D}$, we define the logic $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ as follows. The syntax of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ adds to LTL the operator $\varphi \mathsf{U}_\eta \psi$ (discounting-Until), for every function $\eta \in \mathcal{D}$. Thus, the syntax is given by the following grammar, where p ranges over the set AP of atomic propositions and $\eta \in \mathcal{D}$.

$$\varphi := \mathtt{True} \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{U}_\eta \varphi.$$

The semantics of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ is defined with respect to a *computation* $\pi = \pi^0, \pi^1, \ldots \in (2^{AP})^\omega$. Given a computation π and an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ, the truth value of φ in π is a value in $[0,1]$, denoted $[\![\pi, \varphi]\!]$. The value is defined by induction on the structure of φ as follows, where $\pi^i = \pi_i, \pi_{i+1}, \ldots$.

- $[\![\pi, \mathtt{True}]\!] = 1$.
- $[\![\pi, \varphi \vee \psi]\!] = \max\{[\![\pi, \varphi]\!], [\![\pi, \psi]\!]\}$.
- $[\![\pi, p]\!] = \begin{cases} 1 & \text{if } p \in \pi_0, \\ 0 & \text{if } p \notin \pi_0. \end{cases}$
- $[\![\pi, \neg\varphi]\!] = 1 - [\![\pi, \varphi]\!]$.
- $[\![\pi, \mathsf{X}\varphi]\!] = [\![\pi^1, \varphi]\!]$.
- $[\![\pi, \varphi \mathsf{U} \psi]\!] = \sup_{i\geq 0}\{\min\{[\![\pi^i, \psi]\!], \min_{0\leq j<i}\{[\![\pi^j, \varphi]\!]\}\}\}$.
- $[\![\pi, \varphi \mathsf{U}_\eta \psi]\!] = \sup_{i\geq 0}\{\min\{\eta(i)[\![\pi^i, \psi]\!], \min_{0\leq j<i}\{\eta(j)[\![\pi^j, \varphi]\!]\}\}\}$.

The intuition is that events that happen in the future have a lower influence, and the rate by which this influence decreases depends on the function η. [1] For example, the satisfaction value of a formula $\varphi \mathsf{U}_\eta \psi$ in a computation π depends on the best (supremum) value that ψ can get along the entire computation, while considering the discounted satisfaction of ψ at a position i, as a result of multiplying it by $\eta(i)$, and the same for the value of φ in the prefix leading to the i-th position.

We add the standard abbreviations $\mathsf{F}\varphi \equiv \mathtt{True}\mathsf{U}\varphi$, and $\mathsf{G}\varphi = \neg\mathsf{F}\neg\varphi$, as well as their quantitative counterparts: $\mathsf{F}_\eta\varphi \equiv \mathtt{True}\mathsf{U}_\eta\varphi$, and $\mathsf{G}_\eta\varphi = \neg\mathsf{F}_\eta\neg\varphi$. We denote by $|\varphi|$ the number of subformulas of φ.

A computation of the form $\pi = u \cdot v^\omega$, for $u, v \in (2^{AP})^*$, with $v \neq \epsilon$, is called a *lasso computation*. We observe that since a specific lasso computation has only finitely many distinct suffixes, the $\inf$ and $\sup$ in the semantics of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ can be replaced with $\min$ and $\max$, respectively, when applied to lasso computations.

The semantics is extended to *Kripke structures* by taking the path that admits the lowest satisfaction value. Formally, for a Kripke structure $\mathcal{K}$ and an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ we have that $[\![\mathcal{K}, \varphi]\!] = \inf\{[\![\pi, \varphi]\!] : \pi \text{ is a computation of } \mathcal{K}\}$.

Example 2.1 Consider a lossy-disk: every moment in time there is a chance that some bit would flip its value. Fixing flips is done by a global error-correcting procedure. This procedure manipulates the entire content of the disk, such that initially it causes more errors in the disk, but the longer it runs, the more bits it fixes.

Let $init$ and $terminate$ be atomic propositions indicating when the error-correcting procedure is initiated and terminated, respectively. The quality of the disk (that is, a measure of the amount of correct bits) can be specified by the formula $\varphi = \mathsf{GF}_\eta(init \wedge \neg\mathsf{F}_\mu terminate)$ for some appropriate discounting functions η and μ. Intuitively, φ gets a higher satisfaction value the shorter the waiting time is between initiations of the error-correcting procedure, and the longer the procedure runs (that is, not terminated) in between these initiations. Note that the "worst case" nature of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ fits here. For instance, running the procedure for a very short time, even once, will cause many errors. □

3 $\mathrm{LTL}^{\mathbf{disc}}[\mathcal{D}]$ Model Checking

In the Boolean setting, the model-checking problem asks, given an LTL formula φ and a Kripke structure $\mathcal{K}$, whether $[\![\mathcal{K}, \varphi]\!] = \mathtt{True}$. In the quantitative setting, the model-checking problem is to compute $[\![\mathcal{K}, \varphi]\!]$, where φ is now an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula. A simpler version of this problem is the threshold model-checking problem: given φ, $\mathcal{K}$, and a threshold $v \in [0, 1]$, decide whether $[\![\mathcal{K}, \varphi]\!] \geq v$. In this section we show how we can solve the latter.

Our solution uses the automata-theoretic approach, and consists of the following steps. We start by translating φ and v to an alternating weak automaton $\mathcal{A}_{\varphi,v}$ such that $L(\mathcal{A}_{\varphi,v}) \neq \emptyset$ iff there exists a computation π such that $[\![\pi, \varphi]\!] > v$. The challenge here is that φ has infinitely many satisfaction values, naively implying an infinite-state automaton. We show that using the threshold and the discounting behavior of the eventualities, we can restrict attention to a finite resolution of satisfaction values, enabling the construction of a finite

[1] Observe that in our semantics the satisfaction value of future events tends to 0. One may think of scenarios where future events are discounted towards another value in $[0, 1]$ (e.g. discounting towards $\frac{1}{2}$ as ambivalence regarding the future). We address this in Section 5.3.

automaton. Complexity-wise, the size of $\mathcal{A}_{\varphi,v}$ depends on the functions in $\mathcal{D}$. In Section 3.3, we analyze the complexity for the case of exponential-discounting functions.

The second step is to construct a nondeterministic Büchi automaton $\mathcal{B}$ that is equivalent to $\mathcal{A}_{\varphi,v}$. In general, alternation removal might involve an exponential blowup in the state space [21]. We show, by a careful analysis of $\mathcal{A}_{\varphi,v}$, that we can remove its alternation while only having a polynomial state blowup.

We complete the model-checking procedure by composing the nondeterministic Büchi automaton $\mathcal{B}$ with the Kripke structure $\mathcal{K}$, as done in the traditional, automata-based, model-checking procedure.

The complexity of model-checking an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula depends on the discounting functions in $\mathcal{D}$. Intuitively, the faster the discounting tends to 0, the less states there will be. For exponential-discounting, we show that the complexity is NLOGSPACE in the system (the Kripke structure) and PSPACE in the specification (the $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula and the threshold), staying in the same complexity classes of standard LTL model-checking.

We conclude the section by showing how to use the generated nondeterministic Büchi automaton for addressing threshold satisfiability and synthesis.

3.1 Alternating Weak Automata

For a given set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas `True` and `False`. For $Y \subseteq X$, we say that Y *satisfies* a formula $\theta \in \mathcal{B}^+(X)$ iff the truth assignment that assigns *true* to the members of Y and assigns *false* to the members of $X \setminus Y$ satisfies θ. An *alternating Büchi automaton on infinite words* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $q_{in} \in Q$ is an initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, and $\alpha \subseteq Q$ is a set of accepting states. We define runs of $\mathcal{A}$ by means of (possibly) infinite DAGs (directed acyclic graphs). A run of $\mathcal{A}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots \in \Sigma^\omega$ is a (possibly) infinite DAG $\mathcal{G} = \langle V, E \rangle$ satisfying the following (note that there may be several runs of $\mathcal{A}$ on w).

- $V \subseteq Q \times \mathbb{N}$ is as follows. Let $Q_l \subseteq Q$ denote all states in level l. Thus, $Q_l = \{q : \langle q, l \rangle \in V\}$. Then, $Q_0 = \{q_{in}\}$, and Q_{l+1} satisfies $\bigwedge_{q \in Q_l} \delta(q, \sigma_l)$.
- For every $l \in \mathbb{N}$, Q_l is minimal with respect to containment.
- $E \subseteq \bigcup_{l \geq 0} (Q_l \times \{l\}) \times (Q_{l+1} \times \{l+1\})$ is such that for every state $q \in Q_l$, the set $\{q' \in Q_{l+1} : E(< q, l >, < q', l+1 >)\}$ satisfies $\delta(q, \sigma_l)$.

Thus, the root of the DAG contains the initial state of the automaton, and the states associated with nodes in level $l+1$ satisfy the transitions from states corresponding to nodes in level l. The run $\mathcal{G}$ accepts the word w if all its infinite paths satisfy the acceptance condition α. Thus, in the case of Büchi automata, all the infinite paths have infinitely many nodes $\langle q, l \rangle$ such that $q \in \alpha$ (it is not hard to prove that every infinite path in $\mathcal{G}$ is part of an infinite path starting in level 0). A word w is accepted by $\mathcal{A}$ if there is a run that accepts it. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of infinite words that $\mathcal{A}$ accepts.

When the formulas in the transition function of $\mathcal{A}$ contain only disjunctions, then $\mathcal{A}$ is nondeterministic, and its runs are DAGs of width 1, where at each level there is a single node.

The alternating automaton $\mathcal{A}$ is *weak*, denoted AWA, if its state space Q can be partitioned into sets $Q_1, \ldots, Q_k$, such that the following hold: First, for every $1 \leq i \leq k$ either $Q_i \subseteq \alpha$,

in which case we say that Q_i is an accepting set, or $Q_i \cap \alpha = \emptyset$, in which case we say that Q_i is rejecting. Second, there is a partial-order $\leq$ over the sets, and for every $1 \leq i, j \leq k$, if $q \in Q_i$, $s \in Q_j$, and $s \in \delta(q, \sigma)$ for some $\sigma \in \Sigma$, then $Q_j \leq Q_i$. Thus, transitions can lead only to states that are smaller in the partial order. Consequently, each run of an AWA eventually gets trapped in a set Q_i and is accepting iff this set is accepting.

3.2 From $\mathrm{LTL}^{\mathbf{disc}}[\mathcal{D}]$ to AWA

Our model-checking algorithm is based on translating an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ to an AWA. Intuitively, the states of the AWA correspond to assertions of the form $\psi > t$ or $\psi < t$ for every subformula ψ of φ, and for certain thresholds $t \in [0, 1]$. A lasso computation is then accepted from state $\psi > t$ iff $[\![\pi, \psi]\!] > t$. The assumption about the computation being a lasso is needed only for the "only if" direction, and it does not influence the proof's generality since the language of an automaton is non-empty iff there is a lasso witness for its non-emptiness. By setting the initial state to $\varphi > v$, we are done.

Defining the appropriate transition function for the AWA follows the semantics of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ in the expected manner. A naive construction, however, yields an infinite-state automaton (even if we only expand the state space on-the-fly, as discounting formulas can take infinitely many satisfaction values). As can be seen in the proof of Theorem 3.4, the "problematic" transitions are those that involve the discounting operators. The key observation is that, given a threshold v and a computation π, when evaluating a discounted operator on π, one can restrict attention to two cases: either the satisfaction value of the formula goes below v, in which case this happens after a bounded prefix, or the satisfaction value always remains above v, in which case we can replace the discounted operator with a Boolean one. This observation allows us to expand only a finite number of states on-the-fly.

Before describing the construction of the AWA, we need the following lemma, which reduces an extreme satisfaction of an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula, meaning satisfaction with a value of either 0 or 1, to a Boolean satisfaction of an LTL formula. The proof proceeds by induction on the structure of the formulas.

Lemma 3.1 *Given an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ*, there exist* LTL *formulas* φ^+ *and* $\varphi^{<1}$ *such that* $|\varphi^+|$ *and* $|\varphi^{<1}|$ *are both* $O(|\varphi|)$ *and the following hold for every computation* π*.*

1. *If* $[\![\pi, \varphi]\!] > 0$ *then* $\pi \models \varphi^+$*, and if* $[\![\pi, \varphi]\!] < 1$ *then* $\pi \models \varphi^{<1}$*.*
2. *If* π *is a lasso, then if* $\pi \models \varphi^+$ *then* $[\![\pi, \varphi]\!] > 0$ *and if* $\pi \models \varphi^{<1}$ *then* $[\![\pi, \varphi]\!] < 1$*.*

Henceforth, given an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ, we refer to φ^+ as in Lemma 3.1.

Consider an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ. By Lemma 3.1, if there exists a computation π such that $[\![\pi, \varphi]\!] > 0$, then φ^+ is satisfiable. Conversely, since φ^+ is a Boolean LTL formula, then by [27] we know that φ^+ is satisfiable iff there exists a lasso computation π that satisfies it, in which case $[\![\pi, \varphi]\!] > 0$. We conclude with the following.

Corollary 3.2 *Consider an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ*. There exists a computation* π *such that* $[\![\pi, \varphi]\!] > 0$ *iff there exists a lasso computation* π' *such that* $[\![\pi', \varphi]\!] > 0$*, in which case* $\pi' \models \varphi^+$ *as well.*

Remark 3.3 The curious reader may wonder why we do not prove that $[\![\pi, \varphi]\!] > 0$ iff $\pi \models \varphi^+$ for every computation π. As it turns out, a translation that is valid also for

computations with no period is not always possible. For example, as is the case with the prompt-eventuality operator of [**?**], the formula $\varphi = \mathsf{G}(\mathsf{F}_\eta p)$ is such that the set of computations π with $[\![\pi, \varphi]\!] > 0$ is not ω-regular, thus one cannot hope to define an LTL formula φ^+.

We start with some definitions. For a function $f : \mathbb{N} \to [0,1]$ and for $k \in \mathbb{N}$, we define $f^{+k} : \mathbb{N} \to [0,1]$ as follows. For every $i \in \mathbb{N}$ we have that $f^{+k}(i) = f(i+k)$.

Let φ be an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula over AP. We define the *extended closure* of φ, denoted $xcl(\varphi)$, to be the set of all the formulas ψ of the following *classes*:

1. ψ is a subformula of φ.
2. ψ is a subformula of θ^+ or $\neg\theta^+$, where θ is a subformula of φ.
3. ψ is of the form $\theta_1 \mathsf{U}_{\eta^{+k}} \theta_2$ for $k \in \mathbb{N}$, where $\theta_1 \mathsf{U}_\eta \theta_2$ is a subformula of φ.

Observe that $xcl(\varphi)$ may be infinite, and that it has both $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas (from Classes 1 and 3) and LTL formulas (from Class 2).

Theorem 3.4 *Given an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ *and a threshold* $v \in [0,1]$*, there exists an* AWA $\mathcal{A}_{\varphi,v}$ *such that for every computation* π *the following hold.*

1. *If* $[\![\pi, \varphi]\!] > v$*, then* $\mathcal{A}_{\varphi,v}$ *accepts* π*.*
2. *If* $\mathcal{A}_{\varphi,v}$ *accepts* π *and* π *is a lasso computation, then* $[\![\pi, \varphi]\!] > v$*.*

Proof: We construct $\mathcal{A}_{\varphi,v} = \langle Q, 2^{AP}, Q_0, \delta, \alpha \rangle$ as follows.

The state space Q consists of two types of states. Type-1 states are assertions of the form $(\psi > t)$ or $(\psi < t)$, where $\psi \in xcl(\varphi)$ is of Class 1 or 3 and $t \in [0,1]$. Type-2 states correspond to LTL formulas of Class 2. Let S be the set of Type-1 and Type-2 states for all $\psi \in xcl(\varphi)$ and thresholds $t \in [0,1]$. Then, Q is the subset of S constructed on-the-fly according to the transition function defined below. We later show that Q is indeed finite.

The transition function $\delta : Q \times 2^{AP} \to \mathcal{B}^+(Q)$ is defined as follows. For Type-2 states, the transitions are as in the standard translation from LTL to AWA [27] (see Appendix A.2 for details). For the other states, we define the transitions as follows. Let $\sigma \in 2^{AP}$.

- $\delta((\mathtt{True} > t), \sigma) = \begin{cases} \mathtt{True} & \text{if } t < 1, \\ \mathtt{False} & \text{if } t = 1. \end{cases}$
- $\delta((\mathtt{False} > t), \sigma) = \mathtt{False}$.
- $\delta((\mathtt{True} < t), \sigma) = \mathtt{False}$.
- $\delta((\mathtt{False} < t), \sigma) = \begin{cases} \mathtt{True} & \text{if } t > 0, \\ \mathtt{False} & \text{if } t = 0. \end{cases}$
- $\delta((p > t), \sigma) = \begin{cases} \mathtt{True} & \text{if } p \in \sigma \text{ and } t < 1, \\ \mathtt{False} & \text{otherwise.} \end{cases}$
- $\delta((p < t), \sigma) = \begin{cases} \mathtt{False} & \text{if } p \in \sigma \text{ or } t = 0, \\ \mathtt{True} & \text{otherwise.} \end{cases}$
- $\delta((\psi_1 \vee \psi_2 > t), \sigma) = \delta((\psi_1 > t), \sigma) \vee \delta((\psi_2 > t), \sigma)$.

- $\delta((\psi_1 \vee \psi_2 < t), \sigma) = \delta((\psi_1 < t), \sigma) \wedge \delta((\psi_2 < t), \sigma)$.
- $\delta((\neg\psi_1 > t), \sigma) = \delta((\psi_1 < 1 - t), \sigma)$
- $\delta((\neg\psi_1 < t), \sigma) = \delta((\psi_1 > 1 - t), \sigma)$.
- $\delta((\mathsf{X}\psi_1 > t), \sigma) = (\psi_1 > t)$.
- $\delta((\mathsf{X}\psi_1 < t), \sigma) = (\psi_1 < t)$.
- $\delta((\psi_1 \mathsf{U} \psi_2 > t), \sigma) = \begin{cases} \delta((\psi_2 > t), \sigma) \vee [\delta((\psi_1 > t), \sigma) \wedge (\psi_1 \mathsf{U} \psi_2 > t)] & \text{if } 0 < t < 1, \\ \texttt{False} & \text{if } t \geq 1, \\ \delta(((\psi_1 \mathsf{U} \psi_2)^+), \sigma) & \text{if } t = 0. \end{cases}$
- $\delta((\psi_1 \mathsf{U} \psi_2 < t), \sigma) = \begin{cases} \delta((\psi_2 < t), \sigma) \wedge [\delta((\psi_1 < t), \sigma) \vee (\psi_1 \mathsf{U} \psi_2 < t)] & \text{if } 0 < t \leq 1, \\ \texttt{True} & \text{if } t > 1, \\ \texttt{False} & \text{if } t = 0. \end{cases}$
- $\delta((\psi_1 \mathsf{U}_\eta \psi_2 > t), \sigma) = \begin{cases} \delta((\psi_2 > \frac{t}{\eta(0)}), \sigma) \vee [\delta((\psi_1 > \frac{t}{\eta(0)}), \sigma) \wedge (\psi_1 \mathsf{U}_{\eta^{+1}} \psi_2 > t)] & \text{if } 0 < \frac{t}{\eta(0)} < 1, \\ \texttt{False} & \text{if } \frac{t}{\eta(0)} \geq 1, \\ \delta(((\psi_1 \mathsf{U}_\eta \psi_2)^+), \sigma) & \text{if } \frac{t}{\eta(0)} = 0 \text{ (i.e., } t = 0). \end{cases}$
- $\delta((\psi_1 \mathsf{U}_\eta \psi_2 < t), \sigma) = \begin{cases} \delta((\psi_2 < \frac{t}{\eta(0)}), \sigma) \wedge [\delta((\psi_1 < \frac{t}{\eta(0)}), \sigma) \vee (\psi_1 \mathsf{U}_{\eta^{+1}} \psi_2 < t)] & \text{if } 0 < \frac{t}{\eta(0)} \leq 1, \\ \texttt{True} & \text{if } \frac{t}{\eta(0)} > 1, \\ \texttt{False} & \text{if } \frac{t}{\eta(0)} = 0 \text{ (i.e., } t = 0). \end{cases}$

We provide some intuition for the more complex parts of the transition function: consider, for example, the transition $\delta((\psi_1 \mathsf{U}_\eta \psi_2 > t), \sigma)$. Since η is decreasing, the highest possible satisfaction value for $\psi_1 \mathsf{U}_\eta \psi_2$ is $\eta(0)$. Thus, if $\eta(0) \leq t$ (equivalently, $\frac{t}{\eta(0)} \geq 1$), then it cannot hold that $\psi_1 \mathsf{U}_\eta \psi_2 > t$, so the transition is to `False`. If $t = 0$, then we only need to ensure that the satisfaction value of $\psi_1 \mathsf{U}_\eta \psi_2$ is not 0. To do so, we require that $(\psi_1 \mathsf{U}_\eta \psi_2)^+$ is satisfied. By Corollary 3.2, this is equivalent to the satisfiability of the former. So the transition is identical to that of the state $(\psi_1 \mathsf{U}_\eta \psi_2)^+$. Finally, if $0 < t < \eta(0)$, then (slightly abusing notation) the assertion $\psi_1 \mathsf{U}_\eta \psi_2 > t$ is true if either $\eta(0)\psi_2 > t$ is true, or both $\eta(0)\psi_1 > t$ and $\psi_1 \mathsf{U}_{\eta+1} \psi_2 > t$ are true.

The initial state of $\mathcal{A}_{\varphi,v}$ is $(\varphi > v)$. The accepting states are these of the form $(\psi_1 \mathsf{U} \psi_2 < t)$, as well as accepting states that arise in the standard translation of Boolean LTL to AWA (in Type-2 states). Note that each path in the run of $\mathcal{A}_{\varphi,v}$ eventually gets trapped in a single state. Thus, $\mathcal{A}_{\varphi,v}$ is indeed an AWA. The intuition behind the acceptance condition is as follows. Getting trapped in state of the form $(\psi_1 \mathsf{U} \psi_2 < t)$ is allowed, as the eventuality is satisfied with value 0. On the other hand, getting stuck in other states (or Type-1) is not allowed, as they involve eventualities that are not satisfied in the threshold promised for them.

This concludes the definition of $\mathcal{A}_{\varphi,v}$. Finally, observe that while the construction as described above is infinite (indeed, uncountable), only finitely many states are reachable from the initial state $(\varphi > v)$, and we can compute these states in advance. Intuitively, it follows from the fact that once the proportion between t and $\eta(i)$ goes above 1, for Type-1 states associated with threshold t and sub formulas with a discounting function η, we do not have to generate new states.

A detailed proof of $\mathcal{A}$'s finiteness and correctness is given in the appendix. □

Since $\mathcal{A}_{\varphi,v}$ is a Boolean automaton, then its language is not empty iff it accepts a lasso computation. Combining this observation with Theorem 3.4, we conclude with the following.

Corollary 3.5 *For an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ *and a threshold* $v \in [0,1]$*, it holds that* $L(\mathcal{A}_{\varphi,v}) \neq \emptyset$ *iff there exists a computation* π *such that* $[\![\pi, \varphi]\!] > v$*.*

3.3 Exponential Discounting

The size of the AWA generated as per Theorem 3.4 depends on the discounting functions. In this section, we analyze its size for the class of *exponential discounting* functions, showing that it is singly exponential in the specification formula and in the threshold. This class is perhaps the most common class of discounting functions, as it describes what happens in many natural processes (e.g., temperature change, capacitor charge, effective interest rate, etc.) [8, 25].

For $\lambda \in (0,1)$ we define the *exponential-discounting* function $\exp_\lambda : \mathbb{N} \to [0,1]$ by $\exp_\lambda(i) = \lambda^i$. For the purpose of this section, we restrict to $\lambda \in (0,1) \cap \mathbb{Q}$. Let $E = \{\exp_\lambda : \lambda \in (0,1) \cap \mathbb{Q}\}$, and consider the logic $\mathrm{LTL}^{\mathrm{disc}}[E]$.

For an $\mathrm{LTL}^{\mathrm{disc}}[E]$ formula φ we define the set $F(\varphi)$ to be $\{\lambda_1, ..., \lambda_k :$ the operator $\mathsf{U}_{\exp_\lambda}$ appears in $\varphi\}$. Let $|\langle\varphi\rangle|$ be the length of the description of φ. That is, in addition to $|\varphi|$, we include in $|\langle\varphi\rangle|$ the length, in bits, of describing $F(\varphi)$.

Theorem 3.6 *Given an* $\mathrm{LTL}^{disc}[E]$ *formula* φ *and a threshold* $v \in [0,1] \cap \mathbb{Q}$*, there exists an* AWA $\mathcal{A}_{\varphi,v}$ *such that for every computation* π *the following hold.*

1. *If* $[\![\pi, \varphi]\!] > v$*, then* $\mathcal{A}_{\varphi,v}$ *accepts* π*.*
2. *If* $\mathcal{A}_{\varphi,v}$ *accepts* π *and* π *is a lasso computation, then* $[\![\pi, \varphi]\!] > v$*.*

Furthermore, the number of states of $\mathcal{A}_{\varphi,v}$ *is singly exponential in* $|\langle\varphi\rangle|$ *and in the description of* v*.*

The proof follows from the following observation. Let $\lambda \in (0,1)$ and $v \in (0,1)$. When discounting by $\exp_\lambda$, the number of states in the AWA constructed as per Theorem 3.4 is proportional to the maximal number i such that $\lambda^i > v$, which is at most $\log_\lambda v = \frac{\log v}{\log \lambda}$, which is polynomial in the description length of v and λ. A similar (yet more complicated) consideration is applied for the setting of multiple discounting functions and negations.

3.4 From $\mathcal{A}_{\varphi,v}$ to an NBA

Every AWA can be translated to an equivalent nondeterministic Büchi automaton (NBA, for short), yet the state blowup might be exponential BKR10,MH84. By carefully analyzing the AWA $\mathcal{A}_{\varphi,v}$ generated in Theorem 3.4, we show that it can be translated to an NBA with only a polynomial blowup.

The idea behind our complexity analysis is as follows. Translating an AWA to an NBA involves alternation removal, which proceeds by keeping track of entire levels in a run-DAG. Thus, a run of the NBA corresponds to a sequence of subsets of Q. The key to the reduced state space is that the number of such subsets is only $|Q|^{O(|\varphi|)}$ and not $2^{|Q|}$. To see why, consider a subset S of the states of $\mathcal{A}$. We say that S is *minimal* if it does not include two states of the form $\varphi < t_1$ and $\varphi < t_2$, for $t_1 < t_2$, nor two states of the form $\varphi \mathsf{U}_{\eta^{+i}} \psi < t$ and $\varphi \mathsf{U}_{\eta^{+j}} \psi < t$, for $i < j$, and similarly for ">". Intuitively, sets that are not minimal hold

redundant assertions, and can be ignored. Accordingly, we restrict the state space of the NBA to have only minimal sets.

Lemma 3.7 *For an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ *and* $v \in [0,1]$*, the* AWA $\mathcal{A}_{\varphi,v}$ *constructed in Theorem 3.4 with state space* Q *can be translated to an* NBA *with* $|Q|^{O(|\varphi|)}$ *states.*

3.5 Decision Procedures for $\mathrm{LTL}^{\mathbf{disc}}[\mathcal{D}]$

Model checking and satisfiability. Consider a Kripke structure $\mathcal{K}$, an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ, and a threshold v. By checking the emptiness of the intersection of $\mathcal{K}$ with $\mathcal{A}_{\neg\varphi,1-v}$, we can solve the threshold model-checking problem. Indeed, $L(\mathcal{A}_{\neg\varphi,1-v}) \cap L(\mathcal{K}) \neq \emptyset$ iff there exists a lasso computation π that is induced by $\mathcal{K}$ such that $[\![\pi,\varphi]\!] < v$, which happens iff it is not true that $[\![\mathcal{K},\varphi]\!] \geq v$.

The complexity of the model-checking procedure depends on the discounting functions in $\mathcal{D}$. For the set of exponential-discounting functions E, we provide the following concrete complexities, showing that it stays in the same complexity classes of standard LTL model-checking.

Theorem 3.8 *For a Kripke structure* $\mathcal{K}$*, an* $\mathrm{LTL}^{disc}[E]$ *formula* φ*, and a threshold* $v \in [0,1] \cap \mathbb{Q}$*, the problem of deciding whether* $[\![\mathcal{K},\varphi]\!] > v$ *is in NLOGSPACE in the number of states of* $\mathcal{K}$*, and in PSPACE in* $|\langle\varphi\rangle|$ *and in the description of* v*.*

Proof: By Theorem 3.6 and Lemma 3.7, the size of an NBA $\mathcal{B}$ corresponding to φ and v is singly exponential in $|\langle\varphi\rangle|$ and in the description of v. Hence, we can check the emptiness of the intersection of $\mathcal{K}$ and $\mathcal{B}$ via standard "on the fly" procedures, getting the stated complexities. □

Note that the complexity in Theorem 3.8 is only NLOGSPACE in the system, since our solution does not analyze the Kripke structure, but only takes its product with the specification's automaton. This is in contrast to the approach of model checking temporal logic with (non-discounting) accumulative values, where, when decidable, involves a doubly-exponential dependency on the size of the system [4].

Finally, observe that the NBA obtained in Lemma 3.7 can be used to solve the threshold-satisfiability problem: given an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ and a threshold $v \in [0,1]$, we can decide whether there is a computation π such that $[\![\pi,\varphi]\!] \sim v$, for $\sim \in \{<,>\}$, and return such a computation when the answer is positive. This is done by simply deciding whether there exists a word that is accepted by the NBA.

Threshold synthesis Consider an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ, and assume a partition of the atomic propositions in φ to input and output signals, we can use the NBA $\mathcal{A}_{\varphi,v}$ in order to address the *synthesis* problem, as stated in the following theorem (see Appendix A.6 for the proof).

Theorem 3.9 *Consider an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ*. If there exists a transducer* $\mathcal{T}$ *all of whose computations* π *satisfy* $[\![\pi,\varphi]\!] > v$*, then we can generate a transducer* $\mathcal{T}$ *all of whose computations* τ *satisfy* $[\![\tau,\varphi]\!] \geq v$*.*

4 Adding Propositional Quality Operators

As model checking is decidable for $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$, one may wish to push the limit and extend the expressive power of the logic. In particular, of great interest is the combining of discounting with propositional quality operators [1].

4.1 Adding the Average Operator

A well-motivated extension is the introduction of the average operator $\oplus$, with the semantics $[\![\pi, \varphi \oplus \psi]\!] = \frac{[\![\pi,\varphi]\!]+[\![\pi,\psi]\!]}{2}$. The work in [1] proves that extending LTL by this operator, as well as with other propositional quantitative operators, enables clean specification of quality and results in a logic for which the model-checking problem can be solved in PSPACE.

We show that adding the $\oplus$ operator to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ gives a logic, denoted $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$, for which the satisfiability and model-checking problems are undecidable, both in their strict and non-strict versions. That is, for every $\sim \in \{<, \leq, =\geq, >\}$, it is undecidable, given a formula φ, a system $\mathcal{K}$, and a threshold v, whether $[\![K, \varphi]\!] \sim v$, and whether there exists a computation π such that $[\![\pi, \varphi]\!] \sim v$.

The proofs are arranged in the following structure. We start by showing that the validity problem for $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ is undecidable, and then extract ideas from the proof, which are later used to show that the rest of the problems are undecidable. Our proofs apply to $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ with every nonempty set of discounting functions $\mathcal{D}$.

The validity problem asks, given an $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ formula φ over the atomic propositions AP and a threshold $v \in [0, 1]$, whether $[\![\pi, \varphi]\!] > v$ for every $\pi \in (2^{AP})^\omega$.

In the first undecidability proof, we show a reduction from the 0-halting problem for two-counter machines. A *two-counter machine* $\mathcal{M}$ is a sequence $(l_1, \ldots, l_n)$ of commands involving two counters x and y. We refer to $\{1, \ldots, n\}$ as the *locations* of the machine. There are five possible forms of commands:

$$\textsc{inc}(c),\ \textsc{dec}(c),\ \textsc{goto}\ l_i,\ \textsc{if}\ c{=}0\ \textsc{goto}\ l_i\ \textsc{else goto}\ l_j,\ \textsc{halt},$$

where $c \in \{x, y\}$ is a counter and $1 \leq i, j \leq n$ are locations. Since we can always check whether $c = 0$ before a $\textsc{dec}(c)$ command, we assume that the machine never reaches $\textsc{dec}(c)$ with $c = 0$. That is, the counters never have negative values. Given a counter machine $\mathcal{M}$, deciding whether $\mathcal{M}$ halts is known to be undecidable [20]. Given $\mathcal{M}$, deciding whether $\mathcal{M}$ halts with both counters having value 0, termed the 0-*halting problem*, is also undecidable: given a counter machine $\mathcal{M}$, we can replace every HALT command with a code that clears the counters before halting. In fact, from this we see that the promise problem of deciding whether $\mathcal{M}$ 0-halts given the promise that either it 0-halts, or it does not halt at all, is also undecidable.

Theorem 4.1 *The validity problem for* $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ *is undecidable.*

Proof: We start by showing a reduction from the 0-halting problem for two-counter machines to the following problem: given an $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ formula φ over the atomic propositions AP, whether there exists a path $\pi \in AP^\omega$ such that $[\![\pi, \varphi]\!] \geq \frac{1}{2}$. We dub this the $\frac{1}{2}$-*co-validity* problem. We will later reduce this problem to the (complement of the) validity problem.

Let $\mathcal{M}$ be a two-counter machine with commands $(l_1, \ldots, l_n)$. A *halting run* of a two-counter machine with commands from the set $L = \{l_1, \ldots, l_n\}$ is a sequence $\rho = \rho_1, \ldots, \rho_m \in (L \times \mathbb{N} \times \mathbb{N})^*$ such that the following hold.

1. $\rho_1 = \langle l_1, 0, 0 \rangle$.
2. For all $1 < i \leq m$, let $\rho_{i-1} = (l_k, \alpha, \beta)$ and $\rho_i = (l', \alpha', \beta')$. Then, the following hold.
   - If l_k is an INC(x) command (resp. INC(y)), then $\alpha' = \alpha + 1$, $\beta' = \beta$ (resp. $\beta' = \beta + 1$, $\alpha' = \alpha$), and $l' = l_{k+1}$.
   - If l_k is a DEC(x) command (resp. DEC(y)), then $\alpha' = \alpha - 1$, $\beta' = \beta$ (resp. $\beta' = \beta - 1$, $\alpha' = \alpha$), and $l' = l_{k+1}$.
   - If l_k is a GOTO l_s command, then $\alpha' = \alpha$, $\beta' = \beta$, and $l' = l_s$.
   - If l_k is an IF x=0 GOTO l_s ELSE GOTO l_t command, then $\alpha' = \alpha$, $\beta' = \beta$, and $l' = l_s$ if $\alpha = 0$, and $l' = l_t$ otherwise.
   - If l_k is an IF y=0 GOTO l_s ELSE GOTO l_t command, then $\alpha' = \alpha, \beta' = \beta$, and $l' = l_s$ if $\beta = 0$, and $l' = l_t$ otherwise.
   - If l' is a HALT command, then $i = m$. That is, a run does not continue after HALT.
3. $\rho_m = \langle l_k, \alpha, \beta \rangle$ such that l_k is a HALT command.

Observe that the machine $\mathcal{M}$ is deterministic. We say that $\mathcal{M}$ 0-halts if there exists $l \in L$ that is a HALT command, such that the run of $\mathcal{M}$ ends in $\langle l, 0, 0 \rangle$.

We say that a sequence of commands $\tau \in L^*$ *fits* a run ρ, if τ is the projection of ρ on its first component.

We construct from $\mathcal{M}$ an $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ formula φ such that $\mathcal{M}$ 0-halts iff there exists a computation π such that $[\![\pi, \varphi]\!] \geq \frac{1}{2}$. The idea behind the construction is as follows. The formula φ is interpreted over computations over the atomic propositions $AP = \{1, ..., n, \#, x, y\}$, where $1, ..., n$ are the commands of M. The computation over which φ is interpreted corresponds to a description of a run of $\mathcal{M}$, where every triplet $\langle l_i, \alpha, \beta \rangle$ is encoded as the string $ix^\alpha y^\beta \#$. We ensure that computations that satisfy φ with value greater than 0 are such that in every position only a single atomic proposition is true.

Example 4.2 Consider the following machine $\mathcal{M}$:

l_1: INC(x)

l_2: IF x=0 GOTO l_6 ELSE GOTO l_3

l_3: INC(y)

l_4: DEC(x)

l_5: GOTO (l_2)

l_6: DEC(y)

l_7: HALT

The command sequence that represents the run of this machine is

$$\langle l_1, 0, 0\rangle, \langle l_2, 1, 0\rangle, \langle l_3, 1, 0\rangle, \langle l_4, 1, 1\rangle, \langle l_5, 0, 1\rangle, \langle l_2, 0, 1\rangle, \langle l_6, 0, 1\rangle, \langle l_7, 0, 0\rangle$$

and the encoding of it as a computation is

$$1\#2x\#3x\#4xy\#5y\#2y\#6y\#7\#$$

□

The formula φ "states" (recall that the setting is quantitative, not Boolean) the following properties of the computation π:

1. The first configuration in π is the initial configuration of $\mathcal{M}$ ($\langle l_1, 0, 0\rangle$, or $1\#$ in our encoding).
2. The last configuration in π is $\langle l, 0, 0\rangle$ (or k in our encoding), where l can be any line whose command is HALT.
3. π represents a legal run of $\mathcal{M}$, up to the consistency of the counters between transitions.
4. The counters are updated correctly between configurations.

Properties 1-3 are can easily be specified by a LTL formulas, such that computations which satisfy properties 1-3 get satisfaction value 1. Property 4 utilizes the expressive power of $\text{LTL}^{\text{disc}\oplus}[\mathcal{D}]$, as we now demonstrate. The intuition behind property 4 is the following. We need to compare the value of a counter before and after a command, such that the formula takes a low value if a violation is encountered, and a high value otherwise. Specifically, the formula we construct takes value $\frac{1}{2}$ if no violation occurred, and a lower value if a violation did occur.

We start with a simpler case, to demonstrate the point. Let $\eta \in \mathcal{D}$ be a discounting function. Consider the formula $CountA := a\mathsf{U}_\eta \neg a$ and the computation $a^i b^j \#^\omega$. It holds that $[\![a^i b^j, CountA]\!] = \eta(i)$. Similarly, it holds that $[\![a^i b^j \#^\omega, a\mathsf{U}(b\mathsf{U}_\eta \neg b)]\!] = \eta(j)$. Denote the latter by $CountB$. Let

$$CompareAB := (CountA \oplus \neg CountB) \wedge (\neg CountA \oplus CountB).$$

We now have that

$$[\![a^i b^j \#^\omega, CompareAB]\!] = \min\left\{\frac{\eta(i) + 1 - \eta(j)}{2}, \frac{\eta(j) + 1 - \eta(i)}{2}\right\} = \frac{1}{2} - \frac{|\eta(i) - \eta(j)|}{2}$$

and observe that the latter is $\frac{1}{2}$ iff $i = j$, and is less than $\frac{1}{2}$ otherwise. This is because η is strictly decreasing, and in particular an injection.

Thus, we can compare counters. To apply this technique to the encoding of a computation, we only need some technical formulas to "parse" the input and find consecutive occurrences of a counter.

We now dive into the technical definition of φ. The atomic propositions are $AP = \{1, ..., n, \#, x, y\}$ (where $l_1, ..., l_n$ are the commands of $\mathcal{M}$). We let $\varphi := CheckCmds \wedge CheckInit \wedge CheckFinal \wedge CheckCounters \wedge ForceSingletons$

ForceSingletons: This formula ensures that for a computation to get a value of more than 0, every letter in the computation must be a singleton. Formally,

$$ForceSingletons := \mathsf{G}\left(\bigvee_{p \in AP}\left(p \wedge \bigwedge_{q \in AP \setminus \{p\}} \neg q\right)\right).$$

CheckInit and CheckFinal: These formulas check that the initial and final configurations are correct, and that after the final configuration, there are only $\#$s.

$$CheckInit := 1 \wedge \mathsf{X}\#.$$

Let $I = \{i : l_i = \text{HALT}\}$, we define

$$CheckFinal := \mathsf{G}((\bigvee_{i \in I} i) \rightarrow \mathsf{XG}\#).$$

Note that $CheckFinal$ also ensures that there counters are 0.

CheckCmds: This formula verifies that the local transitions follow the instructions in the counter machine, ignoring the consistency of the counter values, but enforcing that a jump behaves according to the counters. We start by defining, for every $i \in \{1, ..., n\}$, the formula:

$$waitfor(i) := (x \vee y)\mathsf{U}(\# \wedge \mathsf{X}i).$$

Intuitively, a computation satisfies this formula (i.e., gets value 1) iff it reads counter descriptions until the next delimiter, and the next command is l_i.

Now, for every $i \in \{1, ..., n\}$ we define ψ_i as follows.

- If $l_i = \text{GOTO } l_j$, then $\psi_i := \mathsf{X}waitfor(j)$.
- If $l_i \in \{\text{INC}(c), \text{DEC}(c) : c \in \{x, y\}\}$, then $\psi_i := \mathsf{X}waitfor(i+1)$. [2]
- If $l_i = \text{IF } x\text{=0 GOTO } l_j \text{ ELSE GOTO } l_k$ then $\psi_i := \mathsf{X}((x \rightarrow waitfor(k)) \wedge ((\neg x) \rightarrow waitfor(j)))$.
- If $l_i = \text{IF } y\text{=0 GOTO } l_j \text{ ELSE GOTO } l_k$ then $\psi_i := \mathsf{X}(((x\mathsf{U}y) \wedge waitfor(k)) \vee ((x\mathsf{U}\#) \wedge waitfor(j)))$.
- If $l_i = \text{HALT}$ we do not really need additional constraints, due to $CheckFinal$. Thus we have $\psi_i = \texttt{True}$.

Finally, we define $CheckCmds := \mathsf{G} \bigwedge_{i \in \{1,...,n\}} (i \rightarrow \psi_i)$.

CheckCounters: This is the heart of the construction. The formula checks whether consecutive occurrences of the counters match the transition between the commands. We start by defining $countX := x\mathsf{U}_\eta \neg x$ and $countY := x\mathsf{U}(y\mathsf{U}_\eta \neg y)$. Similarly, we have $countX^{-1} = x\mathsf{U}_\eta \mathsf{X} \neg x$ and $countY^{-1} = x\mathsf{U}(y\mathsf{U}_\eta \mathsf{X} \neg y)$.

We need to define a formula to handle some edge cases.

Let $I_{\text{HALT}} = \{i : l_i = \text{HALT}\}$, and similarly define $I_{\text{DEC}(x)}$ and $I_{\text{DEC}(y)}$. We define

$$\begin{aligned} Last := & \bigvee_{i \in I_{\text{DEC}(x)}} i \wedge \mathsf{X}(x \wedge \mathsf{X}(\# \wedge \mathsf{X} \bigvee_{k \in I_{\text{HALT}}} k)) \vee \\ & \bigvee_{i \in I_{\text{DEC}(y)}} i \wedge \mathsf{X}(y \wedge \mathsf{X}(\# \wedge \mathsf{X} \bigvee_{k \in I_{\text{HALT}}} k)) \vee \\ & \bigvee_{i \in \{1,...,n\}} i \wedge \mathsf{X}(\# \wedge \mathsf{X} \bigvee_{k \in I_{\text{HALT}}} k) \end{aligned}$$

[2] if $i = n$ then this line can be omitted from the initial machine, so w.l.o.g this does not happen.

Intuitively, the formula $Last$ holds exactly in the last transition, that is, before the final 0-halting configuration.

Testing the counters involves six types of comparisons: checking equality, increase by 1, and decrease by 1 for each of the two counters. We define the formulas below for these tests. To explain the formulas, consider for example the formula $Comp(x,=)$. This formula compares the number of x's in the current configuration, with the number of x's in the next configuration. The comparison is based on the comparison we explained above, and is augmented by some parsing, as we need to reach the next configuration before comparing.

- $Comp(x,=) := (countX \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}\neg countX))))$
  $\wedge ((\neg countX) \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}countX))))$.
- $Comp(y,=) := (countY \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}\neg countY))))$
  $\wedge ((\neg countY) \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}countY))))$.
- $Comp(x,+1) := \big(countX \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}\neg countX^{-1})))\big)$
  $\wedge \big((\neg countX) \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}countX^{-1})))\big)$.
- $Comp(y,+1) := \big(countY \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}\neg countY^{-1})))\big)$
  $\wedge \big((\neg countY) \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}countY^{-1})))\big)$.
- $Comp(x,-1) := \big(countX^{-1} \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}\neg countX)))\big)$
  $\wedge \big((\neg countX^{-1}) \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}countX)))\big)$.
- $Comp(y,-1) := \big(countY^{-1} \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}\neg countY)))\big)$
  $\wedge \big((\neg countY^{-1}) \oplus (x\mathsf{U}(y\mathsf{U}(\# \wedge \mathsf{XX}countY)))\big)$.

Now, for every $i \in \{1,...,n\}$, we define ξ_i as follows.

- If $l_i \in \{\text{GOTO } l_j, \text{IF } c\text{=0 GOTO } l_j \text{ ELSE GOTO } l_k : c \in \{x,y\}\}$, then we need to make sure that the value of the counters do not change. Accordingly, we define $\xi_i := (X(comp(x,=) \wedge comp(y,=)) \vee Last$.
- If $l_i = \text{INC}(x)$, then we need to make sure that x increases and y does not change. Accordingly, we define $\xi_i := (X(comp(x,+1) \wedge comp(y,=)) \vee Last$.
- If $l_i = \text{INC}(y)$, then $\xi_i := (X(comp(x,=) \wedge comp(y,+1)) \vee Last$.
- If $l_i = \text{DEC}(x)$, then $\xi_i := (X(comp(x,-1) \wedge comp(y,=)) \vee Last$.
- If $l_i = \text{DEC}(y)$, then $\xi_i := (X(comp(x,=) \wedge comp(y,-1)) \vee Last$.
- If $l_i = \text{HALT}$, then we do not need additional constraints, due to $CheckFinal$. Accordingly, we define $\xi_i = \texttt{True}$.

Finally, we define $CheckCounters := \mathsf{G}\bigwedge_{i\in\{1,...,n\}}(i \rightarrow \xi_i)$.

The correctness of the construction is obvious, once one verifies that the defined formulas indeed test what they claim to. Thus, we conclude that $\mathcal{M}$ 0-halts iff there exists a computation π such that $[\![\pi,\varphi]\!] \geq \frac{1}{2}$.

Finally, we reduce the $\frac{1}{2}$-co-validity problem to the complement of the validity problem: given a formula φ, the reduction outputs $\langle \neg\varphi, \frac{1}{2} \rangle$. Now, there exists a computation π such that $[\![\pi, \varphi]\!] \geq \frac{1}{2}$ iff there exists a computation π such that $[\![\pi, \neg\varphi]\!] \leq \frac{1}{2}$, iff it is not true that $[\![\pi, \neg\varphi]\!] > \frac{1}{2}$ for every computation π. Thus, φ is $\frac{1}{2}$-co-valid iff $\neg\varphi$ is not valid for threshold $\frac{1}{2}$. We conclude that the validity problem is undecidable.

□

The *model-checking* problem (resp. *strict* model-checking problem) is to decide, given a Kripke structure $\mathcal{K}$, a formula φ, and a threshold v, whether $[\![K, \varphi]\!] \geq v$ (resp. $[\![\mathcal{K}, \varphi]\!] > v$). We now show that both variants of the model-checking problem are undecidable for $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$. For this, we first pinpoint the essential technical details in the reduction in the proof of Theorem 4.1.

Lemma 4.3 *Given a two-counter machine $\mathcal{M}$ that is promised to either 0-halt or not to halt at all, there exists an $\mathrm{LTL}^{disc\oplus}[\mathcal{D}]$ formula φ such that for every computation π that represents a computation of $\mathcal{M}$, the following hold.*

1. *If π is a legal halting computation of $\mathcal{M}$, then $[\![\pi, \varphi]\!] = \frac{1}{2}$.*
2. *If π cheats in a transition between commands, then $[\![\pi, \varphi]\!] = 1$.*
3. *If π cheats in the counter values, then $[\![\pi, \varphi]\!] = \frac{1}{2} + \epsilon$ such that $\epsilon \geq \frac{1}{2}(\eta(i) - \eta(i+1))$ for the minimal difference $\eta(i) - \eta(i+1)$ where i is a counter value in π.*

Before turning to the proofs, let us briefly explain why these results are nontrivial. For the non-strict model-checking problem, there does not seem to be an immediate reduction from the validity problem. Indeed, in the proof of Theorem 4.1 we have that $[\![\mathcal{K}, \varphi]\!] = \frac{1}{2}$ always (for the system $\mathcal{K}$ that generates every computation).

For the strict model-checking problem, it is tempting to say that $[\![\mathcal{K}, \varphi]\!] > v$ iff for every computation π of $\mathcal{K}$ it holds that $[\![\pi, \varphi]\!] > v$. However, this is incorrect, since it may be the case that $[\![\pi, \varphi]\!] > v$ for every computation π, yet these values can get arbitrarily close to v, and then $[\![\mathcal{K}, \varphi]\!] = v$. This convergence of the satisfaction value is at the heart of the problem, and avoiding it is the crux in our proofs.

We now turn to show that the $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ model-checking and strict model-checking problems are undecidable. The proofs apply to every nonempty set of discounting functions $\mathcal{D}$.

Theorem 4.4 *The strict model-checking problem for $\mathrm{LTL}^{disc\oplus}[\mathcal{D}]$ is undecidable.*

Proof: Assume by way of contradiction that the strict model-checking problem is decidable. We show how to decide the 0-halting promise problem for two-counter machines, thus reaching a contradiction.

Given a two-counter machine $\mathcal{M}$ that is promised to either 0-halt, or not halt at all, construct the formula φ as per Lemma 4.3, and consider the Kripke structure $\mathcal{K}$ that generates every computation. Observe that by Lemma 4.3 it holds that $[\![\mathcal{K}, \varphi]\!] \geq \frac{1}{2}$.

Decide whether $[\![\mathcal{K}, \varphi]\!] > \frac{1}{2}$. If $[\![\mathcal{K}, \varphi]\!] > \frac{1}{2}$, then for every computation π it holds that $[\![\pi, \varphi]\!] > \frac{1}{2}$, and by Lemma 4.3 we conclude that $\mathcal{M}$ does not halt.

If $[\![\mathcal{K}, \varphi]\!] \leq \frac{1}{2}$, then $[\![\mathcal{K}, \varphi]\!] = \frac{1}{2}$. We observe that there are now two possible cases:

1. $\mathcal{M}$ halts.

2. $\mathcal{M}$ does not halt, and for every n, there are computations that reach HALT while cheating in counter values larger than n, and not cheating in the commands.

We show how to distinguish between cases 1 and 2.
Consider the $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ formula

$$\psi = \mathtt{True} \oplus (\mathsf{G}(x\mathsf{U}_\eta \neg x) \wedge \mathsf{G}(y\mathsf{U}_\eta \neg y)).$$

It is not hard to verify that for every computation π that represents a computation of $\mathcal{M}$, it holds that $[\![\pi, \psi]\!] = \frac{1}{2} + \frac{1}{2}\epsilon$, where $\epsilon = \inf(\eta(i) : i$ is the value of a counter in $\pi)$. Let $\xi = \varphi \vee \psi$.

If $\mathcal{M}$ halts (case 1), then for every computation π we have one of the following.

a. π describes a legal halting run of $\mathcal{M}$, in which case $[\![\pi, \varphi]\!] = \frac{1}{2}$ and $[\![\pi, \psi]\!] > \frac{1}{2} + \epsilon$ for some $\epsilon > 0$ (independent of π), since the counters are bounded. Thus, $[\![\pi, \xi]\!] > \frac{1}{2} + \epsilon$.

b. π cheats in the commands, in which case $[\![\pi, \varphi]\!] = 1$, so $[\![\pi, \xi]\!] = 1$.

c. π cheats in the counters, in which case, since the counters are bounded, the first cheat must occur with small counters, and thus $[\![\pi, \varphi]\!] > \frac{1}{2} + \epsilon$ for some $\epsilon > 0$ independent of π. So $[\![\pi, \xi]\!] > \frac{1}{2} + \epsilon$.

In all three cases, we get that $[\![\pi, \xi]\!] > \frac{1}{2} + \epsilon$, so $[\![\mathcal{K}, \xi]\!] > \frac{1}{2}$.

If $\mathcal{M}$ does not halt (case 2), then for every n and for every computation π that cheats with counters larger than n, it holds that $[\![\pi, \varphi]\!] < \frac{1}{2} + \epsilon$ where $\epsilon = \epsilon(n) \to 0$ as $n \to \infty$, and since the counters in π are large, it also holds that $[\![\pi, \psi]\!] < \frac{1}{2} + \epsilon'$, where $\epsilon' = \epsilon'(n) \to 0$ as $n \to \infty$. We conclude that there exists a sequence of computations whose satisfaction values in ξ tend to $\frac{1}{2}$, and thus $[\![\mathcal{K}, \xi]\!] = \frac{1}{2}$.

Thus, in order to distinguish between cases 1 and 2, it is enough to decide whether $[\![\mathcal{K}, \xi]\!] > \frac{1}{2}$.

To conclude, the algorithm for deciding whether $\mathcal{M}$ 0-halts is as follows. Start by constructing φ. If $[\![\mathcal{K}, \varphi]\!] > \frac{1}{2}$, then $\mathcal{M}$ does not halt. Otherwise, construct ξ. If $[\![\mathcal{K}, \xi]\!] > \frac{1}{2}$, then $\mathcal{M}$ halts, and otherwise $\mathcal{M}$ does not halt. □

Theorem 4.5 *The model-checking problem for* $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$ *is undecidable.*

Proof: Recall that for every Kripke structure $\mathcal{K}$ and formula φ it holds that $[\![\mathcal{K}, \varphi]\!] \geq v$ iff there does not exist a computation π of $\mathcal{K}$ such that $[\![\pi, \neg\varphi]\!] > 1 - v$.

We show that the latter problem is undecidable even if we fix $\mathcal{K}$ to be the system that generates every computation.

We show a reduction from the 0-halting promise problem to the latter problem. Given a two-counter machine $\mathcal{M}$ that is promised to either 0-halt, or not halt at all, construct the formula φ as per Lemma 4.3 and the formula ψ such that for every computation π we have that $[\![\pi, \psi]\!] = \frac{1}{2} + \frac{1}{2}\epsilon$, where $\epsilon = \inf(\eta(i) - \eta(i+1) : i$ is the value of a counter in $\pi)$. The formula ψ can be defined as

$$\psi = \mathtt{True} \oplus (\mathsf{G}((x \vee \mathsf{X}x)\mathsf{U}_\eta(\neg x \wedge \neg\mathsf{X}x) \wedge (y \vee \mathsf{X}y)\mathsf{U}_\eta(\neg y \wedge \neg\mathsf{X}y))$$

.

Let $\theta = (\neg\varphi) \oplus \psi$. We claim that $\mathcal{M}$ halts iff there exists a computation π such that $[\![\pi, \theta]\!] > \frac{1}{2}$.

If $\mathcal{M}$ halts, then for the computation π that describes the halting run of $\mathcal{M}$ it holds that $[\![\pi, \varphi]\!] = \frac{1}{2}$, and thus $[\![\pi, \neg\varphi]\!] = \frac{1}{2}$. Since the counters in π are bounded (as the run is halting), then $[\![\pi, \psi]\!] > \frac{1}{2}$, and thus $[\![\pi, \theta]\!] > \frac{1}{2}$.

If $\mathcal{M}$ does not halt, consider a computation π.

- If π cheats in the commands, then $[\![\pi, \neg\varphi]\!] = 0$, so $[\![\pi, \theta]\!] = 0 + \frac{1}{2}[\![\pi, \psi]\!] \leq \frac{1}{2}$.
- If π cheats in the counters, then $[\![\pi, \neg\varphi]\!] = \frac{1}{2} - \frac{1}{2}\epsilon$ and $[\![\pi, \psi]\!] = \frac{1}{2} + \frac{1}{2}\epsilon'$, where $\epsilon \geq \frac{1}{2}(\eta(i) - \eta(i+1)) = \epsilon'$ for the smallest difference $\eta(i) - \eta(i+1)$ in π. Thus, $[\![\pi, \theta]\!] \leq \frac{1}{2}$.

□

Finally, using simple reductions, we can obtain the following.

Theorem 4.6 *For every $\sim \in \{<, \leq, =, \geq, >\}$, the following problems are undecidable.*

- *Model checking: given a formula φ, a system $\mathcal{K}$, and a threshold v, whether $[\![K, \varphi]\!] \sim v$.*
- *Satisfiability: given a formula φ, a system $\mathcal{K}$, and a threshold v, whether there exists a computation π such that $[\![\pi, \varphi]\!] \sim v$.*
- *Validity: given a formula φ, a system $\mathcal{K}$, and a threshold v, whether $[\![\pi, \varphi]\!] \sim v$ for every computation π.*

Proof: We state the main ideas, leaving the technical details to the reader. All the results are simple reductions, using Theorems 4.4, 4.5, and the following observations:

1. From Theorem 4.1 we get that the validity problem for the "$>$" case is undecidable.
2. Theorem 4.1 also shows that the satisfiability problem for the "$=$" case is undecidable.
3. The proofs of Theorems 4.1,4.4, and 4.5 use a Kripke structure that generates every computation.

To demonstrate the ideas in the proof, we show the undecidability of some of the cases.

- The model-checking problem for the "$<$" case is undecidable, since it's the complement of the model-checking for the "$\geq$" case.
- The satisfiability problem for the $<$ case is undecidable, since, by Observation 3, it is equivalent to model-checking for the "$<$" case.
- The satisfiability problem for the $>$ case is undecidable, since it amounts to deciding the "$<$" case of satisfiability for the formula $\neg\varphi$ and the threshold $1 - v$.

Similar short arguments proves the undecidability of the other cases.

□

4.2 Adding Unary Multiplication Operators

As we have seen in Section 4.1, adding the operator $\oplus$ to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ makes model checking undecidable. One may still want to find propositional quality operators that we can add to the logic preserving its decidability. In this section we describe one such operator. We extend $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ with the operator $\triangledown_\lambda$, for $\lambda \in (0,1)$, with the semantics $[\![\pi, \triangledown_\lambda \varphi]\!] = \lambda \cdot [\![\pi, \varphi]\!]$. This operator allows the specifier to manually change the satisfaction value of certain subformulas. This can be used to express importance, reliability, etc. of subformulas. For example, in $\mathsf{G}(\mathit{request} \rightarrow (\mathit{response} \vee \triangledown_{\frac{2}{3}} \mathsf{X}\mathit{response})$, we limit the satisfaction value of computations in which a response is given with a delay to $\frac{2}{3}$.

Note that the operator $\triangledown_\lambda$ is similar to a one-time application of $\mathsf{U}_{\exp_\lambda^{+1}}$, thus $\triangledown_\lambda \varphi$ is equivalent to $\mathtt{False}\mathsf{U}_{\exp_\lambda^{+1}}\psi$. In practice, it is better to handle $\triangledown_\lambda$ formulas directly, by adding the following transitions to the construction in the proof of Theorem 3.4.

$$\delta(\triangledown_\lambda \varphi > t, \sigma) = \begin{cases} \delta(\varphi > \frac{t}{\lambda}, \sigma) & \text{if } \frac{t}{\lambda} < 1, \\ \mathtt{False} & \text{if } \frac{t}{\lambda} \geq 1, \end{cases} \quad \delta(\triangledown_\lambda \varphi < t, \sigma) = \begin{cases} \delta(\varphi < \frac{t}{\lambda}, \sigma) & \text{if } \frac{t}{\lambda} \leq 1, \\ \mathtt{True} & \text{if } \frac{t}{\lambda} > 1. \end{cases}$$

5 Extensions

5.1 $\mathrm{LTL}^{\mathbf{disc}}[\mathcal{D}]$ with Past Operators

One of the well-known augmentations of LTL is the addition of *past operators* [18]. These operators enable the specification of exponentially more succinct formulas, while preserving the PSPACE complexity of model checking. In this section, we add *discounting-past* operators to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$, and show how to perform model-checking on the obtained logic.

We add the operators $\mathsf{Y}\varphi$, $\varphi \mathsf{S} \psi$, and $\varphi \mathsf{S}_\eta \varphi$ (for $\eta \in \mathcal{D}$) to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$, and denote the extended logic $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$, with the following semantics. For $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas φ, ψ, a function $\eta \in \mathcal{D}$, a computation π, and an index $i \in \mathbb{N}$, we have

- $[\![\pi^i, \mathsf{Y}\varphi]\!] = [\![\pi^{i-1}, \varphi]\!]$ if $i > 0$, and 0 otherwise.
- $[\![\pi^i, \varphi \mathsf{S} \psi]\!] = \max_{0 \leq j \leq i} \left\{ \min \left\{ [\![\pi^j, \psi]\!], \min_{j < k \leq i} \left\{ [\![\pi^k, \varphi]\!] \right\} \right\} \right\}$.
- $[\![\pi^i, \varphi \mathsf{S}_\eta \psi]\!] = \max_{0 \leq j \leq i} \left\{ \min \left\{ \eta(i-j)[\![\pi^j, \psi]\!], \min_{j < k \leq i} \left\{ \eta(i-k)[\![\pi^k, \varphi]\!] \right\} \right\} \right\}$.

Observe that since the past is finite, then the semantics for past operators can use $\min$ and $\max$ instead of $\inf$ and $\sup$.

As in $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$, our solution for the $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ model-checking problem is by translating $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas to automata. The construction extends the construction for the Boolean case, which uses 2-way weak alternating automata (2AWA). The details of the construction appear in Appendix A.7. The use of the obtained automata in decision procedures is similar to that in Section 3. In particular, it follows that the model-checking problem for $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ with exponential discounting is in PSPACE.

5.2 Weighted Systems

A central property of the logic $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ is that the verified system need not be weighted in order to get a quantitative satisfaction – it stems from taking into account the delays in

satisfying the requirements. Nevertheless, $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ also naturally fits weighted systems, where the atomic propositions have a value between 0 and 1.

A *weighted Kripke structure* is a tuple $\mathcal{K} = \langle AP, S, I, \rho, L\rangle$, where AP, S, I, and ρ are as in Boolean Kripke structures, and $L : S \to [0,1]^{AP}$ maps each state to a weighted assignment to the atomic propositions. Thus, the value $L(s)(p)$ of an atomic proposition $p \in AP$ in a state $s \in S$ is a value in $[0,1]$. The semantics of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ with respect to a weighted computation coincides with the one for non-weighted systems, except that for an atomic proposition p, we have that $[\![\pi, p]\!] = L(\pi_0)(p)$.

It is possible to extend the construction of $\mathcal{A}_{\varphi,v}$ described in Section 3.2 to an alphabet W^{AP}, where W is a set of possible values for the atomic propositions. Indeed, we only have to adjust the transition for states that correspond to atomic propositions, as follows: for $p \in AP$, $v \in [0,1]$, and $\sigma \in W^{AP}$, we have that

- $\delta(p > v, \sigma) = \begin{cases} \mathtt{True} & \text{if } \sigma(p) > v, \\ \mathtt{False} & \text{otherwise.} \end{cases}$
- $\delta(p < v, \sigma) = \begin{cases} \mathtt{True} & \text{if } \sigma(p) < v, \\ \mathtt{False} & \text{otherwise.} \end{cases}$

5.3 Changing the Tendency of Discounting

One may observe that in our discounting scheme, the value of future formulas is discounted toward 0. This, in a way, reflects an intuition that we are pessimistic about the future, or at least we are impatient. While in some cases this fits the needs of the specifier, it may well be the case that we are ambivalent to the future. To capture this notion, one may want the discounting to tend to $\frac{1}{2}$. Other values are also possible. For example, it may be that we are optimistic about the future, say when a system improves its performance while running and we know that components are likely to function better in the future. We may then want the discounting to tend, say, to $\frac{3}{4}$.

To capture this notion, we define the operator $\mathsf{O}_{\eta,z}$, parameterized by $\eta \in \mathcal{D}$ and $z \in [0,1]$, with the following semantics. $[\![\pi, \varphi \mathsf{O}_{\eta,z} \psi]\!] =$
$\sup_{i \geq 0}\{\min\{\eta(i)[\![\pi^i, \psi]\!] + (1-\eta(i))z, \min_{0 \leq j < i} \eta(j)[\![\pi^j, \varphi]\!] + (1-\eta(j))z\}\}$. The discounting function η determines the rate of convergence, and z determines the limit of the discounting. The longer it takes to fulfill the "eventuality", the closer the satisfaction value gets to z. We observe that $\varphi \mathsf{U}_\eta \psi \equiv \varphi \mathsf{O}_{\eta,0} \psi$.

Example 5.1 Consider a process scheduler. The scheduler decides which process to run at any given time. The scheduler may also run a defragment tool, but only if it is not in expense of other processes. This can be captured by the formula $\varphi = \mathtt{True}\mathsf{O}_{\eta,\frac{1}{2}}\mathrm{defrag}$. Thus, the defragment tool is a "bonus": if it runs, then the satisfaction value is above $\frac{1}{2}$, but if it does not run, the satisfaction value is $\frac{1}{2}$. Treating 1 as "good" and 0 as "bad" means that $\frac{1}{2}$ is ambivalent. □

We claim that Theorem 3.4 holds under the extension of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ with the operator O. Indeed, the construction of the AWA is augmented as follows. For $\varphi = \psi_1 \mathsf{O}_{\eta,z} \psi_2$, denote $\frac{t-(1-\eta(0))z}{\eta(0)}$ by τ. One may observe that the conditions on $\frac{t}{\eta(0)}$ correspond to conditions on τ when dealing with O. Accordingly, the transitions from the state $(\psi_1 \mathsf{O}_{\eta,z} \psi_2 > t)$ are defined as follows.

First, if $t = z$, then $\tau = t$ and we identify the state $(\psi_1 \mathsf{O}_{\eta,z} \psi_2 > t)$ with the state $(\psi_1 \mathsf{U} \psi_2 > t)$. Otherwise, $z \neq t$ and we define:

- $\delta((\psi_1 \mathsf{O}_{\eta,z} \psi_2 > t), \sigma) = \begin{cases} \delta((\psi_2 > \tau), \sigma) \vee \\ \quad [\delta((\psi_1 > \tau), \sigma) \wedge (\psi_1 \mathsf{O}_{\eta+1,z} \psi_2 > t)] & \text{if } 0 \leq \tau < 1, \\ \texttt{False} & \text{if } \tau \geq 1, \\ \texttt{True} & \text{if } \tau < 0. \end{cases}$

- $\delta((\psi_1 \mathsf{O}_{\eta,z} \psi_2 < t), \sigma) = \begin{cases} \delta((\psi_2 < \tau), \sigma) \wedge \\ \quad [\delta((\psi_1 < \tau), \sigma) \vee (\psi_1 \mathsf{O}_{\eta+1,z} \psi_2 < t)] & \text{if } 0 < \tau \leq 1, \\ \texttt{True} & \text{if } \tau > 1, \\ \texttt{False} & \text{if } \tau \leq 0. \end{cases}$

The correctness of the construction is proved in Appendix A.8.

6 Discussion

An ability to specify and to reason about quality would take formal methods a significant step forward. Quality has many aspects, some of which are propositional, such as prioritizing one satisfaction scheme on top of another, and some are temporal, for example having higher quality for implementations with shorter delays. In this work we provided a solution for specifying and reasoning about temporal quality, augmenting the commonly used linear temporal logic (LTL). A satisfaction scheme, such as ours, that is based on elapsed times introduces a big challenge, as it implies infinitely many satisfaction values. Nonetheless, we showed the decidability of the model-checking problem, and for the natural exponential-decaying satisfactions, the complexity remains as the one for standard LTL, suggesting the interesting potential of the new scheme. As for combining propositional and temporal quality operators, we showed that the problem is, in general, undecidable, while certain combinations, such as adding priorities, preserve the decidability and the complexity.

Acknowledgement. We thank Eleni Mandrali for pointing to an error in an earlier version of the paper.

References

[1] S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. In *40th ICALP*, 2013.

[2] S. Almagor, Y. Hirshfeld, and O. Kupferman. Promptness in ω-regular automata. In *8th ATVA, LNCS 6252*, pages 22–36, 2010.

[3] M. Bojańczyk and T. Colcombet. Bounds in ω-regularity. In *21st LICS*, pages 285–296, 2006.

[4] U. Boker, K. Chatterjee, T.A. Henzinger, and O. Kupferman. Temporal Specifications with Accumulative Values. In *26th LICS*, pages 43–52, 2011.

[5] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[6] M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *TCS*, 126:77–96, 1994.

[7] L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *TCS*, 345(1):139–170, 2005.

[8] L. de Alfaro, T. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *30th ICALP, LNCS 2719*, pages 1022–1037, 2003.

[9] M. Droste and G. Rahonis. Weighted automata and weighted logics with discounting. *TCS*, 410(37):3481–3494, 2009.

[10] M. Droste and H. Vogler. Weighted automata and multi-valued logics over arbitrary bounded lattices. *TCS*, 418:14–36, 2012.

[11] M. Faella, A. Legay, and M. Stoelinga. Model checking quantitative linear time logic. *Electr. Notes Theor. Comput. Sci.*, 220(3):61–77, 2008.

[12] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *13th CAV, LNCS 2102*, pages 53–65. Springer, 2001.

[13] S.H. Kan. Metrics and Models in Software Quality Engineering. *Addison-Wesley Longman Publishing Co.*, 2002.

[14] O. Kupferman, N. Piterman, and M.Y. Vardi. From Liveness to Promptness. In *19th CAV, LNCS 4590*, pages 406–419. Springer, 2007.

[15] D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3):405–425, 1994.

[16] M. Kwiatkowska. Quantitative verification: models techniques and tools. In *ESEC/SIGSOFT FSE*, pages 449–458, 2007.

[17] F. Laroussinie and P. Schnoebelen. A hierarchy of temporal logics with past. In *11th STACS*, pages 47–58, 1994.

[18] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Logics of Programs, LNCS 193*, pages 196–218. Springer, 1985.

[19] E. Mandrali. Weighted LTL with discounting. In *CIAA, LNCS 7381*, pages 353–360, 2012.

[20] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1 edition, 1967.

[21] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *TCS*, 32:321–330, 1984.

[22] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[23] S. Moon, K. Lee, and D. Lee. Fuzzy branching temporal logic. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(2):1045–1055, 2004.

[24] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.

[25] L. Shapley. Stochastic games. In *Proc. of the National Academy of Science*, vol. 39, 1953.

[26] D. Spinellis. Code Quality: The Open Source Perspective. *Addison-Wesley Professional*, 2006.

[27] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata, LNCS 1043*, pages 238–266, 1996.

[28] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *1st LICS*, pages 332–344, 1986.

A Proofs

A.1 Proof of Lemma 3.1

We construct φ^+ and $\varphi^{<1}$ by induction on the structure of φ as follows. In all cases but the U case we do not use the assumption that $\pi = u \cdot v^\omega$ and prove an "iff" criterion.

- If φ is of the form $\texttt{True}, \texttt{False}$, or p, for an atomic proposition p, then $\varphi^+ = \varphi$ and $\varphi^{<1} = \neg\varphi$. Correctness is trivial.
- If φ is of the form $\psi_1 \vee \psi_2$, then $\varphi^+ = {\psi_1}^+ \vee {\psi_2}^+$ and $\varphi^{<1} = {\psi_1}^{<1} \wedge {\psi_2}^{<1}$. Indeed, for every computation π we have that $[\![\pi, \varphi]\!] > 0$ iff either $[\![\pi, \psi_1]\!] > 0$ or $[\![\pi, \psi_2]\!] > 0$, and $[\![\pi, \varphi]\!] < 1$ iff both $[\![\pi, \psi_1]\!] < 1$ and $[\![\pi, \psi_2]\!] < 1$.
- If φ is of the form $\mathsf{X}\psi_1$, then $\varphi^+ = \mathsf{X}({\psi_1}^+)$ and $\varphi^{<1} = \mathsf{X}({\psi_1}^{<1})$. Correctness is trivial.
- If φ is of the form $\psi_1 \mathsf{U} \psi_2$, then $\varphi^+ = {\psi_1}^+ \mathsf{U} {\psi_2}^+$ and $\varphi^{<1} = \neg((\neg({\psi_1}^{<1}))\mathsf{U}(\neg({\psi_2}^{<1})))$.

  We start with φ^+. For every computation π we have that $[\![\pi, \varphi]\!] > 0$ iff there exists $i \geq 0$ such that $[\![\pi^i, \psi_2]\!] > 0$ and for every $0 \leq j < i$ it holds that $[\![\pi^j, \psi_1]\!] > 0$. This happens iff π satisfies ${\psi_1}^+ \mathsf{U} {\psi_2}^+$.

  Before we turn to the case of $\varphi^{<1}$, let us note that readers familiar with the release (R) operator of LTL may find it clearer to observe that $\varphi^{<1} = {\psi_1}^{<1} \mathsf{R} {\psi_2}^{<1}$, which perhaps gives a clearer intuition for the correctness of the construction.

  Now, if $[\![\pi, \varphi]\!] < 1$, then for every $i \geq 0$ it holds that either $[\![\pi^i, \psi_2]\!] < 1$ or $[\![\pi^j, \psi_1]\!] < 1$ for some $0 \leq j < i$. Thus, for every $i \geq 0$, either $\pi^i \models {\psi_2}^{<1}$, or $\pi^j \models {\psi_1}^{<1}$ for some $0 \leq j < i$. So for every $i \geq 0$, either $\pi^i \not\models \neg({\psi_2}^{<1})$, or $\pi^j \not\models \neg({\psi_1}^{<1})$ for some $0 \leq j < i$. It follows that $\pi \not\models (\neg({\psi_1}^{<1}))\mathsf{U}(\neg({\psi_2}^{<1}))$. Equivalently, $\pi \models \neg((\neg({\psi_1}^{<1}))\mathsf{U}(\neg({\psi_2}^{<1})))$.

  Conversely, if $\pi = u \cdot v^\omega$ and $\pi \models \neg((\neg({\psi_1}^{<1}))\mathsf{U}(\neg({\psi_2}^{<1})))$, then $\pi \not\models (\neg({\psi_1}^{<1}))\mathsf{U}(\neg({\psi_2}^{<1}))$, so for every $i \geq 0$, either $\pi^i \models {\psi_2}^{<1}$ or $\pi^j \models {\psi_1}^{<1}$ for some $0 \leq j < i$. By the induction hypothesis, for every $i \geq 0$, either $[\![\pi^i, \psi_2]\!] < 1$ or $[\![\pi^j, \psi_1]\!] < 1$ for some $0 \leq j < i$. We now use the assumption that $\pi = u \cdot v^\omega$ to observe that the $\sup$ in the expression for $[\![\pi, \varphi]\!]$ is attained as a $\max$, as there are only finitely many distinct suffixes for π (namely $\pi^0, ..., \pi^{|u|+|v|-1}$). Thus, since all the elements in the $\max$ are strictly smaller than 1, we conclude that $[\![\pi, \varphi]\!] < 1$.
- If $\varphi = \neg\psi$, then $\varphi^+ = \psi^{<1}$ and $\varphi^{<1} = \psi^+$. Again, correctness is trivial.
- If $\varphi = \psi_1 \mathsf{U}_\eta \psi_2$ for $\eta \in \mathcal{D}$, then $\varphi^+ = {\psi_1}^+ \mathsf{U} {\psi_2}^+$. Indeed, since $\eta(i) > 0$ for all $i \geq 0$, then $\varphi^+ = \psi_1 \mathsf{U} {\psi_2}^+$.

  Now, $\varphi^{<1}$ is defined as follows. First, if $\eta(0) < 1$, then $\varphi^{<1} = \texttt{True}$. If $\eta(0) = 1$, then $\varphi^{<1} = {\psi_2}^{<1}$. Indeed, since η is strictly decreasing, the only chance of φ to have $[\![\pi, \varphi]\!] = 1$ is when both $\eta(0) = 1$ and $[\![\pi^0, \psi_2]\!] = 1$. Since a satisfaction value cannot exceed 1, the latter happens iff $\eta(0) = 1$ and $\pi \not\models {\psi_2}^{<1}$ (where the "only if" direction is valid when π is a lasso, as is assumed).

Finally, it is easy to see that $|\varphi^+|$ and $|\varphi^{<1}|$ are both $O(|\varphi|)$.

A.2 The standard translation of LTL to AWA

For completeness, we bring here the construction of the translation from LTL to AWA, which we use in Theorem 3.4. For the correctness proof, see e.g. [27].

Given an LTL formula φ over the atomic propositions AP, we construct an AWA $\mathcal{A}_\varphi = \langle Q, 2^{AP}, Q_0, \delta, \alpha \rangle$ as follows. The state space Q consists of all the subformulas of φ, and their negations (we identify $\neg\neg\psi$ with $\neg\psi$). The initial state is φ, and the accepting states are all the formulas of the form $\neg(\psi_1 \mathsf{U} \psi_2)$. It remains to define the transition function.

We start with a few notations. For a Boolean formula θ over Q, we define its *dual formula* $\overline{\theta}$ by induction over the construction of θ, as follows.

- For $\psi \in Q$ we have $\overline{\psi} = \neg\psi$.
- $\overline{\texttt{True}} = \texttt{False}$ and $\overline{\texttt{False}} = \texttt{True}$.
- $\overline{\psi_1 \vee \psi_2} = \overline{\psi_1} \wedge \overline{\psi_2}$ and $\overline{\psi_1 \wedge \psi_2} = \overline{\psi_1} \vee \overline{\psi_2}$

The transition function can now be defined as follows. Let $\psi \in Q$ and $\sigma \in 2^{AP}$.

- If $\psi = p \in AP$, then $\delta(\psi, \sigma) = \begin{cases} \texttt{True} & p \in \sigma, \\ \texttt{False} & p \notin \sigma, \end{cases}$
- If $\psi = \zeta_1 \vee \zeta_2$, then $\delta(\psi, \sigma) = \delta(\zeta_1, \sigma) \vee \delta(\zeta_2, \sigma)$,
- If $\psi = \neg\zeta$, then $\delta(\psi, \sigma) = \overline{\delta(\zeta, \sigma)}$,
- If $\psi = \mathsf{X}\zeta$, then $\delta(\psi, \sigma) = \zeta$,
- If $\psi = \zeta_1 \mathsf{U} \zeta_2$, then $\delta(\psi, \sigma) = \delta(\zeta_2, \sigma) \vee (\delta(\zeta_1, \sigma) \wedge \zeta_1 \mathsf{U} \zeta_2)$.

A.3 Continuation of the Proof of Theorem 3.4

We continue the proof that is given in the main text, showing that the constructed AWA $\mathcal{A}_{\varphi,v}$ is indeed finite and correct.

We first make some notations and observations regarding the structure of $\mathcal{A}_{\varphi,v}$. For every state $(\psi > t)$ (resp. $(\psi < t)$) we refer to ψ and t as the state's *formula* and *threshold*, respectively. If the outermost operator in ψ is a discounting operator, then we refer to its discounting function as the state's *discounting function*. For states of Type-2 we refer to their *formula* only (as there is no threshold).

First observe that the only cycles in $\mathcal{A}_{\varphi,v}$ are self-loops. Indeed, consider a transition from state q to state $s \neq q$. Let ψ_q, ψ_s be the formulas of q and s, respectively. Going over the different transitions, one may see that either ψ_s is a strict subformula of ψ_q, or s is a Type-2 state, or both ψ_q and ψ_s have an outermost discounting operator with discounting functions η and η^{+1} respectively. By induction over the construction of φ, this observation proves that there are only self-cycles in $\mathcal{A}_{\varphi_v}$.

We now observe that in every run of $\mathcal{A}_{\varphi,v}$ on an infinite word w, every infinite branch (i.e., a branch that does not reach `True`) must eventually be in a state of the form $\psi_1 \mathsf{U} \psi_2 > t$, $\psi_1 \mathsf{U} \psi_2 < t$, $\psi_1 \mathsf{U} \psi_2$ or $\neg(\psi_1 \mathsf{U} \psi_2)$ (if it's a Type-2 state). Indeed, these states are the only states that have a self-loop, and the only cycles in the automaton are self-loops.

We start by proving that there are finitely many states in the construction. First, all the sub-automata that correspond to Type-2 states have $O(|\varphi|)$ states. This follows immediately from Lemma 3.1 and from the construction of an AWA from an LTL formula.

Next, observe that the number of possible state-formulas, up to differences in the discounting function, is $O(|\varphi|)$. Indeed, this is simply the standard closure of φ. It remains to prove that the number of possible thresholds and discounting functions is finite.

We start by claiming that for every threshold $t > 0$, there are only finitely many reachable states with threshold t. Indeed, for every discounting function $\eta \in \mathcal{D}$ (that appears in φ), let $i_{t,\eta} = \max\left\{i : \frac{t}{\eta(i)} \leq 1\right\}$. The value of $i_{t,\eta}$ is defined, since the functions tend to 0. Observe that in every transition from a state with threshold t, if the next state is also with threshold t, then the discounting function (if relevant) is either some $\eta' \in \mathcal{D}$, or η^{+1}. There are only finitely many functions of the former kind. As for the latter kind, after taking η^{+1} $i_{t,\eta}$ times, we have that $t/\eta^{+i_{t,\eta}}(0) > 1$. By the definition of δ, in this case the transitions are to the Boolean-formula states (i.e., $\texttt{True}, \texttt{False}$, or some ψ^+), from which there are finitely many reachable states. We conclude that for every threshold, there are only finitely many reachable states with this threshold.

Next, we claim that there are only finitely many reachable thresholds. This follows immediately from the claim above. We start from the state $\varphi > v$. From this state, there are only finitely many reachable discounting functions. The next threshold that can be encountered is either $1 - v$, or $\frac{v}{\eta(0)}$ for η that is either in $\mathcal{D}$ or one of the η^{+i} for $i \leq i_{v,\eta}$. Thus, there are only finitely many such thresholds. Further observe that if a different threshold is encountered, then by the definition of δ, the state's formula is deeper in the generating tree of φ. Thus, there are only finitely many times that a threshold can change along a single path. So by induction over the depth of the generating tree, we can conclude that there are only finitely many reachable thresholds.

We conclude that the number of states of the automaton is finite.

Next, we prove the correctness of the construction. From Lemma 3.1 and the correctness of the standard translation of LTL to AWA, it remains to prove that for every path π and for every state $(\psi > v)$ (resp. $(\psi < v)$):

1. If $[\![\pi, \psi]\!] > v$ (resp. $[\![\pi, \psi]\!] < v$), then π is accepted from $(\psi > v)$ (resp. $(\psi < v)$).
2. If $\pi = u \cdot v^\omega$ and π is accepted from state $(\psi > v)$ (resp. $(\psi < v)$) then $[\![\pi, \psi]\!] > v$ (resp. $[\![\pi, \psi]\!] < v$).

The proof is by induction over the construction of φ, and is fairly trivial given the definition of δ.

A.4 Proof of Lemma 3.7

We start by defining *generalized Büchi automata*. An NGBA is $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, \alpha\rangle$, where Q, Σ, δ, Q_0 are as in NBA. The acceptance condition is $\alpha = \{F_1, ..., F_k\}$ where $F_i \subseteq Q$ for every $1 \leq i \leq k$. A run r of $\mathcal{A}$ is accepting if for every $1 \leq i \leq k$, r visits F_i infinitely often.

We now proceed with the proof.

Consider the AWA $\mathcal{A}$ obtained from φ using the construction of Section 3.2.

In the translations of AWA to NBA using the method of [12], the AWA is translated to an NGBA whose states are the subset-construction of the AWA. This gives an exponential

blowup in the size of the automaton. We claim that in our translation, we can, in a sense, avoid this blowup.

Intuitively, each state in the NGBA corresponds to a conjunction of states of the AWA. Consider such a conjunction of states of $\mathcal{A}$. If the conjunction contains two states $(\psi < t_1)$ and $(\psi < t_2)$, and we have that $t_1 < t_2$, then by the correctness proof of Theorem 3.4, it holds that a path π is accepted from both states, iff π is accepted from $(\psi < t_1)$. Thus, in every conjunction of states from $\mathcal{A}$, there is never a need to consider a formula with two different "$<$" thresholds. Dually, every formula can appear with at most one "$>$" threshold.

Next, consider conjunctions that contain states of the form $(\psi_1 \mathsf{U}_\eta \psi_2 < t)$ and $(\psi_1 \mathsf{U}_{\eta^{+k}} \psi_2 < t)$. Again, since the former assertion implies the latter, there is never a need to consider two such formulas. Similar observations hold for the other discounting operators.

Thus, we can restrict the construction of the NGBA to states that are conjunctions of states from the AWA, such that no discounting operator appears with two different "offsets".

Further observe that by the construction of the AWA, the threshold of a discounting formula does not change, with the transition to the same discounting formula, only the offset changes. That is, from the state $(\psi_1 \mathsf{U}_\eta \psi_2 < t)$, every reachable state whose formula is $\psi_1 \mathsf{U}_{\eta^{+k}} \psi_2$ has threshold t as well. Accordingly, the possible number of thresholds that can appear with the formula $\psi_1 \mathsf{U}_\eta \psi_2$ in the subset construction of $\mathcal{A}$, is the number of times that this formula appears as a subformula of φ, which is $O(|\varphi|)$.

We conclude that each state of the obtained NGBA is a function that assigns each subformula[3] of φ two thresholds. The number of possible thresholds and offsets is linear in the number of states of $\mathcal{A}$, thus, the number of states of the NGBA is $|\mathcal{A}|^{O(|\varphi|)}$.

Finally, translating the NGBA to an NBA requires multiplying the size of the state space by $|\mathcal{A}|$, so the size of the obtained NBA is also $|\mathcal{A}|^{O(|\varphi|)}$.

A.5 Proof of Theorem 3.6

We construct an AWA $\mathcal{A} = \mathcal{A}_{\varphi,v}$ as per Section 3.2, with some changes.

Recall that the "interesting" states in $\mathcal{A}$ are those of the form $\psi_1 \mathsf{U}_{\eta^{+i}} \psi_2$. Observe that for the function $\exp_\lambda$ it holds that $\exp_\lambda^{+i} = \lambda^i \cdot \exp_\lambda$. Accordingly, we can replace a state of the form $\psi_1 \mathsf{U}_{\exp_\lambda^{+i}} \psi_2 < t$ with the state $\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < \frac{t}{\lambda^i}$, as they express the same assertion. Finally, notice that $\exp_\lambda(0) = 1$. Thus, we can simplify the construction of $\mathcal{A}$ with the following transitions:

Let $\sigma \in 2^{AP}$, then we have that

- $\delta((\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 > t), \sigma) =$
$$\begin{cases} \delta((\psi_2 > t), \sigma) \vee \\ \quad [\delta((\psi_1 > t), \sigma) \wedge (\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 > \frac{t}{\lambda})] & \text{if } 0 < t < 1, \\ \texttt{False} & \text{if } t \geq 1, \\ \delta(((\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2)^+), \sigma) & \text{if } t = 0. \end{cases}$$
- $\delta((\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < t), \sigma) =$
$$\begin{cases} \delta((\psi_2 < t), \sigma) \wedge \\ \quad [\delta((\psi_1 < t), \sigma) \vee (\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < \frac{t}{\lambda})] & \text{if } 0 < t \leq 1, \\ \texttt{True} & \text{if } t > 1, \\ \texttt{False} & \text{if } t = 0\,. \end{cases}$$

[3] where a subformula may have several occurences, e.g., in the formula $p \wedge \mathsf{X}p$ we have two occurences of the subformula p

- $\delta((\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 > t), \sigma) =$
$$\begin{cases} \delta((\psi_2 > t), \sigma) \wedge \\ \quad [\delta((\psi_1 > t), \sigma) \vee (\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 > \frac{t}{\lambda})] & \text{if } 0 < t < 1, \\ \texttt{False} & \text{if } t \geq 1, \\ \delta(((\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2)^+), \sigma) & \text{if } t = 0. \end{cases}$$

- $\delta((\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < t), \sigma) =$
$$\begin{cases} \delta((\psi_2 < t), \sigma) \vee \\ \quad [\delta((\psi_1 < t), \sigma) \wedge (\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < \frac{t}{\lambda})] & \text{if } 0 < t \leq 1, \\ \texttt{True} & \text{if } t > 1, \\ \texttt{False} & \text{if } t = 0. \end{cases}$$

The correctness and finiteness of the construction follows from Theorem 3.4, with the observation above. We now turn to analyze the number of states in $\mathcal{A}$.

For every state $(\psi > t)$ (resp. $(\psi < t)$) we refer to ψ and t as the state's *formula* and *threshold*, respectively. Observe that the number of possible state-formulas is $O(|\varphi|)$. Indeed, the formulas in the states are either in the closure of φ, or are of the form ψ^+, where ψ is in the closure of φ. This is because in the new transitions we do not carry the offset, but rather change the threshold, so the state formula does not change.

It remains to bound the number of possible thresholds. Consider a state with threshold t. In every succeeding state[4], the threshold (if exists) can either remain t, or change to $1 - t$ (in case of negation), or t/λ, where $\lambda \in F(\varphi)$, providing $t < 1$.

Initially, we ignore negations. In this case, the number of states that can be reached from a threshold t is bounded by the size of the set

$$\left\{ \prod_{i=1}^{k} \lambda_i : \prod_{i=1}^{k} \lambda_i > t,\ k \in \mathbb{N}, \forall i\ \lambda_i \in F(\varphi) \right\}$$

This length of the products can be bounded by $\log_\mu t = \frac{\log t}{\log \mu} = O(\log t)$, where $\mu = \max F(\varphi)$. Thus, the number of possible values is bounded by $(\log_\mu t)^{|F(\varphi)|}$. From here we denote $\log_\mu t$ by ℓ.

Next, we consider negations. Observe that in every path, there are at most $|\varphi|$ negations. In every negation, if the current state has threshold s, the threshold changes to $1 - s$. We already proved that from threshold t we can get at most $(\ell)^m$ states. Furthermore, every such state has a threshold of the form

$$\frac{t}{\prod_{i=1}^{\ell} \lambda_i}$$

where $\lambda_i \in F(\varphi) \cup \{1\}$ (we allow 1 instead of allowing shorter products).

Consider a negation state with threshold $s = \frac{t}{\prod_{i=1}^{\ell} \lambda_i}$. If $s = 1$ then in the next state the threshold is 0, and every reachable state is part of a Boolean AWA corresponding to an LTL formula, and thus has polynomially many reachable states.

Otherwise, we have that $s < 1$. Denote $F(\varphi) = \{\lambda_1, ..., \lambda_m\}$, and assume that $t, \lambda_1, ..., \lambda_m$ can be written as $t = \frac{p_t}{q_t}$ and for all i, $\lambda_i = \frac{p_i}{q_i}$ (which is possible as they are in $\mathbb{Q}$), then we have that

[4]This is almost correct. In fact, since δ is defined inductively, we may go through several transitions.

$\log(1-s) = \log(1-\frac{t}{\prod_{i=1}^{\ell}\lambda_i}) = \log(\prod_{i=1}^{\ell}\lambda_i - t) - \log(\prod_{i=1}^{\ell}\lambda_i) > \log(\prod_{i=1}^{\ell}\lambda_i - t)$,
where the last transition is because $\log(\prod_{i=1}^{\ell}\lambda_i) < 0$.

Let $q_{max} = \max\{q_1, ..., q_m\}$, we now observe that

$$\prod_{i=1}^{\ell}\lambda_i - t = \prod_{i=1}^{\ell}\frac{p_i}{q_i} - \frac{p_t}{q_t} = \frac{\prod_{i=1}^{\ell} p_i q_t - \prod_{i=1}^{\ell} q_i p_t}{\prod_{i=1}^{\ell} q_i q_t} \geq \frac{1}{\prod_{i=1}^{\ell} q_i q_t} \geq \frac{1}{q_{max}^{\ell} q_t}$$

where the last transitions are because all the numbers are natural, and $s > 1$, so the numerator must be a positive integer. From this we get that

$$\log(1-s) > \log\left(\frac{1}{q_{max}^{\ell} q_t}\right) = -(\ell \cdot (\log q_{max} + \log q_t))$$

We see that we can bound $1-s$ from below by a rational number whose representation is linear in that of t and $|\langle\varphi\rangle|$. Denote this linear function by $f(n)$. Thus, continuing in this manner through $O(\varphi)$ negations, starting from the threshold v, we have that the number of thresholds reachable from every state is bounded by $\underbrace{(f(f(...f(\log v))))}_{O(\varphi)}^{m}$ which is single exponential in $|\langle\varphi\rangle|$ and the description of v.

We conclude that the number of states of the automaton is single-exponential.

A.6 Proof of Theorem 3.9

Recall that if π is a computation such that $[\![\pi, \varphi]\!] > v$, hen $\mathcal{A}_{\varphi,v}$ accepts π. The converse however, is not true. Still, by carefully examining the construction in Theorem 3.2, we observe that if $\mathcal{A}_{\varphi,v}$ accepts a computation π, then $[\![\pi, \varphi]\!] \geq v$ (note the non-strict inequality).

Assume a partition of the letters in AP to input and output signals, denoted I and O, respectively. By following standard (Boolean) procedures for synthesis (see [24]), we can generate from $\mathcal{A}_{\varphi,v}$ a deterministic tree automaton $\mathcal{D}$ that accepts a 2^O-labeled 2^I-tree iff all the paths along the tree are accepted in $\mathcal{A}_{\varphi,v}$. A tree that is accepted in this manner represents a transducer that realizes φ with value at least v. Accordingly, if there exists a transducer $\mathcal{T}$, all of whose computations satisfy $[\![\pi, \varphi]\!] > v$, then the tree that represents such a transducer is accepted in $\mathcal{D}$. Thus, the language of $\mathcal{D}$ is non-empty, and a non-emptiness witness is a transducer (tree) $\mathcal{T}'$, all of whose computations satisfy $[\![\pi, \varphi]\!] \geq v$.

We remark that this solution is only partial, as there might be an I/O-transducer whose computations π all satisfy $[\![\pi, \varphi]\!] \geq v$, but we will not find it.

A.7 Translating $\mathrm{PLTL}^{\mathbf{disc}}[\mathcal{D}]$ formulas to 2AWA

As we now show, the construction we use when working with the "infinite future" U_η operator is similar to that of the one we use for the "finite past" S_η operator. The key for this somewhat surprising similarity is the fact our construction is based on a threshold. Under this threshold, we essentially bound the future that needs to be considered, thus the fact that it is technically infinite plays no role.

A 2AWA is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ where Σ, Q, q_0, α are as in AWA. The transition function is $\delta : Q \times \Sigma \to \mathcal{B}^+(\{-1, 1\} \times Q)$. That is, positive Boolean formulas over atoms of the form $\{-1, 1\} \times Q$, describing both the state to which the automaton moves and the direction in which the reading head proceeds.

As in $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$, the construction extends the construction for the Boolean case. It is not hard to extend Lemma 3.1 and generate Boolean PLTL formulas for satisfaction values in $\{0, 1\}$.

Given a $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ and a threshold $t \in [0, 1)$, we construct a 2AWA as in Theorem 3.4 with the following additional transitions:[5].

- $\delta((\mathsf{Y}\psi > t), \sigma) = \langle -1, (\psi > t) \rangle$
- $\delta((\mathsf{Y}\psi < t), \sigma) = \langle -1, (\psi < t) \rangle$.
- $\delta((\psi_1 \mathsf{S} \psi_2 > t), \sigma) = \delta((\psi_2 > t), \sigma) \vee (\delta((\psi_1 > t), \sigma) \wedge \langle -1, (\psi_1 \mathsf{S} \psi_2 > t) \rangle)$.
- $\delta((\psi_1 \mathsf{S} \psi_2 < t), \sigma) = \delta((\psi_2 < t), \sigma) \wedge (\delta((\psi_1 < t), \sigma) \vee \langle -1, (\psi_1 \mathsf{S} \psi_2 < t) \rangle)$.
- $\delta((\psi_1 \mathsf{S}_\eta \psi_2 > t), \sigma) =$
$$\begin{cases} \delta((\psi_2 > \frac{t}{\eta(0)}), \sigma) \vee [\delta((\psi_1 > \frac{t}{\eta(0)}), \sigma) \wedge \langle -1, (\psi_1 \mathsf{S}_{\eta^{+1}} \psi_2 > t) \rangle] & \text{if } 0 < \frac{t}{\eta(0)} < 1, \\ \texttt{False} & \text{if } \frac{t}{\eta(0)} \geq 1, \\ \delta(((\psi_1 \mathsf{S}_\eta \psi_2)^+), \sigma) & \text{if } \frac{t}{\eta(0)} = 0. \end{cases}$$
- $\delta((\psi_1 \mathsf{S}_\eta \psi_2 < t), \sigma) =$
$$\begin{cases} \delta((\psi_2 < \frac{t}{\eta(0)}), \sigma) \wedge [\delta((\psi_1 < \frac{t}{\eta(0)}), \sigma) \vee \langle -1, (\psi_1 \mathsf{S}_{\eta^{+1}} \psi_2 < t) \rangle] & \text{if } 0 < \frac{t}{\eta(0)} \leq 1, \\ \texttt{True} & \text{if } \frac{t}{\eta(0)} > 1, \\ \texttt{False} & \text{if } \frac{t}{\eta(0)} = 0\ . \end{cases}$$

The correctness of the construction and the analysis of the blowup are similar to those in Section 3.

A.8 Correctness Proof of the Construction in Section 5.3

Consider the case of $(\psi_1 \mathsf{O}_{\eta,z} \psi_2 > t)$ (the dual case is similar). We check when this assertion holds.

First, if $z = t$, then this assertion is equivalent to $\psi_1 \mathsf{U} \psi_2 > t$ (this follows directly from the semantics). Thus, assume that $z \neq t$.

If $\tau < 0$, then $t < (1 - \eta(0))z$, so in particular, for every value of $[\![\pi^0, \psi_2]\!]$, it holds that $t < \eta(0)[\![\pi^0, \psi_2]\!] + (1 - \eta(0))z$, so the assertion is true, since the first operand in the $\sup$ is greater than t.

If $\tau \geq 1$ then $\eta(0) + (1 - \eta(0))z \leq t$, so both $\eta(0)[\![\pi^0, \psi_2]\!] + (1 - \eta(0))z \leq t$ and $\eta(0)[\![\pi^0, \psi_1]\!] + (1 - \eta(0))z \leq t$. Thus, every operand in the $\sup$ has an element less than t, so the $\sup$ cannot be greater than t, so the assertion is false.

If $0 \leq \tau < 1$, then similarly to the case of U_η - the assertion is equivalent to the following: either $\psi_2 > \tau$, or both $\psi_1 > \tau$ and $(\psi_1 \mathsf{O}_{\eta^{+1},z} \psi_2 > t)$.

[5] In addition, the atoms in the transitions in Theorem 3.4 are adjusted to the 2AWA syntax by replacing each atom q by the atom $\langle 1, q \rangle$.

It remains to show that there are still only finitely many states. Since $z \neq t$, we get that $\lim_{\eta(0)\to 0} \tau = \begin{cases} \infty & t > z \\ -\infty & t < z \end{cases}$. Thus, after a certain number of transitions, τ takes a value that is not in $[0, 1]$, in which case the next state is `True` or `False`, so the number of states reachable from $\varphi > t$ is finite.

We remark that this is not true if $z = t$, which is why we needed to treat this case separately.