Discounting in LTL

Shaull Almagor[1], Udi Boker[2], and Orna Kupferman[1]

[1] The Hebrew University, Jerusalem, Israel.
[2] The Interdisciplinary Center, Herzliya, Israel.

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 the model-checking problem 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) [9–11, 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 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 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 model-checking problem to become undecidable. In fact, model checking becomes 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, most proofs are omitted, and can be found in the full version, in the authors' home pages.

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 $\pi^i = \pi_i, \pi_{i+1}, \ldots$.

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

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 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

[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.

the eventualities, we can restrict attention to a finite resolution of satisfaction values, enabling the construction of a finite 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 1, 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 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 1.

Consider an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ. By Lemma 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 1. *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 1. 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 [14], 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 1. *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 the full version for details). For the other states, we define the transitions as follows. Let $\sigma \in 2^{AP}$.

$$\delta((\mathtt{True} > t), \sigma) = \begin{cases} \mathtt{True} & \text{if } t < 1, \\ \mathtt{False} & \text{if } t = 1. \end{cases} \qquad \delta((\mathtt{False} > t), \sigma) = \mathtt{False}.$$

$$\delta((\mathtt{True} < t), \sigma) = \mathtt{False}. \qquad \delta((\mathtt{False} < t), \sigma) = \begin{cases} \mathtt{True} & \text{if } t > 0, \\ \mathtt{False} & \text{if } t = 0. \end{cases}$$

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

$$\delta((\psi_1 \vee \psi_2 > t), \sigma) = \delta((\psi_1 > t), \sigma) \vee \delta((\psi_2 > t), \sigma).$$

$$\delta((\psi_1 \vee \psi_2 < t), \sigma) = \delta((\psi_1 < t), \sigma) \wedge \delta((\psi_2 < t), \sigma).$$

$$\delta((\neg\psi_1 > t), \sigma) = \delta((\psi_1 < 1 - t), \sigma) \qquad \delta((\neg\psi_1 < t), \sigma) = \delta((\psi_1 > 1 - t), \sigma).$$

$$\delta((\mathsf{X}\psi_1 > t), \sigma) = (\psi_1 > t). \qquad \delta((\mathsf{X}\psi_1 < t), \sigma) = (\psi_1 < t).$$

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

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

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

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

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 $\texttt{False}$. 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 1, 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 a 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 (of 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 full version.

Since $\mathcal{A}_{\varphi,v}$ is a Boolean automaton, then $L(\mathcal{A}) \neq \emptyset$ iff it accepts a lasso computation. Combining this observation with Theorem 1, we conclude with the following.

Corollary 2. *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 1 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 2. *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 1 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 1, 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 2. *For an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ *and* $v \in [0,1]$*, the* AWA $\mathcal{A}_{\varphi,v}$ *constructed in Theorem 1 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. *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 2 and Lemma 2, 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 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 2 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 the full version for the proof).

Theorem 4. *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 validity and model-checking problems are undecidable. 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 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.

Theorem 5. *The validity problem for* $\mathrm{LTL}^{\mathit{disc}\oplus}[\mathcal{D}]$ *is undecidable (for every nonempty set of discounting functions* $\mathcal{D}$*).*

The proof goes along the following lines: 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 $[\![\pi, \varphi]\!] = \frac{1}{2}$. The idea behind the construction is as follows. The computation that φ is verified with corresponds to a description of a run of $\mathcal{M}$, where every triplet $\langle l_i, \alpha, \beta\rangle$ is encoded as the string $ix^\alpha y^\beta\#$.

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

1. The first configuration in π is the initial configuration of $\mathcal{M}$, namely $\langle l_1, 0, 0\rangle$, or $1\#$ in our encoding.
2. The last configuration in π is $\langle \textsc{halt}, 0, 0\rangle$, or k in our encoding, where k is a 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 can easily be captured by an LTL formula. Property 4 utilizes the expressive power of $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$, as we now explain. The intuition behind Property 4 is the following. We compare the value of a counter before and after a command, such that the formula takes a value smaller than $\frac{1}{2}$ if a violation is encountered, and $\frac{1}{2}$ otherwise. Since the value of counters can change by at most 1, the essence of this formula is the ability to test equality of counters.

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 use formulas that "parse" the input and find successive occurrences of a counter.

Since, by considering a Kripke structure that generates all computations, it is easy to reduce the validity problem to the model-checking problem, we can conclude with the following.

Theorem 6. *The model-checking problem for* $\mathrm{LTL}^{\mathit{disc}\oplus}[\mathcal{D}]$ *is undecidable.*

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 1.

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

5 Extensions

$\mathrm{LTL}^{\mathbf{disc}}[\mathcal{D}]$ **with Past Operators** A useful augmentation of LTL is the addition of *past operators* [18]. These operators enable the specification of clearer and more succinct

formulas while preserving the PSPACE complexity of model checking. In the full version, we add *discounting-past* operators to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ and show how to perform model checking on the obtained logic. The solution goes via 2-way weak alternating automata and preserves the complexity of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$.

Weighted Systems In $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$, 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 values in $[0, 1]$. In the full version we extend the semantics of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ to *weighted Kripke structures*, whose computations assign weights in $[0, 1]$ to every atomic proposition. We solve the corresponding model-checking problem by properly extending the construction of the automaton $\mathcal{A}_{\varphi,v}$.

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. 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 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. In the full version, we show how to augment the construction of $\mathcal{A}_{\varphi,v}$ with the operator O in order to solve the model-checking problem.

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.