Discounting in LTL

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

[1] The Hebrew University, Jerusalem, Israel.
[2] IST Austria, Klosterneuburg, Austria.

Abstract. Traditional formal methods are based on a Boolean satisfaction notion – a reactive system satisfies, or not, a given specification. In recent years, there is growing need and interest in formalizing and reasoning about the quality of systems. One direction in this effort is a refinement of the "eventually" operators of temporal logics 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. A second direction is an extension of temporal logic by *propositional quality operators* that enable the specifier to grade different satisfying possibilities. 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 parameterized by a set $\mathcal{D}$ of discounting functions (e.g., linear decaying, exponential decaying, etc.) and includes, for each function $\eta \in \mathcal{D}$, a discounting-"until" operator U_η.
We solve the model-checking 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 $t \in [0, 1]$, our model-checking algorithm decides whether the satisfaction value of φ in $\mathcal{K}$ is at least t. Our automata-based algorithm copes with the infinitely many satisfaction values that φ may have by restricting attention to values that may influence the particular threshold. We show that for standard discounting functions, such as exponential decaying, the problem is PSPACE-complete – not more complex than standard LTL. We then augment $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ with propositional quality operators. Two such basic 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. 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. We further extend $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ to refer to the past, via discounting-"since" operators, to handle weighted systems, where the quantitative reasoning is combined with quantitative aspects of the system, and to allow not only discounting functions, but also functions that converge to an arbitrary value in $[0, 1]$. All these extensions do not increase the complexity of the model-checking problem.

1 Introduction

One of the main obstacles to the development of complex computerized 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 the system, decide whether the model satisfies the formula [4]. 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) [8–11, 14, 21], but is applied also with respect to Boolean systems, where it origins from the semantics of the specification formalism itself [1, 6].

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 should be a difference between positive and negative responses, or cases in which no request is issued. It may also be useful to relate the positivity of the request with the waiting time.

As the example above demonstrates, we can distinguish between two components of the quality of satisfaction. The first, to which we refer here 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. One may try to reduce "temporal quality" to "propositional quality", using the fact that an eventuality involves a repeated choice between satisfying it in the present or delaying its satisfaction to the strict future. This attempt, however, requires unboundedly many applications of the propositional choice, and is similar to a repeated use of the X ("next") operator rather than a use of eventuality operators. Clearly, the use of X is a very limited solution, as it partitions the future into finitly many zones, all of which are in the "near future", except for a single, unbounded, "far future". A more involved approach to distinguish between the "near" and "far" future includes bounded (prompt) eventualities [2, 3]. There, one distinguishes between eventualities whose waiting time is bounded and ones that have no bound.

The weakness of both approaches is not surprising – correctness of LTL is Boolean, and thus has inherent dichotomy between satisfaction and dissatisfaction. The distinction between "near" and "far", however, is not dichotomous. This suggests that in order to capture these concepts, one must extend LTL to an unbounded setting. Realizing this, researchers have suggested to augment temporal logics with *future discounting* [7]. 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 discounting-"until" (U_η) and discounting-"dual until" ($\widetilde{\mathsf{U}}_\eta$) operators, 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 [24]. The discounted setting gives rise to infinitely many satisfaction values. This poses a big algorithmic challenge, as model-checking algorithms, and in particular these 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* [20]. Unfortunately, these extensively-studied models are complicated and many problems become undecidable for them [13]. We show that for threshold problems, we can translate $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas into (Boolean) nondeterministic Büchi automata. We cope with the infinitely many possible satisfaction values by using 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 to other decision procedures, like threshold satisifiability and 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 [23]. In the area of formal verification, it was suggested in [7] 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, [7] shows how to evaluate discounted μ-calculus formulas with arbitrary precision. Formulas of LTL can be translated to the μ-calculus, thus [7] can be used in order to model-check discounted LTL formulas. The translation from LTL to the μ-calculus, however, is complicated and involves an exponential blow-up [5]. Moreover, $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ allows for arbitrary discounting functions, and our algorithm returns an exact solution to the threshold model-checking problem, which is more difficult than the approximation problem.

Closer to our work is [6], where CTL is augmented with discounting and weighted-average operators. The motivation in [6] is to introduce a logic whose semantics is not too sensitive to small perturbations in the model. Accordingly, formulas are evaluated on weighted-system 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 semantics, the expressive power of discounted CTL is limited.

Perhaps closest to our approach is [17], where a version of discounted-LTL was introduced. Semantically, there are two main differences between the logics. The first is that they use 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 [17] is a comprehensive study of the mathematical properties of the logic. Yet, they do not look into algorithmic questions relating 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 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 [6].

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 as well. Second, we add to $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ *past* operators and their discounting version (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 [16, 15]. 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 $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$

The linear temporal logic $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ generalizes LTL by introducing 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 operators $\varphi \mathsf{U}_\eta \psi$ (discounting-Until) and $\varphi \widetilde{\mathsf{U}}_\eta \psi$ (discounting-dual 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 $\eta \in \mathcal{D}$.

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

The semantics of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ is defined with respect to a *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 $\pi^i = \pi_i, \pi_{i+1}, \ldots$.

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

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 that is used.

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 $\mathsf{G}_\eta\varphi = \neg\mathsf{F}_\eta\neg\varphi$.

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 system that grants locks (e.g., for a data structure). When a process requests a grant, it is desirable that the grant is given as soon as possible. Keeping the process waiting incurs a multiplicative cost λ. We can specify this requirement by the formula $\mathsf{G}(\mathrm{req} \rightarrow \mathsf{F}_\eta \mathrm{grant})$, where $\eta(i) = \lambda^i$. Then, a system that always grants requests immediately satisfies the formula with value of 1, a system that sometime waits for k steps satisfies it with value λ^k, and a system where the waiting time can be arbitrarily long (e.g., a system that grants the i-th request after i steps) satisfies it with value 0 (even if indeed a grant is eventually given for every request).

3 $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ 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 $[\![\mathcal{K}, \varphi]\!] > v$. In this section we show how we can solve the latter, as well as other decision problems for $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$. Our solution uses the automata-theoretic approach and we first define alternating weak automata.

3.1 Alternating Weak Automata

Given an alphabet Σ, an *infinite word over* Σ is an infinite sequence $w = \sigma_0 \cdot \sigma_1 \cdots \sigma_2 \cdots$ of letters in Σ. 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 infinite DAGs (directed acyclic graphs). A run of $\mathcal{A}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots$ is an 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 $\bigwedge_{q \in Q_l} \delta(q, \sigma_l)$.
- $E \subseteq \bigcup_{l \geq 0} (Q_l \times \{l\}) \times (Q_{l+1} \times \{l+1\})$ is such that $E(\langle q, l \rangle, \langle q', l+1 \rangle)$ iff $Q_{l+1} \setminus \{q'\}$ does not satisfy $\delta(q, \sigma_l)$.

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. For a set $S \subseteq Q$, a node $\langle q, i \rangle \in V$ is an S-node if $q \in S$. 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 α-nodes. 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 AWW, 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 $q \in Q_i$ and $s \in Q_j$ 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 a AWW eventually gets trapped in a set Q_i and is accepting iff this set is accepting. A useful property of AWW is that it is very easy to complement the language of a given AWW $\mathcal{A}$: we only have to dualize its transitions (that is, swap disjunctions with conjunctions, and trues with falses) and dualize its acceptance condition (that is, swap accepting and rejecting sets).

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

Our model-checking algorithm is based on translating an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula to an AWW.

Before describing the construction of the AWW, we need the following lemma, which reduces satisfaction values in $\{0, 1\}$ of an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula to Boolean satisfaction of LTL formulas. 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 for every computation* π *it holds that* $\pi \models \varphi^+$ *iff* $[\![\pi, \varphi]\!] > 0$ *and* $\pi \models \varphi^{<1}$ *iff* $[\![\pi, \varphi]\!] < 1$.

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 the atomic propositions 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^+, where θ is a subformula of φ.
3. ψ is of the form $\theta_1 \mathsf{U}_{\eta^{+k}} \theta_2$ or $\theta_1 \widetilde{\mathsf{U}}_{\eta^{+k}} \theta_2$ for $k \in \mathbb{N}$, where $\theta_1 \mathsf{U}_\eta \theta_2$ or $\theta_1 \widetilde{\mathsf{U}}_\eta \theta_2$ 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* AWW $\mathcal{A}_{\varphi,v}$ *such that for every computation* π *it holds that* $\mathcal{A}_{\varphi,v}$ *accepts* π *iff* $[\![\pi, \varphi]\!] > v$.

Proof. Let $v \in [0,1]$, we construct an AWW $\mathcal{A}_{\varphi,v} = \langle Q, 2^{AP}, Q_0, \delta, \alpha \rangle$ such that for every computation π it holds that $\mathcal{A}_{\varphi,v}$ accepts π iff $[\![\pi, \varphi]\!] > v$.

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 above. Let S be the set of Type-1 and Type-2 states for every $\psi \in xcl(\varphi)$ and for every threshold $t \in [0,1]$, then Q is a subset of S which is 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. Let $\sigma \in 2^{AP}$, for Type-2 states, the transitions are as in the standard translation from LTL to AWW [22]. For the rest of the states, we define the transitions as follows.

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

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

- $\delta((\psi_1 \mathsf{U}_\eta \psi_2 < t), \sigma) =$

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

– $\delta((\psi_1 \widetilde{\mathsf{U}}_\eta \psi_2 > t), \sigma) =$

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

– $\delta((\psi_1 \widetilde{\mathsf{U}}_\eta \psi_2 < t), \sigma) =$

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

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

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

The initial state of $\mathcal{A}_{\varphi,v}$ is $(\varphi > v)$.

The accepting states are those whose formula is 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 AWW (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 a WAA. 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 (of Type-1 or Type-3) is not allowed, as they involve eventualities that are not satisfied in the threshold promised for them.

This concludes the definition of $\mathcal{A}_{\varphi,v}$. It is not hard to see that it is indeed an AWW, with each state consisting a singleton set. Observe, however, that the construction as described above is infinite (indeed, uncountable). Yet, 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 provided in the appendix.

3.3 From $\mathcal{A}_{\varphi,v}$ to an NBW

Every AWW can be translated to an equivalent nondeterministic Büchi automaton (NBW, for short) [19]. In this section we analyze the size of the NBW obtained from the AWW $\mathcal{A}_{\varphi,v}$.

It is clear from the construction in Section 3.2 that the number of states in the AWW depends on the discounting functions in $\mathcal{D}$ as well as on the threshold v. Intuitively, the faster the discounting tends to 0, the less states there will be. Therefore, a concrete complexity analysis requires prior knowledge on $\mathcal{D}$.

We start with the following general result that does not depend on $\mathcal{D}$. In Section 3.4, we focus in exponential discounting. The idea behind our complexity analysis is as follows. Translating an AWW to an NBW involves alternation removal, which proceeds

by keeping track of all entire levels in a run-DAG. Thus, a run of the NBW corresponds to a sequence of subsets of Q. The key to the proof is showing that the number of such subsets is only $|Q|^{O(|\varphi|)}$. 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 NBW to have only minimal sets, which bounds their number as follows.

Lemma 2. *The* AWW $\mathcal{A}_{\varphi,v}$ *constructed in Theorem 1 can be translated to an* NBW *with* $|Q|^{O(|\varphi|)}$ *states.*

3.4 Exponential Discounting

In this section, we analyze the size of the AWW for a specific class of discounting functions, namely *exponential discounting*. This class is perhaps the most common class of discounting functions, as it describes what happens in many natural processes (e.g., Markov chains, capacitor charge) [7, 23].

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 $\mathcal{D} = \{\exp_\lambda : \lambda \in (0,1) \cap \mathbb{Q}\}$, and consider the logic $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$.

For an $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ formula φ we define the set $F(\varphi)$ to be $\{\lambda_1, ..., \lambda_k :$ the operators $\mathsf{U}_{\exp_\lambda}$ or $\widetilde{\mathsf{U}}_{\exp_\lambda}$ appear in $\varphi\}$. Let $|\varphi|$ be the number of subformulas of φ, and let $|\langle\varphi\rangle|$ be the length of the description of φ. That is, $|\langle\varphi\rangle|$ includes the length, in bits, of describing $F(\varphi)$.

Theorem 2. *Given an* $\mathrm{LTL}^{disc}[\mathcal{D}]$ *formula* φ *and a threshold* $v \in [0,1] \cap \mathbb{Q}$*, there exists an* AWW $\mathcal{A}_{\varphi,v}$ *such that for every computation* π *it holds that* $\mathcal{A}_{\varphi,v}$ *accepts* π *iff* $[\![\pi, \varphi]\!] > v$*. Furthermore, the number of states of* $\mathcal{A}_{\varphi,v}$ *is single-exponential in* $|\langle\varphi\rangle|$ *and 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 AWW 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.

From Theorem 2 and Lemma 2 we conclude that the size of an NBW corresponding to φ and v is also single-exponential in $|\langle\varphi\rangle|$ and the description of v.

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

The AWW $\mathcal{A}_{\varphi,v}$ constructed as per Section 3.2 accepts a path π iff $[\![\varphi, \pi]\!] > v$. By dualyzing (and hence, complementing) $\mathcal{A}_{\varphi,v}$, we can obtain an AWW $\widetilde{\mathcal{A}}_{\varphi,v}$ such that $L(\widetilde{\mathcal{A}}_{\varphi,v}) = \{\pi : [\![\pi, \varphi]\!] \leq v\}$.

Consider a Kripke structure $\mathcal{K}$. By checking the emptiness of the intersection of $\mathcal{K}$ with $\widetilde{\mathcal{A}}_{\varphi,v}$, we can solve the threshold model-checking problem. Using Theorem 1 and

this observation on the formula $\neg\varphi$ and the threshold $1 - v$, we can also decide whether there exist a path π such that $[\![\pi, \varphi]\!] \geq v$, and whether there exists a path π such that $[\![\pi, \varphi]\!] < v$. Finally, by adjusting the initial states of $\mathcal{A}_{\varphi,v}$ and its complement automaton, we can decide whether there exists a path π such that $[\![\pi, \varphi]\!] \in I$ for every (continuous) interval $I \subseteq [0, 1]$, either closed, open, or half-open.

Threshold satisfiability and synthesis: The NBW obtained in Lemma 2 can be used to solve the threshold variant of additional classical decision problems for temporal logics. In particular, 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 $\sim \in \{\leq, <, \geq, >\}$, and return such a computation when the answer is positive. Also, assuming a partition of the atomic propositions in φ to input and output signals, we can use $\mathcal{A}_{\varphi,v}$ in order to generate an I/O-transducer all of whose computations π satisfy $[\![\pi, \varphi]\!] \sim v$.

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 are propositional quality operators.

4.1 Adding the Average Operator

A well motivated extension is the introduction of the average operator $\oplus$, with the semantics $[\![\pi, \varphi \oplus \psi]\!] = \frac{[\![\pi,\varphi]\!]+[\![\pi,\psi]\!]}{2}$. Our work in [1] proves that extending LTL by this operator, in fact also a more general one, where average is weighted, 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 $\pi \in AP^\omega$.

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 [18]. Given $\mathcal{M}$, deciding whether $\mathcal{M}$ halts with both counters having value 0 is also undecidable. Indeed, given a counter machine $\mathcal{M}$, we can replace every HALT command with code that clears the counters before halting. Thus, the halting problem can be reduced to the latter problem, termed the 0-*halting problem*.

Theorem 3. *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 $[\![\pi, \varphi]\!] = 1$. The idea behind the construction is as follows. The computation that φ reads corresponds to a description of a run of $\mathcal{M}$, where every triplet $\langle l_i, \alpha, \beta\rangle$ is encoded as the string $ix^{\alpha}y^{\beta}\#$.

Example 2. Consider the following machine $\mathcal{M}$:

l_1: INC(x)	l_4: DEC(x)
l_2: IF x=0 GOTO l_6 ELSE GOTO l_3	l_5: GOTO (l_2)
l_3: INC(y)	l_6: HALT

The command sequence that represents the run of this machine is

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

and the encoding of it as a computation is $1\#2x\#3x\#4xy\#5y\#2y\#6y\#$.

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 $\mathcal{M}$, namely $\langle l_1, 0, 0\rangle$, or $1\#$ in our encoding.
2. The last configuration in π is $\langle \mathrm{HALT}, 0, 0\rangle$, 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 are 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 need to compare the value of a counter before and after a command, such that the formula takes a negative value if a violation is encountered, and a positive value 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

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

and observe that the latter is 1 iff $i = j$ (and is less than 1 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 provide some technical formulas to "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 4. *The model-checking problem for* $\mathrm{LTL}^{disc\oplus}[\mathcal{D}]$ *is undecidable.*

4.2 Adding Unary Multiplication Operators

As we have seen in Section 4.1, adding the operator $\oplus$ to the logic 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 $\mathrm{LTL}^{\text{disc}}[\mathcal{D}]$ 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.

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

5 Extensions

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

One of the well-known augmentations of LTL is the addition of *past operators* [16]. 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 $\mathrm{LTL}^{\text{disc}}[\mathcal{D}]$, 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 $\mathrm{LTL}^{\text{disc}}[\mathcal{D}]$, denoting the extended logic $\mathrm{PLTL}^{\text{disc}}[\mathcal{D}]$ with the following semantics. For $\mathrm{PLTL}^{\text{disc}}[\mathcal{D}]$ 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 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.

We now wish to translate $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ formulas to automata. In the Boolean case, there are two approaches to this: one is to use 2-way weak alternating automata (2AWW), and the other goes through nondeterministic automata. To comply with our use of AWW in Theorem 1, we work with 2AWW.

A 2AWW is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$ where Σ, Q, q_0, F are as in AWW. 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, so we need to extend Lemma 1 and generate Boolean PLTL formulas for satisfaction values in $\{0, 1\}$. It is not hard to see that defining $(\psi_1 \mathsf{S}_\eta \psi_2)^+ = ({\psi_1}^+) \mathsf{S} ({\psi_2}^+)$ and $(\mathsf{Y}\psi)^+ = \mathsf{Y}(\psi^+)$ works.

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

- $\delta((\mathsf{Y}\psi > t), \sigma) = \langle -1, (\psi > t) \rangle$ - $\delta((\mathsf{Y}\psi < t), \sigma) = \langle -1, (\psi < t) \rangle$.
- $\delta((\psi_1 \mathsf{S} \psi_2 > t), \sigma) = \delta((\psi_2 > t), \sigma) \vee (\delta((\psi_1 > t), \sigma) \wedge \langle -1, (\psi_1 \mathsf{S} \psi_2 > t) \rangle)$.
- $\delta((\psi_1 \mathsf{S} \psi_2 < t), \sigma) = \delta((\psi_2 < t), \sigma) \wedge (\delta((\psi_1 < t), \sigma) \vee \langle -1, (\psi_1 \mathsf{S} \psi_2 < t) \rangle)$.
- $\delta((\psi_1 \mathsf{S}_\eta \psi_2 > t), \sigma) =$

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

- $\delta((\psi_1 \mathsf{S}_\eta \psi_2 < t), \sigma) =$

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

The operator $\widetilde{\mathsf{S}}_\eta$ (dual-since), with the semantics $[\![\pi^i, \varphi \widetilde{\mathsf{S}}_\eta \psi]\!] = \min_{0 \leq j \leq i} \{\max \{\eta(i-j)\, [\![\pi^j, \psi]\!], \max_{j<k\leq i} \{\eta(i-k) [\![\pi^k, \varphi]\!]\}\}\}$, can be added in a similar manner.

The correctness of the construction, the analysis of the blow up, and the use in decision procedures are similar to these in the Section 3. In particular, it follows that the model-checking problem for $\mathrm{PLTL}^{\mathrm{disc}}[\mathcal{D}]$ is in PSPACE.

5.2 Weighted Systems

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

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

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, and want the discounting to tend to $\frac{3}{4}$. For example, when a system is constantly under maintenance and we know that components are likely to function slightly better in the future.

To capture this notion, we define the operators $\mathsf{O}_{\eta,z}$ and $\widetilde{\mathsf{O}}_{\eta,z}$ for $\eta \in \mathcal{D}$ and $z \in [0,1]$ with the following semantics.

$$[\![\pi, \varphi \mathsf{O}_{\eta,z} \psi]\!] = \sup_{i \geq 0} \{\min\{\eta(i)[\![\pi^i, \psi]\!] + (1-\eta(i))z, \min_{0 \leq j < i} \eta(j)[\![\pi^j, \varphi]\!] + (1-\eta(j))z\}\}.$$

$$[\![\pi, \varphi \widetilde{\mathsf{O}}_{\eta,z} \psi]\!] = \inf_{i \geq 0} \{\max\{\eta(i)[\![\pi^i, \psi]\!] + (1-\eta(i))z, \max_{0 \leq j < i} \eta(j)[\![\pi^j, \varphi]\!] + (1-\eta(j))z\}\}.$$

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.

Example 3. 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 $\varphi = \texttt{True}\mathsf{O}_{\eta,\frac{1}{2}}\mathrm{defrag}$. 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.

One may verify that $\neg(\varphi \mathsf{O}_{\eta,z} \psi) \equiv (\neg\varphi)\widetilde{\mathsf{O}}_{\eta,1-z}(\neg\psi)$, so it is enough to include $O_{\eta,z}$ in the logic. Also, clearly $\varphi \mathsf{U}_\eta \psi \equiv \varphi \mathsf{O}_{\eta,0} \psi$ and $\varphi \widetilde{\mathsf{U}}_\eta \psi = \varphi \widetilde{\mathsf{O}}_{\eta,1} \psi$.

We claim that Theorem 1 holds under the extension of $\mathrm{LTL}^{\mathrm{disc}}[\mathcal{D}]$ with the operator O. Indeed, the construction of the AWW 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

- $\delta((\psi_1 \mathsf{O}_{\eta,z} \psi_2 > t), \sigma) =$

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

– $\delta((\psi_1 \mathsf{O}_{\eta,z} \psi_2 < t), \sigma) =$

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

References

1. S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. *Submitted*.
2. S. Almagor, Y. Hirshfeld, and O. Kupferman. Promptness in omega-regular automata. In *8th ATVA*, volume 6252, pages 22–36, 2010.
3. M. Bojańczyk and T. Colcombet. Bounds in ω-regularity. In *21st LICS*, pages 285–296, 2006.
4. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
5. M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *TCS*, 126:77–96, 1994.
6. L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *TCS*, 345(1):139–170, 2005.
7. L. de Alfaro, T. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *30th ICALP, LNCS 2719*, pages 1022–1037, 2003.
8. M. Droste, W. Kuich, and G. Rahonis. Multi-valued mso logics over words and trees. *Fundamenta Informatica*, 84(3–4):305–327, 2008.
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. 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.
14. M. Kwiatkowska. Quantitative verification: models techniques and tools. In *ESEC/SIGSOFT FSE*, pages 449–458, 2007.
15. F. Laroussinie and P. Schnoebelen. A hierarchy of temporal logics with past. In *11th STACS*, pages 47–58, 1994.
16. O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Logics of Programs, LNCS 193*, pages 196–218. Springer, 1985.
17. E. Mandrali. Weighted LTL with discounting. In *CIAA, LNCS 7381*, pages 353–360, 2012.
18. M. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1 edition, 1967.
19. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *TCS*, 32:321–330, 1984.
20. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
21. 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.
22. D. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *3rd LICS*, pages 422–427, 1988.
23. L. Shapley. Stochastic games. In *Proc. of the National Academy of Science*, volume 39, 1953.
24. 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 1

We construct φ^+ and $\varphi^{<1}$ by induction on the structure of φ.

- 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 $\psi_1 \vee \psi_2$, $\mathsf{X}\psi_1$, or $\psi_1 \mathsf{U} \psi_2$, then φ^+ is ${\psi_1}^+ \vee {\psi_2}^+$, $\mathsf{X}{\psi_1}^+$, ${\psi_1}^+ \mathsf{U} {\psi_2}^+$, respectively. Dually, $\varphi^{<1}$ is $\neg(\varphi^+)$ (which is sound since we already defined φ^+ for this case). Again, correctness is immediate.
- If $\varphi = \neg\psi$ we define $\varphi^+ = \psi^{<1}$ 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, for every computation π we have that $[\![\pi, \varphi]\!] > 0$ iff there exists $i \in \mathbb{N}$ such that $\eta(i)[\![\pi^i, \psi_2]\!] > 0$ and for every $0 \leq j < i$ it holds that $\eta(j)[\![\pi^j, \psi_1]\!] > 0$, which happens iff there exists $i \in \mathbb{N}$ such that $\pi^i \models {\psi_2}^+$ and for every $0 \leq j < i$ it holds that $\pi^j \models {\psi_1}^+$, which holds iff $\pi \models {\psi_1}^+ \mathsf{U} {\psi_2}^+$.
- If $\varphi = \psi_1 \widetilde{\mathsf{U}}_\eta \psi_2$ for $\eta \in \mathcal{D}$, then $\varphi^+ = {\psi_2}^+ \mathsf{U} {\psi_1}^+$. Indeed, the value of $\psi_1 \widetilde{\mathsf{U}}_\eta \psi_2$ is greater than 0 iff ψ_1 actually holds at some point, and until then the satisfaction value of ψ_2 is positive.

A.2 Continuation of the Proof of Theorem 1

We continue the proof that is given in the main text, showing that the constructed AWW $\mathcal{A}_{\varphi_v}$ 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_s, or s is a Type-2 state, or both ψ_t 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 $\texttt{True}$) 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 1 and from the construction of an AWW 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., `True`, `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 1 and the correctness of the standard translation of LTL to AWW, it remains to prove that for every path π, π is accepted from state $(\psi > v)$ (resp. $(\psi < v)$) iff $[\![\pi, \psi]\!] > v$ (resp. $[\![\pi, \psi]\!] < v$).

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

A.3 Proof of Lemma 2

We start by defining *generalized Büchi automata*. An NGBW is $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, \alpha\rangle$, where Q, Σ, δ, Q_0 are as in NBW. 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 AWW $\mathcal{A}$ obtained from φ using the construction of Section 3.2.

In the translations of AWW to NBW using the method of [12], the AWW is translated to an NGBW whose states are the subset-construction of the AWW. 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 NGBW corresponds to a conjunction of states of the AWW. 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 1, 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 NGBW to states that are conjunctions of states from the AWW, such that no discounting operator appears with two different "offsets".

Further observe that by the construction of the AWW, 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 NGBW 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 NGBW is $|\mathcal{A}|^{O(|\varphi|)}$.

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

A.4 Proof of Theorem 2

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

Recall that the "interesting" states in $\mathcal{A}$ are those of the form $\psi_1 \mathsf{U}_{\eta^{+i}} \psi_2$ (resp. $\psi_1 \widetilde{\mathsf{U}}_{\eta^{+i}} \psi_2$). 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 (and similarly for $\widetilde{\mathsf{U}}$, and for $> t$). 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

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

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

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

[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

$$\delta(\not\vdash\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < t), \sigma) =$$
$$\begin{cases} \delta((\psi_2 < t), \sigma) \vee [\delta((\psi_1 < t), \sigma) \wedge (\psi_1 \mathsf{U}_{\exp_\lambda} \psi_2 < \frac{t}{\lambda})] & \text{if } 0 < t \leq 1, \\ \texttt{True} & \text{if } t > 1, \\ \texttt{False} & \text{if } t = 0. \end{cases}$$

The correctness and finiteness of the construction follows from Theorem 1, 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 AWW 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

$$\log(1 - s) = \log(1 - \frac{t}{\prod_{i=1}^{\ell} \lambda_i}) = \log(\prod_{i=1}^{\ell} \lambda_i - t) - \log(\prod_{i=1}^{\ell} \lambda_i) > \log(\prod_{i=1}^{\ell} \lambda_i - t)$$

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

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.5 Proof of Theorem 3

We show a reduction from the 0-halting problem for two-counter machines to the following problem: given an $\text{LTL}^{\text{disc}\oplus}[\mathcal{D}]$ formula φ over the atomic propositions AP, whether there exists a path $\pi \in AP^{\omega}$ such that $[\![\pi, \varphi]\!] = 1$. We dub this the 1-*co-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 a 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 a 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 a machine $\mathcal{M}$ 0-halts if its run 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 $[\![\pi, \varphi]\!] = 1$. The idea behind the construction is as follows. The computation that φ reads corresponds to a description of a run of $\mathcal{M}$, where every triplet $\langle l_i, \alpha, \beta\rangle$ is encoded as the string $ix^\alpha y^\beta\#$.

Example 4. 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: HALT

The command sequence that represents the run of this machine is

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

and the encoding of it as a computation is

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

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 $\langle$HALT$, 0, 0\rangle$ (or k in our encoding, where k 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 captured by an LTL formula. Property 4 utilizes the expressive power of $\mathrm{LTL}^{\mathrm{disc}\oplus}[\mathcal{D}]$, 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 negative value if a violation is encountered, and a positive value 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

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

and observe that the latter is 1 iff $i = j$ (and less than 1 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,

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

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

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

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

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

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 $\psi_i := \mathsf{X}waitfor(i+1)$. [5]
- 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

$$CheckCmds := \mathsf{G} \bigwedge_{i \in \{1,...,n\}} \psi_i$$

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

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

$$orHalt := \bigvee_{i\in I_{\text{DEC}(x)}} i \wedge \mathsf{X}(x \wedge \mathsf{X}(\# \wedge \mathsf{X} \bigvee_{k\in I_{\text{HALT}}} k))\vee$$

$$\bigvee_{i\in I_{\text{DEC}(y)}} i \wedge \mathsf{X}(y \wedge \mathsf{X}(\# \wedge \mathsf{X} \bigvee_{k\in I_{\text{HALT}}} k)) \vee \bigvee_{i\in\{1,\ldots,n\}} i \wedge \mathsf{X}(\# \wedge \mathsf{X} \bigvee_{k\in I_{\text{HALT}}} k)$$

Intuitively, this formula simply states that it is the last step, and the counters behave the way they should.

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 following formulas for these tests.

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

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

- If $l_i \in \{\text{GOTO } l_j, \text{IF } c\text{=0 GOTO } l_j \text{ ELSE GOTO } l_k : c \in \{x, y\}\}$, we need to make sure the value of the counters do not change. We define

$$\xi_i := i \rightarrow (X(comp(x,=) \wedge comp(y,=)) \vee orHalt.$$

- If $l_i = \text{INC}(x)$, we need to make sure that x increases and y does not change. We define

$$\xi_i := i \rightarrow (X(comp(x,+1) \wedge comp(y,=)) \vee orHalt.$$

- If $l_i = \text{INC}(y)$, we define

$$\xi_i := i \rightarrow (X(comp(x,=) \wedge comp(y,+1)) \vee orHalt.$$

– If $l_i = \text{DEC}(x)$, we define

$$\xi_i := i \rightarrow (X(comp(x, -1) \wedge comp(y, =)) \vee orHalt.$$

– If $l_i = \text{DEC}(y)$, we define

$$\xi_i := i \rightarrow (X(comp(x, =) \wedge comp(y, -1)) \vee orHalt.$$

– If $l_i = \text{HALT}$ we do not need additional constraints, due to $CheckFinal$. Thus we have $\psi_i = \texttt{True}$.

Finally, we define

$$CheckCounters := \mathsf{G} \bigwedge_{i \in \{1,\ldots,n\}} \xi_i$$

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 $[\![\pi, \varphi]\!] = 1$.

Finally, we reduce the 1-co-validity problem to the complement of the validity problem: given a formula φ, the reduction outputs $\neg(\varphi)$. Now, there exists a computation π such that $[\![\varphi, \pi]\!] = 1$ iff there exists a computation π such that $[\![\pi, \neg\varphi]\!] = 0$, iff not all the computations of $\neg\varphi$ give strictly positive satisfaction value. Thus, φ is 1-co-valid iff $\neg\varphi$ is not valid for threshold 0. Thus, the validity problem is undecidable.

A.6 Correctness Proof of the Construction in Section 5.3

Consider the case of $(\psi_1 \mathsf{O}_{\eta,z} \psi_2 > t)$. 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 $(\psi_1 \mathsf{O}_{\eta^{+1},z} \psi_2 > t)$.

The case of $\widetilde{\mathsf{O}}$ is proved similarly.

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.