A Counting Semantics for Monitoring LTL Specifications over Finite Traces

Ezio Bartocci[1], Roderick Bloem[2], Dejan Nickovic[3] and Franz Roeck[2]

[1] TU Wien, Vienna, Austria
[2] Graz University of Technology, Graz, Austria
[3] Austrian Institute of Technology GmbH, Vienna, Austria

Abstract. We consider the problem of monitoring a Linear Time Logic (LTL) specification that is defined on infinite paths, over finite traces. For example, we may need to draw a verdict on whether the system satisfies or violates the property "p holds infinitely often." The problem is that there is always a continuation of a finite trace that satisfies the property and a different continuation that violates it.
We propose a two-step approach to address this problem. First, we introduce a counting semantics that computes the number of steps to witness the satisfaction or violation of a formula for each position in the trace. Second, we use this information to make a prediction on inconclusive suffixes. In particular, we consider a *good* suffix to be one that is shorter than the longest witness for a satisfaction, and a *bad* suffix to be shorter than or equal to the longest witness for a violation. Based on this assumption, we provide a verdict assessing whether a continuation of the execution on the same system will presumably satisfy or violate the property.

1 Introduction

Alice is a verification engineer and she is presented with a new exciting and complex design. The requirements document coming with the design already incorporates functional requirements formalized in Linear Temporal Logic (LTL) [13]. The design contains features that are very challenging for exhaustive verification and her favorite model checking tool does not terminate in reasonable time.

Runtime Verification. Alice decides to tackle this problem using runtime verification (RV) [3], a light, yet rigorous verification method. RV drops the exhaustiveness of model checking and analyzes individual traces generated by the system. Thus, it scales much better to the industrial-size designs. RV enables automatic generation of monitors from formalized requirements and thus provides a systematic way to check if the system traces satisfy (violate) the specification.

This work was partially supported by the European Union (IMMORTAL project, grant no. 644905), the Austrian FWF (National Research Network RiSE/SHiNE S11405-N23 and S11406-N23), the SeCludE project (funded by UnivPM) and the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking under Grant Agreement no. 692455. This Joint Undertaking receives support from the European Unions HORIZON 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

Motivating Example. In particular, Alice considers the following specification:

$$\psi \equiv \mathsf{G}(\mathsf{request} \rightarrow \mathsf{F}\ \mathsf{grant})$$

This LTL formula specifies that every request coming from the environment must be granted by the design in some finite (but unbounded) future. Alice realizes that she is trying to check a *liveness* property over a set of *finite* traces. She looks closer at the executions and identifies the two interesting examples trace τ_1 and trace τ_2, depicted in Table 1.

Table 1: Unbounded response property example.

trace	time	1	2	3	4	5	6	7
τ_1	request	⊤	–	–	⊤	–	–	–
	grant	–	–	⊤	–	–	–	–
τ_2	request	⊤	–	–	⊤	–	–	⊤
	grant	–	–	⊤	–	–	⊤	–

We use "–" instead of "⊥" to improve the trace readability.

The monitoring tool reports that both τ_1 and τ_2 presumably violate the unbounded response property. This verdict is against Alice's intuition. The evaluation of trace τ_1 seems right to her – the request at Cycle 1 is followed by a grant at Cycle 3, however the request at Cycle 4 is never granted during that execution. There are good reasons to suspect a bug in the design. Then she looks at τ_2 and observes that after every request the grant is given exactly after 2 cycles. It is true that the last request at Cycle 7 is not followed by a grant, but this seems to happen because the execution ends at that cycle – the past trace observations give reason to think that this request would be followed by a grant in cycle 9 if the execution was continued. Thus, Alice is not satisfied by the second verdict.

Alice looks closer at the way that the LTL property is evaluated over finite traces. She finds out that temporal operators are given *strength* – *eventually* and *until* are declared as *strong* operators, while *always* and *weak until* are defined to be *weak* [9]. A strong temporal operator requires all outstanding obligations to be met before the end of the trace. In contrast, a weak temporal operator must not witness any outstanding obligation violation before the end of the trace. Under this interpretation, both τ_1 and τ_2 violate the unbounded response property.

Alice explores another popular approach to evaluate future temporal properties over finite traces – the 3-valued semantics for LTL [4]. In this setting, the Boolean set of verdicts is extended with a third unknown (or maybe) value. A finite trace satisfies (violates) the 3-valued LTL formula if and only if all the infinite extensions of the trace satisfy (violate) the same LTL formula under its classical interpretation. In all other cases, we say that the satisfaction of the formula by the trace is unknown. Alice applies the 3-valued interpretation of LTL on the traces τ_1 and τ_2 to evaluate the unbounded response property. In both situations, she ends up with the unknown verdict. Once again, this is not what she expects and it does not meet her intuition about the satisfaction of the formula by the observed traces.

Alice desires a semantics that evaluates LTL properties on finite traces by taking previous observations into account.

Contributions. In this paper, we study the problem of LTL evaluation over finite traces encountered by Alice and propose a solution. We introduce a new counting semantics for LTL that takes into account the intuition illustrated by the

example from Table 1. This semantics computes for every position of a trace two values – the distances to the nearest satisfaction and violation of the co-safety, respectively safety, part of the specification. We use this quantitative information to make *predictions* about the (infinite) suffixes of the finite observations. We infer from these values the maximum time that we expect for a future obligation to be fulfilled. We compare it to the value that we have for an open obligation at the end of the trace. If the latter is greater (smaller) than the expected maximum value, we have a good indication of a *presumed violation (satisfaction)* that we report to the user. In particular, our approach will indicate that τ_1 is likely to violate the specification and should be further inspected. In contrast, it will evaluate that τ_2 most likely satisfies the unbounded response property.

Organization of the paper. The rest of the paper is organized as follows. We discuss the related work in Section 2 and we provide the preliminaries in Section 3. In Section 4 we present our new counting semantics for LTL, while in Section 5 we show how to make *predictions* about the (infinite) suffixes of the finite observations. Section 6 shows the application of our approach to some examples. Finally in Section 7 we draw our conclusions.

2 Related Work

The finitary interpretation of LTL was first considered in [11], where the authors propose to enrich the logic with the *weak* next operator that is dual to the (strong) next operator defined on infinite traces. While the strong next requires the existence of a next state, the weak next trivially evaluates to true at the end of the trace. In [9], the authors propose a more semantic approach with *weak* and *strong* views for evaluating future obligations at the end of the trace. In essence the empty word satisfies (violates) every formula according to the weak (strong) view. These two approaches result in the violation of the specification ψ by both traces τ_1 and τ_2.

The authors in [4] propose a 3-valued finitary LTL interpretation of LTL, in which the set $\{\mathsf{true}, \mathsf{false}\}$ of verdicts is extended with a third $\mathsf{inconclusive}$ verdict. According to the 3-valued LTL, a finite trace satisfies (violates) a specification iff all its infinite extensions satisfy (violate) the same property under the classical LTL interpretation. Otherwise, it evaluates to $\mathsf{inconclusive}$. The main disadvantage of the 3-valued semantics is the dominance of the $\mathsf{inconclusive}$ verdict in the evaluation of many interesting LTL formulas. In fact, both τ_1 and τ_2 from Table 1 evaluate to $\mathsf{inconclusive}$ against the unbounded response specification ψ.

In [5], the authors combine the weak and strong operators with the 3-valued semantics to refine the $\mathsf{inconclusive}$ with $\{\mathsf{presumably\ true}, \mathsf{presumably\ false}\}$. The strength of the remaining future obligation dictates the presumable verdict. The authors in [12] propose a finitary semantics for each of the LTL (safety, liveness, persistence and recurrence) hierarchy classes that asymptotically converges to the infinite traces semantics of the logic. In these two works, the specification ψ also evaluates to the same verdict for both the traces τ_1 and τ_2.

To summarize, none of the related work handles the unbounded response example from Table 1 in a satisfactory manner. This is due to the fact that these approaches decide about the verdict based on the specification and its remaining future obligations at the end of the trace. In contrast, we propose an approach in which the past observations within the trace are used to predict the future and derive the appropriate verdict. In particular, the application of our semantics for the evaluation of ψ over τ_1 and τ_2 results in presumably true and presumably false verdicts.

In [17], the authors propose another predictive semantics for LTL. In essence, this work assumes that at every point in time the monitor is able to precisely predict a segment of the trace that it has not observed yet and produce its outcome accordingly. In order to ensure such predictive power, this approach requires a white-box setting in which instrumentation and some form of static analysis of the systems are needed in order to foresee in advance the upcoming observations. This is in contrast to our work, in which the monitor remains a passive participant and predicts its verdict only based on the past observations.

In a different research thread [15], the authors introduce the notion of *monitorable* specifications that can be positively or negatively determined by a finite trace. The monitorability of LTL is further studied in [14, 6]. This classification of specifications is orthogonal to our work. We focus on providing a sensible evaluation to all LTL properties, including the non-monitorable ones (e.g., $\mathsf{GF}\, p$).

We also mention the recent work on statistical model checking for LTL [8]. In this work, the authors assume a gray-box setting, where the system-under-test (SUT) is a Markov chain with the known minimum transition probability. This is in contrast to our work, in which we passively observe existing finite traces generated by the SUT, i.e., we have a blackbox setting.

In [1], the authors propose extending LTL with a discounting operator and study the properties of the augmented logic. The LTL specification formalism is extended with path-accumulation assertions in [7]. These LTL extensions are motivated by the need for a more quantitative and refined analysis of the systems. In our work, the motivation for the counting semantics is quite different. We use the quantitative information that we collect during the execution of the trace to predict the future behavior of the system and thus improve the quality of the monitoring verdict.

3 Preliminaries

We first introduce *traces* and Linear Temporal Logic (LTL) that we interpret over 3-valued semantics.

Definition 1 (Trace). *Let P a finite set of* propositions *and let $\Pi = 2^P$. A (finite or infinite)* trace π *is a sequence $\pi_1, \pi_2, \ldots \in \Pi^* \cup \Pi^\omega$. We denote by $|\pi| \in \mathbb{N} \cup \{\infty\}$ the* length *of π. We denote by $\pi \cdot \pi'$ the concatenation of $\pi \in \Pi^*$ and $\pi' \in \Pi^* \cup \Pi^\omega$.*

Definition 2 (Linear Temporal Logic). *In this paper, we consider linear temporal logic (LTL) and we define its syntax by the grammar:*

$$\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}\, \phi_2,$$

where $p \in P$. We denote by Φ the set of all LTL formulas.

From the basic definition we can derive other standard Boolean and temporal operators as follows:

$$\top = p \vee \neg p,\ \bot = \neg\top,\ \phi \wedge \psi = \neg(\neg\phi \vee \neg\psi),\ \mathsf{F}\,\phi = \top \,\mathsf{U}\, \phi,\ \mathsf{G}\,\phi = \neg\,\mathsf{F}\,\neg\phi$$

Let $\pi \in \Pi^\omega$ be an infinite trace and ϕ an LTL formula. The satisfaction relation $(\pi, i) \models \phi$ is defined inductively as follows

$$\begin{array}{ll}
(\pi,i) \models p & \text{iff } p \in \pi_i,\\
(\pi,i) \models \neg\phi & \text{iff } (\pi,i) \not\models \phi,\\
(\pi,i) \models \phi_1 \vee \phi_2 & \text{iff } (\pi,i) \models \phi_1 \text{ or } (\pi,i) \models \phi_2,\\
(\pi,i) \models \mathsf{X}\,\phi & \text{iff } (\pi,i+1) \models \phi,\\
(\pi,i) \models \phi_1 \,\mathsf{U}\, \phi_2 & \text{iff } \exists j \geq i \text{ s.t. } (\pi,j) \models \phi_2 \text{ and } \forall i \leq k < j, (\pi,k) \models \phi_1.
\end{array}$$

We now recall the 3-valued semantics from [4]. We denote by $[\pi \models_3 \phi]$ the evaluation of ϕ with respect to the trace $\pi \in \Pi^*$ that yields a value in $\{\top, \bot, ?\}$.

$$[\pi \models_3 \phi] = \begin{cases} \top & \forall \pi' \in \Pi^\omega, \pi \cdot \pi' \models \phi,\\ \bot & \forall \pi' \in \Pi^\omega, \pi \cdot \pi' \not\models \phi,\\ ? & \text{otherwise.} \end{cases}$$

We now restrict LTL to a fragment without explicit $\top$ and $\bot$ symbols and with the explicit F operator that we add to the syntax. We provide an alternative 3-valued semantics for this fragment, denoted by $\mu_\pi(\phi, i)$ where $i \in \mathbb{N}_{>0}$ indicates a position in or outside the trace. We assume the order $\bot < ? < \top$, and extend the Boolean operations to the 3-valued domain with the rules $\neg_3\top = \bot$, $\neg_3\bot = \top$ and $\neg_3? = ?$ and $\phi_1 \vee_3 \phi_2 = max(\phi_1, \phi_2)$. We define the semantics inductively as follows:

$$\begin{array}{ll}
\mu_\pi(p,i) & = \begin{cases} \top & \text{if } i \leq |\pi| \text{ and } p \in \pi_i,\\ \bot & \text{else if } i \leq |\pi| \text{ and } p \notin \pi_i,\\ ? & \text{otherwise,} \end{cases}\\
\mu_\pi(\neg\phi, i) & = \neg_3 \mu_\pi(\phi, i),\\
\mu_\pi(\phi_1 \vee \phi_2, i) & = \mu_\pi(\phi_1, i) \vee_3 \mu_\pi(\phi_2, i),\\
\mu_\pi(\mathsf{X}\,\phi, i) & = \mu_\pi(\phi, i+1),\\
\mu_\pi(\mathsf{F}\,\phi, i) & = \begin{cases} \mu_\pi(\phi, i) \vee_3 \mu_\pi(\mathsf{XF}\,\phi, i) & \text{if } i \leq |\pi|,\\ \mu_\pi(\phi, i) & \text{if } i > |\pi|, \end{cases}\\
\mu_\pi(\phi_1 \,\mathsf{U}\, \phi_2, i) & = \begin{cases} \mu_\pi(\phi_2, i) \vee_3 (\mu_\pi(\phi_1, i) \wedge_3 \mu_\pi(\mathsf{X}(\phi_1 \,\mathsf{U}\, \phi_2), i)) & \text{if } i \leq |\pi|,\\ \mu_\pi(\phi_2, i) & \text{if } i > |\pi|. \end{cases}
\end{array}$$

We note that the adapted semantics allows evaluating a finite trace in polynomial time, in contrast to $[\pi \models_3 \phi]$, which requires a PSPACE-complete algorithm. This improvement in complexity comes at a price – the adapted semantics cannot semantically characterize tautologies and contradiction. We have for example that $\mu_\pi(p \vee \neg p, 1)$ for the empty word evaluates to ?, despite the fact that $p \vee \neg p$ is semantically equivalent to $\top$. The novel semantics that we introduce in the following sections make the same tradeoff.

In the following lemma, we relate the two three-valued semantics.

Lemma 3. *Given an LTL formula and a trace $\pi \in \Pi^*$, $|\pi| \neq 0$, we have that*

$$\mu_\pi(\phi, 1) = \top \Rightarrow [\pi \models_3 \phi] = \top,$$
$$\mu_\pi(\phi, 1) = \bot \Rightarrow [\pi \models_3 \phi] = \bot.$$

Proof. These two statements can be proven by induction on the structure of the LTL formula (see Appendix A.1 in [2]). $[\pi \models_3 \phi] = \ ? \Rightarrow \mu_\pi(\phi, 1) = \ ?$ is the consequence of the first two.

4 Counting Finitary Semantics for LTL

In this section, we introduce the counting semantics for LTL. We first provide necessary definitions in Section 4.1, we present the new semantics in Section 4.2 and finally propose a predictive mapping that transforms the counting semantics into a qualitative 5-valued verdict in Section 4.3.

4.1 Definitions

Let $\mathbb{N}_+ = \mathbb{N}_0 \cup \{\infty, -\}$ be the set of *natural* numbers (incl. 0) extended with the two special symbols ∞ (infinite) and $-$ (impossible) such that $\forall n \in \mathbb{N}_0$, we define $n < \infty < -$. We define the addition $\oplus$ of two elements $a, b \in \mathbb{N}_+$ as follows.

Definition 4 (Operator $\oplus$). *We define the binary operator $\oplus : \mathbb{N}_+ \times \mathbb{N}_+ \to \mathbb{N}_+$ s. t. for $a \oplus b$ with $a, b \in \mathbb{N}_+$ we have $a + b$ if $a, b \in \mathbb{N}_0$ and $\max\{a, b\}$ otherwise.*

We denote by (s, f) a pair of two extended numbers $s, f \in \mathbb{N}_+$. In Definition 5, we introduce several operations on pairs: (1) the *swap* between the two values ($\sim$), (2) the increment by 1 of both values ($\oplus 1$), (3) the *minmax* binary operation ($\sqcup$) that gives the pair consisting of the minimum first value and the maximum second value, and (4) the *maxmin* binary operation ($\sqcap$) that is symmetric to ($\sqcup$).

Definition 7 introduces the counting semantics for LTL that for a finite trace π and LTL formula ϕ gives a pair $(s, f) \in \mathbb{N}_+ \times \mathbb{N}_+$. We call s and f *satisfaction* and *violation witness counts*, respectively. Intuitively, the s (f) value denotes the minimal number of additional steps that is needed to witness the satisfaction (violation) of the formula. The value ∞ is used to denote that the property can be satisfied (violated) only in an infinite number of steps, while $-$ means the property cannot be satisfied (violated) by any continuation of the trace.

Definition 5 (Operations $\sim$, $\oplus 1$, $\sqcup$, $\sqcap$). *Given two pairs $(s, f) \in \mathbb{N}_+ \times \mathbb{N}_+$ and $(s', f') \in \mathbb{N}_+ \times \mathbb{N}_+$, we have:*

$$\begin{aligned} \sim(s, f) &= (f, s), \\ (s, f) \oplus 1 &= (s \oplus 1, f \oplus 1), \\ (s, f) \sqcup (s', f') &= (\min(s, s'), \max(f, f')), \\ (s, f) \sqcap (s', f') &= (\max(s, s'), \min(f, f')). \end{aligned}$$

Example 6. Given the pairs $(0, 0)$, $(\infty, 1)$ and $(7, -)$ we have the following:

$$\begin{aligned} \sim(0,0) &= (0,0), & \sim(\infty,1) &= (1,\infty), \\ (0,0) \oplus 1 &= (1,1), & (\infty,1) \oplus 1 &= (\infty,2), \\ (0,0) \sqcup (\infty,1) &= (0,1), & (\infty,1) \sqcup (7,-) &= (7,-), \\ (0,0) \sqcap (\infty,1) &= (\infty,0), & (\infty,1) \sqcap (7,-) &= (\infty,1). \end{aligned}$$

Remark. Note that $\mathbb{N}_+ \times \mathbb{N}_+$ forms a lattice where $(s, f) \trianglelefteq (s', f')$ when $s \geq s'$ and $f \leq f'$ with join $\sqcup$ and meet $\sqcap$. Intuitively, larger values are closer to true.

4.2 Semantics

We now present our finitary semantics.

Definition 7 (Counting finitary semantics). *Let $\pi \in \Pi^*$ be a finite trace, $i \in \mathbb{N}_{>0}$ be a position in or outside the trace and $\phi \in \Phi$ be an LTL formula. We define the counting finitary semantics of LTL as the function $d_\pi : \Phi \times \Pi^* \times \mathbb{N}_{>0} \to \mathbb{N}_+ \times \mathbb{N}_+$ such that:*

$$\begin{aligned} d_\pi(p, i) &= \begin{cases} (0,-) & \text{if } i \leq |\pi| \wedge p \in \pi_i, \\ (-,0) & \text{if } i \leq |\pi| \wedge p \notin \pi_i, \\ (0,0) & \text{if } i > |\pi|, \end{cases} \\ d_\pi(\neg\phi, i) &= \sim d_\pi(\phi, i), \\ d_\pi(\phi_1 \vee \phi_2, i) &= d_\pi(\phi_1, i) \sqcup d_\pi(\phi_2, i), \\ d_\pi(\mathsf{X}\,\phi, i) &= d_\pi(\phi, i+1) \oplus 1, \\ d_\pi(\phi\,\mathsf{U}\,\psi, i) &= \begin{cases} d_\pi(\psi, i) \sqcup \Big(d_\pi(\phi, i) \sqcap d_\pi(\mathsf{X}(\phi\,\mathsf{U}\,\psi), i)\Big) & \text{if } i \leq |\pi|, \\ d_\pi(\psi, i) \sqcup \Big(d_\pi(\phi, i) \sqcap (-, \infty)\Big) & \text{if } i > |\pi|, \end{cases} \\ d_\pi(\mathsf{F}\,\phi, i) &= \begin{cases} d_\pi(\phi, i) \sqcup d_\pi(\mathsf{XF}\,\phi, i) & \text{if } i \leq |\pi|, \\ d_\pi(\phi, i) \sqcup (-, \infty) & \text{if } i > |\pi|. \end{cases} \end{aligned}$$

We now provide some motivations behind the above definitions.

Proposition A proposition is either evaluated before or after the end of the trace. If it is evaluated before the end of the trace and the proposition holds, the satisfaction and violations witness counts are trivially 0 and $-$, respectively. In the case that the proposition does not hold, we have the symmetric witness counts. Finally, we take an optimistic view in case of evaluating a proposition after the end of the trace: The trace can be extended to a trace with i steps s.t. either p holds or p does not hold.

Table 2: Unbounded response property example: $d_\pi(\phi, i)$ with the trace $\pi = \tau_2$.

	1	2	3	4	5	6	7	EOT
r	$\top$	$-$	$-$	$\top$	$-$	$-$	$\top$	
g	$-$	$-$	$\top$	$-$	$-$	$\top$	$-$	
$d_\pi(r, i)$	$(0,-)$	$(-,0)$	$(-,0)$	$(0,-)$	$(-,0)$	$(-,0)$	$(0,-)$	(0,0)
$d_\pi(g, i)$	$(-,0)$	$(-,0)$	$(0,-)$	$(-,0)$	$(-,0)$	$(0,-)$	$(-,0)$	(0,0)
$d_\pi(\neg r, i)$	$(-,0)$	$(0,-)$	$(0,-)$	$(-,0)$	$(0,-)$	$(0,-)$	$(-,0)$	(0,0)
$d_\pi(\mathsf{F}\, g, i)$	$(2,-)$	$(1,-)$	$(0,-)$	$(2,-)$	$(1,-)$	$(0,-)$	$(1,\infty)$	$(0,\infty)$
$d_\pi(r \rightarrow \mathsf{F}\, g, i)$	$(2,-)$	$(0,-)$	$(0,-)$	$(2,-)$	$(0,-)$	$(0,-)$	$(1,\infty)$	$(0,\infty)$
$d_\pi(\mathsf{G}(r \rightarrow \mathsf{F}\, g), i)$	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)

We use "$-$" instead of "$\bot$" in the traces r and g to improve the readability.

Negation Negating a formula simply swaps the witness counts. If we witness the satisfaction of ϕ in n steps, we witness the violation of $\neg\phi$ in n steps, and vice versa.

Disjunction We take the shorter satisfaction witness count, because the satisfaction of one subformula is enough to satisfy the property. And we take the longer violation witness count, because both subformulas need to be violated to violate the property.

Next The next operator naturally increases the witness counts by one step.

Eventually We use the rewriting rule $\mathsf{F}\,\phi \equiv \phi \vee \mathsf{XF}\,\phi$ to define the semantics of the eventually operator. When evaluating the formula after the end of the trace, we replace the remaining obligation ($\mathsf{XF}\,\phi$) by $(-,\infty)$. Thus, $\mathsf{F}\,\phi$ evaluated on the empty word is satisfied by a suffix that satisfies ϕ, and it is violated only by infinite suffixes.

Until We use the same principle for defining the until semantics that we used for the eventually operator. We use the rewriting rule $\phi\,\mathsf{U}\,\psi \equiv \psi \vee (\phi \wedge \mathsf{X}(\phi\,\mathsf{U}\,\psi))$. On the empty word, $\phi\,\mathsf{U}\,\psi$ is satisfied (in the shortest way) by a suffix that satisfies ψ, and it is violated by a suffix that violates both ϕ and ψ.

Example 8. We refer to our motivating example from Table 1 and evaluate the trace τ_2 with respect to the specification ψ. We present the outcome in Table 2. We see that every proposition evaluates to $(0,-)$ when true. The satisfaction of a proposition that holds at time i is immediately witnessed and it cannot be violated by any suffix. Similarly, a proposition evaluates to $(-,0)$ when false. The valuations of $\mathsf{F}\,g$ count the number of steps to positions in which g holds. For instance, the first time at which g holds is $i = 3$, hence $\mathsf{F}\,g$ evaluates to $(2,-)$ at time 1, $(1,-)$ at time 2 and $(0,-)$ at time 3. We also note that $\mathsf{F}\,g$ evaluates to $(0,\infty)$ at the end of the trace – it could be immediately satisfied with the continuation of the trace with g that holds, but could be violated only by an infinite suffix in which g never holds. We finally observe that $\mathsf{G}(r \rightarrow \mathsf{F}\,g)$ evaluates to (∞,∞) at all positions – the property can be both satisfied and violated only with infinite suffixes.

Not all pairs $(s, f) \in \mathbb{N}_+ \times \mathbb{N}_+$ are possible according to the counting semantics. We present the possible pairs in Lemma 9.

Lemma 9. *Let $\pi \in \Pi^*$ be a finite trace, ϕ an LTL formula and $i \in \mathbb{N}_0$ an index. We have that $d_\pi(\phi, i)$ is of the form $(a, -)$, $(-, a)$, (b_1, b_2), (b_1, ∞), (∞, b_2) or (∞, ∞), where $a \leq |\pi| - i$ and $b_j > |\pi| - i$ for $j \in \{1, 2\}$.*

Proof. The proof can be obtained using structural induction on the LTL formula (see Appendix A.2 in [2]).

Finally, we relate our counting semantics to the three valued semantics in Lemma 10.

Lemma 10. *Given an LTL formula and a trace $\pi \in \Pi^*$ where $i \in \mathbb{N}_{>0}$ is an index and ϕ is an LTL formula, we have that*

$$
\begin{aligned}
d_\pi(\phi, i) = (a, -) \;&\leftrightarrow \mu_\pi(\phi, i) = \top, \\
&\quad \textit{and} \;\; \nexists x < a \,.\, \pi' = \pi_i \cdot \pi_{i+1} \cdot \ldots \pi_{i+x}, \mu_{\pi'}(\phi, 1) = \top \\
d_\pi(\phi, i) = (-, a) \;&\leftrightarrow \mu_\pi(\phi, i) = \bot, \\
&\quad \textit{and} \;\; \nexists x < a \,.\, \pi' = \pi_i \cdot \pi_{i+1} \cdot \ldots \pi_{i+x}, \mu_{\pi'}(\phi, 1) = \bot \\
d_\pi(\phi, i) = (b_1, b_2) \;&\leftrightarrow \mu_\pi(\phi, i) = ?,
\end{aligned}
$$

where $a \leq |\pi| - i$ and b_j is either ∞ or $b_j > |\pi| - i$ for $j \in \{1, 2\}$.

Intuitively, Lemma 10 holds because we only introduce the symbol "$-$" within the trace when a satisfaction (violation) is observed. And the values of a pair only propagate into the past (and never into the future).

4.3 Evaluation

We now propose a mapping that predicts a qualitative verdict from our counting semantics. We adopt a 5-valued set consisting of true ($\top$), presumably true ($\top_P$), inconclusive ($?$), presumably false ($\bot_P$) and false ($\bot$) verdicts. We define the following order over these five values: $\bot < \bot_P < ? < \top_P < \top$. We equip this 5-valued domain with the negation ($\neg$) and disjunction ($\vee$) operations, letting $\neg\top = \bot$, $\neg\top_P = \bot_P$, $\neg? = ?$, $\neg\bot_P = \top_P$, $\neg\bot = \top$ and $\phi_1 \vee \phi_2 = \max\{\phi_1, \phi_2\}$. We define other Boolean operators such as conjunction by the usual logical equivalences ($\phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2)$, etc.).

We evaluate a property on a trace to $\top$ ($\bot$) when the satisfaction (violation) can be fully determined from the trace, following the definition of the three-valued semantics μ. Intuitively, this takes care of the case in which the safety (co-safety) part of a formula has been violated (satisfied), at least for properties that are intentionally safe (intentionally co-safe, resp.) [10].

Whenever the truth value is not determined, we distinguish whether $d_\pi(\phi, i)$ indicates the possibility for a satisfaction, respective violation, in finite time or not. For possible satisfactions, respective violations, in finite time we make a prediction on whether past observations support the believe that the trace is going to satisfy or violate the property. If the predictions are not inconclusive and not contradicting, then we evaluate the trace to the (presumable) truth value $\top_P$ or$\bot_P$. If we cannot make a prediction to a truth value, we compute

the truth value recursively based on the operator in the formula and the truth values of the subformulas (with temporal operators unrolled).

We use the predicate pred_π to give the prediction based on the observed witnesses for satisfaction. The predicate $\text{pred}_\pi(\phi, i)$ becomes ? when no witness for satisfaction exists in the past. When there exists a witness that requires at least the same amount of additional steps as the trace under evaluation then the predicate evaluates to $\top$. If all the existing witnesses (and at least one exists) are shorter than the current trace, then the predicate evaluates to $\bot$. For a prediction on the violation we make a prediction on the satisfaction of $d_\pi(\neg\phi, i)$, i.e., we compute $\text{pred}_\pi(\neg\phi, i)$.

Definition 11 (Prediction predicate).

Let s, f denote natural numbers and let $s_\pi(\phi, i), f_\pi(\phi, i) \in \mathbb{N}_+$ such that $d_\pi(\phi, i) = \big(s_\pi(\phi, i), f_\pi(\phi, i)\big)$. We define the 3-valued predicate $pred_\pi$ as

$$pred_\pi(\phi, i) = \begin{cases} \top & \text{if } \exists j < i \,.\, d_\pi(\phi, j) = (s', -) \text{ and } s_\pi(\phi, i) \leq s', \\ ? & \text{if } \nexists j < i \,.\, d_\pi(\phi, j) = (s', -), \\ \bot & \text{if } \exists j < i \,.\, d_\pi(\phi, j) = (s', -) \text{ and }, \\ & \quad s_\pi(\phi, i) > \max_{0 \leq j < i}\{s' \mid d_\pi(\phi, j) = (s', -)\}, \end{cases}$$

For the evaluation we consider a case split among the possible combinations of values in the pairs.

Definition 12 (Predictive evaluation). *We define the* predictive evaluation *function $e_\pi(\phi, i)$, with $a \leq |\pi| - i$ and $b_j > |\pi| - i$ for $j \in \{1, 2\}$ and $a, b_j \in \mathbb{N}_0$, for the different cases of $d_\pi(\phi, i)$:*

$d_\pi(\phi, i)$		$e_\pi(\phi, i)$
$(a, -)$		$\top$
(b_1, b_2)	*if* $pred_\pi(\phi, i) > pred_\pi(\neg\phi, i)$	$\top_P$
	if $pred_\pi(\phi, i) = pred_\pi(\neg\phi, i)$	$r_\pi(\phi, i)$
	if $pred_\pi(\phi, i) < pred_\pi(\neg\phi, i)$	$\bot_P$
(b_1, ∞)	*if* $pred_\pi(\phi, i) = \top$	$\top_P$
	if $pred_\pi(\phi, i) = ?$	$r_\pi(\phi, i)$
	if $pred_\pi(\phi, i) = \bot$	$\bot_P$
(∞, b_1)		$e_\pi(\neg\phi, i)$
(∞, ∞)		$r_\pi(\phi, i)$
$(-, a)$		$\bot$

where $r_\pi(\phi, i)$ is an auxiliary function defined inductively as follows:

Table 3: Unbounded response property example with $\pi = \tau_2$.

	1	2	3	4	5	6	7	EOT
r	$\top$	$-$	$-$	$\top$	$-$	$-$	$\top$	
g	$-$	$-$	$\top$	$-$	$-$	$\top$	$-$	
$d_\pi(r,i)$	$(0,-)$	$(-,0)$	$(-,0)$	$(0,-)$	$(-,0)$	$(-,0)$	$(0,-)$	(0,0)
$e_\pi(r,i)$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	?
$d_\pi(g,i)$	$(-,0)$	$(-,0)$	$(0,-)$	$(-,0)$	$(-,0)$	$(0,-)$	$(-,0)$	(0,0)
$e_\pi(g,i)$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	?
$d_\pi(\mathsf{F}\,g,i)$	$(2,-)$	$(1,-)$	$(0,-)$	$(2,-)$	$(1,-)$	$(0,-)$	$(1,\infty)$	$(0,\infty)$
$e_\pi(\mathsf{F}\,g,i)$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top_P$	$\top_P$
$d_\pi(r \to \mathsf{F}\,g,i)$	$(2,-)$	$(0,-)$	$(0,-)$	$(2,-)$	$(0,-)$	$(0,-)$	$(1,\infty)$	$(0,\infty)$
$e_\pi(r \to \mathsf{F}\,g,i)$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top_P$	$\top_P$
$d_\pi(\mathsf{G}(r \to \mathsf{F}\,g),i)$	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)
$e_\pi(\mathsf{G}(r \to \mathsf{F}\,g),i)$	$\top_P$	$\top_P$	$\top_P$	$\top_P$	$\top_P$	$\top_P$	$\top_P$	$\top_P$

We use "$-$" instead of "$\bot$" in the traces r and g to improve the readability.

$$
\begin{aligned}
r_\pi(p,i) &= ? \\
r_\pi(\neg\phi,i) &= \neg e_\pi(\phi,i) \\
r_\pi(\phi_1 \vee \phi_2,i) &= e_\pi(\phi_1,i) \vee e_\pi(\phi_2,i) \\
r_\pi(\mathsf{X}^n\,\phi,i) &= e_\pi(\phi,i+n) \\
r_\pi(\mathsf{F}\,\phi,i) &= \begin{cases} e_\pi(\phi,i) \vee r_\pi(\mathsf{X}\,\mathsf{F}\,\phi,i) & \text{if } i \le |\pi| \\ e_\pi(\phi,i) & \text{if } i > |\pi| \end{cases} \\
r_\pi(\phi_1\,\mathsf{U}\,\phi_2,i) &= \begin{cases} e_\pi(\phi_2,i) \vee (e_\pi(\phi_2,i) \wedge e_\pi(\mathsf{X}(\phi_1\,\mathsf{U}\,\phi_2),i) & \text{if } i \le |\pi| \\ e_\pi(\phi_2,i) & \text{if } i > |\pi| \end{cases}
\end{aligned}
$$

The predictive evaluation function is symmetric. Hence, $e_\pi(\phi,i) = \neg e_\pi(\neg\phi,i)$ holds.

Example 13. The outcome of evaluating τ_2 from Table 1 is shown in Table 3. Subformula $r \to \mathsf{F}\,g$ is predicted to be $\top_P$ at $i = 7$ because there exists a longer witness for satisfaction in the past (e.g., at $i = 1$). Thus, the trace evaluates to $\top_P$, as expected.

In Figure 1 we visualize the evaluation of a pair $d_\pi(\phi,i) = (s,f)$ for a fixed ϕ and a fixed position i. On the x-axis is the witness count s for a satisfaction and on the y-axis is the witness count f for a violation. For a value s, respectively f, that is smaller than the length of the suffix starting at position i (with the other value of the pair always being $-$), the evaluation is either $\top$ or $\bot$. Otherwise the evaluation depends on the values s_{max} and f_{max}. These two values represent the largest witness counts for a satisfaction and a violation in the past, i.e., for positions smaller than i in the trace. Based on the prediction function $\text{pred}_\pi(\phi,i)$ the evaluation becomes $\top_P$, ? or $\bot_P$, where ? indicates that the auxiliary function $r_\pi(\phi,i)$ has to be applied. Starting at an arbitrary point in the

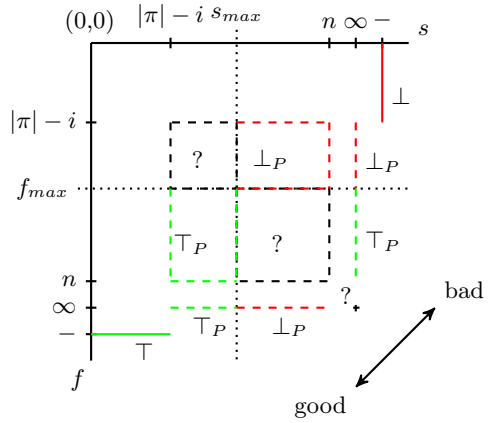

Fig. 1: Lattice for (s, f) with ϕ and $i < |\pi|$ fixed.

diagram and moving to the right increases the witness count for a satisfaction while the witness count for a violation remains constant. Thus, moving to the right makes the pair "more false". The same holds when keeping the witness count for a satisfaction constant and moving up in the diagram as this decrease the witness count for a violation. Analogously, moving down and/or left makes the pair "more true" as the witness count for a violation gets larger and/or the witness count for a satisfaction gets smaller.

Our 5-valued predictive evaluation refines the 3-valued LTL semantics.

Theorem 14. *Let ϕ be an LTL formula, $\pi \in \Pi^*$ and $i \in \mathbb{N}_{>0}$. We have*

$$\begin{aligned}
\mu_\pi(\phi, i) = \top &\leftrightarrow e_\pi(\phi, i) = \top, \\
\mu_\pi(\phi, i) = \bot &\leftrightarrow e_\pi(\phi, i) = \bot, \\
\mu_\pi(\phi, i) = ? \ &\leftrightarrow e_\pi(\phi, i) \in \{\top_P, \bot_P, ?\}.
\end{aligned}$$

Theorem 14 holds, because the evaluation to $\top$ and $\bot$ is simply the mapping of a pair that contains the symbol "$-$", which we have shown in Lemma 10.

Remember that $\mathbb{N}_+ \times \mathbb{N}_+$ is partially ordered by $\trianglelefteq$. We now show that having a trace that is "more true" than another is correctly reflected in our finitary semantics. To define "more true", we first need the polarity of a proposition in an LTL formula.

Example 15. Note that g has positive polarity in $\phi = \mathsf{G}(r \rightarrow \mathsf{F}\, g)$. If we define τ_2' to be as τ_2, except that $g \in \tau_2'(i)$ for $i \in \{1, \ldots, 6\}$, we have $e_{\tau_2'}(\phi, i) = \bot_P$, whereas $e_{\tau_2}(\phi, i) = \top_P$.

Definition 16 (Polarity). *Let $\#\neg$ be the number of negation operators on a specific path in the parse tree of ϕ starting at the root. We define the polarity as*

ϕ	π	$d_\pi(\phi,1)$	$e_\pi(\phi,1)$
p	$-$	$(-,0)$	$\bot$
	$\top$	$(0,-)$	$\top$
$p \wedge \mathsf{X}\,\mathsf{F}\,p$	$- - -$	$(-,0)$	$\bot$
	$\top - -$	$(3,\infty)$	$\bot_P$
$\mathsf{G}\,p$	$-\top\top$	$(-,0)$	$\bot$
	$\top\top\top$	$(\infty,3)$	$\top_P$
$\mathsf{F}\,p$	$- - -$	$(3,\infty)$	$\bot_P$
	$\top - -$	$(0,-)$	$\top$

ϕ	π	$d_\pi(\phi,1)$	$e_\pi(\phi,1)$
$\mathsf{F}\,\mathsf{G}\,p$	$\top - \top - \top$	(∞,∞)	$\bot_P$
	$\top - \top\top\top$	(∞,∞)	$\top_P$
$\mathsf{G}\,\mathsf{F}\,p$	$- - \top - -$	(∞,∞)	$\top_P$
	$\top - \top - -$	(∞,∞)	$\bot_P$
$p \vee \mathsf{X}\,\mathsf{G}\,p$	$-\top\top$	$(\infty,3)$	$\top_P$
	$\top\top\top$	$(0,-)$	$\top$

Table 4: Making a system "more true".

the function pol(p) with proposition p in an LTL formula ϕ as follows:

$$pol(p) = \begin{cases} pos, & \text{if } \#\neg \text{ on all paths to a leaf with proposition } p \text{ is even,} \\ neg, & \text{if } \#\neg \text{ on all paths to a leaf with proposition } p \text{ is odd,} \\ mixed, & \text{otherwise.} \end{cases}$$

With the polarity defined, we now define the constraints for a trace to be "more true" with respect to an LTL formula ϕ.

Definition 17 ($\pi \sqsubseteq_\phi \pi'$)**.** *Given two traces π and π' of equal length and an LTL formula ϕ over proposition p, we define that $\pi \sqsubseteq_\phi \pi'$ iff*

$$\begin{aligned} \forall i \forall p\,.\; pol(p) = mixed \Rightarrow p \in \pi_i \leftrightarrow p \in \pi'_i \text{ and} \\ pol(p) = pos \Rightarrow p \in \pi_i \rightarrow p \in \pi'_i \text{ and} \\ pol(p) = neg \Rightarrow p \in \pi_i \leftarrow p \in \pi'_i. \end{aligned}$$

Whenever one trace is "more true" than another, this is correctly reflected in our finitary semantics.

Theorem 18. *For two traces π and π' of equal length and an LTL formula ϕ over proposition p, we have that*

$$\pi \sqsubseteq_\phi \pi' \Rightarrow d_{\pi'}(\phi,1) \trianglelefteq d_\pi(\phi,1).$$

Therefore, we have for $\pi \sqsubseteq_\phi \pi'$ that

$$\begin{aligned} e_\pi(\phi,1) = \top \Rightarrow e_{\pi'}(\phi,1) = \top, \text{ and} \\ e_\pi(\phi,1) = \bot \Leftarrow e_{\pi'}(\phi,1) = \bot. \end{aligned}$$

Theorem 18 holds, because we have that replacing an arbitrary observed value in π by one with positive polarity in π' always results with $d_\pi(\phi,1) = (s,f)$ and $d_{\pi'}(\phi,1) = (s',f')$ in $s' \leq s$ and $f' \geq f$, as with $\pi \sqsubseteq_\phi \pi'$ we have that π' witnesses a satisfaction of ϕ not later than π and π' also witness a violation of ϕ not earlier than π.

In Table 4 we give examples to illustrate the transition of one evaluation to another one. Note that it is possible to change from $\top_P$ to $\bot_P$. However, this is only the predicated truth value that becomes "worse", because we have strengthened the prefix on which the prediction is based on, the values of $d_\pi(\phi,i)$ do not change and remain the same is such a case.

5 Examples

We demonstrate the strengths and weaknesses of our approach on the examples of LTL specifications and traces shown in Table 5. We fully develop these examples in Appendix B in [2].

Specifications	Traces	
$\psi_1 \equiv \mathsf{F}\,\mathsf{X}\,g$	$\pi_1 : g :\ \bot\bot\bot\bot$	$\pi_5 : r :\ \bot\top\top\top\top\bot\top\top$
$\psi_2 \equiv \mathsf{G}\,\mathsf{X}\,g$	$\pi_2 : g :\ \top\top\top\top$	$g :\ \bot\top\bot\bot\bot\bot\top\bot$
$\psi_3 \equiv \mathsf{G}(r \rightarrow \mathsf{F}\,g)$	$\pi_3 : r :\ \bot\top\bot\bot\top\bot$	$\pi_6 : g :\ \top\top\bot\bot\top\top\bot\bot\top\top\bot\bot\top$
$\psi_4 \equiv \bigwedge_{i\in\{1,2\}} \mathsf{G}(r_i \rightarrow \mathsf{F}\,g_i)$	$g :\ \bot\bot\top\bot\bot\bot$	$\pi_7 : g :\ \top\top\bot\bot\top\top\bot\bot\top\top\top\top\top$
$\psi_5 \equiv \mathsf{G}((\mathsf{X}\,r)\ \mathsf{U}\ (\mathsf{X}\,\mathsf{X}\,g))$	$\pi_4 : r_1 :\ \top\bot\top\bot\top\bot\top$	$\pi_8 : r :\ \top\top\top\top\bot\bot$
$\psi_6 \equiv \mathsf{F}\,\mathsf{G}\,g \vee \mathsf{F}\,\mathsf{G}\,\neg g$	$g_1 :\ \bot\top\bot\top\bot\top\bot$	$g :\ \top\bot\top\bot\top\bot$
$\psi_7 \equiv \mathsf{G}(\mathsf{F}\,r \vee \mathsf{F}\,g)$	$r_2 :\ \bot\top\bot\top\bot\top\bot$	
$\psi_8 \equiv \mathsf{G}\,\mathsf{F}(r \vee g)$	$g_2 :\ \top\bot\top\bot\top\bot\top$	
$\psi_9 \equiv \mathsf{G}\,\mathsf{F}\,r \vee \mathsf{G}\,\mathsf{F}\,g$		

Table 5: Examples of LTL specifications and traces

Table 6 summarizes the evaluation of our examples. The first and the second column denote the evaluated specification and trace. We use these examples to compare LTL with counting semantics (c-LTL) presented in this paper, to the other two popular finitary LTL interpretations, the 3-valued LTL semantics [4] (3-LTL) and LTL on trucated paths [9] (t-LTL). We recall that in t-LTL there is a distinction between a weak and a strong next operator. We denote by t-LTL-s (t-LTL-w) the specifications from our examples in which X is interpreted as the strong (weak) next operator and assume that we always give a strong interpretation to U and F and a weak interpretation to G.

There are two immediate observations that we can make regarding the results presented in Table 6. First, the 3-valued LTL gives for all the examples an *inconclusive* verdict, a feedback that after all has little value to a verification engineer. The second observation is that the verdicts from c-LTL and t-LTL can differ quite a lot, which is not very surprising given the different strategies to interpret the unseen future. We now further comment on these examples, explaining in more details the results and highlighting the intuitive outcomes of c-LTL for a large class of interesting LTL specifications.

Spec.	Trace	c-LTL	3-LTL	t-LTL-s	t-LTL-w
ψ_1	π_1	$\bot_P$	?	$\bot$	$\top$
ψ_2	π_2	$\top_P$	?	$\bot$	$\top$
ψ_3	π_3	$\bot_P$	?	$\bot$	$\bot$
ψ_4	π_4	$\top_P$	?	$\bot$	$\bot$
ψ_5	π_5	$\top_P$	?	$\bot$	$\top$

Spec.	Trace	c-LTL	3-LTL	t-LTL-s	t-LTL-w
ψ_6	π_6	$\bot_P$	?	$\top$	$\top$
ψ_6	π_7	$\top_P$	?	$\top$	$\top$
ψ_7	π_8	$\bot_P$	?	$\bot$	$\bot$
ψ_8	π_8	$\bot_P$	?	$\bot$	$\bot$
ψ_9	π_8	$\top_P$	?	$\bot$	$\bot$

Table 6: Comparison of different verdicts with different semantics

Effect of Nested Next We evaluate with ψ_1 and ψ_2 the effect of nesting X in an F and an G formula, respectively. We make a prediction on $\mathsf{X}\,g$ at the end of the trace before evaluating F and G. As a consequence, we find that (ψ_1, π_1) evaluates to presumably false, while (ψ_2, π_2) evaluates to presumably true. In t-LTL, this class of specification is very sensitive to the weak/strong interpretation of next, as we can see from the verdicts.

Request/Grants We evaluate the request/grant property ψ_3 from the motivating example on the trace π_3. We observe that r at cycle 2 is followed by g at cycle 3, while r at cycle 5 is not followed by g at cycle 6. Hence, (ψ_3, π_3) evaluates to presumably false.

Concurrent Request/Grants We evaluate the specification ψ_4 against the trace π_4. In this example r_1 is triggered at even time stamps and r_2 is triggered at odd time stamps. Every request is granted in one cycle. It follows that regardless of the time when the trace ends, there is one request that is not granted yet. We note that ψ_4 is a conjunction of two basic request/grant properties and we make independent predictions for each conjunct. Every basic request/grant property is evaluated to presumably true, hence (ψ_4, π_4) evaluates to presumably true. At this point, we note that in t-LTL, every request that is not granted by the end of the trace results in the property violation, regardless of the past observations.

Until We use the specification ψ_5 and the trace π_5 to evaluate the effect of U on the predictions. The specification requires that $\mathsf{X}\,r$ continuously holds until $\mathsf{X}\,\mathsf{X}\,g$ becomes true. We can see that in π_5 $\mathsf{X}\,r$ is witnessed at cycles $1-4$, while $\mathsf{X}\,\mathsf{X}\,g$ is witnessed at cycle 5. We can also see that $\mathsf{X}\,r$ is again witnessed from cycle 6 until the end of the trace at cycle 8. As a consequence, (ψ_5, π_5) is evaluated to presumably true.

Stabilization The specification ψ_6 says that the value of g has to eventually stabilize to either true or false. We evaluate the formula on two traces π_6 and π_7. In the trace π_6, g alternates between true and false every two cycles and becomes true in the last cycle. Hence, there is no sufficiently long witness of trace stabilization (ψ_6, π_6) evaluates to presumably false. In the trace π_7, g also alternates between true and false every two cycles, but in the last four cycles g remains continuously true. As a consequence, (ψ_6, π_7) evaluates to presumably true. This example also illustrates the importance of when the trace truncation occurs. If both π_6 and π_7 were truncated at cycle 5, both (ψ_6, π_6) and (ψ_6, π_7) would evaluate to presumably false. We note that ψ_6 is satisfied by all traces in t-LTL.

Sub-formula Domination The specification ψ_7 exposes a weakness of our approach. It requires that in every cycle, either r or g is witnessed in some unbounded future. With our approach, (ψ_7, π_8) evaluates to presumably false. This is against our intuition because we have observed that g becomes regularly true very second time step. However, in this example our prediction for $\mathsf{F}\,r$ dominates

over the prediction for $\mathsf{F}\, g$, leading to the unexpected presumably false verdict. On the other hand, t-LTL interpretation of the same specification is dependent only on the last value of r and g.

Semantically Equivalent Formulas We now demonstrate that our approach may give different answers for semantically equivalent formulas. For instance, both ψ_8 and ψ_9 are semantically equivalent to ψ_7. We have that (ψ_8, π_8) evaluates to presumably false, while (ψ_9, π_8) evaluates to presumably true. We note that t-LTL verdicts are stable for semantically different formulas.

6 Conclusion

We have presented a novel finitary semantics for LTL that uses the history of satisfaction and violation in a finite trace to predict whether the co-safety and safety aspects of a formula will be satisfied in the extension of the trace to an infinite one. We claim that the semantics closely follow human intuition when predicting the truth value of a trace. The presented examples (incl. non-monitorable LTL properties) illustrate our approach and support this claim.

Our definition of the semantics is trace-based, but it is easily extended to take an entire database of traces into account, which may make the approach more precise. Our approach currently uses a very simple form of learning to predict the future. We would like to consider more sophisticated statistical methods to make better predictions. In particular, we plan to apply nonparametric statistical methods (i.e., the Wilcoxon signed-rank test [16]), in combination with our counting semantics, to identify and quantify the traces that are outliers.

References

1. Shaull Almagor, Udi Boker, and Orna Kupferman. Discounting in LTL. In *Proc. of TACAS 2014: the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 424–439. Springer, 2014.
2. Ezio Bartocci, Roderick Bloem, Dejan Nickovic, and Franz Roeck. A counting semantics for monitoring LTL specifications over finite traces. *CoRR*, abs/1804.03237, 2018.
3. Ezio Bartocci and Yliès Falcone, editors. *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*. Springer, 2018.
4. Andreas Bauer, Martin Leucker, and Christian Schallhart. Monitoring of real-time properties. In *Proc. of FSTTCS 2006: the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *LNCS*, pages 260–272. Springer, 2006.
5. Andreas Bauer, Martin Leucker, and Christian Schallhart. The good, the bad, and the ugly, but how ugly is ugly? In *Proc. of RV 2007: the 7th International Workshop on Runtime Verification*, volume 4839 of *LNCS*, pages 126–138. Springer, 2007.
6. Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, 2011.

7. Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Logic*, 15(4):27:1–27:25, July 2014.
8. Przemyslaw Daca, Thomas A. Henzinger, Jan Kretínský, and Tatjana Petrov. Faster statistical model checking for unbounded temporal properties. In *Proc. of TACAS 2016: the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 112–129, 2016.
9. Cindy Eisner, Dana Fisman, John Havlicek, Yoad Lustig, Anthony McIsaac, and David Van Campenhout. Reasoning with temporal logic on truncated paths. In *Proc. of CAV 2003: the 15th International Conference on Computer Aided Verification*, volume 2725 of *LNCS*, pages 27–39. Springer, 2003.
10. Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.
11. Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems - specification*. Springer, 1992.
12. Andreas Morgenstern, Manuel Gesell, and Klaus Schneider. An asymptotically correct finite path semantics for LTL. In *Proc. of LPAR-18: the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 7180 of *LNCS*, pages 304–319. Springer, 2012.
13. Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57, 1977.
14. Amir Pnueli and Aleksandr Zaks. PSL model checking and run-time verification via testers. In *Proc. of FM 2006: Formal Methods, 14th International Symposium on Formal Methods*, volume 4085 of *LNCS*, pages 573–586. Springer, 2006.
15. Mahesh Viswanathan and Moonzoo Kim. Foundations for the run-time monitoring of reactive systems - fundamentals of the mac language. In *Proc. of ICTAC 2004: the First International Colloquium on Theoretical Aspects of Computing*, volume 3407 of *LNCS*, pages 543–556. Springer, 2004.
16. Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.
17. Xian Zhang, Martin Leucker, and Wei Dong. Runtime verification with predictive semantics. In *Proc. of NFM 2012: the 4th International Symposium on NASA Formal Methods*, volume 7226 of *LNCS*, pages 418–432. Springer, 2012.