Finite-Trace Linear Temporal Logic: Coinductive Completeness

Grigore Roşu

University of Illinois, USA
grosu@illinois.edu

Abstract. Linear temporal logic (LTL) is suitable not only for infinite-trace systems, but also for finite-trace systems. Indeed, LTL is frequently used as a trace specification formalism in runtime verification. The completeness of LTL with only infinite or with both infinite and finite traces has been extensively studied, but similar direct results for LTL with only finite traces are missing. This paper proposes a sound and complete proof system for finite-trace LTL. The axioms and proof rules are natural and expected, except for one rule of coinductive nature, reminiscent of the Gödel-Löb axiom. A direct decision procedure for finite-trace LTL satisfiability, a PSPACE-complete problem, is also obtained as a corollary.

1 Introduction

Finite execution traces play an important role in several computing fields. For example, Hoare logic [12], which is at the heart of deductive program verification, defines (partial) correctness in terms of finite traces: $\{pre\}P\{post\}$ holds iff any *finite* execution trace of P starting in a state satisfying *pre* ends in a state satisfying *post*. Also, in runtime verification, formal specifications are often used to characterize the bad behaviors of a system. Then the system is monitored against monitors generated from specifications. While infinite-trace specification formalisms have occasionally been used to specify systems' bad behaviors, in the end such bad behaviors occur after a finite number of observed events, so the generated monitors need only be faithful to the finite-trace safety fragment of the property. Consequently, many temporal specification formalisms used in runtime verification (and not only) have finite-trace semantics [5, 8, 11, 13, 15, 20, 23].

Linear temporal logic (LTL) [18] has established itself as one of the major trace specification formalism. With few exceptions (some mentioned above, others shortly below), the semantics of LTL is typically given in terms of infinite traces or of both infinite and finite traces (see, e.g., [17]), and some of the major theoretical results of LTL have only been studied in this context. This is unfortunate, because LTL is just as suitable a specification formalism for properties over only finite traces. For example, we can specify any finite-state machine *FSM* as a finite-trace LTL formula φ_{FSM} (Example 2), so that a word is in the language of *FSM* iff (a variant of) it satisfies φ_{FSM}. Moreover, consider again a Hoare triple $\{pre\}P\{post\}$ and suppose that *FSM*, φ_{pre}, and φ_{post}, respectively, abstract the state-space of the program P (with accepting states precisely where P is terminated), and *pre* and *post*. Then the formula $\varphi_{FSM} \wedge \varphi_{pre} \rightarrow \Diamond\varphi_{post}$ captures the abstract meaning of the Hoare triple quite elegantly.

When giving LTL a finite-trace semantics, one has to decide upon the semantics of the "next" operator on one-state traces, that is, when there is no next state. In a two-valued[1] setting, there are three admittedly meaningful semantics for "next φ" on one-state traces: (1) it always holds; (2) it never holds; (3) it holds iff φ holds itself on the one-state trace. The semantics (3) has the technical advantage that it reduces finite-trace to infinite-trace semantics by repeating the last state of the finite trace indefinitely, so the usual LTL reasoning remains sound. For that reason, for example, it has been used in the context of runtime verification [20], where a finite-trace semantics with sound deduction was needed. However, (3) has a major drawback: the LTL formulae cannot distinguish between terminated traces and traces which (accidentally) repeat their last state. Hence, in our view, (3) does not capture the nature of finite-trace LTL properly, so we here stick to (1) and (2). In fact, (1) and (2) are equivalent and can co-exist: if $\circ$ is the *weak* next of (1) and $\bullet$ is the *strong* next of (2), then it is easy to see that $\circ\varphi \equiv \neg\bullet\neg\varphi$ and $\bullet\varphi \equiv \neg\circ\neg\varphi$.

While first-order logic expressiveness results for LTL variants with finite-trace semantics have been studied [6, 24], at our knowledge no other major theoretical aspects of finite-trace LTL have been investigated. In particular, *direct* decidability and complete deduction results are missing. By "direct" we mean ones that work directly with finite-trace LTL formulae, as opposed to ones based on translations to other logics. As an analogy, an *indirect* complete proof system for infinite-trace LTL, or for equational logic, etc., can be easily obtained by translations of these logics into first-order logic (FOL) and then using the complete proof system for FOL. Practically, such indirect results have at least two drawbacks: first, the size of the translated formulae may be larger than the original formula, thus incurring increased algorithmic complexity to solve the translated problem; second, the meaning and intuitions of the original logic and its formulae may be lost in translation, making assisted proofs more challenging and inconvenient for humans. Theoretically, direct decidable procedures and complete proof systems specialized for the logics of interest are desirable, because they help us better understand the nature of those logics and their specific challenges.

One may think that complete proof systems for finite-trace LTL should easily follow from the infinite-trace variants, because finite traces are particular infinite traces which stutter in the final state after a finite number of states. However, a careful examination reveals that the infinite-trace LTL results heavily rely on the axiom/property $\neg\circ\varphi \leftrightarrow \circ\neg\varphi$, which does *not* hold for finite-trace LTL. Only one implication holds, namely $\neg\circ\varphi \rightarrow \circ\neg\varphi$ (or its equivalent $\bullet\neg\varphi \rightarrow \neg\bullet\varphi$). Therefore, axioms need to be dropped from the infinite-trace LTL proof system. Furthermore, one may think that it suffices to just drop the implication $\circ\neg\varphi \rightarrow \neg\circ\varphi$ from the axioms of infinite-trace LTL (or replace it with a weaker one), like for the LTL variant in [17] with both infinite and finite traces, because all the other axioms and proof rules, including the powerful Induction rule

$$\textbf{\textit{Ind}} \quad \frac{\varphi \rightarrow \circ\varphi}{\varphi \rightarrow \Box\varphi}$$

are sound for finite traces as well, and finite-trace LTL "ought to" be simpler than LTL with both finite and infinite-traces. However, it turns out that new rules are needed in

[1] See [2] for multi-valued variants of LTL.

order to achieve completeness, because finite-trace LTL admits new tautologies which do not hold for infinite-traces, such as $\Diamond\circ\bot$ (every trace eventually terminates).

Conceptually, the main contribution of this paper is the following Coinduction proof rule, which appears to play a central role in finite-trace temporal reasoning:

$$\textbf{\textit{coInd}} \qquad \frac{\circ\varphi \rightarrow \varphi}{\varphi}$$

In words, it states that if we can always prove that a property holds now assuming it holds next, then the property always holds. For example, if φ is "I am happy" and $\circ$ is "tomorrow", then coinduction allows us to infer "I am happy" provided that we are able to prove "if tomorrow I am happy then today I am happy". This may seem counter-intuitive at first, but it makes full sense in the context of finite traces with the weak interpretation of $\circ$. Indeed, suppose that $\circ\varphi \rightarrow \varphi$ holds for all finite traces. Since $\circ\varphi$ always holds on one-element traces, $\circ\varphi \rightarrow \varphi$ implies that all one-element traces satisfy φ. That implies that $\circ\varphi$ always holds on two-element traces, so $\circ\varphi \rightarrow \varphi$ implies that all two-element traces satisfy φ. We can thus inductively show that traces of any length satisfy φ.

As another example of coinduction, consider program verification of partial correctness using operational semantics, as advocated in [4, 19, 21]. There, program partial correctness is framed as (symbolic) reachability: the desired reachability property holds iff it holds on all finite paths starting with the current (symbolic) program configuration. Consider that our property φ in the ***coInd*** rule above is such a reachability property, and suppose that it refers to a loop. Then $\circ\varphi$ corresponds to the same reachability property holding in the next state, which in this approach is obtained by applying an operational semantics step, which in our case means unrolling the loop once. Proving $\circ\varphi \rightarrow \varphi$ corresponds to proving the original loop program assuming the desired loop property to hold after we unroll the loop once. In other words, checking symbolically the loop invariant property. If that holds, then we can safely assume that our original reachability property φ holds, in the partial correctness sense. Indeed, if the loop does not terminate, then any reachability property can be proved for it using ***coInd*** (similar to Hoare logic).

Our ***coInd*** rule is reminiscent of the Gödel-Löb theorem/axiom, which is at the heart of provability logic [1], where the modality means "provable". We are not aware of other uses of a coinductive, Gödel-Löb-style proof rule in the context of program verification.

We show that ***coInd*** is strictly more powerful than ***Ind***, by showing that it is equivalent to ***Ind*** *plus* $\Diamond\circ\bot$ (Proposition 4), and that dropping implication $\circ\neg\varphi \rightarrow \neg\circ\varphi$ from the proof system of infinite-trace LTL and replacing ***Ind*** with ***coInd*** yields a complete proof system for finite-trace LTL. Technically, the contribution is an almost complete reworking of the infinite-trace LTL decidability and completeness results, to adapt them to finite-trace LTL. The general organization and structure of our proofs follow [16].

Section 2 recalls basic facts about propositional, modal, and linear temporal logics. The syntax and semantics of finite-trace LTL are defined in Section 3. Section 5 defines a variant of formula closure and shows the decidability of the satisfiability problem. In fact, the decidability result is an immediate corollary of a major result of the paper, Theorem 1, which characterizes the satisfiable formulae as those admitting *complete* atom traces; this result is crucial not only for decidability, but also for completeness. Section 6 introduces our seven-rule sound proof system and proves several properties of it. Finally, Section 7 proves the completeness of our proof system. Section 8 concludes.

2 Preliminaries

In this section we remind some basic notions and notations about propositional and modal logic, as well as a sound and complete proof system for infinite-trace LTL.

Propositional Logic Propositions are built with propositional variables from a countable set *PVar*, a constant symbol $\bot$ (false), and a binary operation $\rightarrow$ (implication). Other derived operations include: $\neg$ (negation), $\wedge$ (and), $\vee$ (or), $\leftrightarrow$ (equivalence). The proof system in Fig. 1 (with axiom and proof rule *schemata*) is sound and complete for propositional logic (***MP*** stands for *modus ponens*). To distinguish it from other deducibility relations, we let $\vdash_{MP}$ denote the deducibility relation associated to the proof system above. The Deduction Theorem of propositional logic states that $\Gamma \vdash_{MP} \varphi_1 \rightarrow \varphi_2$ iff $\Gamma \cup \{\varphi_1\} \vdash_{MP} \varphi_2$. There are many equivalent proof systems for propositional logic, and all can be used in this paper in a similar way. We let ***Prop*** denote the set of all *theorems* of propositional logic, i.e., ***Prop*** $= \{\varphi \mid \vdash_{MP} \varphi\}$.

$\boldsymbol{A_1}$	$\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_1)$
$\boldsymbol{A_2}$	$(\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)) \rightarrow ((\varphi_1 \rightarrow \varphi_2) \rightarrow (\varphi_1 \rightarrow \varphi_3))$
$\boldsymbol{A_3}$	$(\neg\varphi_1 \rightarrow \neg\varphi_2) \rightarrow (\varphi_2 \rightarrow \varphi_1)$
MP	$\dfrac{\varphi_1 \qquad \varphi_1 \rightarrow \varphi_2}{\varphi_2}$

Fig. 1. Propositional logic proof system

Modal Logic In this paper we build upon the modal logic $\mathcal{K}$ (see [10] for a thorough presentation and history of modal logics, using a modern notation), whose syntax is:

$$\begin{array}{rl} \varphi ::= & \text{propositional logic variables } (\mathit{PVar}) \text{ and constructs} \\ \mid & \Box\varphi \quad (\Diamond\varphi \text{ commonly used as syntactic sugar for } \neg\Box\neg\varphi) \end{array}$$

The $\mathcal{K}$ modal logic is governed by the axiom and proof rule in Fig. 2, which together with the propositional logic proof system in Fig. 1, yield a sound and complete proof system for frame models (not discussed here; see, for example, [10]). $\mathcal{K}$ is typically enriched with additional axioms and/or proof rules. A notable axiom is $\Box\varphi \rightarrow \Box\Box\varphi$, which turns $\mathcal{K}$ into the logic known as S_4.

$\boldsymbol{K}$	$\Box(\varphi \rightarrow \varphi') \rightarrow (\Box\varphi \rightarrow \Box\varphi')$
$\boldsymbol{N}$	$\dfrac{\varphi}{\Box\varphi}$

Fig. 2. Modal logic proof system

An interesting modal logic extension, which is at the core of *provability logic* [1] where $\Box\varphi$ means "φ provable", is with the Gödel-Löb axiom $\Box(\Box\varphi \rightarrow \varphi) \rightarrow \varphi$, abbreviated ***GL***. It can be easily shown that ***GL*** makes the proof rule

$$\frac{\Box\varphi \rightarrow \varphi}{\varphi}$$

sound, but the converse is not true: one cannot prove ***GL*** from $\mathcal{K}$ plus the rule above.

A Proof System for Infinite-Trace LTL Several different proof systems for infinite-trace LTL can be found in the published literature and in class lecture notes at various institutions, with no well-established winner. Our proof system is inspired from the infinite-trace LTL proof system in Fig. 3.

This proof system appears in unpublished lecture notes by Dam and Guelev, reachable from `http://www.csc.kth.se/~mfd`. They credit it to [16] (personal communication), although in our opinion there are several important differences between the two. The proof system in Fig. 3 is in fact quite close to the one in [9], the only difference being that the latter includes a fixed point axiom for $\Box$, in the style of $\boldsymbol{U}_2$ in Fig. 3, which, as shown by Dam and Guelev in their lecture notes, can in fact be derived. Note that ***Ind*** is given as an axiom rather than as a proof rule, but one can show them equivalent. We used the subscript s to the until operator to make it clear that *strict* until is meant. In our proof system for finite-trace LTL we prefer to work with weak until, which allows us to eliminate $\boldsymbol{U}_1$.

proof system of propositional calculus, extended with the following:

$\boldsymbol{K}_\circ$	$\circ(\varphi \to \varphi') \to (\circ\varphi \to \circ\varphi')$
$\boldsymbol{N}_\circ$	$\dfrac{\varphi}{\circ\varphi}$
$\boldsymbol{K}_\Box$	$\Box(\varphi \to \varphi') \to (\Box\varphi \to \Box\varphi')$
$\boldsymbol{N}_\Box$	$\dfrac{\varphi}{\Box\varphi}$
Fun	$\neg\circ\varphi \leftrightarrow \circ\neg\varphi$
$\boldsymbol{U}_1$	$\varphi_1 \mathcal{U}_s \varphi_2 \to \Diamond\varphi_2$
$\boldsymbol{U}_2$	$\varphi_1 \mathcal{U}_s \varphi_2 \leftrightarrow \varphi_2 \vee \varphi_1 \wedge \circ(\varphi_1 \mathcal{U}_s \varphi_2)$
Ind	$\Box(\varphi \to \circ\varphi) \to (\varphi \to \Box\varphi)$

Fig. 3. Infinite-trace LTL proof system

3 Finite-Trace LTL: Syntax and Semantics

Here we introduce the basic elements of finite-trace LTL. For notational simplicity, from here on we refer to finite-trace LTL as $\mathcal{L}$. Its core syntax is the same as that of infinite-trace LTL, that is, it consists of a unary "next" operator and of a binary"until":

$$\begin{array}{rcl} \varphi & ::= & \text{usual propositional constructs} \\ & | & \circ\varphi \quad \text{(next)} \\ & | & \varphi \mathcal{U} \varphi \quad \text{(until)} \end{array}$$

However, the semantics is given in terms of finite-traces, where for technical simplicity both operators are interpreted *weakly*. That is, $\circ\varphi$ means: *if* there is a next state *then* φ holds in that state; and $\varphi_1 \mathcal{U} \varphi_2$ means: either φ_1 holds in all future states or there is some future state in which φ_2 holds and φ_1 holds in each state until then. Formally,

Definition 1. *A **finite trace** is an element of $\mathcal{P}(PVar)^+$, that is, a non-empty finite sequence of sets of propositional variables (each such set can be thought of as a "state"). We inductively define the **satisfaction relation** between finite-traces and formulae:*

$s_1 \dots s_n \models p$ *iff* $p \in s_1$;
$s_1 \dots s_n \not\models \bot$;
$s_1 \dots s_n \models \varphi_1 \to \varphi_2$ *iff* $s_1 \dots s_n \models \varphi_1$ *implies* $s_1 \dots s_n \models \varphi_2$;
$s_1 \dots s_n \models \circ\varphi$ *iff* $n = 1$ *or* $s_2 \dots s_n \models \varphi$;
$s_1 \dots s_n \models \varphi_1 \mathcal{U} \varphi_2$ *iff either* $s_i \dots s_n \models \varphi_1$ *for all* $1 \le i \le n$ *or there is some* $1 \le i \le n$ *such that* $s_i \dots s_n \models \varphi_2$ *and* $s_j \dots s_n \models \varphi_1$ *for all* $1 \le j < i$.

*Formula φ is **satisfiable** iff there exists some finite trace $s_1 \dots s_n$ such that $s_1 \dots s_n \models \varphi$, and is **valid**, or a **tautology**, written $\models \varphi$, iff $s_1 \dots s_n \models \varphi$ for all finite traces $s_1 \dots s_n$.*

We can now extend the syntax with several derived operators:

$\varphi ::= \bullet\varphi$	(strong next)	$\bullet\varphi \equiv \neg\circ\neg\varphi$
$\mid \Box\varphi$	(always)	$\Box\varphi \equiv \varphi \mathcal{U} \bot$
$\mid \Diamond\varphi$	(eventually)	$\Diamond\varphi \equiv \neg\Box\neg\varphi$
$\mid \varphi \mathcal{U}_s \varphi$	(strong until)	$\varphi_1 \mathcal{U}_s \varphi_2 \equiv \Diamond\varphi_2 \wedge \varphi_1 \mathcal{U} \varphi_2$

It can be easily shown that these operators have the expected semantics:

$s_1 s_2 \dots s_n \models \bullet\varphi$ iff $n > 1$ and $s_2 \dots s_n \models \varphi$;
$s_1 \dots s_n \models \Box\varphi$ iff $s_i \dots s_n \models \varphi$ for all $1 \leq i \leq n$;
$s_1 \dots s_n \models \Diamond\varphi$ iff $s_i \dots s_n \models \varphi$ for some $1 \leq i \leq n$;
$s_1 \dots s_n \models \varphi_1 \mathcal{U}_s \varphi_2$ iff there is some $1 \leq i \leq n$ such that $s_i \dots s_n \models \varphi_2$ and $s_j \dots s_n \models \varphi_1$ for all $1 \leq j < i$.

It can also be easily shown that $\models \circ\varphi \leftrightarrow \neg\bullet\neg\varphi$, that is, $\circ$ and $\bullet$ are completely dual to each other. In the rest of the paper some results are easier to formulate and/or prove using the weak version of next, $\circ$, while others using the strong version, $\bullet$. Since we can easily and linearly convert a formula to use either one or the other, we will simply state which one we assume as basic construct at the beginning of each relevant section.

Another relevant and easy to prove tautology is $\models \psi_1 \mathcal{U} \psi_2 \leftrightarrow \psi_2 \vee \psi_1 \wedge \circ(\psi_1 \mathcal{U} \psi_2)$.

Example 1. Consider a system which performs one or more actions a followed by an action b. We want to show that whenever the system terminates, b is eventually reached. We can specify both the system and the property as the following formula:

$$\Box(a \rightarrow \bullet(a \vee b)) \rightarrow (a \rightarrow \Diamond b)$$

In words, the system is described as the formula stating that once an action a takes place then a next step *must* exist, and in that step a or b takes place. If we additionally want to state that the trace must terminate as soon as b takes place, then we write:

$$\Box((a \rightarrow \bullet(a \vee b)) \wedge (b \rightarrow \circ\bot)) \rightarrow (a \rightarrow \Diamond b)$$

For now, we can show that the above formulae are valid using Definition 1 directly. Section 6 gives a proof system which will allow us to formally derive any tautologies. Note that none of these holds under infinite-trace LTL, since a^ω does not satisfy them.

Example 2. We can, in fact, associate a formula $\varphi_{\mathcal{A}}$ over propositional variables $Q \cup A$ to any finite-state machine $FSM = (Q, A, q_0 \in Q, \delta : Q \times A \rightarrow 2^Q, F \subseteq Q)$ as follows:

$$\Box\left(\left(\bigvee_{q \in Q} q\right) \wedge \bigwedge_{q \in Q}\left(\left(q \rightarrow ((q \in F \wedge \circ\bot) \vee \bigvee_{\substack{a \in A \\ \delta(q,a) \neq \emptyset}} a)\right) \wedge \left(\bigwedge_{\substack{a \in A \\ \delta(q,a) \neq \emptyset}} q \wedge a \rightarrow \bullet \bigvee_{q' \in \delta(q,a)} q'\right)\right)\right)$$

In words, it is always the case that: (1) a state in Q is active; (2) for each final state, allow the trace to terminate there ($\circ\bot$ holds iff there is no next step); (3) for each state q which is active, its outgoing edges are also active; and (4) if a state q and an action a of that state are both active, then a step must take place to a state allowed by *FSM*.

It can be shown that a word $a_1 \dots a_n$ is in the language of *FSM* iff there are states $q_1, \dots, q_n \in Q$ such that $q_n \in F$ and $\{q_0, a_1\}\{q_1, a_2\} \dots \{q_{n-1}, a_n\}\{q_n\} \models \varphi_{FSM}$. This allows us to prove properties about *FSM* (either directly using Definition 1 or using the subsequent proof system), such as: $\models \varphi_{\mathcal{A}} \rightarrow \Box(q_0 \rightarrow \Diamond a)$ (that is, a will be reached on any terminating path starting from q_0), or $\models \varphi_{\mathcal{A}} \rightarrow \Box(a \rightarrow \Diamond b)$ (that is, b will be reached on any terminating path starting with a from any state), etc.

4 Relationship to Infinite-Trace LTL

Before we proceed to present our novel results starting with Section 5, it is worth discussing alternative, indirect approaches to reason about finite-trace LTL properties. We do it in this section, at the same time also arguing for a direct approach.

There is a relatively simple way to transform any LTL formula into another LTL formula so that the former is satisfiable under the finite-trace semantics iff the latter is satisfiable under infinite-trace semantics. The idea is to conceptually complete finite traces with infinite suffixes $\$^\omega$, where \$ is a new propositional variable thought of as "nothing". Formally, given φ, let $\overline{\varphi}$ be the formula defined as follows:

$$\begin{array}{rcll} \overline{\bot} & = & \bot & \\ \overline{p} & = & p \wedge \neg\$ & \text{where } p \text{ is a propositional variable} \\ \overline{\varphi_1 \rightarrow \varphi_2} & = & \overline{\varphi_1} \rightarrow \overline{\varphi_2} & \\ \overline{\circ\varphi} & = & \circ(\overline{\varphi} \vee \$) & \\ \overline{\varphi_1 \mathcal{U} \varphi_2} & = & \overline{\varphi_1} \mathcal{U} (\overline{\varphi_2} \vee \$) & \end{array}$$

Then φ is satisfiable in finite-trace LTL iff $(\neg\$)\mathcal{U}_s\$ \wedge \overline{\varphi}$ is satisfiable in infinite-trace LTL. For example, the formula (recall that $\Box\varphi$ is syntactic sugar for $\varphi\mathcal{U}\bot$, and $\Diamond\varphi$ for $\neg\Box\neg\varphi$)

$$\Box(a \rightarrow \bullet(a \vee b)) \rightarrow (a \rightarrow \Diamond b)$$

in Example 1 is satisfiable in finite-trace LTL iff

$$(\neg\$)\mathcal{U}_s\$ \ \wedge \ ((a \wedge \neg\$ \rightarrow \circ((a \vee b) \wedge \neg\$))\mathcal{U}\$ \ \rightarrow \ (a \wedge \neg\$ \rightarrow \neg((\neg(b \wedge \neg\$))\mathcal{U}\$)))$$

is satisfiable in infinite-trace LTL. Therefore, finite-trace LTL is PSPACE-decidable, like infinite-trace LTL [22], and a decidable procedure can be obtained by translation to infinite-trace LTL as above.

Following such a translation approach has, however, an important practical drawback: the size of the formula doubles, and a more complex than needed procedure is applied on the larger formula. Indeed, as seen in Section 5, our specialized decision procedure for finite-trace LTL reduces to checking simple reachability in a graph exponential in the size of φ, as opposed to checking for algorithmically more complex, ultimately periodic sequences in a graph exponential in twice the size of φ, as the translation to infinite-trace LTL approach would require.

Also, it can be shown that φ is valid in finite-trace LTL iff $(\neg\$)\mathcal{U}_s\$ \rightarrow \overline{\varphi}$ is valid in infinite-trace LTL. Therefore, one can indirectly obtain a complete proof system for finite-trace LTL by translation to infinite-trace LTL and then using off-the-shelf proof systems for the latter, for example [16, 17] (see also Section 2). Besides having to prove a twice-larger formula, this translation-to-infinite-trace-LTL approach has the additional drawback that we now have to explicitly reason about \$ and termination of traces, departing ourselves from the basic intuitions of finite-trace LTL. For example, it seems hard to find a proof of the infinite-trace LTL formula corresponding to the finite-trace formula in Example 1, while as shown in Example 3 there is simple, direct and intuitive proof of the original formula using our new proof system.

Arguments like the above, in favor of direct procedures and reasoning systems for specific logics instead of translations to other logics, abound in the literature. Consider, for example, conventional infinite-trace LTL and its well-known translation to (a monadic fragment of) first-order logic (FOL), suggested for the first time by Kamp in his seminal 1968 thesis [14]. Specifically, each LTL formula φ can be inductively translated to an equivalent (in appropriate models) FOL formula $\varphi(x)$ over free variable x; for example,

$$(\varphi\ \mathcal{U}_s\ \psi)(x) \text{ is the FOL formula } \exists z\ .\ x < z \wedge \psi(z) \wedge \forall y\ .\ x < y < z \rightarrow \varphi(y)$$

Then we can use existing or develop new procedures for that fragment of FOL to decide satisfiability of LTL formulae, and we can use the FOL sound and complete proof system to derive any tautology of LTL. Despite the above, significant research and development effort has been spent since 1968 by the formal verification and analysis community to develop specialized, direct decision procedures and proof systems for LTL. Similarly and perhaps even more interestingly, equational logic is a well-established fragment of FOL, yet almost no equational provers are based on FOL reasoning, but on procedures and sound and complete proof systems specifically crafted for equational logic.

To push the argument to extreme, consider the seminal result by Bergstra and Tucker [3]: any computable domain, of any complexity class, is isomorphic to the initial model of a finite set of equations. Therefore, inductive equational proofs are sufficient to reason within any domain, regardless of its complexity. Such results, in spite of their beauty and insights, tend to have little practical relevance and have certainly not stopped, nor even slowed down the research and development of particular decision procedures and proof systems for particular logics. The fact that finite-trace LTL can be translated to infinite-trace LTL falls into the same category and it is, in our view, no more than an interesting observation. While one can attempt to use decision procedures and proof systems for infinite-trace LTL via the translation to infinite-trace LTL discussed above, we believe that finite trace LTL is a pivotal logic for runtime verification and thus deserves our full attention. Decision procedures and specialized sound and complete proof systems for it will provide the runtime verification researchers with understanding and insights that should carry over to other finite-trace specification formalisms, too.

5 Complete Atom Traces

In this section we show our first important result for finite-trace LTL ($\mathcal{L}$): a formula φ is satisfiable iff there is a complete (i.e., finite and terminated) trace in the tableaux of φ,

where the tableaux is constructed in a way specific to the finite-trace semantics. This also gives a direct decision procedure for finite-trace LTL satisfiability, but the result is particularly important for completeness. Here we prefer to work with $\bullet$ and $\mathcal{U}$ as core formula constructs, so we assume that $\mathcal{L}$ formulae are built with: propositional variables in $PVar$, $\bot$, $\rightarrow$, $\bullet$ and $\mathcal{U}$. As notational convenience, we use $\neg\varphi$ as a shortcut for $\varphi \rightarrow \bot$.

Definition 2. *Let $\neg'\varphi$ be either φ' when φ is $\neg\varphi'$, or $\neg\varphi$ otherwise. A set of formulae C is $\{\neg\}$-**closed** when $\varphi \in C$ implies $\neg'\varphi \in C$, is $\{\bullet\}$-**closed** when $\bullet\varphi \in C$ implies $\varphi \in C$, and is **closed** when: (1) is $\{\neg, \bullet\}$-closed; (2) $\varphi_1 \rightarrow \varphi_2 \in C$ implies $\varphi_1, \varphi_2 \in C$; (3) $\bullet\varphi \in C$ implies $\bullet\neg'\varphi \in C$; and (4) $\varphi_1 \mathcal{U} \varphi_2 \in C$ implies φ_1, φ_2, $\bullet(\varphi_1 \mathcal{U} \varphi_2) \in C$. If φ is a formula then $Closure(\varphi)$ is the smallest closed set that includes φ.*

Note that our notion of closure is slightly stronger than the classic Fischer-Ladner closure [7], in that Fischer-Ladner does *not* require condition (3), namely that $\bullet\neg'\varphi$ is included in the closure together with $\bullet\varphi$. Thus, our closures of formulae will be slightly larger than Fischer-Ladner's, but nevertheless still linear in the formula. For example, if $\varphi_1 \mathcal{U} \varphi_2 \in C$ then also $\bullet\neg(\varphi_1 \mathcal{U} \varphi_2) \in C$, which is critical for the proof of Theorem 1.

Definition 3. *Let C be a $\{\bullet\}$-closed set of formulae. The $\{\bullet\}$-**generated transition relation of** C, written $R_C \subseteq \mathcal{P}(C) \times \mathcal{P}(C)$, is defined as follows: for any $A, B \subseteq C$, $(A, B) \in R_C$ iff $\bullet^{-1}A \neq \emptyset$ and $\bullet^{-1}A \subseteq B$, where $\bullet^{-1}A = \{\psi \mid \bullet\psi \in A\}$. A **complete C-trace** is a sequence $A_1 \ldots A_n$ of subsets of C with $(A_i, A_{i+1}) \in R_\varphi$ for all $1 \leq i < n$ and $\bullet^{-1}A_n = \emptyset$.*

The reason C was required to be $\{\bullet\}$-closed in Definition 3, is because we want $\bullet^{-1}A \neq \emptyset$ to imply that there is some $B \subseteq C$ such that $(A, B) \in R_C$. In other words, we want the emptyness of $\bullet^{-1}A$ alone to determine whether A is terminal for R_C or not.

Definition 4. *Let C be a closed set of formulae. A **C-atom** is a set $A \subseteq C$ such that:*

1. *$\bot \notin A$;*
2. *For each $\psi \in C$, either $\psi \in A$ or $\neg'\psi \in A$;*
3. *If $\bullet^{-1}A \neq \emptyset$ then for each $\bullet\psi \in C$, either $\bullet\psi \in A$ or $\bullet\neg'\psi \in A$;*
4. *For each $\psi_1 \rightarrow \psi_2 \in C$, $\psi_1 \rightarrow \psi_2 \in A$ iff ($\psi_1 \in A$ implies $\psi_2 \in A$);*
5. *For each $\psi_1 \mathcal{U} \psi_2 \in C$, $\psi_1 \mathcal{U} \psi_2 \in A$ iff $\psi_2 \in A$ or $\psi_1 \in A$ and $\bullet\neg(\psi_1 \mathcal{U} \psi_2) \notin A$.*

*Let $Atom_C$ be the set of C-atoms. If $C = Closure(\varphi)$ for some formula φ, then we write $Atom_\varphi$ instead of $Atom_{Closure(\varphi)}$ and R_φ instead of $R_{Closure(\varphi)}$. Also, a **complete atom trace of** φ is a complete $Closure(\varphi)$-trace $A_1 \ldots A_n$ such that $A_1, \ldots, A_n \in Atom_\varphi$ and $\varphi \in A_1$.*

The next theorem is a crucial result of finite-trace semantics, which is used to show both the decidability (Corollary 1) and the completeness (Theorem 4) of $\mathcal{L}$.

Theorem 1. *A formula is satisfiable iff it admits a complete atom trace.*

Theorem 1 gives us a straightforward algorithm to test the satisfiability of a formula φ: show that there is at least one node $A \in Atom_\varphi$ in the (finite) graph $(Atom_\varphi, R_\varphi)$ with $\varphi \in A$, such that there is some path from A to a node without any outgoing edges. In other words, the satisfiability problem of φ reduces to the reachability problem in graph $(Atom_\varphi, R_\varphi)$, which is decidable. Thus, like infinite-trace LTL [22], finite-trace LTL is also decidable. Although checking reachability is algorithmically simpler than checking for ultimately periodic sequences as needed for infinite-trace LTL [22], deciding finite-trace LTL satisfiability is still a PSPACE-complete problem:

Corollary 1. *The satisfiability problem for $\mathcal{L}$ is PSPACE-complete.*

6 Proof System

Fig. 4 depicts our proof system for finite trace LTL. In this section we prefer to work with $\circ$ instead of $\bullet$ as core construct (so $\bullet\varphi$ desugars to $\neg\circ\neg\varphi$). We start by inheriting propositional logic and the modal logic rules corresponding to the modalities $\circ$ and $\Box$. Unlike in infinite-trace LTL, $\neg\circ\varphi \leftrightarrow \circ\neg\varphi$ does not hold anymore, as both $\circ\varphi$ and $\circ\neg\varphi$ hold in one-state traces; only the implication $\neg\circ\varphi \rightarrow \circ\neg\varphi$ holds. In interesting multi-modal logics, the various modal operators tend to be connected somehow. In our case, we axiomatize the expected fact that $\varphi_1 \mathcal{U} \varphi_2$ is the fixed-point of the formula $X \leftrightarrow \varphi_2 \vee \varphi_1 \wedge \circ X$. The only unexpected rule is the Coinduction rule for $\circ$. As usual, the axioms and rules are schemata. The fixed-point equivalence of $\Box$, $\Box\varphi \leftrightarrow \varphi \wedge \circ\Box\varphi$, is an instance of ***Fix*** with $\varphi_1 \mapsto \varphi$ and $\varphi_2 \mapsto \bot$. To avoid inventing rule names, from now on we take the liberty to let ***Fix*** also refer to the latter equivalence. In fact, if one prefers a fragment of LTL with only $\circ$ and $\Box$, then one can replace ***Fix*** with the fixed-point equivalence of $\Box$ and the results in this paper still hold.

proof system of propositional calculus, extended with the following:

$\boldsymbol{K}_\circ$	$\circ(\varphi \rightarrow \varphi') \rightarrow (\circ\varphi \rightarrow \circ\varphi')$
$\boldsymbol{N}_\circ$	$\dfrac{\varphi}{\circ\varphi}$
$\boldsymbol{K}_\Box$	$\Box(\varphi \rightarrow \varphi') \rightarrow (\Box\varphi \rightarrow \Box\varphi')$
$\boldsymbol{N}_\Box$	$\dfrac{\varphi}{\Box\varphi}$
$\neg\circ$	$\neg\circ\varphi \rightarrow \circ\neg\varphi$
Fix	$\varphi_1 \mathcal{U} \varphi_2 \leftrightarrow \varphi_2 \vee \varphi_1 \wedge \circ(\varphi_1 \mathcal{U} \varphi_2)$
coInd	$\dfrac{\circ\varphi \rightarrow \varphi}{\varphi}$

Fig. 4. Finite-trace LTL proof system

Comparing our proof system above with the one for infinite-trace LTL in Section 2, we note that the main difference is that the Induction rule has been replaced with the Coinduction rule. Also, the axiom $\circ\neg\varphi \rightarrow \neg\circ\varphi$ has been removed, and since we chose to work with weak instead of strong until we were able to also remove rule $\boldsymbol{U}_1$. We argue, without proof, that our proof system above is minimal. Indeed, the rules $\boldsymbol{K}_\circ$, $\boldsymbol{N}_\circ$, $\boldsymbol{K}_\Box$, and $\boldsymbol{N}_\Box$ say that the $\circ$ and $\Box$ modalities form $\mathcal{K}$ logics, and $\mathcal{K}$ is the poorest modal logic. The axiom $\neg\circ$ captures the specific one-step granularity of $\circ$, which distinguishes it from $\Box$ for example, so it is unlikely to eliminate it. ***Fix*** captures the recursive nature of the until operator, and it is the only axiom which does it, so again it is unlikely to be removed. Finally, note that none of the rules discussed so far is specific to finite traces, because they are in fact consequences of the infinite-trace LTL proof system, so at least one more rule is needed to allow proving finite-trace-specific properties like $\Diamond\circ\bot$. The ***coInd*** rule not only allows proving $\Diamond\circ\bot$, but as shown in Proposition 4 it also allows proving the Induction proof rule of infinite-trace LTL (which therefore also holds for finite-traces), a rule which is considered crucial for LTL and, indeed, no proof system for LTL omits it.

Let $\vdash_{\mathcal{L}}$ denote the induced deducibility relation. Specifically, if Γ is a set of formulae and φ a formula, then $\Gamma \vdash_{\mathcal{L}} \varphi$ denotes that φ is deducible from Γ using the proof system above; $\vdash_{\mathcal{L}} \varphi$ is a shortcut for $\emptyset \vdash_{\mathcal{L}} \varphi$. Let $Th_{\mathcal{L}} = \{\varphi \mid \vdash_{\mathcal{L}} \varphi\}$ be the set of all *theorems* of $\mathcal{L}$. For notational simplicity, we let ***Prop*** also denote the set of all formulae (not only propositions) deducible with the propositional logic proof subsystem; e.g., $\Box\varphi \to (\circ\varphi \to \Box\varphi) \in$ ***Prop*** (instance of $\boldsymbol{A}_1$ with formulae in $\mathcal{L}$). Note that ***Prop*** $= \{\varphi \mid \vdash_{MP} \varphi\} \subset Th_{\mathcal{L}}$.

Theorem 2. ***(Soundness)*** *For any formula φ, $\vdash_{\mathcal{L}} \varphi$ implies $\models \varphi$. In particular, $\bot \notin Th_{\mathcal{L}}$.*

Fig. 5 shows a few basic properties of the next operators, which can be shown using only the $\{\boldsymbol{K}_\circ, \boldsymbol{N}_\circ, \neg\circ\}$ fragment of the proof system.

Proposition 1. *The formulae in Fig. 5 are all derivable, i.e., belong to $Th_{\mathcal{L}}$.*

The following says that the $\Box$ modality is S_4:

Proposition 2. $\vdash_{\mathcal{L}} \Box\varphi \to \Box\Box\varphi$ *for any formula φ.*

$$\circ\top$$
$$\neg\bullet\bot$$
$$\circ\bot \to \circ\varphi$$
$$\bullet\varphi \to \bullet\top$$
$$\circ\varphi \leftrightarrow \bullet\varphi \vee \circ\bot$$
$$\bullet\varphi \leftrightarrow \circ\varphi \wedge \bullet\top$$
$$\bullet\top \to (\circ\varphi \leftrightarrow \bullet\varphi)$$
$$\bullet\top \to (\neg\circ\varphi \leftrightarrow \circ\neg\varphi)$$
$$\circ(\varphi_1 \wedge \varphi_2) \leftrightarrow \circ\varphi_1 \wedge \circ\varphi_2$$
$$\bullet(\varphi_1 \vee \varphi_2) \leftrightarrow \bullet\varphi_1 \vee \bullet\varphi_2$$
$$\circ(\varphi_1 \vee \varphi_2) \leftrightarrow \circ\varphi_1 \vee \circ\varphi_2$$
$$\bullet(\varphi_1 \wedge \varphi_2) \leftrightarrow \bullet\varphi_1 \wedge \bullet\varphi_2$$
$$\circ(\varphi_1 \to \varphi_2) \leftrightarrow (\circ\varphi_1 \to \circ\varphi_2)$$
$$\bullet(\varphi_1 \to \varphi_2) \to (\bullet\varphi_1 \to \bullet\varphi_2)$$

Fig. 5. Properties of $\circ$ and $\bullet$

The deduction theorem of propositional logic, stating that $\Gamma \vdash_{MP} \varphi \to \psi$ iff $\Gamma, \varphi \vdash_{MP} \psi$, is technically unnecessary but quite useful in practice, because it allows us to prove implications by "assuming" their hypothesis and then deriving their conclusion. We would like to also have it in our setting here. However, it is well-known that the deduction theorem does not hold by default in other logics. For example, in first-order logic, it only holds when φ is a closed formula (i.e., it has no free variables). Here we can prove the following variant of the deduction theorem:

Theorem 3. ***(Deduction theorem)*** *$\Gamma \vdash_{\mathcal{L}} \Box\varphi \to \psi$ iff $\Gamma, \varphi \vdash_{\mathcal{L}} \psi$.*

When doing proofs by induction, it is often convenient to assume the property holds in *all* past moments, and not only in the *previous* one, and then prove it holds now. Dually, when doing proofs by coinduction, it is often convenient to assume the property holds in *all* future moments, and not only in the *next* one, and then prove it holds now.

The following proposition establishes that this apparently stronger variant of coinduction is in fact equivalent to the one we have now. It also gives equivalent axiomatic variants of both coinductive proof rules.

Proposition 3. *Keeping all the other axioms and rules unchanged, the rule* ***coInd*** *is equivalent to any of the alternative rules or axioms in Fig. 6.*

$\boldsymbol{coInd}_\Box$	$\dfrac{\circ\Box\varphi \to \varphi}{\varphi}$
$\boldsymbol{GL}_\circ$	$\Box(\circ\varphi \to \varphi) \to \varphi$
$\boldsymbol{GL}_\Box$	$\Box(\circ\Box\varphi \to \varphi) \to \varphi$

Fig. 6. Other coinductive rules

A natural question is what is the relationship between induction and coinduction. Induction is valid for infinite-traces, too, which means that it is not powerful enough to prove $\Diamond\circ\bot$ (each trace terminates); indeed, $\Diamond\circ\bot$ is equivalent to $\bot$ in infinite-trace LTL. On the other hand, coinduction, as formulated here, is only valid for finite traces. We next show that coinduction is actually equivalent to *both* induction and the finite trace axiom $\Diamond\circ\bot$ *together*:

Ind	$\dfrac{\varphi \rightarrow \circ\varphi}{\varphi \rightarrow \Box\varphi}$
Fin	$\Diamond\circ\bot$

Fig. 7. Induction rule and finite-trace axiom

Proposition 4. *Let **Ind** be the induction rule and **Fin** be the finite-trace axiom in Fig. 7. Keeping all the other rules unchanged, **coInd** is equivalent to **Ind** and **Fin** together.*

The properties in Fig. 8 are quite useful in practice.

Proposition 5. *The formulae in Fig. 8 are all derivable, i.e., belong to $Th_{\mathcal{L}}$.*

$$\Box\varphi \leftrightarrow \Box\Box\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\Box\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2$$
$$\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

Fig. 8. Properties of $\Box$ and $\Diamond$

Example 3. Let us prove the property in Example 1, $\Box(a \rightarrow \bullet(a \vee b)) \rightarrow (a \rightarrow \Diamond b)$. By the Deduction Theorem 3, it suffices to show $a \rightarrow \bullet(a \vee b) \vdash_{\mathcal{L}} a \rightarrow \Diamond b$. This follows by the ***coInd*** proof rule, if we can show $a \rightarrow \bullet(a \vee b) \vdash_{\mathcal{L}} \circ(a \rightarrow \Diamond b) \rightarrow (a \rightarrow \Diamond b)$. By propositional reasoning, it suffices to show $\vdash_{\mathcal{L}} \bullet(a \vee b) \wedge \circ(a \rightarrow \Diamond b) \rightarrow \Diamond b$, which follows by $\boldsymbol{K}_{\circ}$, propositional reasoning, and some theorems in Propositions 1 and 5.

7 Completeness

In this section we show that the proof system discussed in Section 6 is complete for finite-trace LTL ($\mathcal{L}$). The general proof scheme adopted in this section is standard: assume that φ is valid but not derivable, which implies that $\neg\varphi$ is consistent, and then use the proof system to construct a model of $\neg\varphi$ within the atom universe of the tableaux, thus contradicting the validity of φ. Like in Section 5, we here also prefer to work with $\bullet$ as a basic "next" construct instead of $\circ$. Recall that $\vdash_{MP}$ is the deducibility relation using only the proof subsystem of propositional logic.

Consistency, maximal consistency and related results are given below, following a pattern common to many logics (propositional logic, FOL, infinite-trace LTL, etc.).

Definition 5. *Γ is **inconsistent** iff $Th_{\mathcal{L}} \cup \Gamma \vdash_{MP} \bot$, and it is **consistent** otherwise. A formula φ is **consistent** (resp. **inconsistent**) iff $\{\varphi\}$ is consistent (resp. inconsistent). Γ is **maximally consistent** iff Γ is consistent and if Γ' consistent with $\Gamma \subseteq \Gamma'$ then $\Gamma = \Gamma'$.*

Therefore, Γ is inconsistent iff we can derive $\bot$ using only propositional reasoning, but all the theorems of finite-trace LTL. Once we can derive $\bot$, we can derive anything:

Proposition 6. *Γ is inconsistent iff $Th_{\mathcal{L}} \cup \Gamma \vdash_{MP} \varphi$ for any formula φ.*

We can always add more formulae to a consistent set of formulae which is not maximal. Once maximal, we cannot add new formulae without breaking consistency:

Proposition 7. *Suppose that Γ is consistent and φ is any formula. Then:*
1. *$\Gamma \cup \{\varphi\}$ is consistent, or $\Gamma \cup \{\neg\varphi\}$ is consistent, or both;*
2. *If Γ is maximally consistent, then either $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$. In particular, $Th_{\mathcal{L}} \subseteq \Gamma$.*

In particular, no new formulae can be derived from a maximally consistent set:

Corollary 2. *If Γ is maximally consistent and φ is any formula, then $\Gamma \vdash_{MP} \varphi$ iff $\varphi \in \Gamma$.*

Proposition 8. *Suppose that Γ is maximally consistent. Then $\varphi_1 \to \varphi_2 \in \Gamma$ iff $\varphi_1 \in \Gamma$ implies $\varphi_2 \in \Gamma$, $\varphi_1 \wedge \varphi_2 \in \Gamma$ iff $\varphi_1 \in \Gamma$ and $\varphi_2 \in \Gamma$, $\varphi_1 \vee \varphi_2 \in \Gamma$ iff $\varphi_1 \in \Gamma$ or $\varphi_2 \in \Gamma$, and $\varphi_1 \mathcal{U} \varphi_2 \in \Gamma$ iff $\varphi_2 \in \Gamma$ or $\varphi_1 \in \Gamma$ and $\bullet\neg(\varphi_1 \mathcal{U} \varphi_2) \notin \Gamma$.*

Any consistent set of formulae can be extended into a maximally consistent one; folklore goes that a result of this kind was first shown for predicate logic by Lindenbaum in late 1920's (according to Taski):

Proposition 9. *Γ consistent implies there is a Γ' maximally consistent with $\Gamma \subseteq \Gamma'$.*

The results above in this section followed a standard pattern to prove completeness in several logics. The remaining results, however, are specific to finite-trace LTL ($\mathcal{L}$).

Recall from Definition 3 that $\bullet^{-1}\Gamma = \{\psi \mid \bullet\psi \in \Gamma\}$. The next proposition tells that $\bullet^{-1}$ preserves consistency. This, with the help of Proposition 9, allows us to start with a special consistent set of formulae and iteratively "derive" it with $\bullet^{-1}$; the difficult part is to show that, for finite-trace LTL, this derivation process *can be* finite. A result similar to Proposition 10 also exists for infinite-trace LTL (see, e.g., [16]), but our proof is more involved, because of the existence of two distinct next operators. In fact, a similar result for the weak next $\circ$ operator is not possible: for example, $\circ\bot$ is consistent but $\bot$ is not.

Proposition 10. *If Γ is consistent then $\bullet^{-1}\Gamma$ is also consistent.*

To prove the completeness, we will show that any consistent formula admits a complete atom trace (see Definition 4), so we can use Theorem 1 to conclude the formula is satisfiable. Like for infinite-trace LTL [16], it is convenient to consider a subset of the atoms of the formula, namely those obtained by intersecting its closure with maximally consistent sets of formulae. Let us define the worlds of a $\{\bullet\}$-closed set:

Definition 6. *Let C be a $\{\bullet\}$-closed set of formulae and let $W_C \subseteq \mathcal{P}(C)$ be the set $\{\Gamma \cap C \mid \Gamma$ maximally consistent $\}$, whose elements are called the **worlds of** C. Also, let $R_C^W \subseteq W_C \times W_C$ be the restriction of $R_C \subseteq \mathcal{P}(C) \times \mathcal{P}(C)$ to W_C.*

Proposition 10 and the $\{\bullet\}$-closedness of C guarantee that for any $w \in W_C$, $\bullet^{-1}w \neq \emptyset$ iff there is some $w' \in W_C$ such that $(w, w') \in R_C^W$. Now let us show that if C is closed then its worlds are indeed particular atoms. In particular, if C is a formula closure then its worlds are among the atoms that appear in the tableaux of the formula (see Section 5).

Proposition 11. *If C is closed then $W_C \subseteq Atom_C$.*

The next result tells that we can formally derive that a world can evolve to its successors, if any. A similar result also exists for infinite-trace LTL (see, e.g., [16]), but like before our proof is more involved due to the two distinct next operators available.

Proposition 12. *Let C be a finite and $\{\neg, \bullet\}$-closed set of formulae, and let $w \in W_C$ such that $\bullet^{-1} w \neq \emptyset$. Then $\vdash_{\mathcal{L}} \widehat{w} \to \bullet \bigvee_{(w,w') \in R_C^W} \widehat{w'}$, where $\widehat{A} = \bigwedge \{\psi \mid \psi \in A\}$ for any $A \subseteq C$.*

Unlike for infinite-trace LTL, where the objective is to show the existence of a ultimately periodic infinite atom trace that satisfies the formula, for finite-trace LTL the challenge is to show the existence of *any* finite trace that satisfies the formula. This is where our proof differs completely from that for infinite-trace LTL: we show that for any world $w \in W_C$, it is impossible to have only infinite R_C^W-sequences starting with w:

Proposition 13. *If C is finite and $\{\neg, \bullet\}$-closed, then for any $w \in W_C$ there exists some complete C-trace (see Definition 3) starting with w whose elements are all in W_C.*

We can now show that formula consistency and satisfiability coincide:

Proposition 14. *A formula is consistent iff it is satisfiable.*

The completeness theorem is now a simple corollary of the above:

Theorem 4. ***(Completeness)*** *For any formula φ, $\models \varphi$ implies $\vdash_{\mathcal{L}} \varphi$.*

8 Conclusion

This paper gave direct decidability and completeness results for finite-trace LTL. Neither the PSPACE-completeness of satisfiability nor the existence of a sound and complete proof system for finite-trace LTL are surprising results in themselves, because similar results exist for other variants of temporal logics. Moreover, the presented proof architecture follows the usual pattern encountered in infinite-trace variants of temporal logic, which itself follows a pattern well-established in first-order and predicate logics (for almost 100 years now). Looked at from that angle, this paper made two contributions, one conceptual and one technical. The conceptual contribution is the Coinduction proof rule, stating that if $\circ\varphi \to \varphi$ is provable then φ is also provable. It surprised the author that it captures so well the essence of finite-trace reasoning and yields its completeness. Its simplicity and elegance suggest that Coinduction may play a central role in finite-trace temporal reasoning. The technical contribution is Proposition 13, together with Proposition 12 on which it relies, saying that a consistent formula cannot admit only infinite-trace models; it must admit some finite-trace models, too, so the formula is finite-trace satisfiable. It may look "obvious" to the hasty reader now, after the fact, but the difficulty of proving these results made the author initially believe that finite-trace LTL may in fact not allow any complete proof system within itself, that is, without translation to other (richer) logics. This would have not been unheard of: equational logic restricted to *unconditional* equalities over regular expressions does *not* admit a finite axiomatization within itself, but it does admit one if we allow *conditional* equations. It could have just as well been the case that finite-trace LTL admitted no finite proof system within itself, in spite of its infinite-trace variants admitting finite proof systems.

References

1. S. N. Artemov and L. D. Beklemishev. Provability logic. In *Handbook of Philosophical Logic, Volume XIII*, pages 181–360. Springer-Verlag, Berlin, 2 edition, 2005.

2. A. Bauer, M. Leucker, and C. Schallhart. Comparing ltl semantics for runtime verification. *J. Log. and Comput.*, 20(3):651–674, June 2010.
3. J. A. Bergstra and J. V. Tucker. Initial and final algebra semantics for data type specifications: Two characterization theorems. *SIAM J. Comput.*, 12(2):366–387, 1983.
4. A. Ştefănescu, c. Ciobâcă, R. Mereuţă, B. M. Moore, T. F. Şerbănuţă, and G. Roşu. All-path reachability logic. In *Proceedings of the 25th Conference on Rewriting Techniques and Applications and 12th Conference on Typed Lambda Calculi and Applications (RTA-TLCA'14)*, volume 8560 of *LNCS*, pages 425–440. Springer, July 2014.
5. M. d'Amorim and G. Rosu. Efficient monitoring of omega-languages. In *CAV*, volume 3576 of *LNCS*, pages 364–378. Springer, 2005.
6. V. Diekert and P. Gastin. Ltl is expressively complete for mazurkiewicz traces. *Journal of Computer and System Sciences*, 64(2):396 – 418, 2002.
7. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
8. D. Giannakopoulou and K. Havelund. Automata-based verification of temporal properties on running programs. In *ASE*, pages 412–416. IEEE Computer Society, 2001.
9. R. Goldblatt. *Logics of Time and Computation*. Number 7 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, CA, 2. edition, 1992.
10. R. Goldblatt. Mathematical modal logic: A view of its evolution. *J. of Applied Logic*, 1(5-6):309–392, Oct. 2003.
11. K. Havelund and G. Rosu. Efficient monitoring of safety properties. *International Journal on Software Tools for Technology Transfer (STTT)*, 6:158–173, 2004.
12. C. A. R. Hoare. An axiomatic basis for comp. programming. *CACM*, 12(10):576–580, 1969.
13. C. Jard and T. Jéron. On-line model checking for finite linear temporal logic specifications. In *Proc. of Automatic Verif. Methods for Finite State Systems*, pages 189–196. Springer, 1990.
14. H. W. Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.
15. I. Lee, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan. Runtime assurance based on formal specifications. In H. R. Arabnia, editor, *PDPTA*, pages 279–287. CSREA Press, 1999.
16. O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.
17. O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In R. Parikh, editor, *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer Berlin Heidelberg, 1985.
18. A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.
19. G. Roşu, A. Ştefănescu, c. Ciobâcă, and B. M. Moore. One-path reachability logic. In *Proceedings of the 28th Symposium on Logic in Computer Science (LICS'13)*, pages 358–367. IEEE, June 2013.
20. G. Rosu and K. Havelund. Rewriting-based techniques for runtime verification. *Automated Software Engineering*, 12:151–197, 2005. 10.1007/s10515-005-6205-y.
21. G. Rosu and A. Stefanescu. Checking reachability using matching logic. In *Proceedings of the 27th Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'12)*, pages 555–574. ACM, 2012.
22. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985.
23. M. Sulzmann and A. Zechner. Constructive finite trace analysis with linear temporal logic. In *Tests and Proofs*, volume 7305 of *LNCS*, pages 132–148. Springer Berlin Heidelberg, 2012.
24. P. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for mazurkiewicz traces. *Information and Computation*, 179(2):230 – 249, 2002.