Another Look at LTL Modulo Theory over Finite and Infinite Traces*

Alberto Bombardelli, Alessandro Cimatti, Alberto Griggio, and Stefano Tonetta

Fondazione Bruno Kessler, Via Sommarive, 18, 38123 Povo TN
{abombardelli,cimatti,griggio,tonettas}@fbk.eu

Abstract. Linear Temporal Logic is a de facto standard for specification of properties of complex systems. Fundamental problems in formal verification include satisfiability checking and model checking. Extensions and variants of LTL have gained significant interest: with LTL_f, the temporal formulas are interpreted over finite traces; with safety fragments of LTL, model checking can be reduced to search for finite trace counterexamples; in the context of Verification Modulo Theories, LTL includes first-order atoms interpreted over background theories. In this paper we propose a symbolic, automata-theoretic approach for these variants of LTL in a general and comprehensive framework, show the correctness of the reduction to liveness and invariant checking, and present a library of open benchmarks and support tools.

1 Introduction

Temporal logics are a fundamental area of computer science, for their ability to represent properties of the behaviours of complex dynamical systems. Application fields include requirements analysis [12,29], formal verification [2], reactive synthesis [13,39,22], agent-based systems [41], planning [20], contract-based design [32], robotics [38], runtime verification [5,31], and test case generation [49]. In the field of design of complex systems (e.g. hardware, software and cyber-physical), key problems are satisfiability, validity and model checking of temporal logic formulae. Satisfiability and validity allow to check the properties of requirements formalized as temporal formulae, and to prove contract refinement. Model checking is used to show that a system, modeled as a transition system S, satisfies the requirements, formalized as a temporal formula φ.

The paradigm of Linear Temporal Logic (LTL), originally proposed in [59], has become over the years a de facto standard [67], if compared to branching time temporal logics [18]. In the seminal works on finite-state model checking, LTL is propositional, and interpreted over infinite traces. The basic idea in LTL model checking is to tackle the problem $S \models \varphi$, where S is a transition system and φ an LTL formula, by checking whether all (infinite) execution traces of

* We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

S satisfy φ. The automata-theoretic approach provides an elegant solution to reduce reasoning about *LTL* to fair reachability problems [43]. This is done by constructing an automaton $S_{\neg\varphi}$ that accepts the traces violating φ. If the language of the product $S \times S_{\neg\varphi}$ is not empty, we have a counterexample to the original problem. We assume that $S_{\neg\varphi}$ is a degeneralized Büchi automaton, i.e. it has a single acceptance condition $\neg q$, sometimes referred to as the fairness condition.

One of the main advantages of the automata-theoretic approach is that one can focus on conceiving specialized algorithms for the pattern FGq, which is the dual of fair reachability. Thus, for example, efficient algorithms such as the nested depth-first search of [48,36] have focused on finding or proving the absence of (infinite) traces that visit $\neg q$ infinitely often.

We focus on a symbolic verification paradigm: following [35], both the system model and the automaton are represented as symbolic transition systems described by means of logical formulae. Popular approaches include Bounded Model Checking [8], liveness to safety [7], and k-liveness [34].

Several extensions and variants of *LTL* have recently gained significant interest. First, we consider *LTL* interpreted over *finite traces*, as in LTL_f [45,70,1]. This interpretation has the advantage that the analysis can be restricted to sets of finite traces (albeit of unbounded length), in contrast to the search for counterexamples of infinite length. Thus, the model checking problem $S \models_{fin} \varphi$ is focused on proving that all finite traces of S satisfy φ. The automata-theoretic approach in this case builds an automaton $S_{\neg\varphi}$ over finite traces, with a final condition $\neg q$ so that $S \models_{fin} \varphi$ if and only if $\neg q$ cannot be reached in $S \times S_{\neg\varphi}$. This is equivalent to the LTL_f pattern Gq and analogously to the case on infinite traces, specialized algorithms for invariant checking can be used.

The interpretation over finite traces is closely related to the semantic notion of safety languages [53], defined under the infinite trace interpretation, where a formula can be decided by simply looking at finite prefixes. The identification of syntactic safety fragments of *LTL* is particularly relevant because it opens up the possibility of using invariant checking instead of liveness checking algorithms, hence completely disregarding the notion of fairness. This also justifies the importance of recent research on relative safety [53,14,46].

Second, we consider the case of *LTL modulo theories*, $LTL(\mathcal{T})$ [24], where atoms are not limited to Boolean variables. Rather, temporal formulae are built over first-order atoms interpreted with respect to a background theory, following the Satisfiability Modulo Theories [4] paradigm. The problem is particularly relevant and challenging when the first-order atoms can constrain the current and next value of individual variables like in the formula $G(v' > v)$, where v' represents the next value of v.

We work in the setting of Verification Modulo Theories [25], where the infinite-state transition system S is symbolically described by means of SMT formulae. This opens up the possibility to model transition systems with state defined by arrays, integer- and real-valued variables, and complex, nonlinear dynamics, and to represent timed automata, hybrid automata, software, and

	Infinite Traces $S \models \varphi$		Finite Traces $S \models_{fin} \varphi$
	LTL	Safety Fragments	LTL_f
Propositional	$S \times S_{\neg\varphi} \models FGq$	$S \times S_{\neg\varphi} \models_{fin} Gq$	
Modulo Theories	$S \times S_{\neg\hat{\varphi}} \times S_{p_i \leftrightarrow \gamma_i(V)} \models FGq$	$S \times S_{\neg\hat{\varphi}} \times S_{p_i \leftrightarrow \gamma_i(V)} \models_{fin} Gq$	
Engine	Liveness Checker	Invariant Checker	

Table 1: Symbolic automata-theoretic approaches.

their combinations. We remark that in the infinite-state case, differently from the finite state case, when a property is violated there is no guarantee that a lasso-shaped counterexample $\alpha \cdot \beta^\omega$ exists. This makes the reduction to invariant checking even more appealing, when possible, because it avoids the analysis of infinite traces, and supports the use of practically effective invariant checkers such as [27,47,52,50].

In this paper we provide a general and comprehensive view of the applicability of the symbolic automata-theoretic approach to LTL and its extensions. See Table 1. We start from the classical symbolic automaton construction [35], used to reduce LTL model checking[1] to fair reachability. In case of safety LTL and LTL_f, this can be reduced to checking if q holds invariantly, i.e. the fairness condition $\neg q$ simplifies to a reachability condition. In the infinite-state case of Verification Modulo Theories, the construction is applied to the Boolean abstraction of φ, denoted $\hat{\varphi}$, which is φ where each theory atom p is replaced by a fresh Boolean variable v_p. The definition constraints S_φ^{lift} are then added to the description of the transition system. As detailed in the paper, additional rewriting is necessary in case of finite words to handle the presence of next variables in the theory atoms.

To illustrate the generality of the approach, we contribute with a library of benchmarks, expressed in the VMT-LIB open format [28], collected from multiple sources from the literature. We also provide a set of tools, integrated in the open source pyVMT framework [60], to reduce the different variants of LTL to the appropriate liveness or invariant checking problems for symbolic transition systems, using a general symbolic automaton construction that supports uniformly LTL, LTL_f and $SafetyLTL$.

The rest of this paper is structured as follows. In Section 2 we give some preliminaries on Satisfiability Modulo Theories and on Verification Modulo Theories. In Section 3 we present the syntax and semantics of LTL and its variants. In Section 4 we describe the symbolic automata-theoretic reduction to liveness/invariant checking. In Section 5 we introduce the benchmark library

[1] The approach also supports LTL validity and satisfiability checking. $\models \varphi$ is reduced to a model checking problem $S_U \models \varphi$, where S_U is the universal transition system. Satisfiability is obtained by dualization, i.e. φ is satisfiable iff $S_U \not\models \neg\varphi$.

and the tools implementing the reduction. Finally, in Section 6 we draw some conclusions and describe some directions for future work.

2 Satisfiability and Verification Modulo Theories

2.1 SMT

In the next section, we define syntax and semantics of $LTL(\mathcal{T})$ and $LTL_f(\mathcal{T})$, which are parametrized by a given first-order theory $\mathcal{T}$. We work in the setting of Satisfiability Modulo Theory (SMT) [4] and *LTL* Modulo Theory (see, e.g., [24]).

First-order formulas are built as usual by logic connectives, a given set of variables V and a first-order signature Σ. Given a formula ϕ, a Σ-structure M that interprets the symbols in Σ, and an assignment μ of the variables V in the domain M, we use the standard notion of $\langle \mu, M \rangle \models \phi$, which indicates that ϕ is satisfied by μ and M. If μ_1 and μ_2 are assignments to two disjoint sets of variables V_1 and V_2, we denote with $\mu_1 \cdot \mu_2$ the assignment over $V_1 \cup V_2$ mapping each variable $v_i \in V_i$ to $\mu_i(v_i)$.

A first-order theory $\mathcal{T}$ determines a first-order signature $\Sigma_{\mathcal{T}}$ of symbols and a set $\mathcal{M}_{\mathcal{T}}$ of interpretations of such symbols. In the following, we assume to be given a background theory $\mathcal{T}$ so that first-order formulas are meant to be $\Sigma_{\mathcal{T}}$-formulas and structures range in $\mathcal{M}_{\mathcal{T}}$.

2.2 States and traces

Given a structure M and a set of variables V, a *state* is an assignment from V to the domain of M. A [finite] *trace* is an infinite sequence $\pi = s_0, s_1, \ldots$ [resp., finite sequence $s_0, s_1, \ldots, s_k$] of states. Given a finite or infinite trace $\pi = s_0 s_1 \ldots$, we denote by $|\pi|$ the size of π. Thus, $|\pi| = +\infty$ if π is infinite. We denote by $\pi_{[i]}$ the $i+1$-th state s_i of the trace and by $\pi_{\geq i}$ the suffix of π starting from $\pi_{[i]}$. Given a finite trace π_1 and a finite or infinite trace π_2, we denote by $\pi_1\pi_2$ their concatenation. Given an infinite trace π, we denote by $\mathit{Pref}(\pi)$ the set of prefixes, i.e., $\mathit{Pref}(\pi) = \{\pi_1 \mid \pi = \pi_1\pi_2\}$.

If π_1 and π_2 are traces of the same length over two disjoint sets of variables V_1 and V_2, we denote with $\pi_1 \cdot \pi_2$ the trace over $V_1 \cup V_2$ such that for all i, $0 \leq i < |\pi_1|$, $(\pi_1 \cdot \pi_2)_{[i]} = \pi_{1[i]} \cdot \pi_{2[i]}$.

Finally, we denote by $\Pi^M(V)$ and $\Pi_f^M(V)$ the set of all possible respectively infinite and finite traces over the variable set V.

2.3 VMT

Let M be given. Let $\cdot'$ be a bijective function that maps a variable v to a variable v', and let $V' = \{v' \mid v \in V\}$. If s is a state over V, s' is the state defined over V' such that $s'(v') = s(v)$ for all v in V. A first-order *Symbolic Transition System* (STS) S is a tuple $S = \langle V, I, T \rangle$, where V is a set of (state) variables, $I(V)$ is a formula representing the initial states, and $T(V, V')$ is a formula representing

the transitions. A trace π defined over V is a finite [resp. infinite] trace of S iff $\langle \pi_{[0]}, M \rangle \models I$ and, for all $0 < i < |\pi|$, $\langle \pi_{[i-1]} \cdot \pi'_{[i]}, M \rangle \models T$. We say that a state s is reachable in S iff there exists a finite trace of S ending in s. We say that a state s is a deadlock state of S iff s is reachable and there does not exist a state $\overline{s}$ such that $\langle s \cdot \overline{s}', M \rangle \models T$. Moreover, we say that S is deadlock free iff there is no deadlock state in S.

Given two transition systems $S_1 = \langle V_1, I_1, T_1 \rangle$ and $S_2 = \langle V_2, I_2, T_2 \rangle$, we denote with $S_1 \times S_2$ the synchronous product $\langle V_1 \cup V_2, I_1 \wedge I_2, T_1 \wedge T_2 \rangle$.

Invariant and liveness properties. Generally speaking, temporal properties are often divided between safety and liveness properties. The former are usually reduced to invariant properties, while the latter to properties about *eventual invariance*, stating that a formula q holds invariantly from a certain point on. Different terminologies exist in the literature for referring to eventual invariance, including *persistence* [21], *progress* [17], or simply *liveness* [34]. In this paper, we follow the latter and call them liveness properties.

Invariant and liveness properties are basic properties, respectively on finite and infinite traces, for which various algorithms exist (see next section). They correspond respectively to a pattern of LTL_f and LTL model checking, which are introduced later, but we use the same notation for clarity.

Given a STS S and a first-order formula q over the variables of S, S satisfies the *invariant property* Gq, denoted by $S \models_{fin} Gq$, iff for all reachable states s in S, $s \models q$. Consequently, a *counterexample* for Gq is a *finite trace* $s_0, \ldots, s_k$ of S such that $s_k \models \neg q$.

S satisfies the *liveness property* FGq, denoted by $S \models FGq$, iff for all infinite traces $s_0, \ldots, s_i, \ldots$ of S, $\exists i. \forall j > i. s_j \models q$. Consequently, a *counterexample* for FGq is an *infinite trace* $s_0, \ldots, s_i, \ldots$ of S such that $\forall i. \exists j > i. s_j \models \neg q$.

2.4 Algorithms for verification modulo theories

Several successful symbolic model checking algorithms for finite-state systems are based on SAT solvers, e.g. [8,64,57,16]. With the significant improvements achieved in SMT solving over the last two decades, many of these algorithms have been adapted and extended for the verification of infinite-state systems, using SMT solvers as reasoning engines. In the following, we provide an overview of the main techniques used for checking invariant and liveness properties.

Invariant checking. In the case of invariant properties, several effective SAT-based algorithms can be naïvely applied to infinite-state systems by simply replacing the underlying decision procedure (i.e., from a SAT to an SMT solver). This is the case for instance of Bounded Model Checking (BMC) [8] and the interpolation-based algorithm of [57]. In contrast, extending the IC3 algorithm [16] to the infinite-state case requires more sophisticated approaches, combining to various extents techniques like predicate abstraction, interpolation, and approximated quantifier elimination, e.g. [27,47,52,50]. Such extensions of IC3 represent

the current state of the art for invariant checking of infinite-state symbolic transition systems. An alternative approach that has become increasingly popular in the last few years consists in expressing an invariant verification problem as a satisfiability checking problem in the formalism of Constrained Horn Clauses (CHC) [10]. It should be noted however that several of the most effective CHC solvers are still based on extensions of IC3 [47,52].

Liveness checking. The liveness property verification problem $S \models FGq$ amounts to show that no infinite trace in S visits $\neg q$ infinitely often. Existing SAT-based procedures work by encoding such problem as an invariant model checking problem, either monolithically [7] or incrementally [34,17]. The liveness to safety reduction of [7] exploits the fact that, in the case of finite-state systems, it is enough to consider *lasso-shaped* traces when searching for counterexamples to a liveness property. The algorithm transforms the system S into a new transition system S_{L2S} with additional variables. It non-deterministically guesses a loop start state, records if $\neg q$ is seen since the loop start, and checks for a lasso-shaped trace visiting $\neg q$ infinitely often. This reduces the liveness checking problem $S \models FGq$ to the invariant checking problem $S_{L2S} \models \neg loop$. The possibility to consider only lasso-shaped traces is exploited also by the Fair algorithm of [17], which works by incrementally strengthening the transition relation with constraints obtained by refuting candidate lasso-shaped traces visiting $\neg q$ infinitely-often. The k-liveness algorithm of [34], instead, exploits another property of finite-state systems, namely the fact that if $S \models FGq$, then there exists a concrete upper bound k on the number of states in which $\neg q$ holds along all traces of S. The liveness verification problem is then reduced to a sequence of invariant verification problems on an extension of S with additional logic that increments a counter every time $\neg q$ holds, in which the properties to check are of the form $counter \leq k$, for increasing values of k.

All the algorithms mentioned above exploit properties that hold for finite-state systems, but which are not true in general in the infinite-state case: in infinite-state systems a violated property is not guaranteed to have a lasso-shaped counterexample, and conversely, even if a liveness property FGq holds, there might exist no concrete upper bound k on the number of times $\neg q$ is true in any trace of the system (for example, such bound may be a value that depends on a, possibly infinite-valued, variable of the transition system, such as an uninitialized parameter). As a consequence, a naïve lifting of liveness to safety and Fair to the infinite-state case could not be used for proving that a property FGq holds, but only to find counterexamples (but only in cases in which a lasso-shaped counterexample exists), whereas k-liveness is still sound but becomes incomplete. In order to tackle such problems, various extensions to the SAT-based algorithms have been proposed, combining liveness to safety with predicate abstraction and ranking function synthesis [37], extending k-liveness to handle symbolic bounds in transition systems where the value of a variable diverges along all traces [26], and addressing the problem of finding non-looping counterexamples [23].

3 *LTL* and LTL_f modulo theories

3.1 Syntax

In this paper, we consider *LTL* [59] extended with past operators [56] as well as first-order atomic formulas. The language $LTL(\mathcal{T})$ is parametrized by a first-order theory $\mathcal{T}$. Given a set of variables V, the atomic formulas are built over the predicate, function, and constant symbols of $\Sigma_{\mathcal{T}}$ and the variables in $V \cup V'$, where V' represents the value of V in the "next state", after an execution step, as formalized below in the semantics subsection.

Definition 1. *Given a first-order theory $\mathcal{T}$ and a set of variables V, $LTL(\mathcal{T})$ formulas φ are defined by the following syntax:*

$$\varphi := \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid X\varphi \mid \varphi U \varphi \mid Y\varphi \mid \varphi S \varphi$$

where p is a first-order formula over the variables $V \cup V'$

$LTL_f(\mathcal{T})$ has the same syntax as $LTL(\mathcal{T})$.

When $\Sigma_{\mathcal{T}}$ is empty and V is a set of Boolean variables, the logic coincides with the usual propositional case, and we denote it by simply *LTL* (or LTL_f).

We use the following standard abbreviations: $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 R \varphi_2 := \neg(\neg\varphi_1 U \neg\varphi_2)$ (φ_1 releases φ_2), $F\varphi := \top U \varphi$ (sometime in the future φ), $G\varphi := \neg F \neg\varphi$ (always in the future φ), $\varphi_1 T \varphi_2 := \neg(\neg\varphi_1 S \neg\varphi_2)$ (φ_1 is triggered by φ), $O\varphi := \top S \varphi$ (once in the past φ), $H\varphi := \neg O \neg\varphi$ (historically in the past φ), $Z\varphi := \neg Y \neg\varphi$ (weak yesterday, i.e., yesterday φ or at initial state), and $N\varphi := \neg X \neg\varphi$ (weak next, i.e., next φ or at final state).

3.2 Semantics

$LTL(\mathcal{T})$ and $LTL_f(\mathcal{T})$ formulas are evaluated by a structure $M \in \mathcal{M}_{\mathcal{T}}$ and respectively infinite and finite traces. To avoid repetitions, we give a uniform definition of the semantics of *LTL* and LTL_f given a trace in $\Pi^M(V) \cup \Pi_f^M(V)$. To do so, we introduce the following notation. Given $i + 1 < |\pi|$ (π_i is not the last state), we define $\pi^+_{[i]}$ as the assignment $\pi_{[i]} \cdot \pi'_{[i+1]}$. If $i + 1 = |\pi|$ ($\pi_{[i]}$ is the last state), we define $\pi^+_{[i]}$ as the assignment that assigns every variable $v \in V$ to $\pi_{[i]}(v)$ and every variable $v' \in V'$ to a default value d in the domain of M.

Definition 2. *Given a structure M and a trace $\pi \in \Pi^M(V) \cup \Pi_f^M(V)$, the semantics of a formula φ is defined as follows:*

- $\pi, M, i \models \top$
- $\pi, M, i \not\models \bot$
- $\pi, M, i \models p$ *iff* $\langle \pi^+_{[i]}, M \rangle \models p$
- $\pi, M, i \models \varphi_1 \wedge \varphi_2$ *iff* $\pi, M, i \models \varphi_1$ *and* $\pi, M, i \models \varphi_2$
- $\pi, M, i \models \neg\varphi$ *iff* $\pi, M, i \not\models \varphi$
- $\pi, M, i \models \varphi_1 U \varphi_2$ *iff there exists* $k, i \leq k < |\pi|$, *such that* $\pi, M, k \models \varphi_2$ *and for all* $l, i \leq l < k, \pi, M, l \models \varphi_1$

- $\pi, M, i \models \varphi_1 S \varphi_2$ *iff there exists* $k, 0 \leq k \leq i$, *such that* $\pi, M, k \models \varphi_2$ *and for all* $l, k < l \leq i, \pi, M, l \models \varphi_1$
- $\pi, M, i \models X\varphi$ *iff* $i < |\pi|$ *and* $\pi, M, i+1 \models \varphi$
- $\pi, M, i \models Y\varphi$ *iff* $i > 0$ *and* $\pi, M, i-1 \models \varphi$

Finally, we have that $\pi, M \models \varphi$ *iff* $\pi, M, 0 \models \varphi$.

We introduce now a couple of normalizations of formulas. A formula is in Negative Normal Form (NNF) if the negation occurs only in front of the atomic formulas. We define two rewritings, NNF for $LTL(\mathcal{T})$ and NNF_f for $LTL_f(\mathcal{T})$. Although the resulting formulas are equivalent in $LTL(\mathcal{T})$ and we could have just used NNF_f, we avoid using N in $LTL(\mathcal{T})$ for clarity.

If we consider the operators $\vee, R, T, Z$ defined by the abbreviations introduced above as primitive, every $LTL(\mathcal{T})$ formula can be converted into an equivalent one in NNF. In particular, $\text{NNF}(\varphi)$ is obtained by applying the following rewriting to every occurrence of negations in φ:

$$\begin{array}{cc}
\text{NNF}(\neg(\varphi_1 \vee \varphi_2)) := \text{NNF}(\neg\varphi_1) \wedge \text{NNF}(\neg\varphi_2), & \text{NNF}(\neg(\varphi_1 \wedge \varphi_2)) := \text{NNF}(\neg\varphi_1) \vee \text{NNF}(\neg\varphi_2), \\
\text{NNF}(\neg(\varphi_1 R \varphi_2)) := \text{NNF}(\neg\varphi_1) U \text{NNF}(\neg\varphi_2), & \text{NNF}(\neg(\varphi_1 U \varphi_2)) := \neg\text{NNF}(\neg\varphi_1) R \text{NNF}(\neg\varphi_2), \\
\text{NNF}(\neg(\varphi_1 T \varphi_2)) := \text{NNF}(\neg\varphi_1) S \text{NNF}(\neg\varphi_2), & \text{NNF}(\neg(\varphi_1 S \varphi_2)) := \text{NNF}(\neg\varphi_1) T \text{NNF}(\neg\varphi_2), \\
\text{NNF}(\neg(Z\varphi)) := Y\text{NNF}(\neg\varphi), & \text{NNF}(\neg(Y\varphi)) := Z\text{NNF}(\neg\varphi), \\
\multicolumn{2}{c}{\text{NNF}(\neg(X\varphi)) := X\text{NNF}(\neg\varphi).}
\end{array}$$

If we also consider N as a primitive operator, then every $LTL_f(\mathcal{T})$ formula can be converted into an equivalent one in NNF. The corresponding rewriting $\text{NNF}_f(\varphi)$ is defined similarly to NNF, with the exception of the last case that is replaced by $\text{NNF}_f(\neg(X\varphi)) := N\text{NNF}_f(\neg\varphi)$ and $\text{NNF}_f(\neg(N\varphi)) := X\text{NNF}_f(\neg\varphi)$.

We remark that the choice of evaluating the next value of a variable on the last state of a trace to a default value is arbitrary, and any other choice has pros and cons. To avoid ambiguity, it is better to guard the use of next variables with $X\top$ so as to enforce the existence of the next state (e.g., $G(X\top \rightarrow v' = c)$). In the next sections, we will use the following rewriting to ensure that the next variables occur in this form. Let $norm(\varphi)$ be the formula obtained from φ by substituting every atomic formula p with $(X\top \wedge p) \vee (\neg X\top \wedge p[V'/d])$, where $p[V'/d]$ means that every occurrence of $v' \in V$ in p is replaced by the default value d.

Theorem 1. *φ and $norm(\varphi)$ are equivalent.*

Proof. Let π be a trace $\in \Pi^M(V) \cup \Pi_f^M(V)$ where M is a structure.

for all $0 \leq i < |\pi| : \pi, i \models \phi \Leftrightarrow \pi, i \models norm(\phi)$.

The statement easily entails the theorem. We prove it by induction on the structure of the formula. We ignore the cases with infinite trace because they trivially holds ($X\top$ is always true for infinite traces).

Base case: $\psi = p$.

$\pi, M, i \models norm(p) \Leftrightarrow \pi, M, i \models (X\top \wedge p) \vee (\neg X\top \wedge p[V', d])$. We consider two cases: if $i = |\pi| - 1$ and $i < |\pi| - 1$. In the first case $\pi, M, i \models norm(p) \Leftrightarrow \pi, M, i \models p[V', d]$. Since i is the last point of the trace, the assignment of the

dotted variable is provided using the default d value for each variable; therefore, $\pi, M, i \models norm(p) \Leftrightarrow \pi, M, i \models p$.

If i is not the last state then $\pi, M, i \models X\top$ which simplifies the formula to p itself completing the proof.

Inductive case: all the inductive case follows from trivial application of induction.

3.3 Satisfiability and Model checking modulo theories

The problem of LTL [resp., LTL_f] satisfiability modulo theory is the problem of deciding if, given a formula φ, there exists a structure M and an infinite [resp., finite] trace π such that $\langle \pi, M \rangle \models \varphi$. As usual, the dual validity problem is the problem of deciding if, given a formula φ, for every structure M and infinite/finite trace π, $\langle \pi, M \rangle \models \varphi$.

In model checking, a temporal property φ of $S = \langle V, I, T \rangle$ is specified over a set $V_\varphi \subseteq V$ of variables. A finite or infinite trace π of S defines a corresponding trace $\pi_{|V_\varphi}$ over V_φ given by the projection of the assignments of π on the subset of variables V_φ.

Given an LTL [resp., LTL_f] formula φ, the LTL model checking problem is the problem to check if $S \models \varphi$ [resp., $S \models_{fin} \varphi$], i.e., if, for every structure M and every infinite [resp., finite] trace π of S, $\langle \pi, M \rangle \models \varphi$.

The validity and model checking problems are equivalent in the sense that one can be reduced to the other and vice versa:

Theorem 2. *If φ is an LTL formula over the variables V, φ is valid in LTL($\mathcal{T}$) [resp., in $LTL_f(\mathcal{T})$] iff $S_U \models \varphi$ [resp., $S_U \models_{fin} \varphi$], where S_U is the universal model defined as $S_U = \langle V, \top, \top \rangle$; vice versa, given an STS $S = \langle V, I, T \rangle$ and an LTL formula φ over V, $S \models \varphi$ [resp., $S \models_{fin} \varphi$] iff $(I \wedge G(X\top \rightarrow T)) \rightarrow \varphi$ is valid in LTL($\mathcal{T}$) [resp., in $LTL_f(\mathcal{T})$].*

Note that $(I \wedge (X\top \rightarrow GT) \rightarrow \varphi$ can be simplified to $(I \wedge GT) \rightarrow \varphi$ in the case of infinite traces.

Another important relation exists between the propositional and SMT case. Let P_φ be the set of atomic predicates that occur in the formula $norm(\varphi)$. Let us consider a set $\hat{V}_\varphi$ containing one fresh Boolean variable v_p for each predicate p in P_φ. Let $\hat{\varphi}$ be the propositional formula obtained from φ by substituting every predicate p with v_p.

Given a trace π over the variables V, we define the trace $\hat{\pi}$ over the Boolean variables $\hat{V}_\varphi$ as follows: for all i, $0 \leq i < |\pi|$, $\hat{\pi}_{[i]}(v_p) = \top$ iff $\pi, i \models p$. Let $\pi^e = \pi \cdot \hat{\pi}$ be the composition of π and $\hat{\pi}$.

Theorem 3. *φ and $\hat{\varphi} \wedge (\bigwedge_{p \in P_\varphi} v_p \leftrightarrow p)$ are equisatisfiable LTL($\mathcal{T}$). φ and $norm(\varphi) \wedge (\bigwedge_{p \in P_\varphi} v_p \leftrightarrow p)$ are equisatisfiable $LTL_f(\mathcal{T})$. More specifically, for every infinite trace π, $\pi \models \varphi$ iff $\pi^e \models \hat{\varphi} \wedge (\bigwedge_{p \in P_\varphi} v_p \leftrightarrow p)$; and for every finite trace π, $\pi \models \varphi$ iff $\pi^e \models norm(\varphi) \wedge (\bigwedge_{p \in P_\varphi} v_p \leftrightarrow p)$.*

Proof (sketch). The proof follows straightforwardly the recursive definition of formulas. The only interesting case is the case in which π is finite and $\varphi = p$ is atomic. For all i, $0 \leq i < |\pi|$, $\pi, i \models p$ iff $\pi^e \models (X\top \wedge p) \vee (\neg X\top \wedge p[V'/d])$, which follows directly from the semantics of $LTL_f(\mathcal{T})$.

Moreover, if the atomic formulas do not contain the "next" variables, the satisfiability problem can be trivially reduced to the propositional case, by creating an equisatisfiable propositional LTL/LTL_f formula with quantifier elimination applied to $\exists V. \bigwedge_{p \in P_\varphi}(v_p \leftrightarrow p)$. In particular, this can be computed with the ALLSMT procedure [54].

In the propositional case, the satisfiability and model checking problems of both LTL and LTL_f are PSPACE-complete [65,45]. In the general case, the satisfiability and model checking problems are undecidable. It is easy in fact to encode two counters by considering for example the theory of integers. The decidability of fragments has been studied for example in [44] for $LTL(\mathcal{T})$ and most of the results can be ported to the $LTL_f(\mathcal{T})$ case.

3.4 Safety fragments of *LTL*

In this section, we adapt the definition of safety [53] to the $LTL(\mathcal{T})$ case.

Informally, a formula is safety if it specifies that nothing bad happens during an execution. If something bad happens, it occurs in a finite prefix of the execution. This is formalized as follows.

Definition 3. *Let φ be an LTL formula over variables V. The formula φ is safety iff for all structures M, for all $\pi \in \Pi^M(V)$, if $\pi \not\models \varphi$, then there exists $\pi_f \in Pref(\pi)$ such that for all $\pi^\omega \in \Pi^M(V)$, $\pi_f \pi^\omega \not\models \varphi$. Furthermore, we denote π_f as a bad prefix of P.*

We then define two complete syntactic fragments, which therefore can express all and only the safety formulas of LTL. SafetyLTL is a fragment of LTL which disallows positive occurrence of until.

Definition 4. *The syntax of SafetyLTL is defined as follows:*

$$\varphi := \top \mid \bot \mid p \mid \neg\beta \mid \varphi \wedge \varphi \mid X\varphi \mid Y\varphi \mid \varphi S \varphi$$
$$\beta := \top \mid \bot \mid p \mid \neg\varphi \mid \beta \wedge \beta \mid X\beta \mid \beta U \beta \mid Y\beta \mid \beta S \beta$$

Note that we include also past operators in SafetyLTL although this is typically defined only with the future ones.

Another safe fragment of LTL is full-past LTL. Although the syntax of the fragment is far stricter than SafetyLTL, the two logics have the same expressive power.

Definition 5. *The syntax of full-past LTL is of the form $\varphi := G\beta$ where β is as follows:*

$$\beta := \beta \wedge \beta \mid \neg\beta \mid Y\beta \mid \beta S \beta \mid p$$

It was proved in [21] that both fragments, SafetyLTL (even without past operators) and full-past LTL express all and only the safety properties of LTL.

4 Symbolic automata-theoretic approach

4.1 Reduction to liveness and invariant checking

The automata-theoretic approach [68] to *LTL* model checking is to transform an *LTL* formula φ over V into a Büchi automaton, which is here represented symbolically by an STS $S^l_{\neg\varphi}$ with a fairness condition $f^l_{\neg\varphi}$ such that, for every infinite trace σ, $\sigma \models \neg\varphi$ iff there exists an infinite trace π of $S^l_{\neg\varphi}$ with $\pi_{|V} = \sigma$ visiting $f^l_{\neg\varphi}$ infinitely often. Thus, $S \models \varphi$ iff $S \times S^l_{\neg\varphi} \models FG\neg f^l_{\neg\varphi}$.

Similarly, for LTL_f, the automata-theoretic approach [45] transforms a formula φ over V into an automaton on finite traces, which is here represented by an STS $S^s_{\neg\varphi}$ with a final condition $f^s_{\neg\varphi}$ such that, for every finite trace σ, $\sigma \models \neg\varphi$ iff there exists a finite π of $S^s_{\neg\varphi}$ with $\pi_{|V} = \sigma$ reaching $f^s_{\neg\varphi}$. Thus, $S_{fin} \models \varphi$ iff $S \times S^s_{\neg\varphi} \models_{fin} G\neg f^s_{\neg\varphi}$.

Finally, for *SafetyLTL*, the automata-theoretic approach [53] transforms a *SafetyLTL* formula φ into an automaton over finite traces, here represented by an STS $S^b_{\neg\varphi}$ with a final condition $f^b_{\neg\varphi}$ such that, for every infinite trace σ, $\sigma \models \neg\varphi$ iff there exists a finite trace π of $S^b_{\neg\varphi}$ with $\pi_{|V} = \sigma$ and with a bad prefix reaching $f^b_{\neg\varphi}$. Thus, if S is deadlock free, $S \models \varphi$ iff $S \times S^b_{\neg\varphi} \models_{fin} G\neg f^b_{\neg\varphi}$.

Note that the above construction where the safety fragment of *LTL* can be reduced to invariant checking over finite traces can be extended also beyond the safety fragment, in particular considering the notion of relative safety, first introduced in [46]. In fact, in [14], it was shown that under specific assumptions, the reduction can be extended to formulas of the form $\alpha \rightarrow \varphi$, where φ is safety.

4.2 From propositional to modulo theory

The above automata-theoretic approach can be lifted naturally to the "modulo theory" case, by considering the Boolean abstraction of the property and adding constraints to the system. More specifically, let P_φ, $\hat{V}_\varphi$, and $\hat{\varphi}$ be defined as above. Let $S^{\text{lift}}_\varphi = \langle V \cup \hat{V}_\varphi, \top, \bigwedge_{p\in P_\varphi} v_p \leftrightarrow p\rangle$. Then, we can prove the following theorems that lift the previous one to the SMT case.

Theorem 4. *$S \models \varphi$ iff $S \times S^l_{\neg\hat{\varphi}} \times S^{lift}_\varphi \models FG\neg f_{\neg\hat{\varphi}}$.*

Theorem 5. *$S \models \varphi$ iff $S \times S^s_{\neg\hat{\varphi_n}} \times S^{lift}_{\varphi_n} \models_{fin} G\neg f_{\neg\hat{\varphi_n}}$, where $\varphi_n = norm(\varphi)$.*

Theorem 6. *If φ is in SafetyLTL and S is deadlock free, $S \models \varphi$ iff $S \times S^b_{\neg\hat{\varphi}} \times S^{lift}_\varphi \models_{fin} G\neg f_{\neg\hat{\varphi}}$.*

Proof of Theorem 4 Let π be a trace of S violating φ. By Theorem 3, $\hat{\pi} \models \neg\hat{\varphi}$. Thus there exists a trace π^l of $S^l_{\neg\hat{\varphi}}$ with $\pi^l_{|\hat{V}_\varphi} = \hat{\pi}$ satisfying $\neg f_{\neg\hat{\varphi}}$ infinitely many times. Let $\pi^\times$ be the trace of $S \times S^l_{\neg\hat{\varphi}} \times S^{lift}_\varphi$ obtained by composing π, $\hat{\pi}$, and π^l. Then $\pi^\times \models \neg FG\neg f_{\neg\hat{\varphi}}$.

Let $\pi^{\times}$ be a trace of $S \times S^{l}_{\neg\hat{\varphi}} \times S^{lift}_{\varphi}$ satisfying $\neg FG\neg f_{\neg\hat{\varphi}}$. Let $\hat{\pi}$ be $\pi^{\times}_{|\hat{V_{\varphi}}}$ and π be $\pi^{\times}_{|V}$. Then $\pi^{\times}_{|V_{\varphi}} \models \neg\hat{\varphi}$. For all $i \geq 0$ $\hat{\pi}, i \models v_p$ iff $\pi, i \models p$. By Theorem 3, $\pi \models \neg\varphi$.

The proof of the other two theorems is similar.

4.3 Symbolic Compilation of Full LTL

The above lifting works with any automaton construction that preserves language of the properties. In symbolic model checking, there are various compilation approaches [35,51,33]. The one proposed back in [35] is still used in tools such as nuXmv. With small variants, this produces an STS that works for all the cases mentioned above.

Definition 6. *Given an LTL formula ϕ, the STS $ltl2sts(\phi) = \langle V_\phi, I_\phi, T_\phi \rangle$ is defined as follows:*

- $V_\phi = V \cup \{v_{X\beta} \mid X\beta \in Sub(\phi)\} \cup \{v_{X(\beta_1 U \beta_2)} \mid \beta_1 U \beta_2 \in Sub(\phi)\} \cup \{v_{Y\beta} \mid Y\beta \in Sub(\phi)\} \cup \{v_{Y\beta_1 S\beta_2} \mid \beta_1 S \beta_2 \in Sub(\phi)\}$
- $I_\phi = Enc(\phi) \wedge \bigwedge_{v_{Y\beta} \in V_{\neg\phi}} \neg v_{Y\beta}$
- $T_\phi = \bigwedge_{v_{X\beta} \in V_\phi} v_{X\beta} \leftrightarrow Enc(\beta)' \wedge \bigwedge_{v_{Y\beta} \in V_{\neg\phi}} Enc(\beta) \leftrightarrow v'_{Y\beta}$

where Sub is a function that maps a formula ϕ to the set of its subformulas, and Enc is defined recursively as:

- $Enc(\top) = \top$
- $Enc(v) = v$
- $Enc(\phi_1 \wedge \phi_2) = Enc(\phi_1) \wedge Enc(\phi_2)$
- $Enc(\neg\phi_1) = \neg Enc(\phi_1)$
- $Enc(X\phi_1) = v_{X\phi_1}$
- $Enc(\phi_1 U \phi_2) = Enc(\phi_2) \vee (Enc(\phi_1) \wedge v_{X(\phi_1 U \phi_2)})$
- $Enc(Y\phi_1) = v_{Y\phi_1}$
- $Enc(\phi_1 S \phi_2) = Enc(\phi_2) \vee (Enc(\phi_1) \wedge v_{Y(\phi_1 S \phi_2)})$

Intuitively, the variables in V_ϕ are proof obligations for the future states in the trace that must satisfy ϕ; they are initially set to a value according to $Enc(\phi)$ so that ϕ is satisfied in the initial state and are propagated by the transition relation if needed.

Let $F_\phi = \{Enc(\beta_1 U \beta_2 \rightarrow \beta_2) \mid \beta_1 U \beta_2 \in Sub(\phi)\}$[2].

Definition 7. *Given a set of formulas F over the variables V, we can build an STS $S_{deg}(F) = \langle V \cup V_{deg}, I_{deg}, T_{deg} \rangle$ called degeneralization of F, which is built as follows:*

- $V_{deg} = \{v_f \mid f \in F\}$ *is a set of Boolean variables*
- $I_{deg} = \bigwedge_{v_f \in V_{deg}} \neg v_f$

[2] It should be noted that it is possible to restrict the amount of fairness constraint considering only $\beta_1 U \beta_2 \in Sub(\phi)$ occurring positively in ϕ.

– $T_{deg} = (\bigwedge_{v_f \in V_{deg}}((\neg v_f \wedge f) \rightarrow v'_f)) \wedge ((\bigwedge_{v_f \in V_{deg}} v_f) \rightarrow (\bigwedge_{v_f \in V_{deg}} \neg v'_f)) \wedge ((\neg \bigwedge_{v_f \in V_{deg}} v_f) \rightarrow (\bigwedge_{v_f \in V_{deg}}(v_f \rightarrow v'_f)))$

We can finally define $S^f_{\neg\varphi} = ltl2sts(\neg\varphi) \times S_{deg}(F_{\neg\varphi})$. Let $f^l_{\neg\varphi} = \bigwedge_{v_f \in V_{deg}} \neg v'_f$.

Theorem 7. *For any infinite trace σ, $\sigma \models \neg\varphi$ iff there exists an infinite trace π of $S^l_{\neg\varphi}$ over σ visiting $f^l_{\neg\varphi}$ infinitely many times.*

In case of LTL_f, the construction is applied to a formula in NNF and the STS is built with a modification on the transition condition, where the double implication is replaced by a single implication.

Definition 8. *Given an LTL formula ϕ, the STS $ltlf2sts(\phi) = \langle V_\phi, I_\phi, T_\phi \rangle$ is defined as follows:*

– $V_\phi = V \cup \{v_{X\beta} \mid X\beta \in Sub(\phi)\} \cup \{v_{X(\beta_1 U \beta_2)} \mid \beta_1 U \beta_2 \in Sub(\phi)\} \cup \{v_{Y\beta} \mid Y\beta \in Sub(\phi)\} \cup \{v_{Y\beta_1 S \beta_2} \mid \beta_1 S \beta_2 \in Sub(\phi)\} \cup \{v_{N\beta} \mid N\beta \in Sub(\phi)\} \cup \{v_{X(\beta_1 R \beta_2)} \mid \beta_1 R \beta_2 \in Sub(\phi)\} \cup \{v_{Z\beta} \mid Z\beta \in Sub(\phi)\} \cup \{v_{Z\beta_1 T \beta_2} \mid \beta_1 T \beta_2 \in Sub(\phi)\}$
– $I_\phi = Enc(\phi) \wedge \bigwedge_{v_{Y\beta} \in V_{\neg\phi}} \neg v_{Y\beta} \wedge \bigwedge_{v_{Z\beta} \in V_{\neg\phi}} v_{Z\beta}$
– $T_\phi = \bigwedge_{v_{X\beta} \in V_\phi} v_{X\beta} \rightarrow Enc(\beta)' \wedge \bigwedge_{v_{Y\beta} \in V_{\neg\phi}} Enc(\beta) \rightarrow v'_{Y\beta} \wedge \bigwedge_{v_{N\beta} \in V_\phi} v_{N\beta} \rightarrow Enc(\beta)' \wedge \bigwedge_{v_{Z\beta} \in V_{\neg\phi}} Enc(\beta) \rightarrow v'_{Z\beta}$

where Sub is a function that maps a formula ϕ to the set of its subformulas, and Enc extends the previous definition as follows:

– $Enc(\phi_1 \vee \phi_2) = Enc(\phi_1) \vee Enc(\phi_2)$
– $Enc(N\phi_1) = v_{N\phi_1}$
– $Enc(\phi_1 R \phi_2) = Enc(\phi_2) \wedge (Enc(\phi_1) \vee v_{N(\phi_1 R \phi_2)})$
– $Enc(Z\phi_1) = v_{Z\phi_1}$
– $Enc(\phi_1 T \phi_2) = Enc(\phi_2) \wedge (Enc(\phi_1) \vee v_{Z(\phi_1 T \phi_2)})$

Let us define $\psi = \text{NNF}_f(\neg\varphi)$ and $S^s_{\neg\varphi} = ltlf2sts(\psi)$. The condition $f^s_{\neg\varphi}$ is then defined as $\bigwedge_{v_{X\beta} \in V_\psi} \neg v_{X\beta}$.

Theorem 8. *For every finite trace σ, $\sigma \models \neg\varphi$ iff there exists a finite trace π of $S^s_{\neg\varphi}$ with $\pi_{|V} = \sigma$ reaching $f^s_{\neg\varphi}$.*

In case of *SafetyLTL*, we still use $ltlf2sts$ but it is applied to a formula in NNF using nnf, which does not introduce the weak next operator. Thus, let us define $\psi = nnf(\neg\varphi)$ and $S^b_{\neg\varphi} = ltlf2sts(\psi)$. The condition $f^b_{\neg\varphi}$ is defined as $\bigwedge_{v_{X\beta} \in V_{\neg\varphi}} \neg v_{X\beta}$.

Theorem 9. *For every infinite trace σ, $\sigma \models \neg\varphi$ iff there exists a finite trace π of S^b_ψ with $\pi_{|V} = \sigma$ and with a bad prefix reaching f^b_ψ.*

4.4 Running example

We showcase the various parts of our construction with a couple of small examples. Let a STS $S = \langle V, I, T\rangle$ with $V := \{i, o, n\}$, $I := \top$ and $T := n' = n \wedge o' = i + 1$. The STS S has an input variable i, an output variable o and a frozen variable n; each transition replicates the value of i incremented by one in the next state.

A liveness example: We now consider the response property $\varphi_f := G(i = n \rightarrow F(o = n + 1))$. It is easy to see that this liveness property holds; the transition relation satisfies the sub-formula $F(o = n + 1)$ in one step every time that i is equal to n. We now apply the construction defined in the previous subsections. First, we define a propositional formula and a "lift" transition system mapping the original property to the propositional one. Subsequently, we construct the symbolic compilation for the negation of the property.

We can easily construct $\hat{\varphi_f}$ by replacing each predicates with fresh Boolean variables as $\hat{\varphi_f} := G(v_{p_0} \rightarrow F v_{p_1})$. Moreover, we define the "lift" transition system as $S^{lift}_{\varphi_f} := \langle V \cup \hat{V}_{\varphi_f}, \top, \hat{T}_{\varphi_f}\rangle$ where $\hat{V}_{\varphi_f} := \{v_{p_0}, v_{p_1}\}$ and $\hat{T}_{\varphi_f} := (v_{p_0} \leftrightarrow i = n) \wedge (v_{p_1} \leftrightarrow o = n + 1)$. With this mapping between the propositional LTL formula and the $LTL(\mathcal{T})$ formula we just need to construct the STS corresponding to the propositional formula to prove the property.

We now take the negation of $\hat{\varphi_f}$ without abbreviations. The resulting $\neg\hat{\varphi_f}$ formula is as follows. $\neg\hat{\varphi_f} := \top U(v_{p_0} \wedge \neg(\top U v_{p_1}))$. Since our formula contains 2 until temporal operator, $V_{\neg\hat{\varphi_f}} := \{v_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))}, v_{X(\top U v_{p_1})}\}$. The initial condition of $STS_{\neg\hat{\varphi_f}}$ is then encoded as $I_{\neg\hat{\varphi_f}} := v_{p_0} \wedge \neg v_{X(\top U v_{p_1})} \vee v_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))}$. Either the right side of the outer until holds in the initial state, or it will hold eventually in the future thanks to the prophecy variable. The transition relates the next value of the subformulae with their corresponding prophecy variables. $T_{\neg\hat{\varphi_f}} := (v_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))} \leftrightarrow v'_{p_0} \wedge \neg v'_{X(\top U v_{p_1})} \vee v'_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))}) \wedge (v_{X(\top U v_{p_1})} \leftrightarrow v'_{p_1} \vee v_{X(\top U v_{p_1})})$. Finally, we add the fairness condition forcing either the until prophecy to be false or to witness its right side. $F_{\neg\hat{\varphi_f}} := \{v_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))} \vee (v_{p_0} \wedge \neg(v_{p_1} \vee v_{X(\top U v_{p_1})}) \rightarrow (\neg(v_{p_1} \vee v_{X(\top U v_{p_1})}), v_{p_1} \vee v_{X(\top U v_{p_1})} \rightarrow v_{p_1}\}$.

An LTL_f example: We now consider a bounded response property (with bound 1) $\varphi_b := G(i = n \rightarrow X(o = n + 1))$. Although with infinite semantics we would expect this property to hold, in finite semantics any finite trace terminating with $i = n$ is a valid counter-example of the property.

As before, we apply the construction defined in the previous subsections tailored for LTL_f. Since no primed variable occurs in φ_b, $norm(\varphi_b)$ is equivalent to φ_b. Thus, we simply use directly φ_b as our target formula. For brevity we skip the generation of the lift STS since it is identical to the one of φ_f.

We now take the negation of $\hat{\varphi_b}$ in negative normal form. The resulting $\hat{\varphi_b}' := \text{NNF}_f(\neg\hat{\varphi_b})$ formula is as follows. $\hat{\varphi_b}' := \top U(v_{p_0} \wedge N(\neg v_{p_1}))$. Since our formula contains an until and a weak next, $V_{\hat{\varphi_b}'} := \{v_{X(\top U(v_{p_0} \wedge N(\neg v_{p_1})))}, v_{N(\neg v_{p_1})}\}$. The initial condition of $STS_{\hat{\varphi_b}'}$ is then encoded as $I^{\text{s}}_{\hat{\varphi_b}'} := v_{p_0} \wedge v_{N(\neg v_{p_1})} \vee$

$v_{X(\top U(v_{p_0} \wedge N(\neg v_{p_1})))}$. The transition relates the next value of the subformulae with their corresponding prophecy variables. Since our formula is in nnf, we do not need double implication. The resulting transition is encoded as $T^s_{\hat{\varphi}_b} := (v_{X(\top U(v_{p_0} \wedge N(\neg v_{p_1})))} \rightarrow v'_{p_0} \wedge v'_{N\neg v_{p_1}} \vee v'_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))}) \wedge (v_{N\neg v_{p_1}} \rightarrow \neg v'_{p_1})$. Finally, we construct the reachability condition $f^s_{\hat{\varphi}_b{}'}$ with the prophecy as $f^s_{\hat{\varphi}_b{}'} := \neg v_{X(\top U(v_{p_0} \wedge \neg(\top U v_{p_1})))}$. It should be noted that, with such reachability condition, any finite trace terminating terminating with v_{p_0} has a corresponding path with the prophecy variable assigned to false.

5 Benchmarks

The need for establishing a collection of benchmark problems, ideally written in a simple yet expressive language widely supported by different tools, has long been recognized in several communities in formal verification and automated reasoning. The success of initiatives such as TPTP [66] and SMT-LIB [3] in automated reasoning, or Aiger [9], Btor [58] and SV-COMP [6] in formal verification has been a key factor for the significant practical advancements of the state of the art in their respective fields over the last decades.

Recently, we have proposed VMT-LIB [28], an extension of the standard SMT-LIB language for SMT solvers, as a language for the representation of invariant checking and liveness checking problems on infinite-state symbolic transition systems. VMT-LIB has a flexible format and clear semantics based on SMT-LIB, and supports the specification of *LTL* and LTL_f properties with the (extended) syntax introduced in §3. More specifically, VMT-LIB has metadata tags `:ltl-property` and `:ltlf-property`, with which *LTL* and LTL_f specifications can be annotated.

A benchmark set for LTL satisfiability and verification. As a first contribution to the creation of a standard benchmark library for *LTL* model checking and satisfiability, we have collected various benchmark sets from different sources from the literature [55,40,63,15,11], and converted them to VMT-LIB. The resulting benchmark library consists of about 3000 instances. We make the benchmarks available at `http://www.vmt-lib.org`.

Tool support. As mentioned above, an important feature of the VMT-LIB format is that it is possible to precisely represent *LTL* and LTL_f properties using the full syntax defined in §3. This ensures that the *LTL* benchmarks we collected can be kept at their original/intended level of abstraction, rather than being reduced to simpler invariant or liveness properties via reductions similar to those described in §4. This is important in order to make the benchmark set useful for evaluating techniques that might exploit alternative reductions and/or specialised encodings for particular patterns or kinds of properties. At the same time, however, we have developed a set of algorithms implementing the reductions of §4 in order to convert arbitrary *LTL* or LTL_f properties into simpler liveness or invariant ones, so that the benchmarks can be used also in the evaluation of core verification

algorithms/tools that only target such special kinds of properties. In particular, we have integrated our implementations in pyVMT [60], an open-source Python-based programmatic framework for interacting with model checkers developed as an extension of the pySMT [42] library for SMT solvers.

6 Conclusions

In this paper we presented a general framework for a symbolic automata-theoretic approach for satisfiability and model checking of linear temporal logics. The approach covers *LTL* modulo theories, and can deal with properties interpreted over finite and infinite traces. We have shown how to leverage symbolic liveness checkers for the general case, and identified conditions (e.g. safety fragments) to resort to less expensive invariant checkers. Furthermore, the techniques presented can be extended to handle more expressive logics, such as RELTL, that includes suffix implication and conjunction [19,30], or to logics with real time aspects and continuous signals, using e.g. the methods of [24].

In this paper we disregarded the important problem of reactive synthesis, i.e. automatically constructing a system satisfying the given specifications [13]. The problem is very well studied in the finite state case. Recently, very interesting works have proposed synthesis modulo theories where the role of boolean abstraction is prominent [61,62].

References

1. A. Artale, L. Geatti, N. Gigante, A. Mazzullo, and A. Montanari. Complexity of safety and cosafety fragments of linear temporal logic. In *AAAI*, pages 6236–6244. AAAI Press, 2023.
2. C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.
3. C. Barrett, P. Fontaine, and C. Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, 2021. `https://smtlib.cs.uiowa.edu/papers/smt-lib-reference-v2.6-r2021-05-12.pdf`.
4. C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1329. IOS Press, 2021.
5. A. Bauer, M. Leucker, and C. Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, 2011.
6. D. Beyer. Competition on software verification - (SV-COMP). In *TACAS*, volume 7214 of *Lecture Notes in Computer Science*, pages 504–524. Springer, 2012.
7. A. Biere, C. Artho, and V. Schuppan. Liveness checking as safety checking. *Electronic Notes in Theoretical Computer Science*, 66(2):160–177, 2002.
8. A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Adv. Comput.*, 58:117–148, 2003.
9. A. Biere, K. Heljanko, and S. Wieringa. AIGER 1.9 and beyond. Technical Report 11/2, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, 2011.

10. N. S. Bjørner, A. Gurfinkel, K. L. McMillan, and A. Rybalchenko. Horn clause solvers for program verification. In *Fields of Logic and Computation II*, volume 9300 of *Lecture Notes in Computer Science*, pages 24–51. Springer, 2015.
11. S. Bliudze, A. Cimatti, M. Jaber, S. Mover, M. Roveri, W. Saab, and Q. Wang. Formal Verification of Infinite-State BIP Models. In *ATVA*, volume 9364 of *LNCS*, pages 326–343. Springer, 2015.
12. R. Bloem, A. Cimatti, K. Greimel, G. Hofferek, R. Könighofer, M. Roveri, V. Schuppan, and R. Seeber. RATSY - A new requirements analysis tool with synthesis. In *CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 425–429. Springer, 2010.
13. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.
14. A. Bombardelli, A. Cimatti, S. Tonetta, and M. Zamboni. Symbolic model checking of relative safety LTL properties. In *iFM*, volume 14300 of *Lecture Notes in Computer Science*, pages 302–320. Springer, 2023.
15. M. Bozzano, A. Cimatti, A. F. Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, and S. Tonetta. Formal design and safety analysis of AIR6110 wheel brake system. In *CAV (1)*, volume 9206 of *Lecture Notes in Computer Science*, pages 518–535. Springer, 2015.
16. A. R. Bradley. SAT-Based Model Checking without Unrolling. In *VMCAI*, volume 6538 of *LNCS*, pages 70–87. Springer, 2011.
17. A. R. Bradley, F. Somenzi, Z. Hassan, and Y. Zhang. An incremental approach to model checking progress properties. In *FMCAD*, pages 144–153. FMCAD Inc., 2011.
18. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10ˆ20 states and beyond. In *LICS*, pages 428–439. IEEE Computer Society, 1990.
19. D. Bustan, A. Flaisher, O. Grumberg, O. Kupferman, and M. Y. Vardi. Regular Vacuity. In *CHARME*, volume 3725 of *Lecture Notes in Computer Science*, pages 191–206. Springer, 2005.
20. A. Camacho, E. Triantafillou, C. J. Muise, J. A. Baier, and S. A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, pages 3716–3724. AAAI Press, 2017.
21. E. Y. Chang, Z. Manna, and A. Pnueli. Characterization of Temporal Property Classes. In *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 474–486. Springer, 1992.
22. A. Cimatti, L. Geatti, N. Gigante, A. Montanari, and S. Tonetta. Fairness, assumptions, and guarantees for extended bounded response LTL+P synthesis. *Softw. Syst. Model.*, 23(2):427–453, 2024.
23. A. Cimatti, A. Griggio, and E. Magnago. LTL falsification in infinite-state systems. *Inf. Comput.*, 289(Part):104977, 2022.
24. A. Cimatti, A. Griggio, E. Magnago, M. Roveri, and S. Tonetta. SMT-based satisfiability of first-order LTL with event freezing functions and metric operators. *Inf. Comput.*, 272:104502, 2020.
25. A. Cimatti, A. Griggio, S. Mover, M. Roveri, and S. Tonetta. Verification modulo theories. *Formal Methods Syst. Des.*, 60(3):452–481, 2022.
26. A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. Verifying LTL Properties of Hybrid Systems with K-Liveness. In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 424–440. Springer, 2014.
27. A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. Infinite-state invariant checking with IC3 and predicate abstraction. volume 49, pages 190–218, 2016.

28. A. Cimatti, A. Griggio, and S. Tonetta. The VMT-LIB language and tools. In *SMT*, volume 3185 of *CEUR Workshop Proceedings*, pages 80–89. CEUR-WS.org, 2022.
29. A. Cimatti, M. Roveri, A. Susi, and S. Tonetta. Validation of requirements for hybrid systems: A formal approach. *ACM Trans. Softw. Eng. Methodol.*, 21(4):22:1–22:34, 2012.
30. A. Cimatti, M. Roveri, and S. Tonetta. Symbolic compilation of PSL. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 27(10):1737–1750, 2008.
31. A. Cimatti, C. Tian, and S. Tonetta. Assumption-based runtime verification. *Formal Methods Syst. Des.*, 60(2):277–324, 2022.
32. A. Cimatti and S. Tonetta. Contracts-refinement proof system for component-based embedded systems. *Sci. Comput. Program.*, 97:333–348, 2015.
33. K. Claessen, N. Eén, and B. Sterin. A circuit approach to ltl model checking. *2013 Formal Methods in Computer-Aided Design*, pages 53–60, 2013.
34. K. Claessen and N. Sörensson. A liveness checking algorithm that counts. *2012 Formal Methods in Computer-Aided Design (FMCAD)*, pages 52–59, 2012.
35. E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at ltl model checking. *Formal Methods in System Design*, 10:47–71, 1994.
36. C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory-efficient algorithms for the verification of temporal properties. *Formal Methods Syst. Des.*, 1(2/3):275–288, 1992.
37. J. Daniel, A. Cimatti, A. Griggio, S. Tonetta, and S. Mover. Infinite-state liveness-to-safety via implicit abstraction and well-founded relations. In *CAV (1)*, volume 9779 of *Lecture Notes in Computer Science*, pages 271–291. Springer, 2016.
38. G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Autom.*, 45(2):343–352, 2009.
39. B. Finkbeiner. Synthesis of reactive systems. In *Dependable Software Systems Engineering*, volume 45 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, pages 72–98. IOS Press, 2016.
40. V. Fionda and G. Greco. The complexity of LTL on finite traces: Hard and easy fragments. In *AAAI*, pages 971–977. AAAI Press, 2016.
41. M. Fisher and M. J. Wooldridge. Temporal reasoning in agent-based systems. In *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*, pages 469–495. Elsevier, 2005.
42. M. Gario and A. Micheli. PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms. In *SMT Workshop 2015*, 2015.
43. R. Gerth, D. A. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *PSTV*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.
44. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Combination Methods for Satisfiability and Model-Checking of Infinite-State Systems. In *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 362–378. Springer, 2007.
45. G. D. Giacomo and M. Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*, pages 854–860. IJCAI/AAAI, 2013.
46. T. A. Henzinger. Sooner is safer than later. *Inf. Process. Lett.*, 43:135–141, 1992.
47. K. Hoder and N. S. Bjørner. Generalized property directed reachability. In *SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2012.
48. G. J. Holzmann, D. A. Peled, and M. Yannakakis. On nested depth first search. In *The Spin Verification System*, volume 32 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 23–31. DIMACS/AMS, 1996.

49. H. S. Hong, I. Lee, O. Sokolsky, and H. Ural. A temporal logic based theory of test coverage and generation. In *TACAS*, volume 2280 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2002.
50. D. Jovanovic and B. Dutertre. Property-directed k-induction. In *FMCAD*, pages 85–92. IEEE, 2016.
51. Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic Verification of Linear Temporal Logic Specifications. In *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1998.
52. A. Komuravelli, A. Gurfinkel, and S. Chaki. SMT-based model checking for recursive programs. *Formal Methods Syst. Des.*, 48(3):175–205, 2016.
53. O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods Syst. Des.*, 19(3):291–314, 2001.
54. S. K. Lahiri, R. Nieuwenhuis, and A. Oliveras. SMT techniques for fast predicate abstraction. In *CAV*, volume 4144 of *Lecture Notes in Computer Science*, pages 424–437. Springer, 2006.
55. J. Li, G. Pu, Y. Zhang, M. Y. Vardi, and K. Y. Rozier. Sat-based explicit ltlf satisfiability checking. *Artif. Intell.*, 289:103369, 2020.
56. O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Workshop on Logic of Programs*, pages 196–218. Springer, 1985.
57. K. L. McMillan. Interpolation and sat-based model checking. In *CAV*, volume 2725 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2003.
58. A. Niemetz, M. Preiner, C. Wolf, and A. Biere. Btor2 , btormc and boolector 3.0. In *CAV (1)*, volume 10981 of *Lecture Notes in Computer Science*, pages 587–595. Springer, 2018.
59. A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.
60. Pyvmt. `https://github.com/pyvmt/pyvmt`, 2022.
61. A. Rodríguez and C. Sánchez. Boolean abstractions for realizability modulo theories. In *CAV (3)*, volume 13966 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2023.
62. A. Rodríguez and C. Sánchez. Adaptive reactive synthesis for LTL and ltlf modulo theories. In *AAAI*, pages 10679–10686. AAAI Press, 2024.
63. V. Schuppan and L. Darmawan. Evaluating LTL satisfiability solvers. In *ATVA*, volume 6996 of *Lecture Notes in Computer Science*, pages 397–413. Springer, 2011.
64. M. Sheeran, S. Singh, and G. Stålmarck. Checking safety properties using induction and a sat-solver. In *FMCAD*, volume 1954 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2000.
65. A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *J. ACM*, 32(3):733–749, 1985.
66. G. Sutcliffe. The TPTP problem library and associated infrastructure - from CNF to th0, TPTP v6.4.0. *J. Autom. Reason.*, 59(4):483–502, 2017.
67. M. Y. Vardi. Branching vs. linear time: Final showdown. In *TACAS*, volume 2031 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2001.
68. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society, 1986.
69. Y. Xia, A. Cimatti, A. Griggio, and J. Li. Avoiding the shoals - a new approach to liveness checking. In *CAV*, Lecture Notes in Computer Science. Springer, 2024.
70. S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi. Symbolic LTLf synthesis. In *IJCAI*, pages 1362–1369. ijcai.org, 2017.

A Proofs

A.1 Proofs of Section 3

Proof of Theorem 2

From validity to model checking:

Proof. To prove the equivalence between the model checking of universal model S_U and the validity, we show that the set of infinite (resp. finite) traces of S_U is equal to $\Pi^M(V)$ (resp. $\Pi_f^M(V)$).

Let $S = \{\pi | \pi \text{is a trace infinite trace of } S_U\}$ and $S_f = \{\pi | \pi \text{is a trace finite trace of } S_U\}$. $S = \Pi^M(V)$ and $S_f = \Pi_f^M(V)$

We split the proof in two parts using set inclusion: one direction is trivial $Q \subseteq \Pi^M(V)$ and $Q^f \subseteq \Pi_f^M(V)$ since the traces of S_U are all traces over V.

We prove the other direction by w.o.c. considering an infinite (resp. finite) trace π (π') s.t. $\pi \in \Pi^M(V)$ ($\pi' \in \Pi_f^M(V)$) and $\pi \notin Q$ ($\pi' \notin Q_f$). Since π (π') is not a trace of S_U then either $\pi(\pi'), 0 \not\models I_U$ or for all $0 \leq i < |\pi|(|\pi'|) - 1$ $\pi(\pi') \not\models T_U$. Since $I_U = T_U = \top$ from the semantics we obtain that no trace violates $\top$ by definition contradicting the claim. Therefore, as expected the universal model contains all the traces over V proving the mapping from validity to model checking.

From model checking to validity:

Proof.($\Rightarrow$)
- We want to prove that: $S \models \varphi \Rightarrow I \wedge GT \rightarrow \varphi$ is valid over $LTL(\mathcal{T})$. By contradiction we say that $S \models \varphi$ and $I \wedge GT \rightarrow \varphi$ is not valid over $LTL(\mathcal{T})$. Which means that there is an infinite trace π that violates the formula i.e. $\pi \models I \wedge GT \wedge \neg\phi$. However, by satisfaction we derive that $\pi, 0 \models I$ and forall $i \geq 0 : \pi, i \models T$ from which we derive that π is a trace of S; however, by assumption the infinite traces of S satisfy φ (contradiction).
- We want to prove that: $S \models_{fin} \varphi \Rightarrow I \wedge G(X\top \rightarrow T) \rightarrow \varphi$ is valid over $LTL_f(\mathcal{T})$.
  By contradiction we say that $S \models_{fin} \varphi$ and $I \wedge G(X\top \rightarrow T) \rightarrow \varphi$ is not valid over $LTL_f(\mathcal{T})$. Which means that there is a finite trace π that violates the formula i.e. $\pi \models I \wedge G(X\top \rightarrow T) \wedge \neg\varphi$. However, by satisfaction we obtain that $\pi, 0 \models I$ and for all $i \leq |\pi| - 1 : \pi, i \models X\top \rightarrow T$. We can simplify the latter one as for all $i < |\pi| - 1$: $\pi, i \models T$ since $X\top$ holds when $i < |\pi| - 1$. As for the case with infinite traces we obtain that π is a trace of S ending up with a contradiction. It should be note that without $X\top$ a finite trace π disproving the property could have existed; indeed, the $X\top$ guard is crucial for the correctness of the theorem.

($\Leftarrow$)
- We want to prove that: $I \wedge GT \rightarrow \varphi$ is valid over $LTL(\mathcal{T}) \Rightarrow S \models \varphi$. By contradiction we say that $I \wedge GT \rightarrow \varphi$ is valid over $LTL(\mathcal{T})$ and there is an infinite trace of S violating φ. If immediately follows that being a trace of S satisfies I in the initial state and T at each transition; thus contradicting the claim.

- We want to prove that: $I \wedge G(X\top \rightarrow T) \rightarrow \varphi$ is valid over $LTL_f(\mathcal{T})$ $\Rightarrow S \models_{fin} \varphi$.
  By contradiction we say that $I \wedge G(X\top \rightarrow T) \rightarrow \varphi$ is valid over $LTL(\mathcal{T})$ and there is a finite trace of S violating φ. If immediately follows that being a trace of S satisfies I in the initial state and T at each transition which formally means: $\pi, 0 \models_{fin} I$ and for all $0 \leq i < |\pi| - 1$ $\pi, i \models T$. Since $X\top$ is valid iff $i < |\pi| - 1$ then for all $0 \leq i \leq |\pi| - 1 : \pi, i \models X\top \rightarrow T$. From which we show the contradiction.

A.2 Symbolic compilation correctness

Proof of Theorem 7

Definition 9. *Let us define the function State taking an LTL formula and a trace σ and returning a set of variables:*

- $State^{sng}(a, \sigma, i) = \emptyset$
- $State^{sgn}(\neg\psi, \sigma, i) = State^{\overline{sgn}}(\psi, \sigma, i)$
- $State^{-}(\psi_1 \vee \psi_2, \sigma, i) = State^{-}(\psi_1, \sigma, i) \cup State^{-}(\psi_2, \sigma, i)$
- $State^{+}(\psi_1 \vee \psi_2, \sigma, i) = \begin{cases} State^{+}(\psi_1, \sigma, i) & \text{if } \sigma, i \models \psi_1 \\ State^{+}(\psi_2, \sigma, i) & \text{else} \end{cases}$
- $State^{+}(\psi_1 U \psi_2, \sigma, i) = \begin{cases} State^{+}(\psi_2, \sigma, i) & \text{if } \sigma, i \models \psi_2 \\ State^{+}(\psi_1, \sigma, i) \cup \{v_{X(\psi_1 U \psi_2)}\} & \text{else} \end{cases}$
- $State^{+}(Y\psi, \sigma, i) = \{v_{Y\psi}\}$
- $State^{+}(\psi_1 S \psi_2, \sigma, i) = \begin{cases} State^{+}(\psi_2, \sigma, i) & \text{if } \sigma, i \models \psi_2 \\ State^{+}(\psi_1, \sigma, i) \cup \{v_{Y(\psi_1 S \psi_2)}\} & \text{else} \end{cases}$
- $State^{-}(\mathcal{OP}1(\psi), \sigma, i) = \emptyset$
- $State^{-}((\psi_1)\mathcal{OP}2(\psi_2), \sigma, i) = \emptyset$

where $sgn \in \{-, +\}$, $\overline{sgn} = -$ if $sgn = +$, $\overline{sgn} = +$ if $sng = -$, $\mathcal{OP}1 \in \{X, Y\}$ and $\mathcal{OP}2 \in \{U, S\}$

Proof.($\Leftarrow$) $\exists \pi$ of $S_{\neg\varphi} \times S^{deg}_{F_{\neg\varphi}}$ visiting $f^{l}_{\neg\varphi}$ infinitely often $\Rightarrow \sigma \models \neg\varphi$.
Given a subformula ψ of $\neg\varphi$, we prove that for all $i \geq 0$, (i) if ψ occurs positively in $\neg\phi$ $\pi, i \models Enc(\psi)$ implies $\pi, i \models \psi$; (ii)if ψ occurs negatively in $\neg\phi$: $\pi, i \models \neg Enc(\psi)$ implies $\pi, i \models \neg\psi$ From this the theorem follows immediately, since the initial state $\pi, 0 \models Enc(\neg\phi)$. We prove the claim by induction on ψ:

- $\psi = p$: $\pi, i \models \psi$ by construction
- $\psi = \psi_1 \vee \psi_2$: trivially by induction
- $\psi = \psi_1 \wedge \psi_2$: Trivial induction
- $\psi = \neg\psi'$. Trivially holds by induction. (if ψ' occurs negative the subformula occurs positively and vice versa).
- $\psi = X\psi'$: From the transition relation $\pi, i \models v_{X\psi'}$ iff $\pi, i+1 \models Enc(\psi')$; thus, $\pi, i + 1 \models \psi'$ by induction.

- $\psi = \psi_1 U \psi_2$: We split the two cases (positive, negative occurrence). In the first case, either $Enc(\psi_2)$ holds or $Enc(\psi_1)$ holds and $v_{X(\psi_1 U \psi_2)}$ holds. The first case is managed by induction over ψ_2. For the second case we apply the induction for ψ_1 and, since $Enc(\psi) \rightarrow Enc(\psi_2) \in F_{\neg\psi}$ then there is a point $k \geq i$ s.t. $\pi, k \models Enc(\psi_2)$ and (from prophecy variable transition relation) for all $i \leq j < k : \pi, j \models Enc(\psi_1)$; from which we can apply induction and derive the result (for the positive occurrence).
  In the other case, $\pi, i \not\models Enc(\psi_2)$ and either $\pi, i \not\models Enc(\psi_1)$ or $\pi, i \not\models v_{X(\psi_1 U \psi_2)}$. By the transition relation we know that $v_{X(\psi_1 U \psi_2)} \leftrightarrow Enc(\psi_2)' \vee Enc(\psi_1)' \wedge v'_{X_{\psi_1 U \psi_2)}}$; therefore, since $v_{X(\psi_1 U \psi_2)}$ is violated either there is a point k in the future s.t. ψ_1 is violated and all the points j between i and j violates $Enc(\psi_2)$ as well or it is never true that $Enc(\psi_2)$ holds. We can prove this claim by w.o.c. suppose that exists a $k \geq i$ s.t. $\pi, k \models Enc(\psi_2)$ and for all $i \leq j < k : \pi, j \models Enc(\psi_1)$. From the transition relation definition we get that $\pi, k-1 \models v_{X(\psi_1 U \psi_2)}$. Since by hypothesis in all the index $i \leq j < k : \pi, j \models Enc(\psi_1)$ with a straightforward inductive reasoning down to i we derive that $\pi, i \models Enc(\psi_1 U \psi_2)$ which contradicts the statement.
  Finally, we apply induction to $Enc(\psi_1)$ and $Enc(\psi_2)$ and we derive the semantics of the negation of until.
- $\psi = Y\psi'$: $\pi, i \models v_{Y\psi}$ iff $i > 0$ (from initial condition) and $\pi, i-1 \models \psi'$ (from transition relation); thus, $\pi, i \models v_{Y\psi'} \Leftrightarrow i > 0$ and $\pi, i-1 \models Enc(\psi')$. Finally from induction we obtain $i > 0$ and $\pi, i-1 \models \psi'$ proving the case.
- $\psi = \psi_1 S \psi_2$: we prove this case by induction on i; for the base case, we consider the index 0; in this case, all variables $v_{Y\beta}$ are false in $\pi, 0$ and the claim follows trivially; suppose the claim holds for $i-1$, we prove it for i: $\pi, i \models v_{Y(\psi_1 S \psi_2)}$ iff $i > 0$ (initial condition) and either $\pi, i-1 \models \psi_2$ and thus $\pi, i-1 \models \psi_2$ by induction, or that $\pi, i-1 \models \psi_1 \wedge v_{Y(\psi_1 S \psi_2)}$ and thus $\pi, i-1 \models \psi$ by induction.

$\Rightarrow$ $\sigma \models \neg\varphi \Rightarrow \exists \pi$ of $S_{\neg\varphi} \times S^{deg}_{F_{\neg\varphi}}$ visiting $f^l_{\neg\varphi}$ infinitely often.
Given a trace σ and an LTL property φ such that $\sigma \models \neg\varphi$. Let us define the sequence of states s_i as follows: $s_0 = State^+(Enc(\neg\varphi))$; for all $i > 0$, $s_i = State^+(\bigwedge_{v_{X\beta} \in s_{i-1}} \beta)$. The sequence $\pi = s_0, s_1, \ldots$ is a path of $S^l_{\neg\varphi}$ over the trace π. It is easy to see that in each point i if each subformula ψ is satisfied by σ then $\pi, i \models State^+(\psi)$. Therefore, for untils $v_{X(\psi_1 U \psi_2)}$ is true iff in the future there is a point satisfying ψ_2 which guarantees that the degeneralized fairness holds infinitely often.

Proof of Theorem 8

Definition 10. *Let us define the function $State_f$ taking an LTL formula and a trace σ and returning a set of variables:*

- $State_f(a, \sigma, i) = \emptyset$

- $State_f(\psi_1 \wedge \psi_2, \sigma, i) = State_f(\psi_1, \sigma, i) \cup State_f(\psi_2, \sigma, i)$
- $State_f(\psi_1 \vee \psi_2, \sigma, i) = \begin{cases} State_f(\psi_1, \sigma, i) & \text{if } \sigma, i \models \psi_1 \\ State_f(\psi_2, \sigma, i) & \text{else} \end{cases}$
- $State_f(X\psi, \sigma, i) = \{v_{X\psi}\}$
- $State_f(N\psi, \sigma, i) = \{v_{N\psi}\}$
- $State_f(\psi_1 U \psi_2, \sigma, i) = \begin{cases} State_f(\psi_2, \sigma, i) & \text{if } \sigma, i \models \psi_2 \\ State_f(\psi_1, \sigma, i) \cup \{v_{X(\psi_1 U \psi_2}\} & \text{else} \end{cases}$
- $State_f(\psi_1 R \psi_2, \sigma, i) = \begin{cases} State_f(\psi_2, \sigma, i) \cup State_f(\psi_1, \sigma, i) & \text{if } \sigma, i \models \psi_2 \wedge \psi_1 \\ State_f(\psi_1, \sigma, i) \cup \{v_{X(\psi_1 R \psi_2}\} & \text{else} \end{cases}$
- $State_f(Y\psi, \sigma, i) = \{v_{Y\psi}\}$
- $State_f(\psi_1 S \psi_2, \sigma, i) = \begin{cases} State_f(\psi_2, \sigma, i) & \text{if } \sigma, i \models \psi_2 \\ State_f(\psi_1, \sigma, i) \cup \{v_{Y(\psi_1 S \psi_2}\} & \text{else} \end{cases}$

Proof.($\Leftarrow$) Given a subformula ψ of ϕ, we prove that for all $i \geq 0$, if $\pi, i \models Enc(\psi)$, then $\pi, i \models \psi$. From this the theorem follows immediately, since the initial state $\pi, 0 \models Enc(\phi)$.

We prove the claim by induction on ψ:

- $\psi = p$: $\pi, i \models \psi$ by construction
- $\psi = \psi_1 \vee \psi_2$: Trivial induction
- $\psi = \psi_1 \wedge \psi_2$: Trivial induction
- $\psi = N\psi' : \pi, i \models v_{N\psi'}$ by construction, then the transition requires, if $i < |\pi| - 1$, that $\pi, i+1 \models_{fin} Enc(\psi')$; thus, $\pi, i+1 \models_{fin} \psi'$ by induction; if $i = |\pi| - 1$ then $\pi, i \models_{fin} N\psi'$ vacuously.
- $\psi = X\psi'$: Identical to the previous if $i < |\pi| - 1$. We can show that when $v_{X\psi'}$ is true i is not the final assignment. That follows from the assumption that π reaches $f^b_{\neg\varphi}$ which is composed of the conjunction of the negation of all the proof obligations including $v_{X\psi'}$
- $\psi = \psi_1 U \psi_2$: we prove this case by induction on i; for the base case, we consider the index $= |\pi| - 1$ as base case; in this case, all the prophecy variables of ϕ are false in π, i and the claim follows trivially; suppose the claim holds for $i+1$ (which is guaranteed to stay in the path bounds), we prove it for i: $\pi, i \models_{fin} v_{X(\psi_1 U \psi_2)}$ by construction (ψ_2 does not hold), then the transition requires either that $\pi, i+1 \models_{fin} Enc(\psi_2)$ and thus $\pi, i+1 \models_{fin} \psi_2$ by induction, or that $\pi i+1 \models_{fin} Enc(\psi_1) \wedge v_{X(\psi_1 U \psi_2))}$ and thus $\pi, i+1 \models_{fin} \psi$ by induction.
- $\psi = \psi_1 R \psi_2$: The proof is similar to the previous case, the only difference is that in this case in the last state you don't need $v_{N(\psi_1 R \psi_2)}$ to be false to prove the property while the inductive case is more or less the same.
- $\psi = Y\psi'$: $\pi, i \models_{fin} v_{Y\psi}$ by construction, then if $i > 0$ the transition requires that $\pi, i-1 \models_{fin} \psi'$; thus, $\pi, i-1 \models_{fin} \psi'$
- $\psi = \psi_1 S \psi_2$: we prove this case by induction on i; for the base case, we consider the index 0; in this case, all variables $v_{Y\beta}$ are false in $\pi, 0$ and the claim follows trivially; suppose the claim holds for $i-1$, we prove it for i: $\pi, i \models_{fin} v_{Y(\psi_1 S \psi_2)}$ by construction, then if $i > 0$ the transition require

either that $\pi, i-1 \models_{fin} \psi_2$ and thus $\pi, i-1 \models_{fin} \psi_2$ by induction, or that $\pi, i-1 \models_{fin} \psi_1 \wedge v_{Y(\psi_1 S \psi_2)}$ and thus $\pi, i-1 \models_{fin} \psi$ by induction.
- the other cases (Z, T) are similar to the previous ones.

($\Rightarrow$) Given a trace σ and an LTL_f property φ with $\phi = \text{NNF}_f(\neg\varphi)$ such that $\sigma \models \phi$. Let us define the sequence of states s_i as follows: $s_0 = State_f(Enc(\neg\varphi))$; for all $i > 0$, $s_i = State(\bigwedge_{v_{X\beta} \in s_{i-1}} \beta)$. The sequence $\pi = s_0, s_1, \ldots$ is a path of $S^b_{\neg\varphi}$ over the trace π. Let d be the size of σ, then $\pi, d \models_{fin} f^b_{\neg\varphi}$

Proof of Theorem 9

Definition 11. *Let us define the function State taking an LTL formula and a trace σ and returning a set of variables:*

- $State(a, \sigma, i) = \emptyset$
- $State(\psi_1 \wedge \psi_2, \sigma, i) = State(\psi_1, \sigma, i) \cup State(\psi_2, \sigma, i)$
- $State(\psi_1 \vee \psi_2, \sigma, i) = \begin{cases} State(\psi_1, \sigma, i) & \text{if } \sigma, i \models \psi_1 \\ State(\psi_2, \sigma, i) & \text{else} \end{cases}$
- $State(X\psi, \sigma, i) = \{v_{X\psi}\}$
- $State(\psi_1 U \psi_2, \sigma, i) = \begin{cases} State(\psi_2, \sigma, i) & \text{if } \sigma, i \models \psi_2 \\ State(\psi_1, \sigma, i) \cup \{v_{X(\psi_1 U \psi_2}\} & \text{else} \end{cases}$
- $State(Y\psi, \sigma, i) = \{v_{Y\psi}\}$
- $State(\psi_1 S \psi_2, \sigma, i) = \begin{cases} State(\psi_2, \sigma, i) & \text{if } \sigma, i \models \psi_2 \\ State(\psi_1, \sigma, i) \cup \{v_{Y(\psi_1 S \psi_2}\} & \text{else} \end{cases}$

Proof.($\Leftarrow$) Given a subformula ψ of ϕ, we prove that for all $i \geq 0$, if $\pi, i \models Enc(\psi)$, then $\pi, i \models \psi$. From this the theorem follows immediately, since the initial state $\pi, 0 \models Enc(\phi')$.

We prove the claim by induction on ψ:
- $\psi = p$: $\pi, i \models \psi$ by construction
- $\psi = \psi_1 \vee \psi_2$: Trivial induction
- $\psi = X\psi'$: $\pi, i \models v_{X\psi'}$ by construction, then the transition requires that $\pi, i+1 \models Enc(\psi')$; thus, $\pi, i+1 \models \psi'$ by induction
- $\psi = \psi_1 U \psi_2$: we prove this case by induction on i; for the base case, we consider the index i in which π, i reaches $f^b_{\neg\varphi}$; in this case, all the prophecy variables of ϕ are false in π, i and the claim follows trivially; suppose the claim holds for $i+1$, we prove it for i: $\pi, i \models v_{X(\psi_1 U \psi_2)}$ by construction (ψ_2 does not hold), then the transition requires either that $\pi(i+1) \models Enc(\psi_2)$ and thus $\pi, i+1 \models \psi_2$ by induction, or that $\pi i+1 \models E(\psi_1 \wedge X(\psi_1 U \psi_2))$ and thus $\pi, i+1 \models \psi$ by induction.
- $\psi = Y\psi'$: $\pi, i \models v_{Y\psi}$ by construction, then if $i > 0$ the transition requires that $\pi, i-1 \models \psi'$; thus, $\pi, i-1 \models \psi'$
- $\psi = \psi_1 S \psi_2$: we prove this case by induction on i; for the base case, we consider the index 0; in this case, all variables $v_{Y\beta}$ are false in $\pi, 0$ and the claim follows trivially; suppose the claim holds for $i-1$, we prove it for i: $\pi, i \models v_{Y(\psi_1 S \psi_2)}$ by construction, then if $i > 0$ the transition require either that $\pi, i-1 \models \psi_2$ and thus $\pi, i-1 \models \psi_2$ by induction, or that $\pi, i-1 \models \psi_1 \wedge v_{Y(\psi_1 S \psi_2)}$ and thus $\pi, i-1 \models \psi$ by induction.

- the other cases (Z, T) are similar to the previous ones.

($\Rightarrow$) Given a trace σ and a safety property φ such that $\sigma \not\models \varphi$ and $\phi := \text{NNF}(\neg\phi)$. Let us define the sequence of states s_i as follows: $s_0 = State(Enc(\phi))$; for all $i > 0$, $s_i = State(\bigwedge_{v_{X\beta} \in s_{i-1}} \beta)$. The sequence $\pi = s_0, s_1, \ldots$ is a path of $S^b_{\neg\varphi}$ over the trace π. Let d be the size of the bad prefix of σ violating ϕ. Then $\pi, d \models f^b_{\neg\varphi}$