ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-513

LTL_f Satisfiability Checking

Jianwen Li [1] and **Lijun Zhang** [2] and **Geguang Pu**[1] and **Moshe Y. Vardi** [3] and **Jifeng He**[1]

Abstract. We consider here Linear Temporal Logic (LTL) formulas interpreted over *finite* traces. We denote this logic by LTL_f. The existing approach for LTL_f satisfiability checking is based on a reduction to standard LTL satisfiability checking. We describe here a novel direct approach to LTL_f satisfiability checking, where we take advantage of the difference in the semantics between LTL and LTL_f. While LTL satisfiability checking requires finding a *fair cycle* in an appropriate transition system, here we need to search only for a finite trace. This enables us to introduce specialized heuristics, where we also exploit recent progress in Boolean SAT solving. We have implemented our approach in a prototype tool and experiments show that our approach outperforms existing approaches.

1 Introduction

Linear Temporal Logic (LTL) was first introduced into computer science as a property language for the verification for non-terminating reactive systems [9]. Following that, many researches in AI have been attracted by LTL's rich expressiveness. Examples of applications of LTL in AI include temporally extended goals in planning [3], plan constraints [1], and user preferences [13].

In a recent paper [5], De Giacomo and Vardi argued that while standard LTL is interpreted over *infinite* traces, cf. [9], AI applications are typically interested only in *finite* traces. For example, temporally extended goals are viewed as finite desirable sequences of states and a plan is correct if its execution succeeds in yielding one of these desirable sequences. Also in the area of business-process modeling, temporal specifications for declarative workflows are interpreted over finite traces [14]. De Giacomo and Vardi, therefore, introduced LTL_f, which has the same syntax as LTL but is interpreted over finite traces.

In the formal-verification community there is by now a rich body of knowledge regarding automated-reasoning support for LTL. On one hand, there are solid theoretical foundations, cf. [15]. On the other hand, mature software tools have been developed, such as SPOT [4]. Extensive research has been conducted to evaluate these tools, cf. [10]. While the basic theory for LTL_f was presented at [5], no tool has yet to be developed for LTL_f, to the best of our knowledge. Our goal in this paper is to address this gap.

Our main focus here is on the *satisfiability problem*, which asks if a given formula has satisfying model. This most basic automated-reasoning problem has attracted a fair amount of attention for LTL over the past few years as a principled approach to *property assurance*, which seeks to eliminate errors when writing LTL properties, cf. [10, 8].

[1] East China Normal University. Geguang Pu is the corresponding author.
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences.
[3] Rice University, USA.

De Giacomo and Vardi studied the computational complexity of LTL_f satisfiability and showed that it is PSPACE-complete, which is the same complexity as for LTL satisfiability [12]. Their proof of the upper bound uses a reduction of LTL_f satisfiability to LTL satisfiability. That is, for an LTL_f formula ϕ, one can create an LTL formula ϕ' such that ϕ is satisfiable iff ϕ' is satisfiable; furthermore, the translation from ϕ to ϕ' involves only a linear blow-up. The reduction to LTL satisfiability problem can, therefore, take advantage of existing LTL satisfiability solvers [11, 8]. On the other hand, LTL satisfiability checking requires reasoning about infinite traces, which is quite nontrivial algorithmically, cf. [2], due to the required fair-cycle test. Such reasoning is not required for LTL_f satisfiability. A reduction to LTL satisfiability, therefore, may add unnecessary overhead to LTL_f satisfiability checking.

This paper approaches the LTL_f satisfiability problem directly. We develop a direct, and more efficient, algorithm for checking satisfiability of LTL_f, leveraging the existing body of knowledge concerning LTL satisfiability checking. The finite-trace semantics for LTL_f is fully exploited, leading to considerable simplification of the decision procedure and significant performance boost. The finite-trace semantics also enables several heuristics that are not applicable to LTL satisfiability checking. We also leverage the power of advanced Boolean SAT solvers in our decision procedure. We have implemented the new approach and experiments show that this approach significantly outperforms the reduction to LTL satisfiability problems.

The paper is organized as follows. we first introduce the definition of LTL_f, the satisfiability problem, and the associated transition system in Section 2. We then propose a direct satisfiability-checking framework in Section 3. We discuss various optimization strategies in Section 4, and present experimental results in Section 5. Section 6 concludes the paper.

2 Preliminaries

2.1 LTL over Finite Traces

The logic LTL_f is a variant of LTL. Classical LTL formulas are interpreted on infinite traces, whereas LTL_f formulas are defined over the finite traces. Given a set $\mathcal{P}$ of atomic propositions, an LTL_f formula ϕ has the form:

$\phi ::= \mathtt{tt} \mid \mathtt{ff} \mid p \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid X\phi \mid X_w\phi \mid \phi U \phi \mid \phi R \phi$

where X(strong Next), X_w(weak Next), U(Until), and R(Release) are temporal operators. We have $X_w\phi \equiv \neg X \neg \phi$ and $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$. Note that in LTL_f, $X\phi \equiv X_w\phi$ is not true, which is however the case in LTL.

For an atom $a \in \mathcal{P}$, we call it or its negation ($\neg a$) a literal. We use the set L to denote the set of literals, i.e. $L = \mathcal{P} \cup \{\neg a | a \in \mathcal{P}\}$. Other boolean operators, such as $\rightarrow$ and $\leftrightarrow$, can be represented by the combination $(\neg, \vee)$ or $(\neg, \wedge)$, respectively, and we denote the constant *true* as $\mathtt{tt}$ and *false* as $\mathtt{ff}$. Moreover, we use the notations $G\phi$

(Global) and $F\phi$ (Eventually) to represent $\mathtt{ff}R\phi$ and $\mathtt{tt}U\phi$. We use ϕ, ψ to represent LTL_f or LTL formulas, and α, β for propositional formulas.

Note that standard LTL_f has the same syntax as LTL, see [5]. Here, however, we introduce the X_w operator, as we consider LTL_f formulas in NNF (Negation Normal Form), which requires all negations to be pushed all the way down to atoms. So a dual operator for X is necessary. In LTL the dual of X is X itself, while in LTL_f it is X_w.
Proviso: In the rest of paper we assume that all formulas (both LTL and LTL_f) are in NNF, and thus there are types of formulas, based on the primary connective: tt, ff, literal, $\wedge$, $\vee$, X (and X_w in LTL_f), U and R.

The semantics of LTL_f formulas is interpreted over finite traces, which is referred to as the LTL_f interpretations [5]. Given an atom set $\mathcal{P}$, we define $\Sigma := 2^L$. Let $\eta \in \Sigma^*$ with $\eta = \omega_0\omega_1 \ldots \omega_n$, we use $|\eta| = n + 1$ to denote the length of η. Moreover, for $1 \leq i \leq n$, we use the notation η^i to represent $\omega_0\omega_1 \ldots \omega_{i-1}$, which is the prefix of η before position i (i is not included). Similarly, we also use η_i to represent $\omega_i\omega_{i+1} \ldots \omega_n$, which is the suffix of η from position i. Then we define η models ϕ, i.e. $\eta \models \phi$ in the following way:

- $\eta \models \mathtt{tt}$ and $\eta \not\models \mathtt{ff}$;
- If $\phi = p$ is a literal, then $\eta \models \phi$ iff $p \in \eta^1$;
- If $\phi = X\psi$, then $\eta \models \phi$ iff $|\eta| > 1$ and $\eta_1 \models \psi$;
- If $\phi = X_w\psi$, then $\eta \models \phi$ iff $|\eta| > 1$ and $\eta_1 \models \psi$, or $|\eta| = 1$;
- If $\phi = \phi_1 U \phi_2$ is an Until formula, then $\eta \models \phi$ iff there exists $0 \leq i < |\eta|$ such that $\eta_i \models \phi_2$, and for every $0 \leq j < i$ it holds $\eta_j \models \phi_1$ as well;
- If $\phi = \phi_1 R \phi_2$ is a Release formula, then $\eta \models \phi$ iff either for every $0 \leq i < |\eta|$ $\eta_i \models \phi_2$ holds, or there exists $0 \leq i < |\eta|$ such that $\eta_i \models \phi_1$ and for all $0 \leq j \leq i$ it holds $\eta_j \models \phi_2$ as well;
- If $\phi = \phi_1 \wedge \phi_2$, then $\eta \models \phi$ iff $\eta \models \phi_1$ and $\eta \models \phi_2$;
- If $\phi = \phi_1 \vee \phi_2$, then $\eta \models \phi$ iff $\eta \models \phi_1$ or $\eta \models \phi_2$.

The difference between the strong Next (X) and the weak Next (X_w) operators is that X requires a next state in the following while X_w may not. Thus $X_w\phi$ is always true in the last state of a finite trace, since no next state is provided. As a result, in LTL_f $X\mathtt{ff}$ is unsatisfiable, while $X_w\mathtt{ff}$ is satisfiable, which is quite different with that in LTL, where neither $X\mathtt{ff}$ nor $\neg X\neg\mathtt{tt}$ are satisfiable.

Let ϕ be an LTL_f formula, we use $CF(\phi)$ to represent the set of conjuncts in ϕ, i.e. $CF(\phi) = \{\phi_i | \phi_i \in I\}$ if $\phi = \bigwedge_{i\in I} \phi_i$, where the root of ϕ_i is not a conjunction. $DF(\phi)$ (the set of disjuncts) is defined analogously.

2.2 The LTL_f Satisfiability Problem

The satisfiability problem is to check whether, for a given LTL_f formua ϕ, there is a finite trace $\eta \in \Sigma^*$ such that $\eta \models \phi$:

Definition 1 (LTL_f Satisfiability Problem)**.** *Given an LTL_f formula ϕ over the alphabet Σ, we say ϕ is satisfiable iff there is a finite trace $\eta \in \Sigma^*$ such that $\eta \models \phi$.*

One approach is to reduce the LTL_f satisfiability problem to that of LTL.

Theorem 1 ([5])**.** *The Satisfiability problem for LTL_f formulas is PSPACE-complete.*

Proof Sketch: It is easy to reduce the LTL_f satisfiability to LTL satisfiability:

1. Introduce a proposition "*Tail*";
2. Require that *Tail* holds at position 0;
3. Require also that *Tail* stays tt until it turns into ff, and after that stays ff forever (*Tail*$U(G\neg$*Tail*)).
4. The LTL_f formula ϕ is translated into a corresponding LTL formula in the following way:
   - $t(p) \rightarrow p$, where p is a literal;
   - $t(\neg\phi) = \neg t(\phi)$;
   - $t(\phi_1 \wedge \phi_2) \rightarrow t(\phi_1) \wedge t(\phi_2)$;
   - $t(\phi_1 \vee \phi_2) \rightarrow t(\phi_1) \vee t(\phi_2)$;
   - $t(X\psi) \rightarrow X($*Tail* $\wedge\, t(\psi))$;
   - $t(\phi_1 U \phi_2) \rightarrow \phi_1 U($*Tail* $\wedge\, t(\phi_2))$;

(The translation here does not require ϕ in NNF. Thus the X_w and R operators can be handled by the rules $X_w\phi \equiv \neg X\neg\phi$ and $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$.) Finally one can refer to [5] that ϕ is satisfiable iff *Tail* $\wedge$ *Tail*$U(G\neg$*Tail*$) \wedge t(\phi)$ is satisfiable. Also, a PSPACE lower bound is shown in [5] by reduction from STRIPS Planning. □

The reduction approach can take advantage of existing LTL satisfiability solvers. But, there may be an overhead as we need to find a *fair cycle* during LTL satisfiability checking, which is not necessary in LTL_f checking.

2.3 LTL_f Transition System

In [8], Li et al. have proposed using transition systems for checking satisfiability of LTL formulas. Here we adapt this approach to LTL_f. First, we define the *normal form* for LTL_f formulas.

Definition 2 (Normal Form)**.** *The normal form of an LTL_f formula ϕ, denoted as $NF(\phi)$, is a formula set defined as follows:*

- $NF(\phi) = \{\phi \wedge X(\mathtt{tt})\}$ *if* $\phi \neq \mathtt{ff}$ *is a literal. If* $\phi = \mathtt{ff}$*, we define* $NF(\mathtt{ff}) = \emptyset$*;*
- $NF(X\phi/X_w\phi) = \{\mathtt{tt} \wedge X(\psi) \mid \psi \in DF(\phi)\}$*;*
- $NF(\phi_1 U \phi_2) = NF(\phi_2) \cup NF(\phi_1 \wedge X(\phi_1 U \phi_2))$*;*
- $NF(\phi_1 R \phi_2) = NF(\phi_1 \wedge \phi_2) \cup NF(\phi_2 \wedge X(\phi_1 R \phi_2))$*;*
- $NF(\phi_1 \vee \phi_2) = NF(\phi_1) \cup NF(\phi_2)$*;*
- $NF(\phi_1 \wedge \phi_2) = \{(\alpha_1 \wedge \alpha_2) \wedge X(\psi_1 \wedge \psi_2) \mid \forall i = 1, 2.\ \alpha_i \wedge X(\psi_i) \in NF(\phi_i)\}$*;*

For each $\alpha_i \wedge X\phi_i \in NF(\phi)$, we say it a clause of $NF(\phi)$.

(Although the normal forms of X and X_w formulas are the same, we do distinguished bethween them through the accepting conditions introduced below.) Intuitively, each clause $\alpha_i \wedge X\phi_i$ of $NF(\phi)$ indicates that the propositionl formula α_i should hold now and then ϕ_i should hold in the next state. For ϕ_i, we can also compute its normal form. We can repeat this procedure until no new states are required.

Definition 3 (LTL_f Transition System)**.** *Let ϕ be the input formula. The labeled transition system T_ϕ is a tuple $\langle Act, S_\phi, \rightarrow, \phi\rangle$ where: 1). ϕ is the initial state; 2). Act is the set of conjunctive formulas over L_ϕ; 3). the transition relation $\rightarrow \subseteq S_\phi \times Act \times S_\phi$ is defined by: $\psi_1 \xrightarrow{\alpha} \psi_2$ iff there exists $\alpha \wedge X(\psi_2) \in NF(\psi_1)$; and 4). S_ϕ is the smallest set of formulas such that $\psi_1 \in S_\phi$, and $\psi_1 \xrightarrow{\alpha} \psi_2$ implies $\psi_2 \in S_\phi$.*

Note that in LTL transition systems the ff state can be deleted, as it can never be part of a fair cycle. This state must be kept in LTL_f transition systems: a finite trace that reach ff may be accepted in

LTL$_f$, cf. $X_w\mathtt{ff}$. Nevertheless, $\mathtt{ff}$ edges are not allowed both in *LTL$_f$* and LTL transition systems.

A *run* of T_ϕ on finite trace $\eta = \omega_0\omega_1 \dots \omega_n \in \Sigma^*$ is a sequence $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots s_n \xrightarrow{\alpha_n} s_{n+1}$ such that $s_0 = \phi$ and for every $0 \le i \le n$ it holds $\omega_i \models \alpha_i$. We say ψ is *reachable* from ϕ iff there is a run of T_ϕ such that the final state is ψ.

3 *LTL$_f$* Satisfiability-Checking Framework

In this section we present our framework for checking satisfiability of *LTL$_f$* formulas. First we show a simple lemma concerning finite sequences of length 1.

Lemma 1. *For a finite trace $\eta \in \Sigma^*$ and LTL$_f$ formula ϕ, if $|\eta| = 1$ then $\eta \models \phi$ holds iff:*

- $\eta \models \mathtt{tt}$ *and* $\eta \not\models \mathtt{ff}$*;*
- *If $\phi = p$ is a literal, then return true if $\phi \in \eta$. otherwise return false;*
- *If $\phi = \phi_1 \wedge \phi_2$, then return $\eta \models \phi_1$ and $\eta \models \phi_2$;*
- *If $\phi = \phi_1 \vee \phi_2$, then return $\eta \models \phi_1$ or $\eta \models \phi_2$;*
- *If $\phi = X\phi_2$, then return false;*
- *If $\phi = X_w\phi_2$, then return true;*
- *If $\phi = \phi_1 U \phi_2$ or $\phi = \phi_1 R \phi_2$, then return $\eta \models \phi_2$.*

Proof. This lemma can be directly proven from the semantics of *LTL$_f$* formulas by fixing $|\eta| = 1$. □

Now we characterize the satisfaction relation for finite sequences:

Lemma 2. *For a finite trace $\eta = \omega_0\omega_1 \dots \omega_n \in \Sigma^*$ and LTL$_f$ formula ϕ,*

1. *If $n = 0$, then $\eta \models \phi$ iff there exists $\alpha_i \wedge X\phi_i \in NF(\phi)$ such that $\omega_0 \models \alpha_i$ and $CF(\alpha_i) \models \phi$;*
2. *If $n \ge 1$, then $\eta \models \phi$ iff there exists $\alpha_i \wedge X\phi_i \in NF(\phi)$ such that $\omega_0 \models \alpha_i$ and $\eta_1 \models \phi_i$;*
3. *$\eta \models \phi$ iff there exists a run $\phi = \phi_0 \xrightarrow{\alpha_0} \phi_1 \xrightarrow{\alpha_1} \phi_2 \dots \xrightarrow{\alpha_n} \phi_{n+1}$ in T_ϕ such that for every $0 \le i \le n$ it holds that $\omega_i \models \alpha_i$ and $\eta_i \models \phi_i$.*

Proof. 1. $CF(\alpha_i)$ is treated to be a finite trace whose length is 1. We prove the first item by structural induction over ϕ.

- If $\phi = p$, then $\eta \models \phi$ iff $\omega_0 \models p$ and $CF(p) \models \phi$ hold, where $p \wedge X\mathtt{tt}$ is actually in $NF(\phi)$;
- If $\phi = \phi_1 \wedge \phi_2$, then $\eta \models \phi$ holds iff $\eta \models \phi_1$ and $\eta \models \phi_2$ hold, and iff by induction hypothesis, there exists $\beta_i \wedge X\psi_i$ in $NF(\phi_i)$ such that $\omega_0 \models \beta_i$ and $CF(\beta_i) \models \phi_i$ $(i = 1, 2)$. Let $\alpha_i = \beta_1 \wedge \beta_2$ and $\phi_i' = \psi_1 \wedge \psi_2$, then according to Definition 2 we know $\alpha_i \wedge X\phi_i'$ is in $NF(\phi)$, and $\omega_0 \models \alpha_i$ and $CF(\alpha_i) \models \phi$ hold; The proof for the case when $\phi = \phi_1 \vee \phi_2$ is similar;
- Note that $\eta \models X\psi$ is always false, and if $\phi = X_w\psi$ then from Lemma 1 it is always true that $\eta \models X_w\psi$ iff $\mathtt{tt} \wedge X\psi \in NF(\phi)$ and $\mathtt{tt} \models X_w\psi$;
- If $\phi = \phi_1 U \phi_2$, then $\eta \models \phi$ holds iff $\eta \models \phi_2$ holds from Lemma 1, and iff by induction hypothesis, there exists $\alpha_i \wedge X\phi_i \in NF(\phi_2)$ such that $\omega_0 \models \alpha_i$ and $CF(\alpha_i) \models \phi_2$, and thus $CF(\alpha_i) \models \phi$ according to *LTL$_f$* semantics. From Definition 2 we know as well that $\alpha_i \wedge X\phi_i$ is in $NF(\phi)$, thus the proof is done; The proof for the case when $\phi = \phi_1 R \phi_2$ is similar;

2. The second item is also proven by structural induction over ϕ.
   - If $\phi = \mathtt{tt}$ or $\phi = p$, then $\eta \models \phi$ iff $\omega_0 \models \phi$ and $\eta_1 \models \mathtt{tt}$ hold, where $\phi \wedge X\mathtt{tt}$ is actually in $NF(\phi)$;
   - If $\phi = X\phi_2$ or $\phi = X_w\phi_2$, since $|\eta| > 1$ so it is obviously true that $\eta \models \phi$ iff $\omega_0 \models \mathtt{tt}$ and $\eta_1 \models \phi_2$ hold according to *LTL$_f$* semantics, and obviously $\mathtt{tt} \wedge X\phi_2$ is in $NF(\phi)$;
   - If $\phi = \phi_1 \wedge \phi_2$, then $\eta \models \phi$ iff $\eta \models \phi_1$ and $\eta \models \phi_2$, and iff by induction hypothesis, there exists $\beta_i \wedge X\psi_i \in NF(\phi_i)(i = 1, 2)$ such that $\omega_0 \models \beta_i$ and $\eta_1 \models \psi_i$ hold, and iff $\omega_0 \models \beta_1 \wedge \beta_2$ and $\eta_1 \models \psi_1 \wedge \psi_2$ hold, in which $(\beta_1 \wedge \beta_2) \wedge X(\psi_1 \wedge \psi_2)$ is indeed in $NF(\phi)$; The case when $\phi = \phi_1 \vee \phi_2$ is similar;
   - If $\phi = \phi_1 U \phi_2$, then $\eta \models \phi$ iff $\eta \models \phi_2$ or $\eta \models (\phi_1 \wedge X\phi)$. If $\eta \models \phi_2$ holds, then by induction hypothesis iff there exists $\alpha_i \wedge X\phi_i \in NF(\phi_2)$ such that $\omega_0 \models \alpha_i$ and $\eta_1 \models \phi_i$. According to Definition 2 we know $\alpha_i \wedge X\phi_i$ is also $NF(\phi_2)$. On the other hand, if $\eta \models \phi_1 \wedge X\phi$ holds, the proofs for $\wedge$ formulas are already done. Thus, it is true that $\eta \models \phi$ iff there exists $\alpha_i \wedge X\phi_i \in NF(\phi_2)$ such that $\omega_0 \models \alpha_i$ and $\eta_1 \models \phi_i$; The case when $\phi = \phi_1 R \phi_2$ is similar to prove.
3. Applying the first item if $n = 0$ and recursively applying the second item if $n \ge 1$, we can prove the third item.

□

Lemma 2 states that, to check whether a finite trace $\eta = \omega_0\omega_1 \dots \omega_n$ satisfies the *LTL$_f$* formula ϕ, we can find a run of T_ϕ on η such that η can finally reach the transition $\phi_n \xrightarrow{\alpha_n} \phi_{n+1}$ and satisfies $\omega_n \models \alpha_n$, and moreover $CF(\alpha_n) \models \phi_n$. Now we can give the main theorem of this paper.

Theorem 2. *Given an LTL$_f$ formula ϕ and a finite trace $\eta = \omega_0 \dots \omega_n (n \ge 0)$, we have that $\eta \models \phi$ holds iff there exists a run of T_ϕ on η which ends at the transition $\psi_1 \xrightarrow{\alpha} \psi_2$ satisfying $CF(\alpha) \models \psi_1$.*

Proof. Combine the first and third items in Lemma 2, and we can easily prove this theorem. □

We say the state ψ_1 in T_ϕ is *accepting*, if there exists a transition $\psi_1 \xrightarrow{\alpha} \psi_2$ such that $CF(\alpha) \models \psi_1$. Theorem 2 implies that, the formula ϕ is satisfiable if and only if there exists an accepting state ψ_1 in T_ϕ which is reachable from the initial state ϕ. Based on this observation, we now propose a simple on-the-fly satisfiability-checking framework for *LTL$_f$* as follows:

1. If ϕ equals $\mathtt{tt}$, return ϕ is *satisfiable*;
2. The checking is processed on the transition system T_ϕ on-the-fly, i.e. computing the reachable states step by step with the DFS (Depth First Search) manner, until an accepting one is reached: Here we return *satisfiable*;
3. Finally we return *unsatisfiable* if all states in the whole transition system are explored.

The complexity of our algorithm mainly depends on the size of constructed transition system. The system construction is the same as the one for LTL proposed in [8]. Given an *LTL$_f$* formula ϕ, the constructed transition system T_ϕ has at worst the size of $2^{cl(\phi)}$, where $cl(\phi)$ is the set of subformulas of ϕ.

4 Optimizations

In this section we propose some optimization strategies by exploiting SAT solvers. First we study the relationship between the satisfiability problems for *LTL$_f$* and LTL formulas.

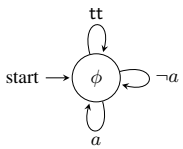

Figure 1: The transition system of $\phi = GFa \wedge GF\neg a$.

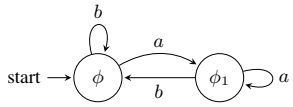

Figure 2: The transition system of $\phi = G(aUb)$. Note $\phi_1 = \phi \wedge aUb$.

4.1 Relating to LTL Satisfiability

In this section we discuss some connections between LTL_f and LTL formulas. We say an LTL_f formula ϕ is X_w*-free* iff ϕ does not have the X_w operator. Note thatLTL_f formulas may contain the X_w operator, while standard LTL ones do not. Here we consider X_w-free formulas, in which LTL_f and LTL have the same syntax. First the following lemma shows how to extend a finite trace into an infinite one but still preserve the satisfaction from LTL_f to LTL:

Lemma 3. *Let $\eta = \omega_0$ and ϕ an LTL_f formula which is X_w-free, then $\eta \models \phi$ implies $\eta^\omega \models \phi$ when ϕ is considered as an LTL formula.*

Proof. We prove it by structural induction over ϕ:

- If ϕ is a literal p, then $\eta \models p$ implies $p \in \eta$. Thus $\eta^\omega \models \phi$ is true; And if ϕ is $\mathtt{tt}$, then $\eta^\omega \models \mathtt{tt}$ is obviously true;
- If $\phi = \phi_1 \wedge \phi_2$, then $\eta \models \phi$ implies $\eta \models \phi_1$ and $\eta \models \phi_2$. By induction hypothesis we have $\eta^\omega \models \phi_1$ and $\eta^\omega \models \phi_2$. So $\eta^\omega \models \phi_1 \wedge \phi_2$; The proof is similar when $\phi = \phi_1 \vee \phi_2$;
- If $\phi = X\psi$, then according to Lemma 1 we know $\eta \models \phi$ cannot happen; And since ϕ is X_w-free, so ϕ cannot be a X_w formula;
- If $\phi = \phi_1 U \phi_2$, then $\eta \models \phi$ implies $\eta \models \phi_2$ according to Lemma 1. By induction hypothesis we have $\eta^\omega \models \phi_2$. Thus $\eta^\omega \models \phi$ is true from the LTL semantics; Similarly when $\phi = \phi_1 R \phi_2$, we know for every $i \geq 0$ it is true that $(\xi_i = \eta^\omega) \models \phi_2$. Thus $\eta^\omega \models \phi$ holds from the LTL semantics; The proof is done.

□

We showed earlier that LTL_f satisfiability can be reduced to LTL satisfiability problem. We show that the satisfiability of some LTL_f formulas implies satisfiability of LTL formulas:

Theorem 3. *Let ϕ be an X_w-free formula. If ϕ is satisfiable as an LTL_f formula, then ϕ is also satisfiable as an LTL formula.*

Proof. Assume ϕ is a X_w-free LTL_f formula, and is satisfiable. Let $\eta = \omega_0 \dots \omega_n$ such that $\eta \models \phi$. Now we interpret ϕ as an LTL formula. Combining Lemma 2 and Lemma 3, we get that $\xi \models \phi$ where $\xi = \omega_0 \dots \omega_{n-1}(\omega_n)^\omega$. □

Equivalently, if ϕ is an LTL formula and ϕ is unsatisfiable, then the LTL_f formula ϕ is also unsatisfiable. Note here the LTL_f formula ϕ is X_w-free since it can be considered as an LTL formula.

Example 1. • *Consider the X_w-free formula $\phi = GFa \wedge GF\neg a$, whose transition system is shown in Figure 1. If ϕ is treated as an LTL formula, then we know that the infinite trace $(\{a\}\{\neg a\})^\omega$ satisfies ϕ. However, if ϕ is considered to be an LTL_f formula, then we know from that no accepting state exists in the transition system, so it is unsatisfiable. It is due to the fact that no transition $\psi_1 \xrightarrow{\alpha} \psi_2$ in T_ϕ satisfies the condition $CF(\alpha) \models \psi_1$.*

- *Consider another example formula $\phi = G(aUb)$, whose transition system is shown in Figure 2. Here we can find an accepting state (ϕ, as $\phi \xrightarrow{b} \phi$ and $CF(b) \models \phi$ hold). Thus we know that ϕ is satisfiable, interpreted over both finite or infinite traces.*

4.2 Obligation Formulas

For an LTL formula ϕ, Li et al. [7] have defined its *obligation formula of* (ϕ) and show that if *of* (ϕ) is satisfiable then ϕ is satisfiable. Since *of* (ϕ) is essentially a boolean formula, so we can check it efficiently using modern SAT solvers. However this cannot apply to LTL_f directly, which we illustrate in the following example.

Example 2. *Consider $\phi = GXa$, where α is a satisfiable propositional formula. It is easy to see that it is satisfiable if it is an LTL formula (with respect to some word a^ω), while unsatisfiable when it is an LTL_f formula (because no finite trace can end with the point satisfying Xa). From [7], the obligation formula of ϕ is of $(\phi) = a$, which is obviously satisfiable. So the satisfiability of obligation formula implies the satisfiability of LTL formulas, but not that of LTL_f formulas.*

We now show how to handle of Next operators (X and X_w) after the Release operators. For a formula ϕ, we define three obligation formulas:

Definition 4 (Obligation Formulas)**.** *Given an LTL_f formula ϕ, we define three kinds of obligation formulas: global obligation formula, release obligation formula, and general obligation formula–denoted as $ofg(\phi)$, $ofr(\phi)$ and $off(\phi)$, by induction over ϕ. (We use ofx as a generic reference to ofg, ofr, and off.)*

- $ofx(\phi) = \mathtt{tt}$ *if* $\phi = \mathtt{tt}$*; and* $ofx(\phi) = \mathtt{ff}$ *if* $\phi = \mathtt{ff}$*;*
- *If $\phi = p$ is a literal, then $ofx(\phi) = p$;*
- *If $\phi = \phi_1 \wedge \phi_2$, then $ofx(\phi) = ofx(\phi_1) \wedge ofx(\phi_2)$;*
- *If $\phi = \phi_1 \vee \phi_2$, then $ofx(\phi) = ofx(\phi_1) \vee ofx(\phi_2)$;*
- *If $\phi = X\phi_2$, then $off(\phi) = off(\phi_2)$, $ofr(\phi) = \mathtt{ff}$ and $ofg(\phi) = \mathtt{ff}$;*
- *If $\phi = X_w\phi_2$, then $off(\phi) = off(\phi_2)$, $ofr(\phi) = \mathtt{ff}$ and $ofg(\phi) = \mathtt{tt}$;*
- *If $\phi = \phi_1 U \phi_2$, then $ofx(\phi) = ofx(\phi_2)$.*
- *If $\phi = \phi_1 R \phi_2$, then $off(\phi) = ofr(\phi)$, $ofr(\phi) = ofr(\phi_2)$ and $ofg(\phi) = ofg(\phi_2)$*

For example in the third item, the equation represents actually three: $off(\phi) = off(\phi_1) \wedge off(\phi_2)$, $ofr(\phi) = ofr(\phi_1) \wedge ofr(\phi_2)$ and $ofg(\phi) = ofg(\phi_1) \wedge ofg(\phi_2)$.

For $off(\phi)$, the changes in comparison to [7] are the definition for release formulas, and introducing the X_w operator. For example, we have that $off(GXa)$ is $\mathtt{ff}$ rather than a. Moreover, since the LTL_f formula GX_wa is satisfiable, the definition of $ofg(\phi)$ is required to identify this situation. (Below we show a fast satisfiability-checking strategy that uses global obligation formulas.)

The obligation-acceleration optimization works as follows:

Theorem 4 (Obligation Acceleration)**.** *For an LTL_f formula ϕ, if $off(\phi)$ is satisfiable then ϕ is satisfiable.*

Proof. Since $off(\phi)$ is satisfiable, there exists $A \in \Sigma$ such that $A \models off(\phi)$. We prove that there exists $\eta = A^n$ where $n \geq 1$ such that $\eta \models \phi$, by structural induction over ϕ. Note the cases $\phi = \mathtt{tt}$ or $\phi = p$ are trivial. For other cases:

- If $\phi = \phi_1 \wedge \phi_2$, then $\mathit{off}(\phi) = \mathit{off}(\phi_1) \wedge \mathit{off}(\phi_2)$ from Definition 4. So $\mathit{off}(\phi)$ is satisfiable implies that there exists $A \models \mathit{off}(\phi_1)$ and $A \models \mathit{off}(\phi_2)$. By induction hypothesis there exists $\eta_i = A^{n_i}$ ($n_i \geq 0$) such that $\eta_i \models \phi_i$ ($i = 1, 2$). Assume $n_1 \geq n_2$, then let $\eta = \eta_1$. Then, $\eta \models \phi_1 \wedge \phi_2$. The case when $\phi = \phi_1 \vee \phi_2$ can be proved similarly;
- If $\phi = X\phi_2$ or $\phi = X_w\phi_2$, then $\mathit{off}(\phi)$ is satisfiable iff $\mathit{off}(\phi_2)$ is satisfiable. So there exists A models ϕ_2. By induction hypothesis, there exists n such that $A^n \models \phi_2$, thus according to LTL_f semantics, we know $A^{n+1} \models \phi$;
- If $\phi = \phi_1 R \phi_2$, then $\mathit{off}(\phi) = \mathit{ofr}(\phi_2)$. Thus $\mathit{ofr}(\phi_2)$ is also satisfiable. So there exists $A \models \mathit{ofr}(\phi_2)$, based on which we can show that $A \models \phi_2$ by structural induction over ϕ_2 by a similar proof. Thus Let $\eta = A$ and according to Lemma 1 we know $\eta \models \phi_2$ implies $\eta \models \phi$. The case for Until can be treated in a similar way, thus the proof is done.

□

4.3 A Complete Acceleration Technique for Global Formulas

The obligation-acceleration technique (Theorem 4) is sound but not complete, see the formula $\phi = a \wedge GF(\neg a)$, in which $\mathit{off}(\phi)$ is unsatisfiable, while ϕ is, in fact, satisfiable. In the following, we prove that both soundness and completeness hold for the *global* LTL_f formulas, which are formulas of the form of $G\psi$, where ψ is an arbitrary LTL_f formula.

Theorem 5 (Obligation Acceleration for Global formulas)**.** *For a global LTL_f formula $\phi = G\psi$, we have that ϕ is satisfiable iff $\mathit{ofg}(\psi)$ is satisfiable.*

Proof. For the forward direction, assume that ϕ is satisfiable. It implies that there is a finite trace η satisfying ϕ. According to Theorem 2, η can run on T_ϕ and reaches an accepting state ψ_1, i.e., $\psi_1 \xrightarrow{\alpha} \psi_2$ and $CF(\alpha) \models \psi_1$. Since ϕ is a global formula and ψ_1 is reachable from ϕ, it is not hard to prove that $CF(\phi) \subseteq CF(\psi_1)$ from Definition 3. So $CF(\alpha) \models \phi$ is also true. Since ϕ is a global formula so $CF(\alpha) \models \psi$ holds from Lemma 1. Then one can prove that $CF(\alpha) \models \mathit{ofg}(\psi)$ by structural induction over ψ (it is left to readers here), which implies that $\mathit{ofg}(\psi)$ is satisfiable.

For the backward direction, assume $\mathit{ofg}(\psi)$ is satisfiable. So there exists $A \in \Sigma$ such that $A \models \mathit{ofg}(\psi)$. Then one can prove $A \models \phi$ is also true by structural induction over ψ ($\phi = G\psi$). For paper limit, this proof is left to readers. So ϕ is satisfiable. The proof is done. □

4.4 Acceleration for Unsatisfiable Formulas

Theorem 3 indicates that if an LTL formula ϕ (of course X_w-free) is unsatisfiable, then the LTL_f formula ϕ is also unsatisfiable. As a result, optimizations for unsatisfiable LTL formulas, for instance those in [7], can be used directly to check unsatisfiable X_w-free LTL_f formulas.

5 Experiments

In this section we present an experimental evaluation. The algorithms are implemented in the *aalta* tool[4]. We have implemented three optimization strategies. They are 1). *off*: the obligation acceleration technique for LTL_f(Theorem 4); 2). *ofg*: the obligation acceleration for global LTL_f formula (Theorem 5); 3). *ofp*: the acceleration for unsatisfiable formulas (Section 4.4). Note that all three optimizations can benefit from the power of modern SAT solvers.

We compare our algorithm with the approach using off-the-shelf tools for checking LTL satisfiability. We choose the tool *Polsat*, a portfolio LTL solver, which was introduced in [6]. One main feature of *Polsat* is that it integrates most existing LTL satisfiability solvers (see [6]); consequently, it is currently the best-of-breed LTL satisfiability solver. The input of *aalta* is directly an LTL_f formula ϕ, while that of *Polsat* should be $\mathit{Tail} \wedge \mathit{TailUG}(\neg \mathit{Tail}) \wedge t(\phi)$, which is the LTL formula that is equi-satisfiable with the LTL_f formula ϕ.

The experimental platform of this paper is the BlueBiou cluster [5] at Rice university. The cluster consists of 47 IBM Power 755 nodes, each of which contains four eight-core POWER7 processors running at 3.86GHz. In our experiments, both *aalta* and *Polsat* occupy a unique node, and *Polsat* runs all its integrated solvers in parallel by using independent cores of the node. The time is measured by Unix time command, and each test case has the maximal limitation of 60 seconds.

Since LTL formulas are also LTL_f formulas, we use existing LTL benchmarks to test the tools. We compare the results from both tools, and no inconsistency occurs.

5.1 Schuppan-collected Formulas

We consider first the benchmarks introduced in previous works [11]. The benchmark suite there include earlier benchmark suites (e.g., [10]), and we refer to this suite as *Schuppan-collected*. The *Schuppan-collected* suite has a total amount of 7448 formulas. The different types of benchmarks are shown in the first column of Table 1.

Table 1: Experimental results on Schuppan-collected formulas.

Formula type	*aalta*(sec.)	*Polsat*(sec.)	*Polsat*/*aalta*
/acacia/example	1.5	3.3	2.2
/acacia/demo-v3	1.4	604.7	431.9
/acacia/demo-v22	2.0	1.3	0.65
/alaska/lift	23.0	7319.6	318.2
/alaska/szymanski	1.2	7.3	6.1
/anzu/amba	2120.9	2052.9	0.97
/anzu/genbuf	3606.9	3717.9	1.0
/rozier/counter	1840.3	3009.3	1.6
/rozier/formulas	552.9	467.0	0.8
/rozier/pattern	22.9	49.9	2.1
/schuppan/O1formula	2.9	7.1	2.4
/schuppan/O2formula	3.1	1265.0	408.1
/schuppan/phltl	226.3	602.5	2.6
/trp/N5x	10.5	42.0	4.0
/trp/N5y	2764.9	2777.4	1.0
/trp/N12x	22.8	24061.1	1055.3
/trp/N12y	4040.2	4049.2	1.0
Total	15244.2	50038.2	3.2

Table 1 shows the experimental results on *Schuppan-collected* benchmarks. The fourth column of the table shows the speed-up of *aalta* relative to *Polsat*. One can see that the results from *aalta* outperforms those from *Polsat*, often by several orders of magnitudes. We explain some of them.

The formulas in "Schuppan-collected/alaska/lift" are mostly unsatisfiable, which can be handled by the *ofg* technique of *aalta*. On the other side, *Polsat* needs more than 300 times to finish the checking. The same happens on the "Schuppan-collected/trp/N12x" patterns, in which *aalta* is more than 1000 times faster. For the "Schuppan-collected/schuppan/O2formula" pattern formulas, *aalta* scales better due to the *ofp* technique.

[4] www.lab205.org/aalta

[5] http://www.rcsg.rice.edu/sharecore/bluebiou/

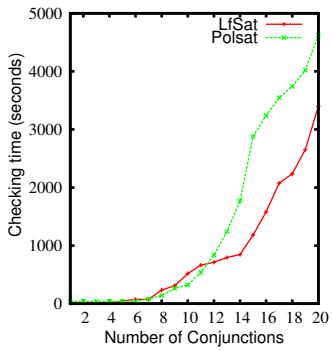

Figure 3: Experimental results on random conjunction formulas.

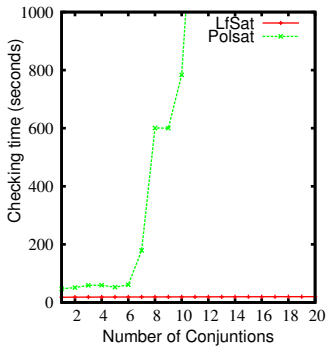

Figure 4: Experimental results on global random conjunction formulas.

Among the results from *aalta*, totally 5879 out of 7448 formulas in the benchmark are checked by using the *off* technique. This indicates the *off* technique is very efficient. Moreover, 84 of them are finished by exploring whole system in the worst time, which requires further improvement. Overall, we can see *Polsat* is three times slower on this benchmark suite than *aalta*.

5.2 Random Conjunction Formulas

Random conjunction formulas have the form of $\bigwedge_{1\leq i\leq n} P_i$, where P_i is randomly selected from typical small pattern formulas widely used in model checking [8]. By randomly choosing the that atoms the small patterns use, a large number of random conjunction formulas can be generated. More specially, to evaluate the performance on global formulas, we also fixed the selected P_i by a random global pattern, and thus create a set of global formulas. In our experiments, we test 10,000 cases each for both random conjunction and global random conjunction formulas, with the number of conjunctions varying from 1 to 20 and 500 cases for each number.

Figure 3 shows the comparison results on random conjunction formulas. On average *aalta* earns about 10% improving performance on this kind of formulas. Among all the 10,000 cases, 8059 of them are checked by the *off* technique; 1105 of them are obtained by the *ofg* technique; 508 are acquired by the *ofp* technique; and another 107 are from an accepting state. There are also 109 formulas equivalent to `tt` or `ff`, which can be directly checked. In the worst case, 76 formulas are finished by exploring the whole transition system. About 36 formulas fail to be checked within 60 seconds by *aalta*. Statistics above show the optimizations are very useful.

Moreover, one can conclude from Figure 4 that, *aalta* dominates *Polsat* when performing on the global random conjunction formulas. As the *ofg* technique is both sound and complete for global formulas and invokes SAT solvers only once, so *aalta* performs almost constant time for checking both satisfiable and unsatisfiable formulas. Compared with that, *Polsat* takes an ordinary checking performance for this kind of special formulas. Indeed, the *ofg* technique is considered to play the crucial role on checking global LTL_f formulas.

6 Conclusion

In this paper we have proposed a novel LTL_f satisfiability-checking framework based on the LTL_f *transition system*. Meanwhile, three different optimizations are introduced to accelerate the checking process by using the power of modern SAT solvers, in which particularly the *ofg* optimization plays the crucial role on checking global formulas. The experimental results show that, the checking approach proposed in this paper is clearly superior to the reduction to LTL satisfiability checking.

7 Acknowledgement

We thank anonymous reviewers for the useful comments. Geguang Pu is partially supported by Shanghai Knowledge Service Platform No. ZF1213. Jianwen Li is partially supported by SHEITC Project 130407 and NSFC Project No. 91118007. Jifeng He is partially supported by NSFC Project No. 61021004. Lijun Zhang is supported by NSFC project No. 61361136002. Moshe Vardi is supported in part by NSF grants CNS 1049862 and CCF-1139011, by NSF Expeditions in Computing project "ExCAPE: Expeditions in Computer Augmented Program Engineering", by BSF grant 9800096, and by gift from Intel.

REFERENCES

[1] F. Bacchus and F. Kabanza, 'Using temporal logic to express search control knowledge for planning', *Artificial Intelligence*, **116**(1–2), 123–191, (2000).
[2] C. Courcoubetis, M.Y. Vardi, P. Wolper, and M. Yannakakis, 'Memory efficient algorithms for the verification of temporal properties', *Formal Methods in System Design*, **1**, 275–288, (1992).
[3] G. De Giacomo and M.Y. Vardi, 'Automata-theoretic approach to planning for temporally extended goals', in *Proc. European Conf. on Planning*, Lecture Notes in AI 1809, pp. 226–238. Springer, (1999).
[4] A. Duret-Lutz and D. Poitrenaud, 'SPOT: An extensible model checking library using transition-based generalized büchi automata', in *Proc. 12th Int'l Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 76–83. IEEE Computer Society, (2004).
[5] G. De Giacomo and M. Vardi, 'Linear temporal logic and linear dynamic logic on finite traces', in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI'13, pp. 2000–2007. AAAI Press, (2013).
[6] J. Li, G. Pu, L. Zhang, M. Y. Vardi, and J. He, 'Polsat: A portfolio LTL satisfiability solver', *CoRR*, **abs/1311.1602**, (2013).
[7] J. Li, G. Pu, L. Zhang, M. Y. Vardi, and J. He, 'Fast LTL satisfiability checking by sat solvers', *CoRR*, **abs/1401.5677**, (2014).
[8] J. Li, L. Zhang, G. Pu, M. Vardi, and J. He, 'LTL satisfiability checking revisited', in *The 20th International Symposium on Temporal Representation and Reasoning*, pp. 91–98, (2013).
[9] A. Pnueli, 'The temporal logic of programs', in *Proc. 18th IEEE Symp. on Foundations of Computer Science*, pp. 46–57, (1977).
[10] K.Y. Rozier and M.Y. Vardi, 'LTL satisfiability checking', *Int'l J. on Software Tools for Technology Transfer*, **12**(2), 1230–137, (2010).
[11] V. Schuppan and L. Darmawan, 'Evaluating LTL satisfiability solvers', in *Proceedings of the 9th international conference on Automated technology for verification and analysis*, AVTA'11, pp. 397–413. Springer-Verlag, (2011).
[12] A.P. Sistla and E.M. Clarke, 'The complexity of propositional linear temporal logic', *Journal of the ACM*, **32**, 733–749, (1985).
[13] S. Sohrabi, J. A. Baier, and S. A. McIlraith, 'Preferred explanations: Theory and generation via planning', in *Proceedings of the 25th Conference on Artificial Intelligence (AAAI-11)*, pp. 261–267, San Francisco, USA, (August 2011). Accepted as both oral and poster presentation.
[14] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, 'Declarative workflows: Balancing between flexibility and support.', *Computer Science - R&D*, 99–113, (2009).
[15] M.Y. Vardi, 'An automata-theoretic approach to linear temporal logic', in *Logics for Concurrency: Structure versus Automata*, eds., F. Moller and G. Birtwistle, volume 1043 of *Lecture Notes in Computer Science*, pp. 238–266. Springer, (1996).