Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Enhancing unsatisfiable cores for LTL with information on temporal relevance [☆]

Viktor Schuppan

ARTICLE INFO

Article history:
Received 7 July 2015
Accepted 12 January 2016
Available online xxxx

Keywords:
LTL
Unsatisfiable cores
Vacuity
Temporal resolution
Resolution graphs
Parikh images

ABSTRACT

LTL is frequently used to express specifications in many domains such as embedded systems or business processes. Witnesses can help to understand why an LTL specification is satisfiable, and a number of approaches exist to make understanding a witness easier. In the case of unsatisfiable specifications unsatisfiable cores (UCs), i.e., parts of an unsatisfiable formula that are themselves unsatisfiable, are a well established means for debugging. However, little work has been done to help understanding a UC of an unsatisfiable LTL formula. In this paper we suggest to enhance a UC of an unsatisfiable LTL formula with information about the time points at which the subformulas of the UC are relevant for unsatisfiability. In previous work we showed how to obtain a UC in LTL by translating the LTL formula into a clausal normal form, applying temporal resolution, extracting a clausal UC from the resolution proof, and mapping the clausal UC back to a UC in LTL. In this paper we extend that method by extracting information at which time points the clauses of a clausal UC are relevant for unsatisfiability from a resolution proof and by transferring that information to a UC in LTL. We implement our method in `TRP++`, and we experimentally evaluate it.

1. Introduction

1.1. Motivation

Typically, a specification is expected to be satisfiable. If it turns out to be unsatisfiable, finding a reason for unsatisfiability can help with the ensuing debugging. Given the sizes of specifications of real world systems (e.g., [2]) automated support for determining a reason for unsatisfiability of a specification is crucial. For many specification languages it is possible to point out a part of the unsatisfiable specification, which has been obtained by removing or relaxing parts of the unsatisfiable specification and which is by itself unsatisfiable, as a reason for unsatisfiability (e.g., [3–5]). In some domains such as SAT (e.g., [6–8]), SMT (e.g., [9]), declarative specifications (e.g., [10]), and LTL (e.g., [3]) this is called an unsatisfiable core (UC).

LTL (e.g., [11,12]) and its relatives are important specification languages for reactive systems (e.g., [13]) and for business processes (e.g., [14]). Experience in verification (e.g., [15,16]) and in synthesis (e.g., [17]) has led to specifications in LTL becoming objects of analysis themselves. Clearly, determining satisfiability of a specification in LTL is an important check (e.g., [18]), and providing a UC for an unsatisfiable specification can help the user track down the problem (e.g., [19]). Besides checking satisfiability other, less simplistic, ways to examine an LTL specification ϕ exist [20], and understanding

[☆] A preliminary version of this paper appeared in [1].
E-mail address: Viktor.Schuppan@gmx.de.
URL: http://www.schuppan.de/viktor/.

http://dx.doi.org/10.1016/j.tcs.2016.01.014
0304-3975/© 2016 Elsevier B.V. All rights reserved.

their results also benefits from availability of UCs. First, one can ask whether a certain scenario ϕ', given as an LTL formula, is permitted by ϕ. That is the case iff $\phi \wedge \phi'$ is satisfiable. Second, one can check whether ϕ ensures a certain LTL property ϕ''. ϕ'' holds in ϕ iff $\phi \wedge \neg\phi''$ is unsatisfiable. In the first case, if the scenario turns out not to be permitted by the specification, a UC can help to understand which parts of the specification and the scenario are responsible for that. In the second case a UC can show which parts of the specification imply the property. Moreover, if there are parts of the property that are not part of the UC, then those parts of the property could be strengthened without invalidating the property in the specification; i.e., the property is vacuously satisfied (e.g., [15,21–24,16]).

Trying to help users to understand counterexamples in verification, which are essentially witnesses to satisfiable formulas, is a well established research topic (see, e.g., [25] for some references). In particular, it is common to add information to a counterexample on which parts of a counterexample are relevant at which points in time (e.g., [26,25]). According to [25] such explanations are an integral part of every counterexample trace in IBM's verification platform RuleBase PE. Checks for vacuous specifications, which are closely related to UCs [3,27], are an important feature of industrial hardware verification tools (see, e.g., [15,22]). In the academic world UCs are an important part of design methods for embedded systems (e.g., [20]) as well as for business processes (e.g., [19]). Despite this relevance of UCs efforts to provide additional information in the context of UCs or vacuity have remained isolated (e.g., [28]). In this paper we suggest to enhance UCs for LTL with information on the time points at which their subformulas are relevant for unsatisfiability.

1.2. Example

As illustration consider the example in (1). It can be read as "globally p and next time not p" (an alternative verbalization to "globally" is "always"). It is evaluated on infinite words over the alphabet $\{\emptyset, \{p\}\}$; intuitively, a word maps each time point in $\mathbb{N} = 0, 1, 2, \ldots$ to the set of atomic propositions TRUE at that time point. The first conjunct, $\mathbf{G}p$, requires p to be TRUE at all time points, which of course includes time point 1. The second conjunct, $\mathbf{X}\neg p$, requires p to be FALSE at time point 1. Clearly, on any word at most one of the two conjuncts can hold, i.e., (1) is unsatisfiable.

$$(\mathbf{G}p) \wedge \mathbf{X}\neg p \tag{1}$$

When (1) is evaluated on some word π according to the standard semantics of LTL (see Sec. 3), (1) and both of its conjuncts, $\mathbf{G}p$ and $\mathbf{X}\neg p$, are evaluated at time point 0, the operand of the $\mathbf{G}$ operator, p, is evaluated at all time points in $\mathbb{N}$, and the operand of the $\mathbf{X}$ operator, $\neg p$, as well as its operand, p, are evaluated at time point 1. We can include this information into (1) by writing the set of time points at which an operand is evaluated directly below the corresponding operator. Note that in this scheme there is no place for the set of time points at which (1) itself is evaluated; however, (1) (as any LTL formula) will always be evaluated only at time point 0, so this need not be spelled out explicitly. We then obtain (2).

$$(\underset{\mathbb{N}}{\mathbf{G}}\, p) \underset{\{0\},\{0\}}{\wedge} \underset{\{1\}}{\mathbf{X}} \underset{\{1\}}{\neg}\, p \tag{2}$$

Remember that the second conjunct, $\mathbf{X}\neg p$, requires p to be FALSE at time point 1. Therefore, to conclude unsatisfiability of (1) it is sufficient to know that the first conjunct, $\mathbf{G}p$, requires p to be TRUE at time point 1; the fact that $\mathbf{G}p$ requires p to be TRUE also at all time points in $\mathbb{N} \setminus \{1\}$ is immaterial. This means that in the evaluation of $\mathbf{G}p$ the operand p would only need to be evaluated at time point 1. At all other time points in $\mathbb{N} \setminus \{1\}$ it could be replaced with, e.g., TRUE without losing unsatisfiability. Using this information, (2) can be modified by replacing $\mathbb{N}$ below $\mathbf{G}$ with $\{1\}$, obtaining (3). (3) can be seen as a UC of (1).

$$(\underset{\{1\}}{\mathbf{G}}\, p) \underset{\{0\},\{0\}}{\wedge} \underset{\{1\}}{\mathbf{X}} \underset{\{1\}}{\neg}\, p \tag{3}$$

1.3. Contributions

Enhancing UCs for LTL with sets of time points In [3,27] a basic notion of UCs for LTL is used that replaces positive polarity occurrences of subformulas of an LTL formula ϕ with TRUE and negative polarity occurrences of subformulas of ϕ with FALSE provided that the modified formula is still unsatisfiable. After [3] we term that notion of UCs "UCs for LTL via syntax trees". In this paper we extend that notion of UCs by incorporating information on the time points at which the occurrences of the subformulas of a UC, which were not replaced with TRUE or FALSE, are relevant for unsatisfiability.

UCs for LTL with sets of time points can help users to understand why a UC is unsatisfiable by making the following information explicit. (i) Sets of time points can show that invariants only need to hold at certain time points rather than always to guarantee unsatisfiability. For an example see (3). (ii) Sets of time points can make cyclic interaction between subformulas that lead to unsatisfiability clear, including period and offset of the cycle. This is particularly noteworthy because, as is well known, LTL cannot count (e.g., [29]). For an example see (5). (iii) Finally, because positive and negative polarity occurrences of propositions need to interact at the same time points to obtain unsatisfiability, sets of time points can limit the subformulas that need to be taken into account when trying to understand why a UC is unsatisfiable. For an example see (10).