Enhancing Unsatisfiable Cores for LTL with Information on Temporal Relevance

Viktor Schuppan

LTL Specification Validation with Satisfiability

LTL + relatives widely used specification languages; methodologies exist:

- Embedded systems: e.g., [EF06]; [Pil+06].
- Business processes: e.g., [PA06]; [Awa+12].

Examples of satisfiability in validation checks of an LTL specification $\phi$:

- Satisfiability of $\phi$ (e.g., [RV10,Awa+12]).
- Feasibility of LTL scenario $\phi'$ in $\phi$: satisfiability of $\phi \land \phi'$ (e.g., [Pil+06]).
- Implication of desired LTL property $\phi''$ by $\phi$: unsatisfiability of $\phi \land \neg \phi''$ (e.g., [Pil+06]).

An unsatisfiable core (UC) is an unsatisfiable formula $\phi'$ that is derived from another unsatisfiable formula $\phi$. $\phi'$ focuses on a reason for $\phi$ being unsatisfiable.

UCs can help understanding results of validation checks.
Linear Temporal Logic (LTL)

LTL formulas are evaluated on infinite sequences of sets of atomic propositions, i.e., $\pi \in (2^{AP})^\omega$. Constants and Boolean operators as expected.

$$\pi, i \models p \iff p \in \pi[i]$$

$$\pi, i \models X\psi \iff \pi, i + 1 \models \psi$$

$$\pi, i \models F\psi \iff \exists j \geq i. \pi, j \models \psi$$

$$\pi, i \models G\psi \iff \forall i' \geq i. \pi, i' \models \psi$$

$$\pi, i \models \psi U \psi' \iff \exists j \geq i. \pi, j \models \psi' \land \forall i \leq i'' < j. \pi, i'' \models \psi$$
Replace some positive polarity occurrences of subformulas with 1 and some negative polarity occurrences of subformulas with 0 while preserving unsatisfiability ([Sch12b,KV03]).
UCs with Sets of Time Points

In model checking it is common to annotate counterexamples with additional information to help users understanding them (see references in [Bee+09]).

Counterexamples can be annotated with the time points at which its atomic propositions matter.

Almost no comparable work for UCs or vacuity (except first attempts [Sim+10] and ideas [Sch12b]).

In our example, the $p$ operand of the $G$ operator “matters” only at time point 1. Other subformulas also “matter” only at time points 0 or 1.

$$( G_{\{1\}} p ) \land ( X_{\{1\}} \neg_{\{1\}} p )$$

Intuition: replace occurrences of subformulas at specific time points with 1 or 0 depending on polarity (rather than always as before).
1. Introduction

2. LTL with Sets of Time Points

3. Extracting UCs in LTL with S.o.T.P. via Temporal Resolution

4. Implementation and Experimental Evaluation
LTL with Sets of Time Points (LTLp)

Annotate each subformula with a set of time points $\subseteq \mathbb{N}$.

Not a “new logic” but annotations incorporating the required information naturally with well-defined semantics.

Sets of time points of a subformula are attached to the operator of its immediate superformula.

The top level formula is evaluated (only) at time point 0. This is the standard semantics anyway.

Proper subformulas are evaluated at given time points. At other time points they are replaced with 1 or 0 depending on polarity.

Example operators:

$+$: $\langle \pi, i \rangle \models \tau \land \tau' \iff \left( \begin{array}{c} (i \notin I) \lor \left( (\pi, i) \models \tau \right) \right) \land \left( (i \notin I') \lor \left( (\pi, i) \models \tau' \right) \right)$

$-$: $\langle \pi, i \rangle \models G\tau \iff \forall i' \geq i . \left( (i' \in I) \land \left( (\pi, i') \models \tau \right) \right)$
LTLp — A More Complex Example

\[ p \land ((G(p \rightarrow XXp)) \land (F((\neg p) \land X\neg p))) \]

1st and 2nd conjunct: \( p \) must be 1 at even time points

3rd conj.: \( p \) must eventually be 0 two time points in a row

\[ p \land_{\{0\},\{0\}} ((\begin{array}{l}
G_{2\mathbb{N}} (p \rightarrow_{2\mathbb{N},2\mathbb{N}} X_{2\mathbb{N}+1} X_{2\mathbb{N}+2} p)) \land_{\{0\},\{0\}} \bigg((F_{\mathbb{N}} ((\neg_{2\mathbb{N}} p) \land_{2\mathbb{N},2\mathbb{N}+1} X_{2\mathbb{N}+2} X_{2\mathbb{N}+2} p))\bigg) \]
Choice of Solver

**TRP++** [HK03, HK04, trp++] by Boris Konev and Ullrich Hustadt.

Based on Temporal Resolution (TR) [Fis91, FDP01].

Uses BFS [Dix98, Dix97, Dix95] for loop search.

Performed competitive in experimental evaluation of LTL satisfiability solvers [SD11] (in particular also on unsatisfiable instances).

Access to and reasoning about proof is straightforward.

Extended with extraction of UCs without sets of time points “previously” [Sch12a].

Available as source code.
TR works on a clausal normal form called Separated Normal Form (SNF) [FDP01].

Let \( p_1, \ldots, p_n, q_1, \ldots, q_{n'}, l \) with \( 0 \leq n, n' \) be literals such that \( p_1, \ldots, p_n \) and \( q_1, \ldots, q_{n'} \) are pairwise different.

\((p_1 \lor \ldots \lor p_n)\) is an initial clause.

\((G((p_1 \lor \ldots \lor p_n) \lor (X(q_1 \lor \ldots \lor q_{n'}))))\) is a global clause.

\((G((p_1 \lor \ldots \lor p_n) \lor (F(l))))\) is an eventuality clause.

\(\) or \((G())\), denoted \(\Box\), stand for 0 or \(G(0)\) and are called empty clause.

Let \( c_1, \ldots, c_n \) with \( 0 \leq n \) be SNF clauses. Then \( \land_{1 \leq i \leq n} c_i \) is an LTL formula in SNF.

There exists a structure-preserving translation from an LTL formula into an equisatisfiable formula in SNF [FDP01].
A Taste of Temporal Resolution

One part: straightforward extension of propositional resolution. Examples:

\[\frac{(p_1 \lor \ldots \lor p_n \lor l) \quad (G(\neg l \lor q_1 \lor \ldots \lor q_{n'}))}{(p_1 \lor \ldots \lor p_n \lor q_1 \lor \ldots \lor q_{n'})} \text{ init-in}\]

\[\frac{(G(p_1 \lor \ldots \lor p_n \lor l)) \quad (G((q_1 \lor \ldots \lor q_{n'}) \lor (X(\neg l \lor r_1 \lor \ldots \lor r_{n''}))))}{(G((q_1 \lor \ldots \lor q_{n'}) \lor X(p_1 \lor \ldots \lor p_n \lor r_1 \lor \ldots \lor r_{n''}))} \text{ step-nx}\]

Note: time step of 1 between first premise and conclusion.

Other part: for resolving with eventuality clauses. Note: Fixed point check involves subsumption between already derived clauses.
Graph with clauses as vertices and edges from premises to conclusions.

UC w/o sets of time points obtained by taking input clauses backward reachable from empty clause.

Standard in propositional SAT.

Crucial differences to propositional SAT for this paper:

- Time shifting of premises by either 0 or 1 time steps.
- Loops from subsumption checks (makes computation non-straightforward).
Assigning Sets of Time Points 1

TR terminates with result unsatisfiable iff the empty clause is derived.

The empty clause comes in an initial and a universal flavor.

The empty initial clause must be assigned time point 0.

The empty universal clause could be assigned any time point; we pick 0.

Now propagate sets of time points from conclusions to premises, ...

... taking time steps into account.
Blue edges involved time steps of 0, red edges time steps of 1.

Sets of time points for input clauses are obtained by taking contributions from all (reverse) paths from the empty clause into account.

Note that loops prevent us from simply pushing information until a fixed point is reached.
Let $\Sigma$ be a finite alphabet, $\sigma \in \Sigma$ a letter in $\Sigma$, $L \subseteq \Sigma^*$ a language over $\Sigma$, and $w \in L$ a word in $L$.

Define a function from words and letters to naturals $\Psi : \Sigma^* \times \Sigma \rightarrow \mathbb{N}, (w, \sigma) \mapsto m$ where $m$ is the number of occurrences of $\sigma$ in $w$.

$\Psi$ is called Parikh mapping and $\Psi(w, \sigma)$ is called the Parikh image of $\sigma$ in $w$.

The Parikh image of a set of words $W$ is defined in the natural way: $\Psi(W, \sigma) = \{\Psi(w, \sigma) \mid w \in W\}$.

Parikh's theorem [Par66] states that for every context-free language $L$, for every letter $\sigma$, the Parikh image $\Psi(L, \sigma)$ is semilinear.
Computing Sets of Time Points for Input Clauses 1

For each input clause:

– Turn the resolution graph into an NFA over the alphabet \( \{0, 1\} \) as follows.
  
  – The set of states is given by the set of clauses of the resolution graph.
  
  – The single initial state is the empty clause.
  
  – The single final state is the input clause.
  
  – The set of transitions is given by the set of reversed edges of the resolution graph.
  
  – The transitions are labeled with 0 or 1 depending on their time steps.

  – Now the set of time points for the input clause is just the Parikh image of the letter 1 in the regular language given by the NFA.

For \(|C|\) input clauses and a resolution graph with \(|V'|\) vertices backward reachable from the empty clause the sets of time points can be computed in time \( \mathcal{O}(|V'|^3 + |V'|^2 \cdot |C|) \).
Example for input clause \( (\mathcal{G}(\neg a) \lor (Xa)) \).

Accepted language: 00(01)*00.

Parikh image of letter 1 in 00(01)*00: \( \mathbb{N} \).
Implementation

– basis: \( \text{TRP}^{++} \) extended with extraction of UCs \([\text{Sch12a}]\)
– make NFA \( \epsilon \)-free: \([\text{HU79}]\)
– compute Parikh images for unary NFA: optimized versions of
  – algorithm by Gawrychowski \([\text{Gaw11}]\)
  – algorithm by Sawa \([\text{Saw13}]\)

Experimental Setup

– Intel Core i7 M 620 @ 2 GHz
– Ubuntu 10.04
– time limit: 600 seconds
– memory limit: 6 GB
– time and memory measured and bounded with run [run]
## Benchmarks

<table>
<thead>
<tr>
<th>Family</th>
<th>Description</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category application</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alaska_lift</td>
<td>Elevator specifications</td>
<td>71 /</td>
<td>71 /</td>
<td>71</td>
<td>4605</td>
<td>[Har05, Wul+08]</td>
</tr>
<tr>
<td>anzu_genbuf</td>
<td>Generalized buffer</td>
<td>16 /</td>
<td>16 /</td>
<td>16</td>
<td>2676</td>
<td>[Blo+07]</td>
</tr>
<tr>
<td>forobots</td>
<td>Model of a robot with properties</td>
<td>25 /</td>
<td>25 /</td>
<td>25</td>
<td>635</td>
<td>[BDF09]</td>
</tr>
<tr>
<td><strong>Category crafted</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>schup_.O1form.</td>
<td>Exponential behavior in some solvers</td>
<td>21 /</td>
<td>21 /</td>
<td>21</td>
<td>1606</td>
<td>[SD11]</td>
</tr>
<tr>
<td>schup_.O2form.</td>
<td>Exponential behavior in some solvers</td>
<td>8 /</td>
<td>7 /</td>
<td>7</td>
<td>91</td>
<td>[SD11]</td>
</tr>
<tr>
<td>schuppan_phtl</td>
<td>Temporal variant of pigeonhole</td>
<td>4 /</td>
<td>4 /</td>
<td>4</td>
<td>125</td>
<td>[SD11]</td>
</tr>
<tr>
<td><strong>Category random</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rozier_formulas</td>
<td>Obtained by generating a syntax tree</td>
<td>66 /</td>
<td>66 /</td>
<td>66</td>
<td>157</td>
<td>[RV10]</td>
</tr>
<tr>
<td>trp</td>
<td>Obtained by lifting propositional CNF into fixed temporal structure</td>
<td>397 /</td>
<td>397 /</td>
<td>345</td>
<td>1421</td>
<td>[HS02]</td>
</tr>
</tbody>
</table>

a: # solved UC w/o s.o.t.p.  
b: # solved UC w/ s.o.t.p. (Gawrychowski’s alg.)  
c: # solved UC w/ s.o.t.p. (Sawa’s alg.)  
d: |largest solved|
## Occurrences of Sets of Time Points

<table>
<thead>
<tr>
<th>Family</th>
<th>Application</th>
<th>Crafted</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>lift</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>genbuf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>forobots</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O1formula</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2formula</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phltl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>formulas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trp</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overhead of UC Extraction with Sets of Time Points

UC Extraction with Sets of Time Points (Gawrychowski's algorithm)

run time [seconds]  memory [GB]

UC Extraction without Sets of Time Points

Author: V. Schuppan
Summary

Suggested more fine-grained notion of UCs for LTL.

Can yield interesting additional information.

Extraction of UCs with sets of time points incurs acceptable overhead.

Future Work

Use solvers based on SAT or BDDs.

Minimize sets of time points w.r.t. $\subseteq$.

Extend to unrealizable cores.

Instead of using Parikh images solve set of equations of the form $I = I' \cup \ldots \cup (I'' + 1) \cup \ldots$ where $I, I', \ldots, I'', \ldots \subseteq \mathbb{N}$. 

Author: V. Schuppan
Thanks

Thanks to

... you for your attention,

... B. Konev and M. Ludwig for making TRP++ and TSPASS available,

... A. Cimatti for bringing up the subject of temporal resolution and for pointing out that the resolution graph can be seen as a regular language acceptor.

Questions?

http://www.schuppan.de/viktor/qapll13/
References


Gaw11  P. Gawrychowski. Chrobak Normal Form Revisited, with Applications. CIAC’11.

HK03  U. Hustadt and B. Konev. TRP++ 2.0: A Temporal Resolution Prover. CADE’03.


Pil+06  I. Pill, S. Semprini, R. Cavada, M. Roveri, R. Bloem, and A. Cimatti. Formal Analysis of Hardware Requirements. DAC’06.

run  A. Biere and T. Jussila. Benchmark Tool Run. URL: http://fmv.jku.at/run/

RV10  K. Rozier and M. Vardi. LTL Satisfiability Checking. STTT, 12(2), 2010.


trp++  http://www.csc.liv.ac.uk/~konev/software/trp++./
