Enhancing Unsatisfiable Cores for LTL with Information on Temporal Relevance

Viktor Schuppan

QAPL 2013, Rome, Italy, March 23-24, 2013

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 ϕ:

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

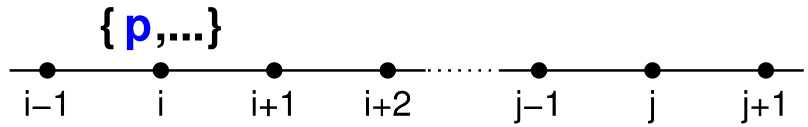
An unsatisfiable core (UC) is an unsatisfiable formula ϕ' that is derived from another unsatisfiable formula ϕ. ϕ' focuses on a reason for ϕ being unsatisfiable.

UCs can help understanding results of validation checks.

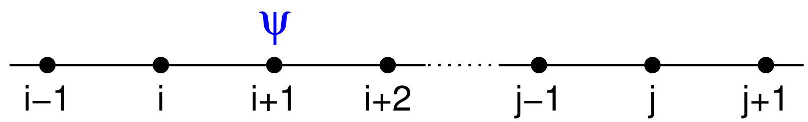
Linear Temporal Logic (LTL)

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.

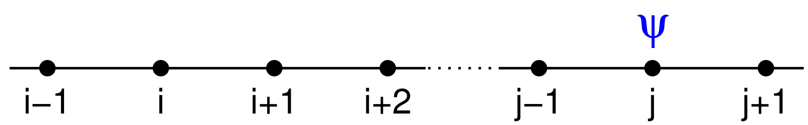
$$\pi, i \models p \Leftrightarrow p \in \pi[i]$$



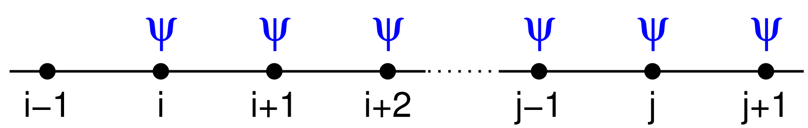
$$\pi, i \models \mathbf{X}\psi \Leftrightarrow \pi, i+1 \models \psi$$



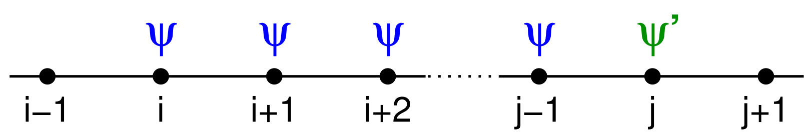
$$\pi, i \models \mathbf{F}\psi \Leftrightarrow \exists j \geq i \; . \; \pi, j \models \psi$$



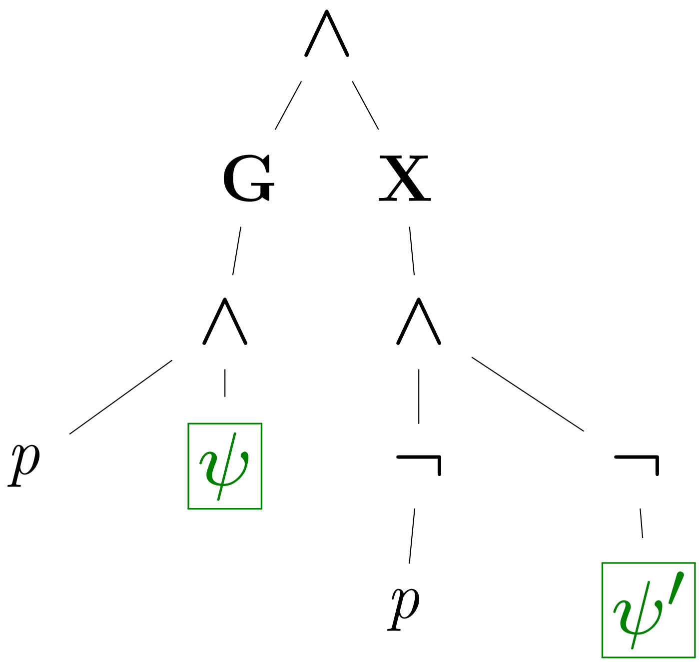
$$\pi, i \models \mathbf{G}\psi \Leftrightarrow \forall i' \geq i \; . \; \pi, i' \models \psi$$

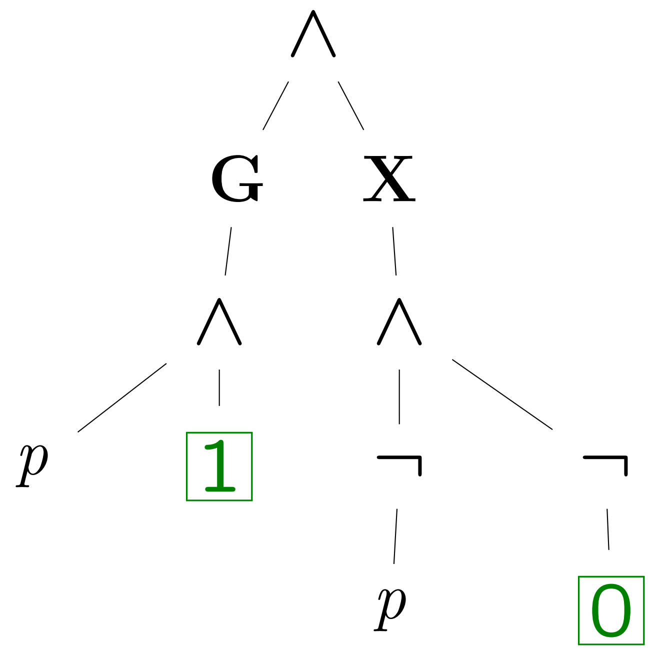


$$\begin{aligned} \pi, i \models \psi \mathbf{U} \psi' \Leftrightarrow\; & \exists j \geq i \; . \\ & \pi, j \models \psi' \qquad \wedge \\ & \forall i \leq i'' < j \; . \; \pi, i'' \models \psi \end{aligned}$$



UCs via Syntax Trees





$(\mathbf{G}(p \wedge \boxed{\psi})) \wedge (\mathbf{X}(\neg p \wedge \neg \boxed{\psi'}))$

$(\mathbf{G}(p \wedge \boxed{1})) \wedge (\mathbf{X}(\neg p \wedge \neg \boxed{0}))$

Replace some positive polarity occurrences of subformulas with 1 and some negative polarity occurrences of subformulas with 0 while preserving unsatisfiability ([Sch12b,KV03]).

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 $\mathbf{G}$ operator “matters” only at time point 1. Other subformulas also “matter” only at time points 0 or 1.

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

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

Contents

LTL with Sets of Time Points (LTL*p*)

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:

$$+\colon (\pi, i) \models \tau \underset{I,I'}{\wedge} \tau' \Leftrightarrow ((i \notin I) \vee ((\pi, i) \models \tau)) \wedge ((i \notin I') \vee ((\pi, i) \models \tau'))$$

$$-: (\pi, i) \models \underset{I}{\mathbf{G}} \tau \Leftrightarrow \forall i' \geq i \;.\; ((i' \in I) \wedge ((\pi, i') \models \tau))$$

$$p \wedge ((\mathbf{G}(p \rightarrow \mathbf{X}\mathbf{X}p)) \wedge (\mathbf{F}((\neg p) \wedge \mathbf{X}\neg p)))$$

1st and 2nd conjunct: p must be 1 at even time points	unsat!
3rd conj.: p must eventually be 0 two time points in a row	

$$p \underset{\{0\},\{0\}}{\wedge} ((\underset{2\cdot\mathbb{N}}{\mathbf{G}} (p \underset{2\cdot\mathbb{N},2\cdot\mathbb{N}}{\rightarrow} \underset{2\cdot\mathbb{N}+1}{\mathbf{X}} \underset{2\cdot\mathbb{N}+2}{\mathbf{X}} p)) \underset{\{0\},\{0\}}{\wedge}$$
$$(\underset{\mathbb{N}}{\mathbf{F}}((\underset{2\cdot\mathbb{N}}{\neg} p) \underset{2\cdot\mathbb{N},2\cdot\mathbb{N}+1}{\wedge} \underset{2\cdot\mathbb{N}+2}{\mathbf{X}} \underset{2\cdot\mathbb{N}+2}{\neg} p)))$$

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.

Separated Normal Form (SNF)

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.

$(p_1 \vee \ldots \vee p_n)$ is an initial clause.

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

$(\mathbf{G}((p_1 \vee \ldots \vee p_n) \vee (\mathbf{F}(l))))$ is an eventuality clause.

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

Let $c_1, \ldots, c_n$ with $0 \leq n$ be SNF clauses. Then $\bigwedge_{1 \leq i \leq n} c_i$ 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:

$$\frac{(p_1 \vee \ldots \vee p_n \vee l) \qquad (\mathbf{G}(\neg l \vee q_1 \vee \ldots \vee q_{n'}))}{(p_1 \vee \ldots \vee p_n \vee q_1 \vee \ldots \vee q_{n'})} \; \boxed{\text{init-in}}$$

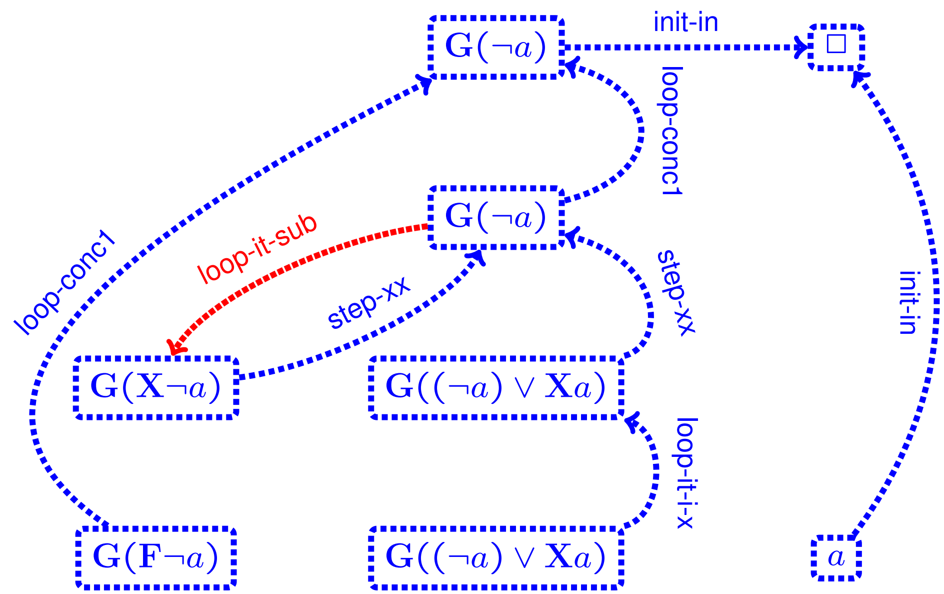
$$\frac{\begin{array}{c}(\mathbf{G}(p_1 \vee \ldots \vee p_n \vee l)) \\ (\mathbf{G}((q_1 \vee \ldots \vee q_{n'}) \vee (\mathbf{X}(\neg l \vee r_1 \vee \ldots \vee r_{n''}))))\end{array}}{(\mathbf{G}((q_1 \vee \ldots \vee q_{n'}) \vee \mathbf{X}(p_1 \vee \ldots \vee p_n \vee r_1 \vee \ldots \vee r_{n''})))} \; \boxed{\text{step-nx}}$$

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.

Resolution Graph, UC w/o Sets of Time Points [Sch12a]

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.

$$\{(a), (\mathbf{G}((\neg a) \vee (\mathbf{X}(a)))), (\mathbf{G}(\mathbf{F}(\neg a)))\}$$

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.

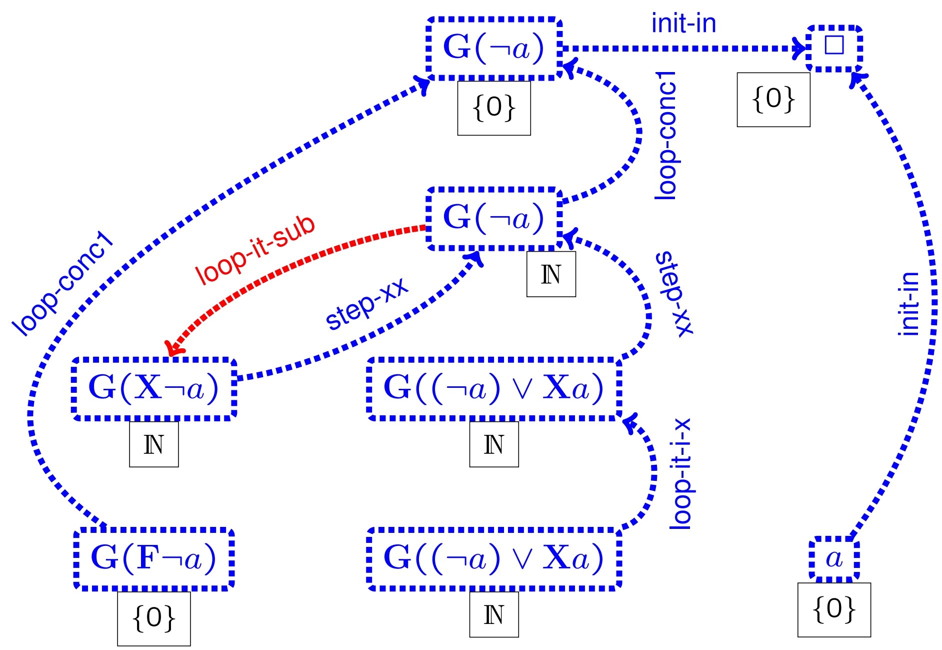
The empty initial clause must be assigned time point 0.

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.

Assigning Sets of Time Points 2



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.

Excursion: Parikh Images

Let Σ be a finite alphabet, $\sigma \in \Sigma$ a letter in Σ, $L \subseteq \Sigma^*$ a language over Σ, and $w \in L$ a word in L.

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

Ψ is called Parikh mapping and $\Psi(w, \sigma)$ is called the Parikh image of σ 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 σ, the Parikh image $\Psi(L, \sigma)$ is semilinear.

Computing Sets of Time Points for Input Clauses 1

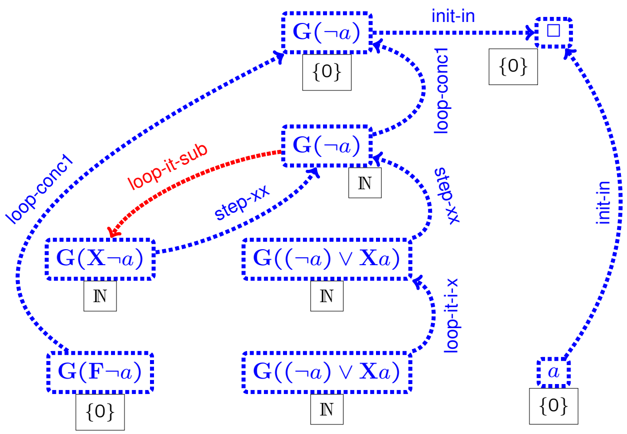
For each input clause:

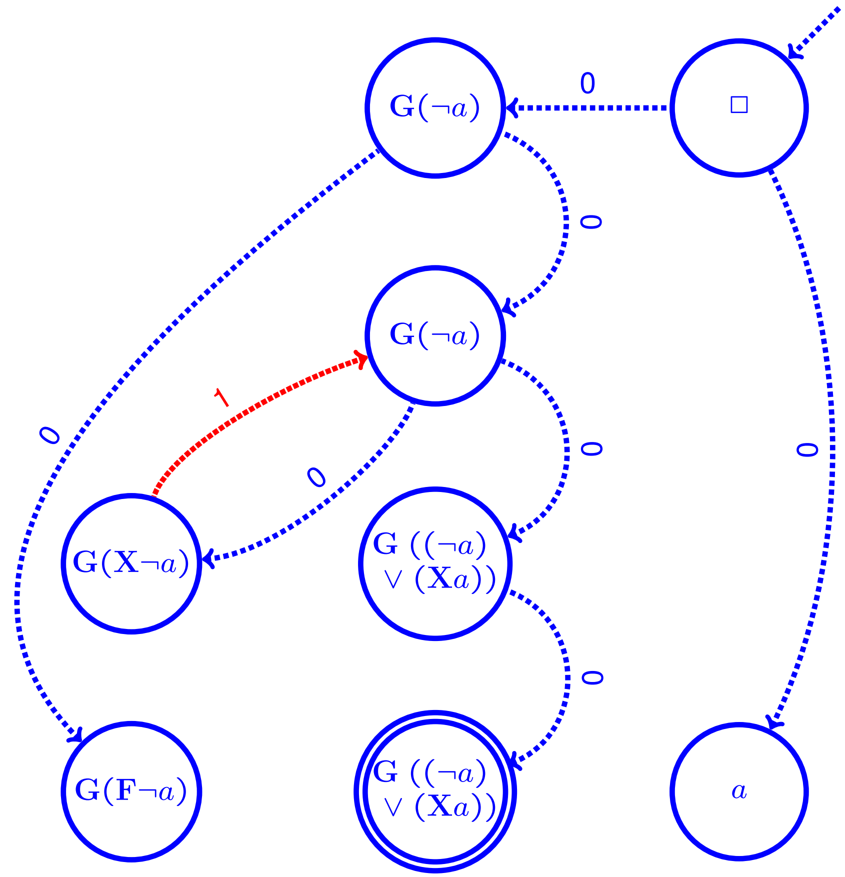
- Turn the resolution graph into an NFA over the alphabet $\{0, 1\}$ as follows.
  - The set of states is given by the set of clauses of the resolution graph.
  - The single initial state is the empty clause.
  - The single final state is the input clause.
  - The set of transitions is given by the set 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 $\mathcal{O}(|V'|^3 + |V'|^2 \cdot |C|)$.

Author: V. Schuppan

Computing Sets of Time Points for Input Clauses 2





Example for input clause $(\mathbf{G}((\neg a) \vee (\mathbf{X}(a))))$.

Accepted language: 00(01)*00.

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

Author: V. Schuppan

Implementation

- basis: `TRP++` extended with extraction of UCs [Sch12a]
- make NFA ϵ-free: [HU79]
- compute Parikh images for unary NFA: optimized versions of
  - algorithm by Gawrychowski [Gaw11]
  - algorithm by Sawa [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

Family	Description	a	b	c	d	Source
	Category application					
alaska_lift	Elevator specifications	71 /	71 /	71	4605	[Har05, Wul+08]
anzu_genbuf	Generalized buffer	16 /	16 /	16	2676	[Blo+07]
forobots	Model of a robot with properties	25 /	25 /	25	635	[BDF09]
	Category crafted					
schup._O1form.	Exponential behavior in some solvers	21 /	21 /	21	1606	[SD11]
schup._O2form.	Exponential behavior in some solvers	8 /	7 /	7	91	[SD11]
schuppan_phltl	Temporal variant of pigeonhole	4 /	4 /	4	125	[SD11]
	Category random					
rozier_formulas	Obtained by generating a syntax tree	66 /	66 /	66	157	[RV10]
trp	Obtained by lifting propositional CNF into fixed temporal structure	397 /	397 /	345	1421	[HS02]

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

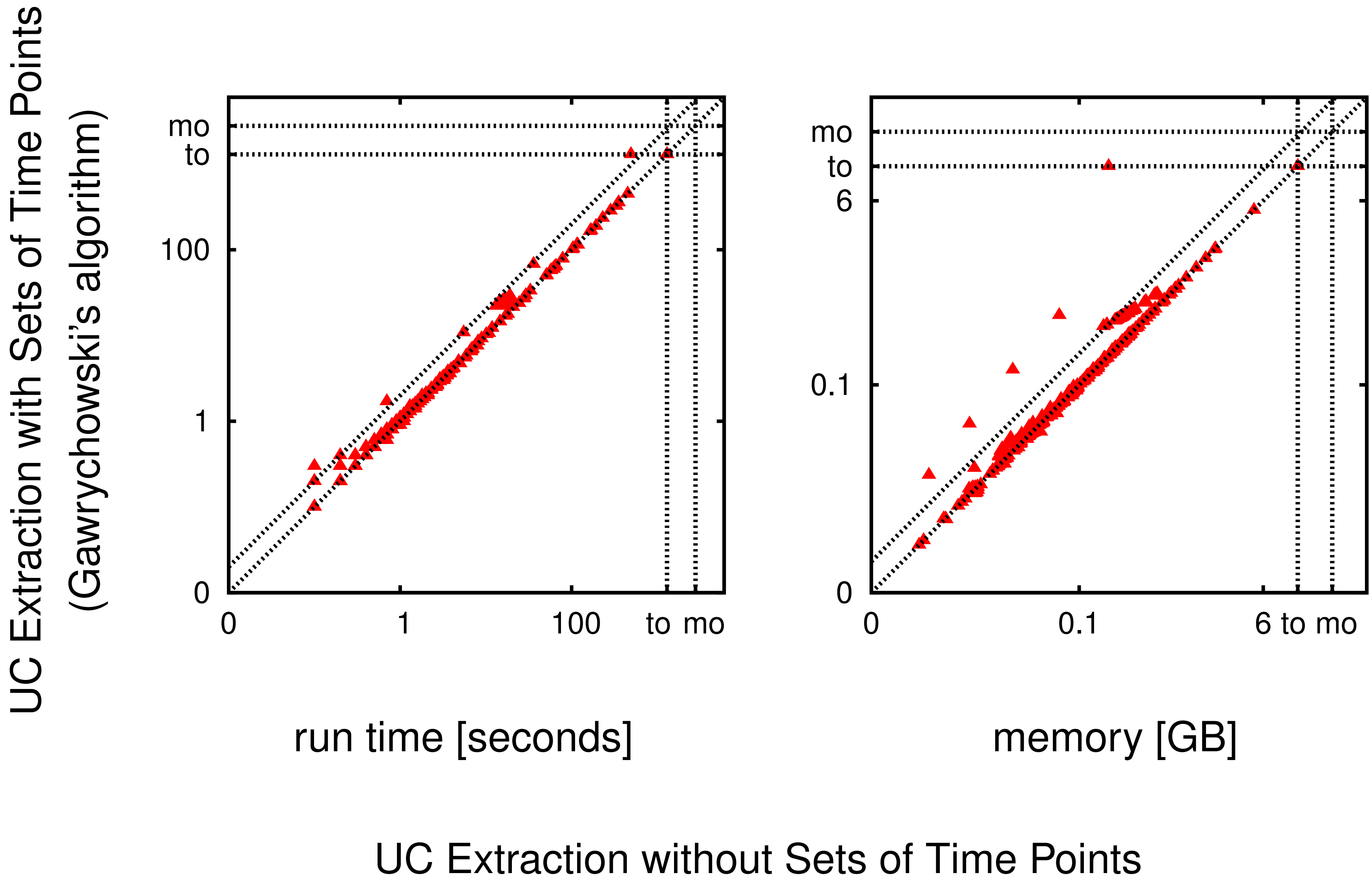
family	$\{0\}$	$\{0,1\}$	$\{0,2\}$	$\{1\}$	$\{1,2\}$	$\{1,2,3\}$	$\{1,3\}$	$\{2\}$	$\{2,3\}$	$\{3\}$
application										
lift	■	■		■	■			■		
genbuf	■			■						
forobots	■	■		■						
crafted										
O1formula	■			■						
O2formula	■									
phltl	■									
random										
formulas	■	■	■	■	■	■	■	■	■	■
trp	■			■						

family	$\mathbb{N}$	$\mathbb{N}+1$	$\mathbb{N}+2$	$\mathbb{N}+3$	$\mathbb{N}+4$	$\mathbb{N}+5$	$\mathbb{N}+6$	$\mathbb{N}+7$	$\mathbb{N}+8$	$\mathbb{N}+9$	$\mathbb{N}+10$	$\{0,\mathbb{N}+2\}$
application												
lift	■	■										
genbuf	■	■	■	■	■	■	■					
forobots	■	■	■	■								
crafted												
O1formula												
O2formula	■	■										
phltl	■	■	■	■	■	■	■	■	■	■	■	
random												
formulas	■	■	■	■	■	■						■
trp	■	■	■	■								

family	$4\cdot\mathbb{N}$	$4\cdot\mathbb{N}+1$	$\{4\cdot\mathbb{N}+1, 4\cdot\mathbb{N}+2\}$	$\{4\cdot\mathbb{N}+1, 4\cdot\mathbb{N}+2, 4\cdot\mathbb{N}+3\}$	$4\cdot\mathbb{N}+2$	$\{4\cdot\mathbb{N}+2, 4\cdot\mathbb{N}+3\}$	$\{4\cdot\mathbb{N}+2, 4\cdot\mathbb{N}+3, 4\cdot\mathbb{N}+4\}$	$4\cdot\mathbb{N}+3$	$\{4\cdot\mathbb{N}+3, 4\cdot\mathbb{N}+4\}$	$4\cdot\mathbb{N}+4$	$4\cdot\mathbb{N}+5$	$\{5\cdot\mathbb{N}+0\},\ldots,\{5\cdot\mathbb{N}+5\}$	$\{12\cdot\mathbb{N}+0\},\ldots,\{12\cdot\mathbb{N}+12\}$
application													
lift													
genbuf	■	■	■	■	■	■	■	■	■	■	■		
forobots													
crafted													
O1formula													
O2formula													
phltl													
random													
formulas													
trp												■	■

Author: V. Schuppan

Overhead of UC Extraction with Sets of Time Points



The End

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 $I, I', \ldots, I'', \ldots \subseteq \mathbb{N}$.

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/qapl13/`

References

Awa+12 A. Awad, R. Goré, Z. Hou, J. Thomson, and M. Weidlich. An Iterative Approach to Synthesize Business Process Templates from Compliance Rules. Inf. Syst. 37.8, 2012.

BDF09 A. Behdenna, C. Dixon, and M. Fisher. Deductive Verification of Simple Foraging Robotic Behaviours. International Journal of Intelligent Computing and Cybernetics, 2009.

Bee+09 I. Beer, S. Ben-David, H. Chockler, A. Orni, and R. Trefler. Explaining Counterexamples Using Causality. CAV'09.

Blo+07 R. Bloem, S. Galler, B. Jobstmann, N. Piterman, A. Pnueli, and M. Weiglhofer. Specify, Compile, Run: Hardware from PSL. COCV'07.

Dix95 C. Dixon. Strategies for Temporal Resolution. PhD thesis. Department of Computer Science, University of Manchester, 1995.

Dix97 C. Dixon. Using Otter for Temporal Resolution. ICTL'97.

Dix98 C. Dixon. Temporal Resolution Using a Breadth-First Search Algorithm. Ann. Math. Artif. Intell. 22.1-2, 1998.

EF06 C. Eisner and D. Fisman. A Practical Introduction to PSL. Springer, 2006.

FDP01 M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. ACM Trans. Comput. Log. 2.1, 2001.

Fis91 M. Fisher. A Resolution Method for Temporal Logic. IJCAI'91.

Gaw11 P. Gawrychowski. Chrobak Normal Form Revisited, with Applications. CIAA'11.

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

HK04 U. Hustadt and B. Konev. TRP++: A Temporal Resolution Prover. Collegium Logicum, Vol. 8, 2004.

Har05 A. Harding. Symbolic Strategy Synthesis For Games With LTL Winning Conditions. PhD thesis. University of Birmingham, 2005.

HS02 U. Hustadt and R. A. Schmidt. Scientific Benchmarking with Temporal Logic Decision Procedures. KR'02.

HU79 J. Hopcroft and J. Ullman. Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 1979.

KV03 O. Kupferman and M. Vardi. Vacuity Detection in Temporal Model Checking. STTT 4.2, 2003.

PA06 M. Pesic and W. van der Aalst. A Declarative Approach for Flexible Business Processes Management. Business Process Management Workshops. 2006.

Par66 R. Parikh. On Context-Free Languages. J. ACM 13.4, 1966.

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.

Saw13 Z. Sawa. Efficient Construction of Semilinear Representations of Languages Accepted by Unary Nondeterministic Finite Automata. To appear in: Fundam. Inform., 2013.

Sch12a V. Schuppan. Extracting Unsatisfiable Cores for LTL via Temporal Resolution. Available at arXiv:1212.3884v1 [cs.LO]. 2012.

Sch12b V. Schuppan. Towards a Notion of Unsatisfiable and Unrealizable Cores for LTL. Sci. Comput. Program. 77.7-8, 2012.

SD11 V. Schuppan and L. Darmawan. Evaluating LTL Satisfiability Solvers. ATVA'11.

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

Sim+10 J. Simmonds, J. Davies, A. Gurfinkel, and M. Chechik. Exploiting Resolution Proofs to Speed Up LTL Vacuity Detection for BMC. STTT 12.5, 2010.

Wul+08 M. De Wulf, L. Doyen, N. Maquet, and J.-F. Raskin. Antichains: Alternative Algorithms for LTL Satisfibility and Model-Checking. TACAS'08.