A New Rule for LTL Tableaux

Mark Reynolds

School of Computer Science and Software Engineering
The University of Western Australia

GandALF 2016, Catania, Italy

GandALF 2016
University of Catania
Thursday 15th September 2016
15:00 until 15:30

Abstract

Propositional linear time temporal logic (LTL) is the standard temporal logic for computing applications and many reasoning techniques and tools have been developed for it. Tableaux for deciding satisfiability have existed since the 1980s. However, the tableaux for this logic do not look like traditional tree-shaped tableau systems and their processing is often quite complicated.

In this paper, we introduce a novel style of tableau rule which supports a new simple traditional-style tree-shaped tableau for LTL. We prove that it is sound and complete. As well as being simple to understand, to introduce to students and to use, it is also simple to implement and is competitive against state of the art systems. It is particularly suitable for parallel implementations.

Outline

Introduction

LTL the most important temporal logic for CS.

Satisfiability (or validity) checking is a basic automated reasoning task for LTL: eg giving a "sanity check" on specifications.

Receiving renewed interest with recent competitive state of the art developments in automata, resolution, tableau and mixed techniques.

We think the new rules here have a part to play in further improving the performance of tableau approaches.

The new system is set out in tech reports, and there is a recent IJCAI paper on a fast and competitive implementation. This conference paper is the first published account of the proof of correctness (soundness, completeness and termination) of the system.

Structures

We assume a countable set AP of propositional atoms, or atomic propositions.

A (transition) structure is a triple (S, R, g) with S a finite set of states, $R \subseteq S \times S$ a binary relation called the transition relation and labelling g tells us which atoms are true at each state: for each $s \in S$, $g(s) \subseteq AP$. Furthermore, R is assumed to be serial: every state has at least one successor $\forall x \in S. \exists y \in S$ s.t.$(x, y) \in R$.
Given a structure (S, R, g), an ω-long sequence of states $\langle s_0, s_1, s_2, ... \rangle$ from S is a *fullpath* (through (S, R, g)) iff for each i, $(s_i, s_{i+1}) \in R$. If $\sigma = \langle s_0, s_1, s_2, ... \rangle$ is a fullpath then we write $\sigma_i = s_i$, $\sigma_{\geq j} = \langle s_j, s_{j+1}, s_{j+2}, ... \rangle$ (also a fullpath).

Syntax

The (well formed) formulas of LTL include the atoms and if α and β are formulas then so are $\neg\alpha$, $\alpha \wedge \beta$, $X\alpha$, and $\alpha U \beta$. We will also include some formulas built using other connectives that are often presented as abbreviations instead. However, before detailing them we present the semantic clauses.

Semantics

Semantics defines truth of formulas on a fullpath through a structure. Write $M, \sigma \models \alpha$ iff the formula α is true of the fullpath σ in the structure $M = (S, R, g)$ defined recursively by:

$M, \sigma \models p$	iff	$p \in g(\sigma_0)$, for $p \in AP$;
$M, \sigma \models \neg\alpha$	iff	$M, \sigma \not\models \alpha$;
$M, \sigma \models \alpha \wedge \beta$	iff	$M, \sigma \models \alpha$ and $M, \sigma \models \beta$;
$M, \sigma \models X\alpha$	iff	$M, \sigma_{\geq 1} \models \alpha$; and
$M, \sigma \models \alpha U \beta$	iff	there is some $i \geq 0$ s.t. $M, \sigma_{\geq i} \models \beta$ and for each j, if $0 \leq j < i$ then $M, \sigma_{\geq j} \models \alpha$.

Standard abbreviations in LTL include the classical $\top \equiv p \vee \neg p$, $\bot \equiv \neg\top$, $\alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta)$, $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$, $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$. We also have the temporal: $F\alpha \equiv (\top U \alpha)$, $G\alpha \equiv \neg F(\neg\alpha)$ read as eventually and always respectively.

A formula α is *satisfiable* iff there is some structure (S, R, g) with some fullpath σ through it such that $(S, R, g), \sigma \models \alpha$. A formula is *valid* iff for all structures (S, R, g) for all fullpaths σ through (S, R, g) we have $(S, R, g), \sigma \models \alpha$. A formula is valid iff its negation is not satisfiable. For example, $\top$, p, Fp, $p \wedge Xp \wedge F\neg p$, Gp are each satisfiable. However, $\bot$, $p \wedge \neg p$, $Fp \wedge G\neg p$, $p \wedge G(p \rightarrow Xp) \wedge F\neg p$ are each not satisfiable.

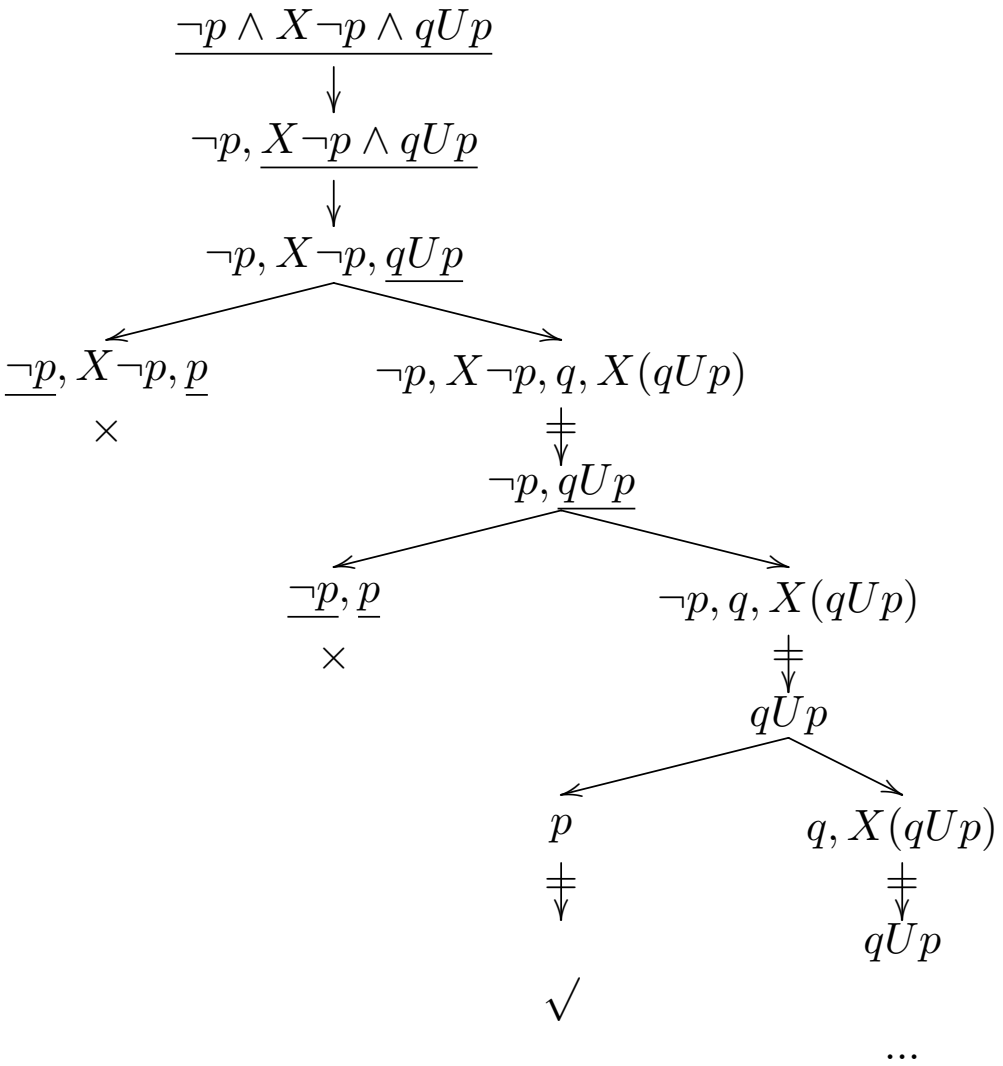
We will fix a particular formula, ϕ say, and describe how a tableau for ϕ is built and how that decides the satisfiability or otherwise, of ϕ. We will use other formula names such as α, β, e.t.c., to indicate arbitrary formulas which are used in labels in the tableau for ϕ.

Axioms, Decidability, Complexity

Decidability, PSPACE-complexity of LTL by [SC85] (via a small model theorem).

Axioms in [GPSS80].

A Tableau



A tableau for $\neg p \wedge X\neg p \wedge (qUp)$

Rules to grow a tableau

The four (static) termination rules:

EMPTY-rule: $\{\} \;/\; \surd$;

$\bot$-rule: $\{\bot\} \uplus \Delta \;/\; \times$;

CONTRADICTION-rule: $\{\alpha, \neg\alpha\} \uplus \Delta \;/\; \times$;

$\neg\top$-rule: $\{\neg\top\} \uplus \Delta \;/\; \times$.

These are the positive static rules:

$\top$-rule: $\{\top\} \uplus \Delta \;/\; \Delta$.

$\wedge$-rule: $\{\alpha \wedge \beta\} \uplus \Delta \;/\; (\Delta \cup \{\alpha, \beta\})$.

U-rule: $\{\alpha U \beta\} \uplus \Delta \;/\; (\Delta \cup \{\beta\} \mid \Delta \cup \{\alpha, X(\alpha U \beta)\})$.

There are also static rules for negations:

$\neg\neg$-rule: $\{\neg\neg\alpha\} \uplus \Delta \;/\; \Delta \cup \{\alpha\}$.

$\neg\wedge$-rule: $\{\neg(\alpha \wedge \beta)\} \uplus \Delta \;/\; (\Delta \cup \{\neg\alpha\} \mid \Delta \cup \{\neg\beta\})$.

$\neg U$-rule: $\{\neg(\alpha U \beta)\} \uplus \Delta \;/\; (\Delta \cup \{\neg\alpha, \neg\beta\} \mid \Delta \cup \{\neg\beta, X\neg(\alpha U \beta)\})$.

Optional/derived rules for some abbreviations

F-rule: $\{F\alpha\} \uplus \Delta \;/\; (\Delta \cup \{\alpha\} \mid \Delta \cup \{XF\alpha\})$.

G-rule: $\{G\alpha\} \uplus \Delta \;/\; \Delta \cup \{\alpha, XG\alpha\}$.

$\neg G$-rule: $\{\neg G\alpha\} \uplus \Delta \;/\; (\Delta \cup \{\neg\alpha\} \mid \Delta \cup \{X\neg G\alpha\})$.

$\neg F$-rule: $\{\neg F\alpha\} \uplus \Delta \;/\; \Delta \cup \{\neg\alpha, X\neg F\alpha\}$.

Loop and Transition (non-static) Rules

Only apply any of the four non-static rules if no static rules apply.

[LOOP]: If a node v with poised label Γ_v has a proper ancestor (i.e., not itself) u with poised label Γ_u such that $\Gamma_u \supseteq \Gamma_v$, and for each X-eventuality $X(\alpha U \beta)$ or $XF\beta$ in Γ_u we have a node w such that $u < w \leq v$ and $\beta \in \Gamma_w$ then v should be a ticked leaf.

[TRANSITION]: If none of the other rules apply to it then a node labelled by poised Γ say, can have one child whose label is: $\Delta = \{\alpha | X\alpha \in \Gamma\} \cup \{\neg\alpha | \neg X\alpha \in \Gamma\}$.

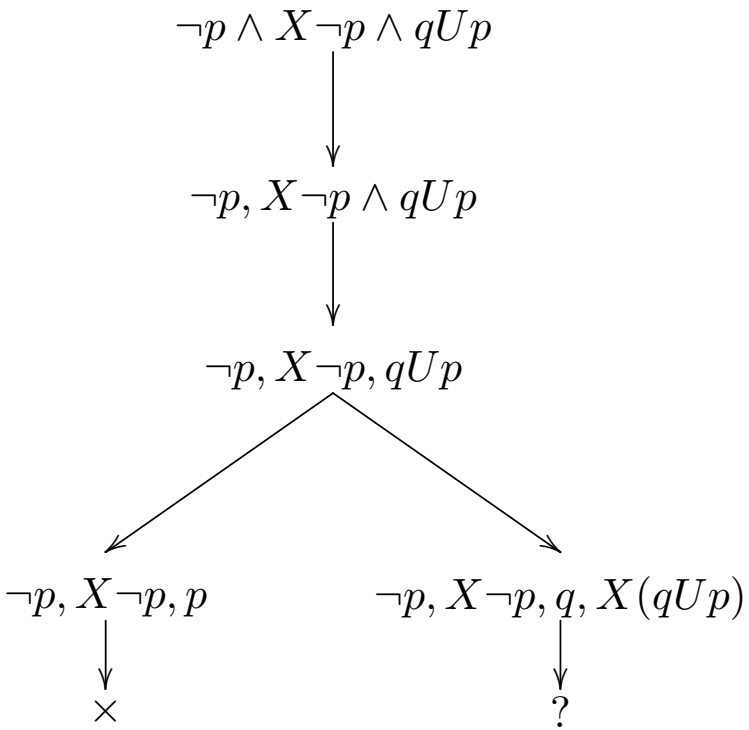
The new Prune Rule

[PRUNE]: Suppose that $u < v < w$ and each of u, v and w have the same poised label Γ. Suppose also that for each X-eventuality $X(\alpha U \beta)$ or $XF\beta$ in Γ, if there is x with $\beta \in \Gamma_x$ and $v < x \leq w$ then there is y such that $\beta \in \Gamma_y$ and $u < y \leq v$. Then w should be a crossed leaf.

(Another optional derived shortcut rule)
[PRUNE$_0$]: Suppose that $u < v$ share the same poised label Γ and Γ contains at least one X-eventuality. Suppose that there is no X-eventuality $X(\alpha U \beta)$ or $XF\beta$ in Γ with a node x such that $\beta \in \Gamma_x$ and $u < x \leq v$. Then v should be a crossed leaf.

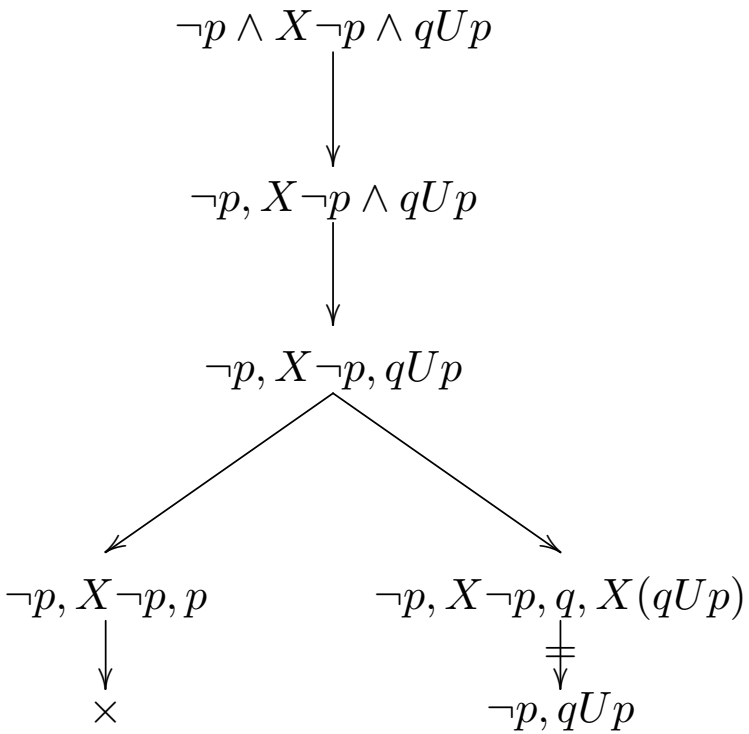
Example in detail

Start with $\neg p \wedge X\neg p \wedge qUp$ labelling the root.
Eventually, break down a formula into elementary ones using the static rules.



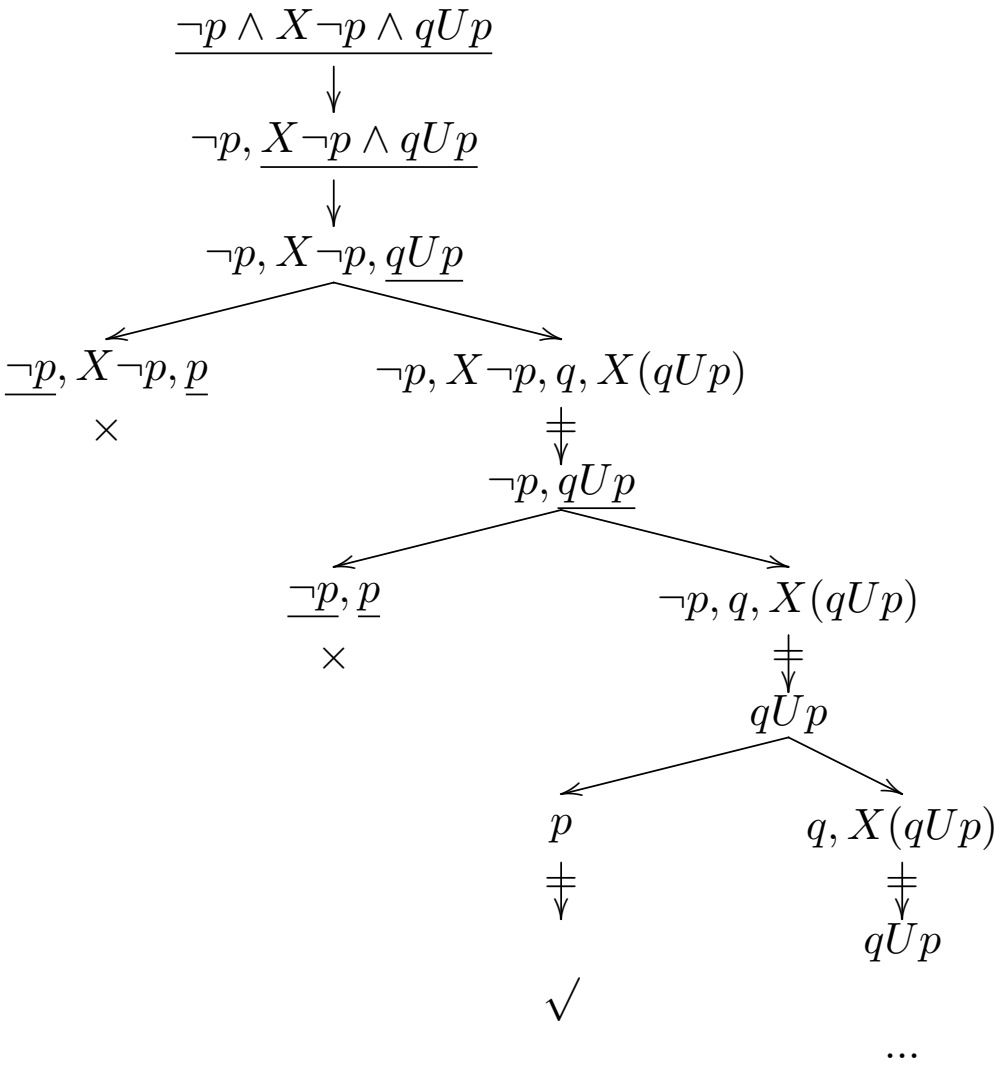
But what to do when we want to move forwards in time?

Using the Transition rule in the example ...



Reasoning switches to the next time point and we carry over only information nested below X and $\neg X$.

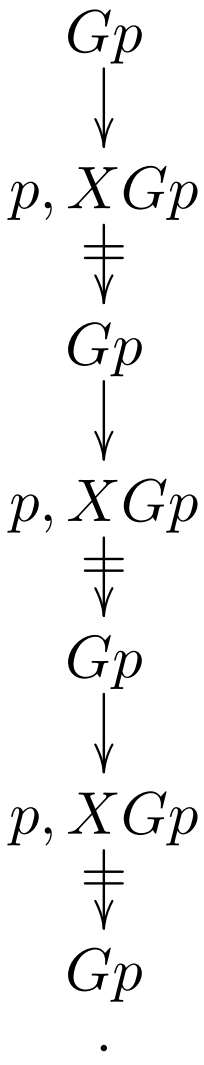
Finished Example



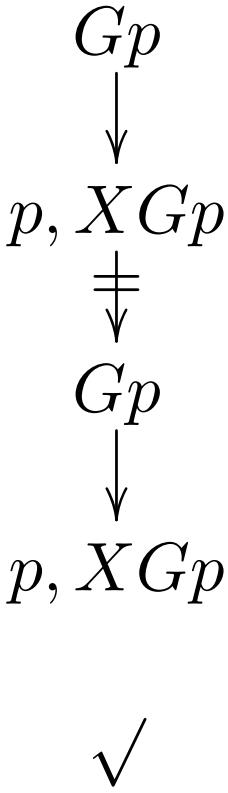
Another example: this one needing LOOP

... to deal with infinite behaviour.

Consider the example *Gp* which, in the absence of LOOP, gives rise to the very repetitive infinite tableau



... but succeeds quickly with LOOP



$\surd$

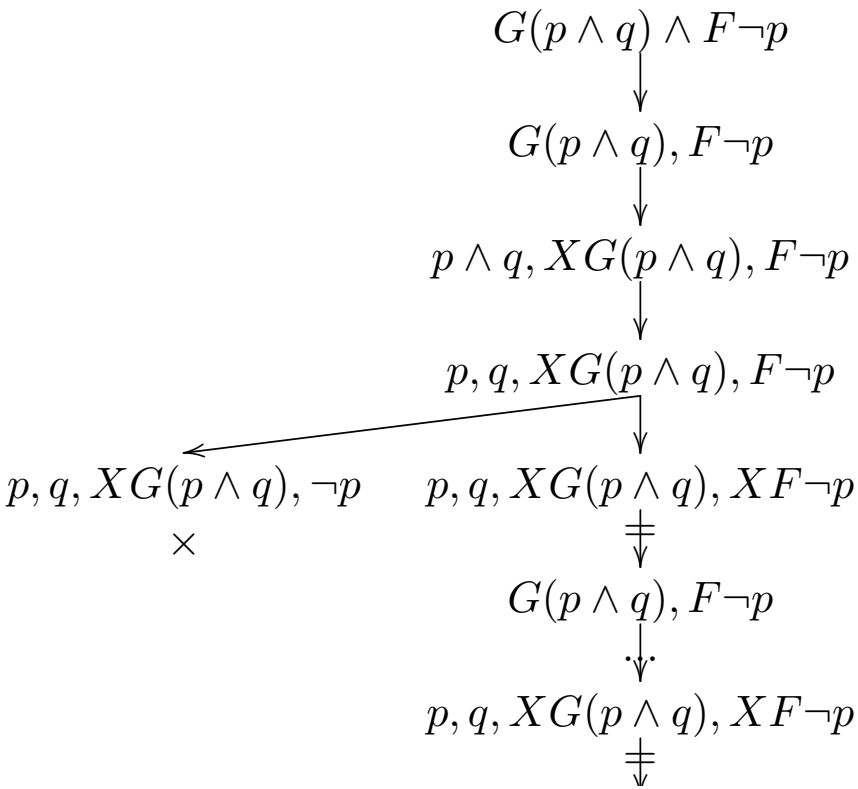
LOOP needs the Eventualities to be fulfilled

Notice that the infinite fullpath that it suggests is a model for Gp as would be a fullpath just consisting of the one state with a self-loop (a transition from itself to itself).

This suggests that we should allow the tableau branch construction to halt if a state is repeated.

However the following example $G(p \wedge q) \wedge F\neg p$ shows that we can not just accept any repetition as demonstrating satisfiability: the tableau for this unsatisfiable formula does have repeated labels ...

Repeats without LOOP



$G(p \wedge q) \wedge F\neg p$ does repeat itself but not all eventualities are fulfilled so the LOOP rule does not apply.

Thus the LOOP rule is useful

Eventualities are eventually satisfied in any (actual) model of a formula: by the semantics of U (or F or $\neg G$).

This motivates the LOOP rule. If a label is repeated along a branch and all eventualities are satisfied in between then we can build a model by looping states. In fact, the ancestor can have a superset and it will work (see the soundness proof in [Rey16]).

But what to do about infinite repeats?

Examples like $G(p \wedge q) \wedge F\neg p$ above and $p \wedge G(p \rightarrow Xp) \wedge F\neg p$ which have branches that go on forever without satisfying eventualities, still present a problem for us.

We need to stop and fail branches so that we can answer “no” correctly and terminate and so that we do not get distracted when another branch may be successful. In fact, no infinite branches should be allowed.

Repetetiveness

The final rule that we consider, and the most novel, is based on observing that these infinite branches are just getting repetitive without making a model. The repetition is clear because there are only a finite set of formulas which can ever appear in labels for a given initial formula ϕ. The *closure set* for a formula ϕ is as follows:

$$\{\psi, \neg\psi | \psi \leq \phi\} \cup \{X(\alpha U \beta), \neg X(\alpha U \beta) | \alpha U \beta \leq \phi\}$$

Here we use $\psi \leq \phi$ to mean that ψ is a subformula of ϕ. The size of closure set is $\leq 4n$ where n is the length of the initial formula. Only formulas from this finite set will appear in labels. So there are only $\leq 2^{4n}$ possible labels.

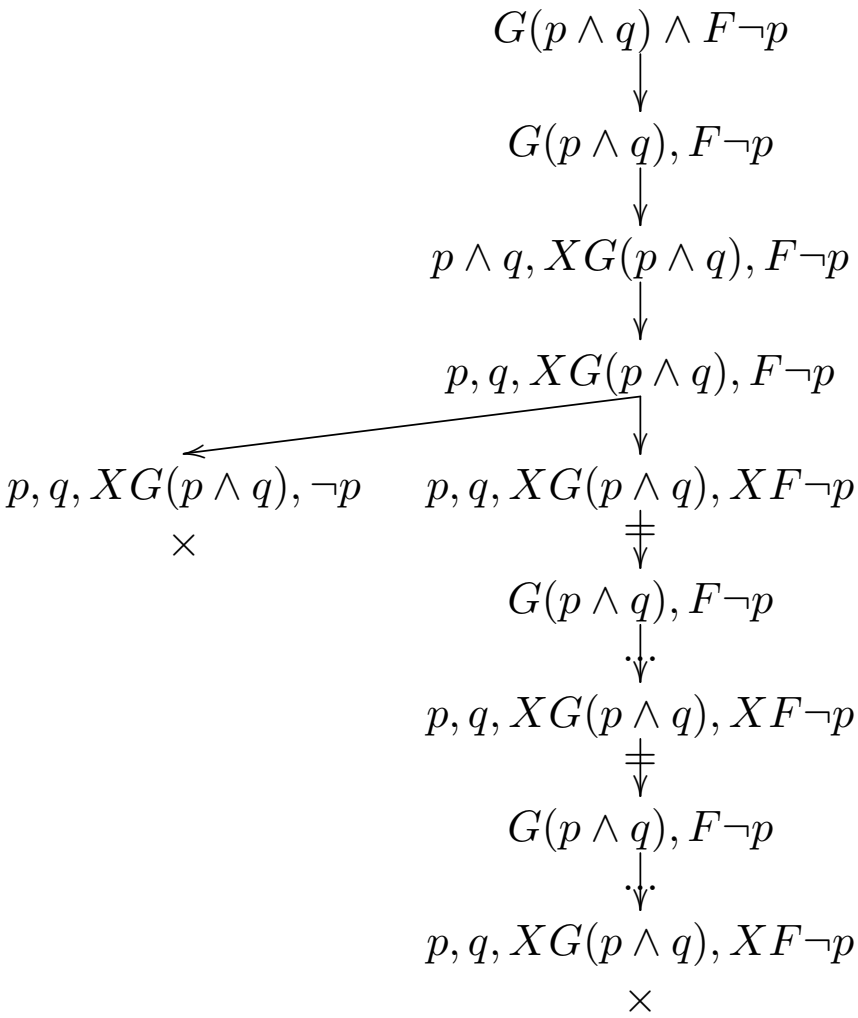
Reminder of the PRUNE rule

The PRUNE rule is as follows. If a node at the end of a branch (of an unfinished tableau) has a label which has appeared already twice above, and between the second and third appearance there are no new eventualities satisfied that were not already satisfied between the first and second appearances then that whole interval of states (second to third appearance) has been useless.

The PRUNE_0 rule applies similar reasoning to an initial repeat in which no eventualities are fulfilled.

In the example again, the PRUNE rule crosses the right hand branch as the only X-eventuality $XF\neg p$ remains unfulfilled as $\neg p$ does not appear in a label despite three repeats of the same label.

Crossed by PRUNE



$G(p \wedge q) \wedge F\neg p$ crossed by PRUNE

Choices of Rules to apply

It should be mentioned that the tableau building process we describe above is non-deterministic in several respects and so really not a proper description of an algorithm. However, we will see in the correctness proof below that the further details of which formula to consider at each step in building the tableau are unimportant.

If we have time ...

Finally a suggestion for a nice example to try.

Try

$$a \wedge G(a \leftrightarrow X\neg a) \wedge GFb_1 \wedge GFb_2 \wedge G(b_1 \rightarrow \neg a) \wedge G(b_2 \rightarrow \neg a) \wedge G\neg(b_1 \wedge b_2).$$

Termination and Soundness

In the long technical report [Rey16], we show full details of the proof of soundness, completeness and termination for the tableau search.

Termination is guaranteed because there can be no infinitely long branches.

Soundness presents no novelty to those familiar with soundness proofs for similar tableaux: construct a model from the successful tableau branch. (Prune rules do not appear in the soundness proof)

Completeness

Completeness proof is novel.

(1) Uses a model to find a branch.
(2) But may need to backtrack up inside the tableau while continuing in the model.
(3) no need to rely on any “fair" expansion strategy amongst eventualities.
(4) $\not\exists$ proof based on the shortest lasso model of a formula

Lemma

Note also that the completeness proof gives us the rather strong result that for a satisfiable formula we will find a successful tableau regardless of the order in which we use the rules and regardless of the order in choosing unfinished branches to extend.

Lemma (Completeness)

Suppose that ϕ is a satisfiable formula of LTL. Then any finished tableau for ϕ will be successful.

Implementation 1

Fast implementation by Matteo Battelo of Udine University is available from `https://github.com/Corralx/leviathan` and described in [BGMR16]. Experiments run using this implementation, on the full set of 3723 standard benchmark formulas [SD11], show comparative speed performance with five state of the art tools (Aalta [LYP+14], TRP++ [HK03], LS4 [SW12], NuSMV [CCG+02], and PLTL) based on automata, resolution, resolution(again), symbolic model checking, the Schwendimann tableau technique respectively. Interestingly the memory usage for the new tableau is significantly less.

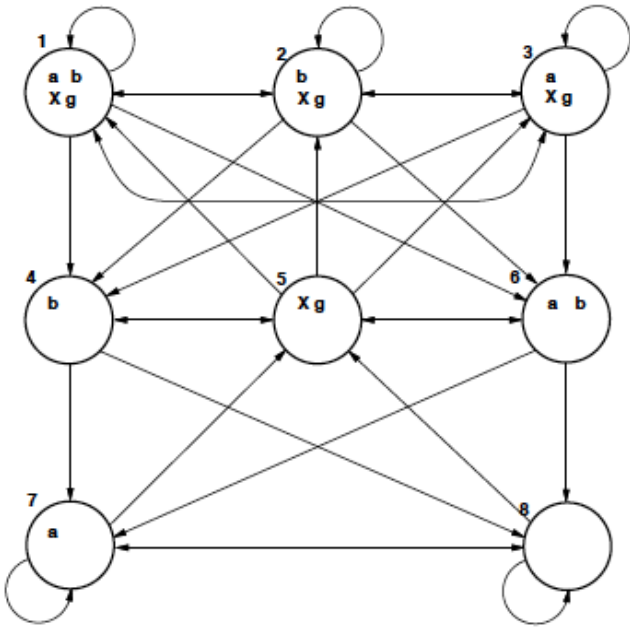
Implementation 2

To support this GandALF paper, a demonstration Java implementation to allow readers to experiment with the way that the tableau works. The program allows comparison with a corresponding implementation of the Schwendimann tableau. The demonstration Java implementation is available at `http://staffhome.ecm.uwa.edu.au/~00054620/research/Online/ltlsattab.html`. This allows users to understand the tableau building process in a step by step way. It is not designed as a fast implementation. However, it does report on how many tableau construction steps were taken.

New tableau needs roughly the same number of steps but saves time on each step (at least as the formulas get longer).

Comparisons

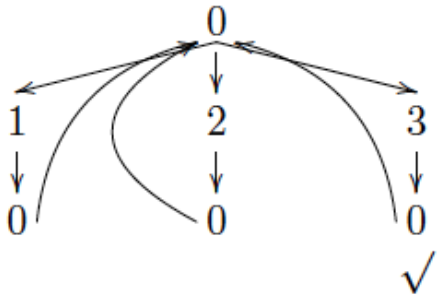
LTL tableaux are not new.
Many are based on a graph shape: not a traditional or usual shape amongst tableaux in general; but this is the traditional shape for LTL tableaux.
This one from [CGH97]



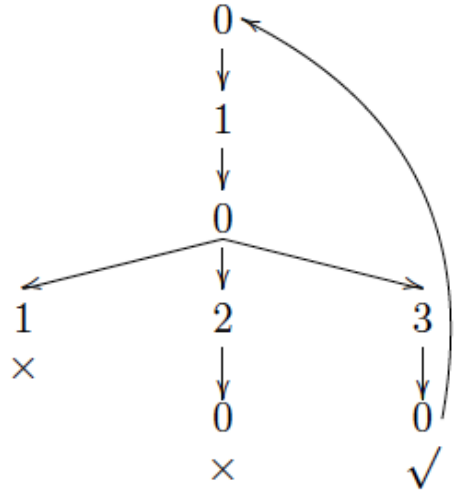
A few are tree shaped such as the the tree-style tableau from Schwendimann [Sch98]. (current state of the art in LTL tableaux).

Comparison Example

$$a \wedge G(a \leftrightarrow X\neg a) \wedge GFb_1 \wedge GFb_2 \wedge G(b_1 \rightarrow \neg a) \wedge G(b_2 \rightarrow \neg a) \wedge G\neg(b_1 \wedge b_2).$$



Schwendimann Example



Same Example with new tableau

Conclusion and Future Work

A novel but traditionally tree-shaped, one-pass tableau system for LTLSAT: no extra notations on nodes; neat to introduce to students; amenable to manual use; and can be implemented efficiently with competitive performance.

Future Work:
(1) one can explore down branches completely independently and further break up the search down individual branches into separate somewhat independent processes. Thus it is particularly suited to parallel implementations.
(2) Because of the simplicity, it also seems to be a good base for more intelligent and sophisticated algorithms: including heuristics for choosing amongst branches and ways of managing sequences of label sets.
(3) The idea of the PRUNE rules potentially have many other applications.

Thank you for listening

Questions? Comments.


Matteo Bertello, Nicola Gigante, Angelo Montanari, and Mark Reynolds.
Leviathan: A new LTL satisfiability checking tool based on a one-pass tree-shaped tableau.
In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 950–956. IJCAI/AAAI Press, 2016.


Kai Brünnler and Martin Lange.
Cut-free sequent systems for temporal logic.
J. Log. Algebr. Program., 76(2):216–225, 2008.


Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella.
Nusmv 2: An opensource tool for symbolic model checking.
In Ed Brinksma and Kim Guldstrand Larsen, editors, *Computer Aided Verification, 14th International Conference, CAV*

2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer, 2002.


Edmund M. Clarke, Orna Grumberg, and Kiyoharu Hamaguchi.
Another look at LTL model checking.
Formal Methods in System Design, 10(1):47–71, 1997.


D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi.
On the temporal analysis of fairness.
In *7th ACM Symposium on Principles of Programming Languages, Las Vegas*, pages 163–173, 1980.


Ullrich Hustadt and Boris Konev.
TRP++2.0: A temporal resolution prover.
In Franz Baader, editor, *Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August 2, 2003, Proceedings*, volume 2741 of *Lecture Notes in Computer Science*, pages 274–278. Springer, 2003.


Jianwen Li, Yinbo Yao, Geguang Pu, Lijun Zhang, and Jifeng He.
Aalta: an LTL satisfiability checker over infinite/finite traces.
In Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey, editors, *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pages 731–734. ACM, 2014.


Mark Reynolds.
A traditional tree-style tableau for LTL.
CoRR, abs/1604.03962, 2016.


A. Sistla and E. Clarke.
Complexity of propositional linear temporal logics.
J. ACM, 32:733–749, 1985.


Stefan Schwendimann.
A new one-pass tableau calculus for PLTL.
In Harrie C. M. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference,*

TABLEAUX '98, Oisterwijk, The Netherlands, May 5-8, 1998, Proceedings, volume 1397 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 1998.


Viktor Schuppan and Luthfi Darmawan.
Evaluating LTL satisfiability solvers.
In Tevfik Bultan and Pao-Ann Hsiung, editors, *ATVA'11*, volume 6996 of *Lecture Notes in Computer Science*, pages 397–413. Springer, 2011.


Martin Suda and Christoph Weidenbach.
A pltl-prover based on labelled superposition with partial model guidance.
In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 537–543. Springer, 2012.


Florian Rainer Widmann.

Tableaux-based decision procedures for fixed point logics.
Thesis, 2010.
Thesis (Ph.D.) – ANU, 2010.