A One-Pass Tree-Shaped Tableau for LTL+Past

Nicola Gigante
University of Udine, Italy

joint work with Angelo Montanari
University of Udine, Italy

and Mark Reynolds
University of Western Australia, Australia

LPAR-21, May 8-12 2017, Maun, Botswana
Introduction
Linear Temporal Logic (LTL) is a propositional modal logic interpreted over infinite, discrete, linear orders.

- $\chi \alpha$ \begin{itemize} \item $\alpha$ will be true at the \textbf{next} state. \end{itemize}
- $\alpha U \beta$ \begin{itemize} \item $\beta$ will eventually be true, and \item $\alpha$ always holds \textbf{until} then. \end{itemize}
- $F \beta \equiv T U \beta$ \begin{itemize} \item $\beta$ will \textbf{eventually} be true. \end{itemize}
- $G \beta \equiv \neg F \neg \beta$ \begin{itemize} \item $\beta$ will \textbf{always} be true. \end{itemize}
Augmenting LTL with past operators

LTL can be augmented with past modalities:

- $Y\alpha$ \hspace{1cm} $\alpha$ was true at the previous state.
- $\alpha S \beta$ \hspace{1cm} $\beta$ has been true in the past, and $\alpha$ always held since then.
- $P \beta \equiv T S \beta$ \hspace{1cm} $\beta$ has been true in the past.
- $H \beta \equiv \neg P \neg \beta$ \hspace{1cm} historically, $\beta$ has always been true.

Why? Past operators do not add expressive power to LTL, but they do allow to express many formulae more succinctly.

Note: formulae are satisfied if they hold at the first state.
LTL satisfiability is the problem of checking whether there exists a model that satisfies a given LTL formula.

- **PSPACE-complete** problem.
- Algorithmic solutions:
  - (Büchi) Automata-based
  - Tableau methods
  - Temporal resolution
  - Reduction to model checking
  - ...

The satisfiability problem for LTL+P is still PSPACE-complete.
Why LTL satisfiability?

LTL is usually used to write specification in model checking, but other applications exist for the satisfiability problem:

- sanity checking of specifications
- temporal reasoning in AI
- ...

Tableaux were among the first methods proposed to solve the LTL satisfiability problem:

- Early tableau methods were graph-shaped and multiple-pass (Wolper 1984).
- Subsequently, Schwendimann [Sch98] introduced a single-pass tableau with a tree-like shape (still a DAG).
A one-pass tree-shaped tableau method for LTL satisfiability was recently proposed.

<table>
<thead>
<tr>
<th>Reynolds 2016</th>
</tr>
</thead>
</table>
A one-pass tree-shaped tableau method for LTL satisfiability was recently proposed, and implemented in a tool.

Bertello et al. 2016


http://www.github.com/corralx/leviathan
A one-pass tree-shaped tableau method for LTL satisfiability was recently proposed, and implemented in a tool.

- Purely tree-shaped rule-based search procedure.
- A single pass is sufficient to determine the acceptance of rejection of a given branch.
- Very simple structure, combining the simplicity of declarative tableaux with the efficiency of one-pass systems.
- Easy to extend!
- Easy to parallelize (work in progress)!
In this paper we extended the method to support LTL+P:

- The extension can be done in a very modular way:
  - it respects the same one-pass tree-shaped structure.
  - new rules are added to the system, with old rules left completely unchanged.

- First evidence of how this tableau can be easy to extend to different logics.
How it works
The tableau for $\phi$ is a tree where each node is labeled by a set of formulae, with the root labeled with $\{\phi\}$.

- The formula starts in Negated Normal Form.
- At each step some rules are applied to a leaf, depending on the contents of the label, possibly generating new children for the current node.
- Some rules can accept a branch, others can reject it.
- If the complete tree contains at least an accepted branch, the formula is satisfiable.
Expansion rules are applied to a node until no other expansion rule can be applied anymore:

- Boolean connectives handled just like in classical propositional tableau.

\[
\begin{array}{c}
\{ \alpha \lor \beta \} \\
\{ \alpha \} \quad \{ \beta \} \\
\{ \alpha \land \beta \} \\
\{ \alpha, \beta \}
\end{array}
\]
Expansion rules

Expansion rules are applied to a node until no other expansion rule can be applied anymore:

- Common expansion rules handle temporal operators:

  \[
  \{ \alpha \cup \beta \} \quad \{ \alpha, X(\alpha \cup \beta) \} \quad \{ F \beta \} \quad \{ G \alpha \} \quad \{ \alpha, X G \alpha \} 
  \]

  \( \beta \) is called an **eventuality**.
Once the current state has been fully expanded, we proceed to the next temporal state by the STEP rule:

$$\{\ldots, X\alpha, \ldots\} \downarrow \{\alpha\}$$
If a label contains contradictions, we **reject** the branch.

\[
\{\ldots, p, \ldots, \neg p, \ldots\}
\]

\(\times\)
If a **STEP** rule results into an empty label, we’re done: the branch is **accepted**.

\[
\{ \ldots, p, \neg q, r, \ldots \} \\
\downarrow \\
\{ \} \\
\checkmark
\]
Finding periodic models - LOOP rule

Some formulae (e.g., GFp) require to satisfy infinitely often the same request, thus the labels may never become empty.

This formulae will have infinite periodic models:

**LOOP rule**

If two nodes $u < v$ with labels $\Gamma_u = \Gamma_v$ are found and all the eventualities in $\Gamma_u$ are fulfilled inbetween, the branch is accepted and the model loops through $u$ and $v$. 
Example

\[ \{ G F(p \land X \neg p) \} \]
Example

\[
\{G F(p \land X \neg p)\} \\
| \\
\{F(p \land X \neg p), X G F(p \land X \neg p)\}
\]
Example

\[
\begin{align*}
\{G \ F(p \land X \neg p)\} \\
\{F(p \land X \neg p), X \ G \ F(p \land X \neg p)\} \\
\ldots \\
\{p, X \neg p, X \ G \ F(p \land X \neg p)\}
\end{align*}
\]
Example

\(\{GF(p \land X \neg p)\}\)

| \(\{F(p \land X \neg p), XGF(p \land X \neg p)\}\)

\(\ldots\)

\(\{p, X \neg p, XGF(p \land X \neg p)\}\)

\(\downarrow\)

\(\{\neg p, GF(p \land X \neg p)\}\)
Example

\[
\{ \text{GF}(p \land X \neg p) \} \\
\downarrow \\
\{ \text{F}(p \land X \neg p), \text{XGF}(p \land X \neg p) \} \\
\downarrow \\
\{ p, X \neg p, \text{XGF}(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, \text{GF}(p \land X \neg p) \}
\]
Example

\[
\{ \text{GF}(p \land X \neg p) \} \\
\{ F(p \land X \neg p), X \text{GF}(p \land X \neg p) \} \\
\ldots \\
\{ p, X \neg p, X \text{GF}(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, \text{GF}(p \land X \neg p) \} \\
\{ \neg p, F(p \land X \neg p), X \text{GF}(p \land X \neg p) \} \\
\]
Example

\[
\{G \ F(p \land X \neg p)\} \\
\downarrow \\
\{\neg p, \ G \ F(p \land X \neg p)\} \\
\downarrow \\
\{\neg p, \ F(p \land X \neg p), \ X \ G \ F(p \land X \neg p)\} \\
\downarrow \\
\{\neg p, \ p, \ X \neg p, \ldots\}
\]
Example

\[
\{ \text{GF}(p \land X \neg p) \} \\
\{ \text{F}(p \land X \neg p), \text{XG} \text{F}(p \land X \neg p) \} \\
\ldots \\
\{ p, X \neg p, \text{XG} \text{F}(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, \text{GF}(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, \text{F}(p \land X \neg p), \text{XG} \text{F}(p \land X \neg p) \} \\
\{ \neg p, p, X \neg p, \ldots \} \{ \neg p, \text{XF}(p \land X \neg p), \text{XG} \text{F}(p \land X \neg p) \} \\
\times
\]
Example

\[
\{ GF(p \land X \neg p) \} \\
\downarrow \\
\{ p, X \neg p, XGF(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, GF(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, F(p \land X \neg p), XGF(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, p, X\neg p, \ldots \} \\
\downarrow \\
\{ F(p \land X \neg p), GF(p \land X \neg p) \} \\
\]
\begin{align*}
\{ \text{GF}(p \land X \neg p) \} \\
\{ \text{F}(p \land X \neg p), \text{XGF}(p \land X \neg p) \} \\
\ldots \\
\{ p, X \neg p, \text{XGF}(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, \text{GF}(p \land X \neg p) \} \\
\downarrow \\
\{ \neg p, \text{F}(p \land X \neg p), \text{XGF}(p \land X \neg p) \} \\
\{ \neg p, p, X \neg p, \ldots \} \\
\{ \neg p, \text{XF}(p \land X \neg p), \text{XGF}(p \land X \neg p) \} \\
\times \\
\{ \text{F}(p \land X \neg p), \text{GF}(p \land X \neg p) \} \\
\end{align*}
Example

\[
\{ \mathsf{GF}(p \land X \neg p) \}
\]

\[
\{ \mathsf{F}(p \land X \neg p), \mathsf{XGF}(p \land X \neg p) \}
\]

\[
\ldots\{ p, X \neg p, \mathsf{XGF}(p \land X \neg p) \}\]

\[
\{ \neg p, \mathsf{GF}(p \land X \neg p) \}
\]

\[
\{ \neg p, \mathsf{F}(p \land X \neg p), \mathsf{XGF}(p \land X \neg p) \}
\]

\[
\{ \neg p, p, X \neg p, \ldots \} \{ \neg p, \mathsf{XF}(p \land X \neg p), \mathsf{XGF}(p \land X \neg p) \}\]

\[
\mathsf{X}
\]

\[
\{ \mathsf{F}(p \land X \neg p), \mathsf{GF}(p \land X \neg p) \}
\]

\[
\ldots\{ p, X \neg p, \mathsf{XGF}(p \land X \neg p) \}\]
Example

\[
\{G F(p \land X \neg p)\}
\]

\[
\{F(p \land X \neg p), X G F(p \land X \neg p)\}
\]

\[
\ldots
\]

\[
\{p, X \neg p, X G F(p \land X \neg p)\}
\]

\[
\downarrow
\]

\[
\{\neg p, G F(p \land X \neg p)\}
\]

\[
\downarrow
\]

\[
\{\neg p, F(p \land X \neg p), X G F(p \land X \neg p)\}
\]

\[
\{\neg p, p, X \neg p, \ldots\}
\]

\[
\{\neg p, X F(p \land X \neg p), X G F(p \land X \neg p)\}
\]

\[
\downarrow
\]

\[
\{F(p \land X \neg p), G F(p \land X \neg p)\}
\]

\[
\ldots
\]

\[
\{p, X \neg p, X G F(p \land X \neg p)\}
\]
Example

\[
\{G F(p \land X \neg p)\} \\
\{F(p \land X \neg p), X GF(p \land X \neg p)\}
\]

\[
\ldots
\{p, X \neg p, X GF(p \land X \neg p)\}
\]

\[
\downarrow
\{\neg p, GF(p \land X \neg p)\}
\]

\[
\{\neg p, F(p \land X \neg p), X GF(p \land X \neg p)\}
\]

\[
\{\neg p, p, X \neg p, \ldots\} \quad \{\neg p, XF(p \land X \neg p), X GF(p \land X \neg p)\}
\]

\[
\land
\{F(p \land X \neg p), GF(p \land X \neg p)\}
\]

\[
\ldots
\checkmark \{p, X \neg p, X GF(p \land X \neg p)\}\]
Unrealizable eventualities

Something is still missing. Consider the following formula:

\[ G \neg p \land q \mathcal{U} p \]

- It is unsatisfiable, but not because of propositional contradictions.
- The requested eventuality is unrealizable.
In these cases we have to stop postponing the eventuality to guarantee termination:

<table>
<thead>
<tr>
<th><strong>PRUNE rule</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>If three occurrences of the same label $\Gamma$ are found in three nodes $u &lt; v &lt; w$ and the set of eventualities fulfilled between $u$ and $v$ is the same of those between $v$ and $w$, the branch is <strong>rejected</strong>.</td>
</tr>
</tbody>
</table>
Example - unsatisfiable formula

\{G \neg p \land q \cup p\}
Example - unsatisfiable formula

\{ G \neg p \land q U p \}\n\mid
\{ G \neg p, q U p \}
Example - unsatisfiable formula

\[ \{ G \neg p \land q \cup p \} \]
| \[ \{ G \neg p, q \cup p \} \]
| \[ \{ \neg p, XG \neg p, p \} \]
| \[ \{ \neg p, XG \neg p, q, X(q \cup p) \} \]
Example - unsatisfiable formula

\[
\begin{align*}
\{ \neg p \wedge q \cup p \} \\
| \\
\{ \neg p, q \cup p \} \\
\{ \neg p, X \neg p, p \} \quad \{ \neg p, X \neg p, q, X(q \cup p) \} \\
\times
\end{align*}
\]
Example - unsatisfiable formula

\[ \{G \neg p \land q \cup p\} \]

\[ \{G \neg p, q \cup p\} \]

\[ \{\neg p, X G \neg p, p\} \{\neg p, X G \neg p, q, X(q \cup p)\} \]

\[ \times \]
Example - unsatisfiable formula

\[
\{ G \neg p \land q \cup p \} \\
\{ G \neg p, q \cup p \} \\
\{ \neg p, XG \neg p, p \} \quad \{ \neg p, XG \neg p, q, X(q \cup p) \} \\
\times \\
\{ G \neg p, q \cup p \}
\]
Example - unsatisfiable formula

\[ \{ G \neg p \land q \cup p \} \]

\[ \begin{array}{c}
\{ G \neg p, q \cup p \} \\
\{ \neg p, XG \neg p, p \} \quad \{ \neg p, XG \neg p, q, X(q \cup p) \}
\end{array} \]

\[ \times \quad \downarrow \]

\[ \{ G \neg p, q \cup p \} \]

\[ \begin{array}{c}
\{ \neg p, XG \neg p, p \} \quad \{ \neg p, XG \neg p, q, X(q \cup p) \}
\end{array} \]

\[ \times \quad \downarrow \]

\[ \{ G \neg p, q \cup p \} \]
Example - unsatisfiable formula

\[\{G \neg p \land q \cup p\}\]

\[\{G \neg p, q \cup p\}\]

\[\{\neg p, XG \neg p, p\} \{\neg p, XG \neg p, q, X(q \cup p)\}\]

\[\times\]

\[\{G \neg p, q \cup p\}\]

\[\{\neg p, XG \neg p, q, X(q \cup p)\}\]

\[\times\]

\[\{G \neg p, q \cup p\}\]

\[\{\neg p, XG \neg p, p\} \{\neg p, XG \neg p, q, X(q \cup p)\}\]

\[\times\]

\[\{G \neg p, q \cup p\}\]

\[\{\neg p, XG \neg p, q, X(q \cup p)\}\]

\[\times\]
Example - unsatisfiable formula

\[
\{ G \neg p \land q \lor p \} \\
\{ G \neg p, q \lor p \} \\
{\neg p, XG \neg p, p} \quad {\neg p, XG \neg p, q, X(q \lor p)} \\
{\neg p, XG \neg p, p} \quad {\neg p, XG \neg p, q, X(q \lor p)} \\
\]
Example - unsatisfiable formula

\[
\{G \neg p \land q U p\} \\
| \\
\{G \neg p, q U p\} \\
\{\neg p, X G \neg p, p\} \quad \{\neg p, X G \neg p, q, X(q U p)\} \\
\times \\
\downarrow \\
\{G \neg p, q U p\} \\
\{\neg p, X G \neg p, p\} \quad \{\neg p, X G \neg p, q, X(q U p)\} \\
\times \\
\downarrow \\
\{G \neg p, q U p\} \\
\{\neg p, X G \neg p, p\} \quad \{\neg p, X G \neg p, q, X(q U p)\} \\
\times \\
\times
\]
To summarize:

• When to **accept** a branch?
  • When the label is **empty**
  • When we are looping while satisfying all the **eventualities**

• When to **reject** a branch?
  • When a label is contradictory
  • When we are looping but unable to satisfy all the eventualities
To summarize:

- When to accept a branch?
  - When the label is empty
  - When we are looping while satisfying all the eventualities

- When to reject a branch?
  - When a label is contradictory
  - When we are looping but unable to satisfy all the eventualities
Supporting past operators
Supporting past operators

Handling the past is trivial in graph-shaped tableaux:

• Just build the graph edges such that each $Y_\alpha$ is satisfied

Our one-pass tableau is different:

• In each branch we are committed to a single history
• How to ensure the satisfaction of past requests if the past is fixed already?
Past temporal operators other than $Y\alpha$ are expanded like their future counterparts:

$$
\begin{align*}
\{\alpha \mathcal{S} \beta\} & \quad \{\beta\} & \quad \{\alpha, Y(\alpha \mathcal{S} \beta)\} \\
\{\alpha, Y(\alpha \mathcal{S} \beta)\} & \quad \{\beta\} & \quad \{Y \mathcal{P} \beta\} \\
\{Y \mathcal{P} \beta\} & \quad \{\alpha, Y \mathcal{H} \alpha\}
\end{align*}
$$

Thus the problem reduces to correctly handling $Y\alpha$ formulae.
Introducing the YESTERDAY rule:

- If $u$ is such that $Y\alpha \in \Gamma_u$ and the STEP rule has never been applied before, then the branch is rejected.
- Otherwise, let $v$ be the node to which we lastly applied the STEP rule.
  - If we cannot find $\alpha$ in $v$ nor in its expanded ancestors, then the branch is rejected.
  - A new child $v'$ is added to $v$, with $\Gamma_{v'} = \Gamma_v \cup \{\alpha\}$
{φ}
Supporting past operators - example

\{\phi\} \leftarrow \{\ldots, XY(p \lor q),\ldots\} \rightarrow \ldots
Supporting past operators - example

\[
\{ \phi \} \\
\downarrow \\
\{ \ldots, X Y(p \lor q), \ldots \} \\
\{ \ldots, Y(p \lor q), \ldots \}
\]
Supporting past operators - example

\{ \ldots, XY(p \lor q), \ldots \} \quad \{ \phi \} \quad \ldots

\downarrow

\{ \ldots, Y(p \lor q), \ldots \}

\times
Supporting past operators - example

\[
\{ \phi \} \\
\{ \ldots, \mathcal{X} \mathcal{Y}(p \lor q), \ldots \} \\
\downarrow \\
\{ \ldots, \mathcal{Y}(p \lor q), \ldots \} \\
\{ \ldots, \mathcal{X} \mathcal{Y}(p \lor q), p \lor q, \ldots \} \:
\]

\[\times \]
Supporting past operators - example

\[ \{\phi\} \]

\[ \{\ldots, XY(p \lor q), \ldots\} \]

\[ \downarrow \]

\[ \{\ldots, Y(p \lor q), \ldots\} \quad \{\ldots, XY(p \lor q), p \lor q, \ldots\} \quad * \]

\[ \times * \]

\[ \{\ldots, XY(p \lor q), p, \ldots\} \]

\[ \ldots \]
Supporting past operators - example

\[
\begin{align*}
\{\phi\} & \\
\{\ldots, XY(p \lor q), \ldots\} & \quad \ldots \\
\downarrow & \\
\{\ldots, Y(p \lor q), \ldots\} & \quad \{\ldots, XY(p \lor q), p \lor q, \ldots\} \\
\times & \quad \times \\
\{\ldots, XY(p \lor q), p, \ldots\} & \quad \ldots \\
\downarrow & \\
\{\ldots, Y(p \lor q), \ldots\}
\end{align*}
\]
Supporting past operators - example

\[
\{ \phi \} \quad \rightarrow \quad \{ \ldots, X Y(p \lor q), \ldots \} \quad \rightarrow \quad \{ \ldots, Y(p \lor q), \ldots \}
\]

\[
\{ \ldots, X Y(p \lor q), p, \ldots \} \quad \rightarrow \quad \{ \ldots, Y(p \lor q), \ldots \}
\]

\[
\{ \ldots, X Y(p \lor q), p \lor q, \ldots \}
\]
Supporting past operators - example
Conclusions
Conclusions

We extended a recent one-pass tree-shaped tableau method for LTL satisfiability to cover past operators:

We provided a very modular extension:

- The extension requires only a single new rule for each new temporal operator.
- We preserve the one-pass rule-based tree search structure of the procedure.
- We provide full soundness and completeness proofs:
  - soundness never appeared before (future-only neither)
  - improved, clarified completeness proof
Future lines of work:

- Add the past to our satisfiability checking tool.
  - Not trivial: our rule causes a lot of backtracking
- Exploit the modular structure of the tableau to extend it to other LTL extensions:
  - LTL on finite traces,
  - LTL with forgettable past,
  - metric extensions of LTL,
  - Alur & Hentzinger TPTL logic [AH94],
  - ...
- Implement these extensions: one tool for a broad family of linear time logics
Thank you!

Questions?
