One-Pass Tree-Shaped Tableau Systems for Timed Temporal Logics

Luca Geatti  
University of Udine, Italy

joint work with Nicola Gigante  
University of Udine, Italy

and Angelo Montanari  
University of Udine, Italy

and Mark Reynolds  
University of Western Australia, Australia

GandALF, 26 - 28 September 2018, Saarbrücken, Germany
Introduction
A real-time system is commonly described as a system that “controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time”.

- their correctness does not depend only on their logical correctness, but also on their response time;
- most of the mission or safety critical systems are real-time: their formal correctness is an aspect that cannot be overlooked.
A real-time system is commonly described as a system that “controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time”.

- their correctness does not depend only on their logical correctness, but also on their response time;
- most of the mission or safety critical systems are real-time: their formal correctness is an aspect that cannot be overlooked.
A real-time system is commonly described as a system that “controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time”.

- their correctness does not depend only on their logical correctness, but also on their response time;
- most of the mission or safety critical systems are real-time: their formal correctness is an aspect that cannot be overlooked.
In classical LTL, we can express the request-response property:

$$\varphi := G(request \rightarrow F response)$$

We do not know the exact times at which the request and the response actually take place: the only thing we know is the temporal ordering between these two events.

**LTL** $\Rightarrow$ qualitative time requirements only, not suitable for real-time properties.
In classical LTL, we can express the request-response property:

$$\varphi := G(request \rightarrow F response)$$

We do not know the exact times at which the request and the response actually take place: the only thing we know is the *temporal ordering* between these two events.

$$\text{LTL} \implies \text{qualitative time requirements only, not suitable for real-time properties.}$$
Timed Propositional Temporal Logic (TPTL [AH94]) allows for quantitative time requirements.

- Syntax:

  
  \[
  \begin{align*}
  (\textit{terms}) \; \pi & := x + c \mid c \\
  (\textit{formulae}) \; \phi & := p \mid \pi_1 \leq \pi_2 \mid \pi_1 \equiv d \; \pi_2 \mid \\
  & \quad \neg \phi_1 \mid \phi_1 \lor \phi_2 \mid \phi_1 \land \phi_2 \mid \\
  & \quad X \phi_1 \mid \phi_1 U \phi_2 \mid \phi_1 R \phi_2 \mid \\
  & \quad x.\phi_1
  \end{align*}
  \]

  where \( x \) is a variable, \( p \) is a proposition letter and \( c,d \in \mathbb{N} \).

- ‘\( x. \)’ is a freeze quantifier: ‘\( x. \)’ freezes the variable \( x \) to the time of the local temporal context.
**Definition (Timed state sequence)**

Let $\sigma = \sigma_0\sigma_1\sigma_2 \ldots$ be an infinite sequence of states (each state is a subset of proposition letters).
Definition (Timed state sequence)

Let $\sigma = \sigma_0\sigma_1\sigma_2 \ldots$ be an infinite sequence of states (each state is a subset of proposition letters). A time sequence $\tau = \tau_0\tau_1\tau_2 \ldots$ is an infinite sequence of times $\tau_i \in \mathbb{N}$, for all $i \geq 0$, that satisfies the following conditions:
Definition (Timed state sequence)

Let \( \sigma = \sigma_0\sigma_1\sigma_2 \ldots \) be an infinite sequence of states (each state is a subset of proposition letters). A time sequence \( \tau = \tau_0\tau_1\tau_2 \ldots \) is an infinite sequence of times \( \tau_i \in \mathbb{N} \), for all \( i \geq 0 \), that satisfies the following conditions:

1. monotonicity: \( \tau_i \leq \tau_{i+1} \), for all \( i \geq 0 \);
2. progress: for all \( t \in \mathbb{N} \), there exists \( i \geq 0 \) such that \( \tau_i > t \).
Definition (Timed state sequence)

Let $\sigma = \sigma_0 \sigma_1 \sigma_2 \ldots$ be an infinite sequence of states (each state is a subset of proposition letters). A time sequence $\tau = \tau_0 \tau_1 \tau_2 \ldots$ is an infinite sequence of times $\tau_i \in \mathbb{N}$, for all $i \geq 0$, that satisfies the following conditions:

1. monotonicity: $\tau_i \leq \tau_{i+1}$, for all $i \geq 0$;
2. progress: for all $t \in \mathbb{N}$, there exists $i \geq 0$ such that $\tau_i > t$.

A timed state sequence $\rho = (\sigma, \tau)$ is a pair consisting of a state sequence $\sigma$ and a time sequence $\tau$. 

Let $E : V \rightarrow \mathbb{N}$ be an interpretation for the variables, that we call environment.
### Definition (Timed state sequence)

Let \( \sigma = \sigma_0 \sigma_1 \sigma_2 \ldots \) be an infinite sequence of states (each state is a subset of proposition letters). A time sequence \( \tau = \tau_0 \tau_1 \tau_2 \ldots \) is an infinite sequence of times \( \tau_i \in \mathbb{N} \), for all \( i \geq 0 \), that satisfies the following conditions:

1. **monotonicity**: \( \tau_i \leq \tau_{i+1} \), for all \( i \geq 0 \);
2. **progress**: for all \( t \in \mathbb{N} \), there exists \( i \geq 0 \) such that \( \tau_i > t \).

A **timed state sequence** \( \rho = (\sigma, \tau) \) is a pair consisting of a state sequence \( \sigma \) and a time sequence \( \tau \).

Let \( \mathcal{E} : \mathcal{V} \rightarrow \mathbb{N} \) be an interpretation for the variables, that we call environment.
We inductively define $\rho^i \models T \phi$, as follows:

1. $\rho^i \models T p$ iff $p \in \sigma_i$

2. $\rho^i \models T \pi_1 \leq \pi_2$ iff $E(\pi_1) \leq E(\pi_2)$

3. $\rho^i \models T \pi_1 \equiv_d \pi_2$ iff $E(\pi_1) \equiv_d E(\pi_2)$

4. $\rho^i \models T x.\phi$ iff $\rho^i \models T x:=\tau_i \phi$

The other operators are interpreted in the same way as in LTL.

Example (classical time bounded request-response property):

$$\phi_{BR} := \text{G} x. (\text{request} \rightarrow \text{F} y. (\text{response} \land y \leq x + 10))$$
We inductively define $\rho^i \models \varepsilon \phi$, as follows:

1. $\rho^i \models \varepsilon p$ iff $p \in \sigma_i$
2. $\rho^i \models \varepsilon \pi_1 \leq \pi_2$ iff $\mathcal{E}(\pi_1) \leq \mathcal{E}(\pi_2)$
3. $\rho^i \models \varepsilon \pi_1 \equiv_d \pi_2$ iff $\mathcal{E}(\pi_1) \equiv_d \mathcal{E}(\pi_2)$
4. $\rho^i \models \varepsilon x.\phi$ iff $\rho^i \models \varepsilon[x:=\tau_i] \phi$

The other operators are interpreted in the same way as in LTL.

**Example** (classical time bounded request-response property):

$$\phi_{BR} := G x. (\text{request} \rightarrow F y. (\text{response} \land y \leq x + 10))$$
The TPTL$_b$+P logic

TPTL$_b$+P is a bounded version of TPTL with past operators:

1. $\rho^i \models _{\xi} X_w \phi_1$    iff    $\tau_{i+1} \leq \tau_i + w$ and $\rho^{i+1} \models _{\xi} \phi_1$
2. $\rho^i \models _{\xi} \phi_1 U_w \phi_2$    iff    there exists $j \geq i$ such that
   (i) $\tau_j \leq \tau_i + w$
   (ii) $\rho^j \models _{\xi} \phi_2$
   (iii) $\rho^k \models _{\xi} \phi_1$ for all $i \leq k < j$

The bounds on the operators allow us to know a priori the bound between two variables. The bounds are similar to the ones of Metric Temporal Logic.
The TPTL\(_b+P\) logic

TPTL\(_b+P\) is a bounded version of TPTL with past operators:

1. \(\rho^i \models_\xi X_w \phi_1\) iff \(\tau_{i+1} \leq \tau_i + w\) and \(\rho^{i+1} \models_\xi \phi_1\)

2. \(\rho^i \models_\xi \phi_1 U_w \phi_2\) iff there exists \(j \geq i\) such that
   (i) \(\tau_j \leq \tau_i + w\)
   (ii) \(\rho^j \models_\xi \phi_2\)
   (iii) \(\rho^k \models_\xi \phi_1\) for all \(i \leq k < j\)

The bounds on the operators allow us to know \textit{a priori} the bound between two variables. The bounds are similar to the ones of Metric Temporal Logic.
The satisfiability problem for TPTL (resp. TPTL\_b+P) is the problem of deciding whether there exists a model satisfying a given TPTL (resp. TPTL\_b+P) formula.

Useful in a number of situations:

- sanity check: it allows one to check whether an input formula is satisfiable before running a model checking algorithm;
- monitoring, synthesis (UNSAT \rightarrow UNREALIZABLE), and, in general, all the steps of a model-based design approach;
- the timeline-based planning problem with bounded temporal constraints can be captured by TPTL\_b+P [Del+17].
The **satisfiability problem** for TPTL (resp. TPTL\(_b\)+P) is the problem of deciding whether there exists a model satisfying a given TPTL (resp. TPTL\(_b\)+P) formula.

Useful in a number of situations:

- **sanity check**: it allows one to check whether an input formula is satisfiable before running a model checking algorithm;
- **monitoring**, **synthesis** (UNSAT $\rightarrow$ UNREALIZABLE), and, in general, all the steps of a model-based design approach;
- the **timeline-based planning problem** with bounded temporal constraints can be captured by TPTL\(_b\)+P [Del+17].
The **satisfiability problem** for TPTL (resp. TPTL\(_b\)+P) is the problem of deciding whether there exists a model satisfying a given TPTL (resp. TPTL\(_b\)+P) formula.

Useful in a number of situations:

- **sanity check**: it allows one to check whether an input formula is satisfiable before running a model checking algorithm;
- **monitoring, synthesis** (UNSAT $\rightarrow$ UNREALIZABLE), and, in general, all the steps of a model-based design approach;
- **the timeline-based planning problem** with bounded temporal constraints can be captured by TPTL\(_b\)+P [Del+17].
The **satisfiability problem** for TPTL (resp. $\text{TPTL}_{b+P}$) is the problem of deciding whether there exists a model satisfying a given TPTL (resp. $\text{TPTL}_{b+P}$) formula.

Useful in a number of situations:

- **sanity check**: it allows one to check whether an input formula is satisfiable before running a model checking algorithm;
- **monitoring, synthesis** ($\text{UNSAT} \rightarrow \text{UNREALIZABLE}$), and, in general, all the steps of a model-based design approach;
- **the timeline-based planning problem** with bounded temporal constraints can be captured by $\text{TPTL}_{b+P}$ [Del+17].
Tableau methods are among the most well known techniques used to solve the satisfiability problem for temporal logics:

- **Two-pass and graph-shaped** [MP95]:
  - first pass → builds the graph encoding all the candidate models;
  - second pass → prunes the graph by removing the wrong candidates;
  - difficult to implement and impractical because of the huge size of the graph.

- **One-pass and tree-shaped** [Ber+16]:
  - in a single pass, we can build a candidate model and decide to either accept or reject it;
  - easy to describe and to implement (since each branch is independent from the others, they are easily parallelisable);
Tableau methods are among the most well known techniques used to solve the satisfiability problem for temporal logics:

- **Two-pass and graph-shaped** [MP95]:
  - first pass $\to$ builds the graph encoding all the candidate models;
  - second pass $\to$ prunes the graph by removing the wrong candidates;
  - difficult to implement and impractical because of the huge size of the graph.

- **One-pass and tree-shaped** [Ber+16]:
  - in a single pass, we can build a candidate model and decide to either accept or reject it;
  - easy to describe and to implement (since each branch is independent from the others, they are easily parallelisable);
Main contribution of the paper: the proposal of two original one-pass and tree-shaped tableau systems for the logics

- TPTL
- TPTL$_b$+P

proving their soundness and completeness and analyzing their complexity (both algorithms run in doubly exponential space).
The Tableau System
The tableau is a tree where each node is labeled by a set of subformulae and a time point belonging to $\mathbb{N}$;

The initial tableau for $z.\phi$ (in Negated Normal Form) is a tree consisting of the following single node (the root):

$$\{z.\phi\}_{\text{TIME}=0}$$
• The tableau is a tree where each node is labeled by a set of subformulae and a time point belonging to $\mathbb{N}$;

• The initial tableau for $z.\phi$ (in Negated Normal Form) is a tree consisting of the following single node (the root):

$$\{z.\phi\}^{TIME=0}$$
The tableau is built recursively and top-down starting from the root, by applying a set of rules to the leaves of the tree (in this order):

1. expansion rules: add one or two children to a leaf of the tree;
2. termination rules: close a branch either by ticking a leaf, and thus accepting the branch (√), or by crossing a leaf, and thus rejecting the branch (✗);
3. step rule: force an advancement in time of the model.

If all the branches of the tableau are closed (that is, either ticked or crossed), we say that the tableau is complete.

Given a complete tableau $T_\phi$, the input formula $\phi$ is satisfiable if and only if there is in $T_\phi$ at least one accepted branch.
The tableau is built recursively and top-down starting from the root, by applying a set of rules to the leaves of the tree (in this order):

1. **expansion rules**: add one or two children to a leaf of the tree;
2. **termination rules**: close a branch either by *ticking* a leaf, and thus accepting the branch (√), or by *crossing* a leaf, and thus rejecting the branch (X);
3. **step rule**: force an advancement in time of the model.

If all the branches of the tableau are closed (that is, either ticked or crossed), we say that the tableau is *complete*.

Given a complete tableau $T_\phi$, the input formula $\phi$ is satisfiable if and only if there is in $T_\phi$ at least one accepted branch.
The tableau is built recursively and top-down starting from the root, by applying a set of rules to the leaves of the tree (in this order):

1. expansion rules: add one or two children to a leaf of the tree;
2. termination rules: close a branch either by ticking a leaf, and thus accepting the branch (√), or by crossing a leaf, and thus rejecting the branch (✗);
3. step rule: force an advancement in time of the model.

If all the branches of the tableau are closed (that is, either ticked or crossed), we say that the tableau is complete.

Given a complete tableau $T_\phi$, the input formula $\phi$ is satisfiable if and only if there is in $T_\phi$ at least one accepted branch.
• The tableau is built recursively and top-down starting from the root, by applying a set of rules to the leaves of the tree (in this order):
  1. expansion rules: add one or two children to a leaf of the tree;
  2. termination rules: close a branch either by ticking a leaf, and thus accepting the branch (✓), or by crossing a leaf, and thus rejecting the branch (✗);
  3. step rule: force an advancement in time of the model.

• If all the branches of the tableau are closed (that is, either ticked or crossed), we say that the tableau is complete.

• Given a complete tableau $T_\phi$, the input formula $\phi$ is satisfiable if and only if there is in $T_\phi$ at least one accepted branch.
The tableau is built recursively and **top-down** starting from the root, by applying a set of rules to the leaves of the tree (in this order):

1. **expansion rules**: add one or two children to a leaf of the tree;
2. **termination rules**: close a branch either by *ticking* a leaf, and thus accepting the branch (✓), or by *crossing* a leaf, and thus rejecting the branch (✗);
3. **step rule**: force an advancement in time of the model.

If all the branches of the tableau are closed (that is, either ticked or crossed), we say that the tableau is **complete**.

Given a complete tableau $T_\phi$, the input formula $\phi$ is satisfiable if and only if there is in $T_\phi$ at least one accepted branch.
Expansion rules
Expansion rules are applied to the leaves of the tree, until no expansion rule can be applied anymore.

- $\psi \rightarrow \Delta_1$
- $\psi \rightarrow \Delta_1 \mid \Delta_2$
Expansion rules are applied to the leaves of the tree, until no expansion rule can be applied anymore.

- $\psi \rightarrow \Delta_1$  \quad z.(\alpha \land \beta) \rightarrow \{z.\alpha, z.\beta\}$
- $\psi \rightarrow \Delta_1 \mid \Delta_2$  \quad z.(\alpha \lor \beta) \rightarrow \{z.\alpha\} \mid \{z.\beta\}$

**Boolean connectives:**

\[
\begin{align*}
\{z.(\alpha \land \beta) \ldots\} & \quad \{z.(\alpha \lor \beta) \ldots\} \\
| & \\
\{z.\alpha, z.\beta \ldots\} & \quad \{z.\alpha \ldots\} \quad \{z.\beta \ldots\}
\end{align*}
\]
Expansion rules

Expansion rules are applied to the leaves of the tree, until no expansion rule can be applied anymore.

- $\psi \rightarrow \Delta_1$
- $\psi \rightarrow \Delta_1 \mid \Delta_2$

$$z. G \alpha \rightarrow \{z.\alpha, z. X G \alpha\}$$
$$z. (\alpha \cup \beta) \rightarrow \{z.\beta\} \mid \{z.\alpha, z. X (\alpha \cup \beta)\}$$
$$z. (F \beta) \rightarrow \{\beta\} \mid \{z. X F \beta\}$$

Temporal connectives:

$$\{z. (\alpha \cup \beta) \ldots \}$$
$$\{z. F \beta \ldots \}$$

$$\{z.\beta \ldots\} \mid \{z.\alpha, z. X(\alpha \cup \beta) \ldots\}$$

$$\{z.\beta \ldots\} \mid \{z.\alpha, z. X F \beta \ldots\}$$

$$\{z. G \alpha \ldots\}$$
$$\{z.\alpha, z. X G \alpha \ldots\}$$
Expansion rules

Expansion rules are applied to the leaves of the tree, until no expansion rule can be applied anymore.

- $\psi \rightarrow \Delta_1$
- $z.y.\alpha \rightarrow z.\alpha[z/y]$
- $\psi \rightarrow \Delta_1 \mid \Delta_2$

Freeze quantifier:

\[
\{z.y.\alpha \ldots\} \\
\mid \\
\{z.\alpha[z/y] \ldots\}
\]
By repeatedly applying expansion rules, we eventually reach a node whose label contains only:

- proposition letters;
- timing constraints;
- formulae of type $z.X\alpha$.

Such a node is called a poised node.
Once we reach a poised node, we can apply the **STEP** rule and advance in a state of the model.

\[
\begin{align*}
\{z.\alpha \ldots\}^{TIME=i} & \\
\{z.\alpha^0 \ldots\}^{TIME=i} & \quad | \\
\{z.\alpha^1 \ldots\}^{TIME=i+1} & \quad | \\
\{z.\alpha^{\delta}\phi \ldots\}^{TIME=i+\delta\phi} & \\
\end{align*}
\]

where \(\delta\phi\) is a value that we can pre-compute from the initial formula \(\phi\) and that does not affect satisfiability.

\(\cdot\)^\(\delta\) is called a temporal shift. For instance:

- \(x. X G y.(p \rightarrow y \leq x + 1)^1 = x. X G y.(p \rightarrow y \leq x)\)
- \(x. X G y.(p \rightarrow y \leq x + 1)^2 = x. X G y.(p \rightarrow \bot)\)
Once we reach a poised node, we can apply the **STEP** rule and advance in a state of the model.

\[
\{z.\alpha^0 \ldots\}^{\text{TIME}=i} \quad \{z.\alpha^1 \ldots\}^{\text{TIME}=i+1} \quad \{z.\alpha^\phi \ldots\}^{\text{TIME}=i+\phi}
\]

where \(\phi\) is a value that we can pre-compute from the initial formula \(\phi\) and that does not affect satisfiability.

\((\cdot)^\delta\) is called a **temporal shift**. For instance:

- \(x.\mathbf{X} \mathbf{G} y.(p \rightarrow y \leq x + 1)^1 = x.\mathbf{X} \mathbf{G} y.(p \rightarrow y \leq x)\)
- \(x.\mathbf{X} \mathbf{G} y.(p \rightarrow y \leq x + 1)^2 = x.\mathbf{X} \mathbf{G} y.(p \rightarrow \bot)\)
Termination rules
Termination rules decide if

- the current branch has to be accepted (✓) (in this case we have found a model);
- the current branch has to be rejected (✗);
- or the current branch must be further explored (i.e., STEP rule).
EMPTY rule:

\[
\{ \ldots, z.p, z.q, \neg z.r, \ldots \} \\
| \\
\{ \} \\
\checkmark
\]

CONTRADICTION rule:

\[
\{ \ldots, z.p, \neg z.p, \ldots \} \\
\times
\]
EMPTY rule:

\[ \{ \ldots, z.p, z.q, \neg z.r, \ldots \} \]

CONTRADICTION rule:

\[ \{ \ldots, z.p, \neg z.p, \ldots \} \]
SYNC rule:

\{ \ldots, x.(x \leq x + 1), \ldots \} 

We can check the truth of this timing constraint by simply checking if $0 \leq 1$.

Remark: thanks to the expansion rule $z.y.\alpha \rightarrow \{z.\alpha[z/y]\}$ and the temporal shift, all the timing constraints that can appear in a label are of the form $z.(z \sim z + c)$, for some operator $\sim$ and some constant $c \in \mathbb{N}$. 
SYNC rule:

\{\ldots, x. (x \leq x + 1), \ldots \}

We can check the truth of this timing constraint by simply checking if $0 \leq 1$.

Remark: thanks to the expansion rule $z.y.\alpha \rightarrow \{z.\alpha[z/y]\}$ and the temporal shift, all the timing constraints that can appear in a label are of the form $z.(z \sim z + c)$, for some operator $\sim$ and some constant $c \in \mathbb{N}$. 
Consider the TPTL formula for the bounded request-response property:

$$\phi_{BR} := G x. (request \rightarrow F y. (response \land y \leq x + 10))$$

Among the models of this formula there are models featuring infinitely many requests, and consequently infinitely many responses.
Consider the TPTL formula for the bounded request-response property:

\[ \phi_{BR} := G x. ( request \rightarrow F y. ( response \land y \leq x + 10 ) ) \]

Among the models of this formula there are models featuring infinitely many requests, and consequently infinitely many responses.

**LOOP rule**

Let \( v \) be a poised leaf, and let \( u < v \) be a poised node, which is a proper ancestor of \( v \), such that \( \Gamma(u) = \Gamma(v) \) and all the eventualities (i.e., \( z. X ( \alpha U \beta ) \) or \( z. X F \beta \)) requested in \( u \) are fulfilled between \( u \) and \( v \) (included). Then,

- if \( \text{time}(u) = \text{time}(v) \), then \( v \) is crossed and the branch rejected;
- if \( \text{time}(u) < \text{time}(v) \), then \( v \) is ticked and the branch accepted.
We cross the branch because the difference between the two timestamps is 0: the candidate model does not satisfy the progress condition.
Consider the formula $G \neg p \land q \mathcal{U} p$. Even though it does not present any propositional contradiction, it is *unsatisfiable* because the eventuality $p$ cannot be fulfilled.

**PRUNE rule**

Let $w$ be a poised leaf. If there exist three poised nodes $u < v < w$ such that $\Gamma(u) = \Gamma(v) = \Gamma(w)$, and each eventuality requested in $u$ and fulfilled between $v$ and $w$ is also fulfilled between $u$ and $v$, then, $w$ is crossed and the branch rejected.
Consider the formula $G \neg p \land q \mathcal{U} p$. Even though it does not present any propositional contradiction, it is *unsatisfiable* because the eventuality $p$ cannot be fulfilled.

**PRUNE rule**

Let $w$ be a poised leaf. If there exist three poised nodes $u < v < w$ such that $\Gamma(u) = \Gamma(v) = \Gamma(w)$, and each eventuality requested in $u$ and fulfilled between $v$ and $w$ is also fulfilled between $u$ and $v$, then, $w$ is crossed and the branch rejected.
PRUNE rule - Example

\[
\{\ldots, X F \alpha, X F \beta, X F \gamma, \ldots \}^{\text{TIME}=i} \\
\{\ldots, \gamma, \ldots \} \\
\{\ldots, \beta, \ldots \} \\
\{\ldots, X F \alpha, X F \beta, X F \gamma, \ldots \}^{\text{TIME}=j}
\]
PRUNE rule - Example

\[
\{\ldots, X F \alpha, X F \beta, X F \gamma, \ldots \}^{\text{TIME}=i}
\]

\[
\{\ldots, \gamma, \ldots \}
\]

\[
\{\ldots, \beta, \ldots \}
\]

\[
\{\ldots, X F \alpha, X F \beta, X F \gamma, \ldots \}^{\text{TIME}=j}
\]

\[
\{\ldots, \gamma, \ldots \}
\]

\[
\{\ldots, \beta, \ldots \}
\]

\[
\{\ldots, X F \alpha, X F \beta, X F \gamma, \ldots \}^{\text{TIME}=k}
\]

\[
X
\]
PRUNE rule - Example

$$\{\ldots, XF\,\alpha, XF\,\beta, XF\,\gamma, \ldots\}^{\text{TIME}=i}$$

$$\{\ldots, \gamma, \ldots\}$$

$$\{\ldots, \beta, \ldots\}$$

$$\{\ldots, XF\,\alpha, XF\,\beta, XF\,\gamma, \ldots\}^{\text{TIME}=j}$$

$$\{\ldots, \gamma, \ldots\}$$

$$\{\ldots, \beta, \ldots\}$$

$$\{\ldots, XF\,\alpha, XF\,\beta, XF\,\gamma, \ldots\}^{\text{TIME}=k}$$

Redundant Cycle

X
PRUNE rule - Intuition

**Intuition**

The PRUNE rule recognizes and prunes the redundant cycles.

Why three occurrences (and not only two)?

- with two nodes we identify one cycle (e.g., LOOP rule). If this cycle does not fulfill all the eventualities, then it is an incomplete cycle, but *not* redundant as it still can fulfill in the future the pending requests;
- with three nodes we identify two cycles. Therefore, if the second cycles fulfills a subset of the eventualities fulfilled by the first, then it is a redundant cycle and we can prune it.
Intuition

The PRUNE rule recognizes and prunes the redundant cycles.

Why three occurrences (and not only two)?

- with two nodes we identify one cycle (e.g., LOOP rule). If this cycle does not fulfill all the eventualities, then it is an incomplete cycle, but *not* redundant as it still can fulfill in the future the pending requests;
- with three nodes we identify two cycles. Therefore, if the second cycles fulfills a subset of the eventualities fulfilled by the first, then it is a redundant cycle and we can prune it.
Intuition
The PRUNE rule recognizes and prunes the redundant cycles.

Why three occurrences (and not only two)?

- with two nodes we identify one cycle (e.g., LOOP rule). If this cycle does not fulfill all the eventualities, then it is an incomplete cycle, but not redundant as it still can fulfill in the future the pending requests;
- with three nodes we identify two cycles. Therefore, if the second cycles fufills a subset of the eventualities fulfilled by the first, then it is a redundant cycle and we can prune it.
Examples
Tableau for TPTL - Example

\{ x. G y. (p \rightarrow y \leq x + 2) \}^{TIME=0}
Tableau for TPTL - Example

\[
\{ x. G y. (p \rightarrow y \leq x + 2) \}^{TIME=0}
\]

\[
\{ x. y. (p \rightarrow y \leq x + 2), x. X G y. (p \rightarrow y \leq x + 2) \}^{TIME=0}
\]
Tableau for TPTL - Example

\{x. G y.(p \rightarrow y \leq x + 2)\}^{\text{TIME}=0}

\{x. G y.(p \rightarrow y \leq x + 2)\}

\{x. y.(p \rightarrow y \leq x + 2), x. X G y.(p \rightarrow y \leq x + 2)\}^{\text{TIME}=0}

\text{FREEZE}

\{x.(p \rightarrow x \leq x + 2), x. X G y.(p \rightarrow y \leq x + 2)\}^{\text{TIME}=0}
Tableau for TPTL - Example

\[
\begin{align*}
\{x. G y. (p \rightarrow y \leq x + 2)\}^{TIME=0} \\
\{x. y. (p \rightarrow y \leq x + 2), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0} \\
\{x. (p \rightarrow x \leq x + 2), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0} \\
\{x. (\neg p), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}\{\ldots, x. (x \leq x + 2), \ldots\}^{TIME=0}
\end{align*}
\]
Tableau for TPTL - Example

\[ \{x. G y.(p \rightarrow y \leq x + 2)\}^{TIME=0} \]

\[ \{x.y.(p \rightarrow y \leq x + 2), x. X G y.(p \rightarrow y \leq x + 2)\}^{TIME=0} \]

\[ \{x.(p \rightarrow x \leq x + 2), x. X G y.(p \rightarrow y \leq x + 2)\}^{TIME=0} \]

\[ \{x.(-p), x. X G y.(p \rightarrow y \leq x + 2)\}^{TIME=0} \]

STEP_0 ↓

\[ \{x. G y.(p \rightarrow y \leq x + 2)\}^{TIME=0} \]

\[ \{\ldots, x.(x \leq x + 2), \ldots \}^{TIME=} \]
Tableau for TPTL - Example

\[ \{x. \text{G} y. (p \to y \leq x + 2)\}^{\text{TIME}=0} \]

\[ \{x. y. (p \to y \leq x + 2), x. \text{X G} y. (p \to y \leq x + 2)\}^{\text{TIME}=0} \]

\[ \{x. (p \to x \leq x + 2), x. \text{X G} y. (p \to y \leq x + 2)\}^{\text{TIME}=0} \]

\[ \{x. \neg p, x. \text{X G} y. (p \to y \leq x + 2)\}^{\text{TIME}=0}\{\ldots, x. (x \leq x + 2), \ldots\}^{\text{TIME}=0}\]

\[ \{x. \text{G} y. (p \to y \leq x + 2)\}^{\text{TIME}=0}\]

\[ \{x. \neg p, x. \text{X G} y. (p \to y \leq x + 2)\}^{\text{TIME}=0} \]
Tableau for TPTL - Example

\{x. G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}

| \{x.y. (p \rightarrow y \leq x + 2), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}
| \{x.(p \rightarrow x \leq x + 2), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}

\{x.(\neg p), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}\{\ldots, x.(x \leq x + 2), \ldots\}^{TIME=0}

↓

\{x. G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}

| \{x.(\neg p), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0}

\times LOOP_1

\gg TPTL_b+P
Tableau for TPTL - Example

\{x.(\neg p), x. X G y.(p \rightarrow y \leq x + 2)\}^{TIME=0}
Tableau for TPTL - Example

\[
\{ x. (\neg p), \ x. X G y. (p \rightarrow y \leq x + 2) \}^{\text{TIME}=0}
\]

\[
\downarrow \text{STEP}_1
\]

\[
\{ x. G y. (p \rightarrow y \leq x + 1) \}^{\text{TIME}=1}
\]
Tableau for TPTL - Example

\[
\{ x. (\neg p), x. X G y. (p \rightarrow y \leq x + 2) \}^{TIME=0}
\]

\[
\downarrow
\]

\[
\{ x. G y. (p \rightarrow y \leq x + 1) \}^{TIME=1}
\]

\[
\downarrow
\]

\[
\{ x. (x \leq x + 1), x. X G y. (p \rightarrow y \leq x + 1) \}^{TIME=1}
\]
Tableau for TPTL - Example

\[ \{ x.(-p), x. X G y. (p \rightarrow y \leq x + 2)\}^{TIME=0} \]

\[ \downarrow \]

\[ \{ x. G y. (p \rightarrow y \leq x + 1)\}^{TIME=1} \]

\[ \{ x. (x \leq x + 1), x. X G y. (p \rightarrow y \leq x + 1)\}^{TIME=1} \]

SYNC
Tableau for TPTL - Example

\{x.(-p), x. \timesG y.(p \rightarrow y \leq x + 2)\}^{TIME=0}
\downarrow
\{x. G y.(p \rightarrow y \leq x + 1)\}^{TIME=1}

\{x.(x \leq x + 1), x. \timesG y.(p \rightarrow y \leq x + 1)\}^{TIME=1}
STEP_1 \downarrow
\{x. G y.(p \rightarrow y \leq x)\}^{TIME=2}
Tableau for TPTL - Example

\[ \{ x.(-p), x. X G y.(p \to y \leq x + 2)\}^{TIME=0} \]

\[ \downarrow \]

\[ \{ x. G y.(p \to y \leq x + 1)\}^{TIME=1} \]

\[ \downarrow \]

\[ \{ x.(x \leq x + 1), x. X G y.(p \to y \leq x + 1)\}^{TIME=1} \]

\[ \downarrow \]

\[ \{ x. G y.(p \to y \leq x)\}^{TIME=2} \]

\[ \downarrow \]

\[ \{ x.(x \leq x), x. X G y.(p \to y \leq x)\}^{TIME=2} \]
Tableau for TPTL - Example

\[\{x.(\neg p), x.\mathsf{XGy} .(p \rightarrow y \leq x + 2)\}\]^{TIME=0}

\[\downarrow\]

\[\{x.\mathsf{Gy} .(p \rightarrow y \leq x + 1)\}\]^{TIME=1}

\[\downarrow\]

\[\{x.(x \leq x + 1), x.\mathsf{XGy} .(p \rightarrow y \leq x + 1)\}\]^{TIME=1}

\[\downarrow\]

\[\{x.\mathsf{Gy} .(p \rightarrow y \leq x)\}\]^{TIME=2}

\[\downarrow\]

\[\{x.(x \leq x), x.\mathsf{XGy} .(p \rightarrow y \leq x)\}\]^{TIME=2}

\[\text{STEP}_1\downarrow\]

\[\{x.\mathsf{Gy} .(p \rightarrow \bot)\}\]^{TIME=3}
Tableau for TPTL - Example

\[
\{x. (\neg p), x. XGy.(p \rightarrow y \leq x + 2)\}^{TIME=0}
\]

\[
\downarrow
\]

\[
\{x. Gy.(p \rightarrow y \leq x + 1)\}^{TIME=1}
\]

\[
\downarrow
\]

\[
\{x. (x \leq x + 1), x. XGy.(p \rightarrow y \leq x + 1)\}^{TIME=1}
\]

\[
\downarrow
\]

\[
\{x. Gy.(p \rightarrow y \leq x)\}^{TIME=2}
\]

\[
\downarrow
\]

\[
\{x. (x \leq x), x. XGy.(p \rightarrow y \leq x)\}^{TIME=2}
\]

\[
\downarrow
\]

\[
\{x. Gy.(p \rightarrow \bot)\}^{TIME=3}
\]

\[
\{\ldots \bot \ldots\}^{TIME=3}
\]
Tableau for TPTL - Example

\[
\{x. (\neg p), x. XG y. (p \rightarrow y \leq x + 2)\}^{TIME=0}
\]

\[
\downarrow
\]

\[
\{x. Gy. (p \rightarrow y \leq x + 1)\}^{TIME=1}
\]

\[
\{x. (x \leq x + 1), x. XG y. (p \rightarrow y \leq x + 1)\}^{TIME=1}
\]

\[
\downarrow
\]

\[
\{x. Gy. (p \rightarrow y \leq x)\}^{TIME=2}
\]

\[
\{x. (x \leq x), x. XG y. (p \rightarrow y \leq x)\}^{TIME=2}
\]

\[
\downarrow
\]

\[
\{x. Gy. (p \rightarrow \bot)\}^{TIME=3}
\]

\[
\{\ldots \bot \ldots\}^{TIME=3}
\]

\[\times \text{ CONTRADICTION}\]
Tableau for TPTL - Example

\[
\begin{align*}
\{ & x. (\neg p), x. X G y. (p \rightarrow y \leq x + 2) \}^{TIME=0} \\
\downarrow \\
\{ & x. G y. (p \rightarrow y \leq x + 1) \}^{TIME=1} \\
\downarrow \\
\{ & x. (x \leq x + 1), x. X G y. (p \rightarrow y \leq x + 1) \}^{TIME=1} \\
\downarrow \\
\{ & x. G y. (p \rightarrow y \leq x) \}^{TIME=2} \\
\downarrow \\
\{ & x. (x \leq x), x. X G y. (p \rightarrow y \leq x) \}^{TIME=2} \\
\downarrow \\
\{ & x. G y. (p \rightarrow \bot) \}^{TIME=3} \\
\downarrow \\
\{ & \ldots \bot \ldots \}^{TIME=3} \\
\downarrow \\
\{ & x. (\neg p), x. X G y. (p \rightarrow \bot) \}^{TIME=3} \\
\downarrow \\
\text{x CONTRADICTION}
\end{align*}
\]
Tableau for TPTL - Example

\{ x.(\neg p), x. X G y.(p \rightarrow y \leq x + 2) \}^{TIME=0}

\downarrow

\{ x. G y.(p \rightarrow y \leq x + 1) \}^{TIME=1}

\{ x.(x \leq x + 1), x. X G y.(p \rightarrow y \leq x + 1) \}^{TIME=1}

\downarrow

\{ x. G y.(p \rightarrow y \leq x) \}^{TIME=2}

\{ x.(x \leq x), x. X G y.(p \rightarrow y \leq x) \}^{TIME=2}

\downarrow

\{ x. G y.(p \rightarrow \bot) \}^{TIME=3}

\{ \ldots \bot \ldots \}^{TIME=3}

\times \text{ CONTRADICTION}

\{ x.(\neg p), x. X G y.(p \rightarrow \bot) \}^{TIME=3}

\text{STEP}_1 \downarrow

\{ x.(\neg p), x. X G y.(p \rightarrow \bot) \}^{TIME=4}
Tableau for TPTL - Example

\{x.(¬p), x. X G y.(p \rightarrow y \leq x + 2)\}^{TIME=0}

\downarrow

\{x. G y.(p \rightarrow y \leq x + 1)\}^{TIME=1}

\{x.(x \leq x + 1), x. X G y.(p \rightarrow y \leq x + 1)\}^{TIME=1}

\downarrow

\{x. G y.(p \rightarrow y \leq x)\}^{TIME=2}

\{x.(x \leq x), x. X G y.(p \rightarrow y \leq x)\}^{TIME=2}

\downarrow

\{x. G y.(p \rightarrow ⊥)\}^{TIME=3}

\{\ldots \perp \ldots\}^{TIME=3}

\times CONTRADICTION

\{x.(¬p), x. X G y.(p \rightarrow ⊥)\}^{TIME=3}

\downarrow

\{x.(¬p), x. X G y.(p \rightarrow ⊥)\}^{TIME=4}

✓ LOOP2
Tableau for TPTL$_b$+P

- It has the same structure of the previous tableau for TPTL.
- Now it is **not** true anymore that $y$ is instantiated always in the future w.r.t. $x$, but we can give a priori a bound to the difference between the timestamps of two variables, thanks to the bounds on the temporal operators. This is crucial for simplifying the timed constraints.
- In order to deal with past modalities, the **YESTERDAY rule** has been introduced:
  
  **YESTERDAY**: it checks if all the past requests made by the formulae of the current node are already satisfied by the previous nodes of the current branch;
  - if this is not the case, the current branch is rejected and the construction of the tableau restarts from a previous state of the branch, assuming that all the past requests are true.
Yesterday rule - Example

\{z.\phi\}^{\text{TIME}=0}
Yesterday rule - Example

\[ \{ z.\phi \}^{TIME=0} \]

\[ \{ \ldots, z.XY(\varphi), \ldots \}^{TIME=0} \]

\[ \ldots \]
Yesterday rule - Example

\[
\{z.\phi\}^{TIME=0}
\]

\[
\{\ldots, z. X Y(\varphi), \ldots\}^{TIME=0}
\]

\[
\text{STEP}_{\delta} \downarrow
\]

\[
\{\ldots, z. Y(\psi), \ldots\}^{TIME=\delta}
\]

\[
\ldots
\]
Yesterday rule - Example

\[ \{ z.\phi \}^{TIME=0} \]

\[ \{ \ldots, z. X Y(\varphi), \ldots \}^{TIME=0} \]

\[ \{ \ldots, z. Y(\psi), \ldots \}^{TIME=\delta} \]

\[ \times \text{YESTERDAY} \]
Yesterday rule - Example

\[ \{ \ldots, z.XY(\varphi), \ldots \}^{\text{TIME}=0} \]

\[ \{ \ldots, z.Y(\psi), \ldots \}^{\text{TIME} = \delta} \]

\[ \times \text{YESTERDAY} \]
Yesterday rule - Example

\[
\{z.\phi\}^{\text{TIME}=0}
\]

\[
\{\ldots, z. X Y(\varphi), \ldots\}^{\text{TIME}=0}
\]

\[\downarrow\]

\[
\{\ldots, z. Y(\psi), \ldots\}^{\text{TIME}=\delta}
\]

\[
\{\ldots, z. X Y(\varphi), z. \psi^{-\delta}, \ldots\}^{\text{TIME}=0}
\]

\[\downarrow\]

\[
\{\ldots, z. Y(\psi), \ldots\}^{\text{TIME}=\delta}
\]

\(\times\) YESTERDAY
Yesterday rule - Example

\[ \{z.\phi\}^{\text{TIME}=0} \]

\[ \{\ldots, z.XY(\varphi), \ldots\}^{\text{TIME}=0} \]

\[ \downarrow \]

\[ \{\ldots, z.Y(\psi), \ldots\}^{\text{TIME}=\delta} \]

\[ \times \text{YESTERDAY} \]

\[ \{\ldots, z.XY(\varphi), z.\psi^{-\delta}, \ldots\}^{\text{TIME}=0} \]

\[ \downarrow \]

\[ \{\ldots, z.Y(\psi), \ldots\}^{\text{TIME}=\delta} \]
Yesterday rule - Example

\[
\{z.\phi\}^{\text{TIME}=0} \\
\{\ldots, z. X Y(\varphi), \ldots\}^{\text{TIME}=0} \downarrow \quad \{\ldots, z. Y(\psi), \ldots\}^{\text{TIME}=\delta} \\
\times \text{YESTERDAY} \quad \{\ldots, z. X Y(\varphi), z. \psi^{-\delta}, \ldots\}^{\text{TIME}=0} \downarrow \quad \{\ldots, z. Y(\psi), \ldots\}^{\text{TIME}=\delta} \downarrow \quad \ldots
\]
Conclusions
Results: we developed two original *one-pass* and *tree-shaped* tableau systems for the logics TPTL and TPTL$_b$+P.

- easy to implement and well suited for parallel implementations;
- no optimality: although the satisfiability problem for these two logics is EXPSPACE-complete, our tableau systems run in *doubly* exponential space (logarithmic encoding for the constants).
Results: we developed two original one-pass and tree-shaped tableau systems for the logics TPTL and TPTL$_b$+P.

- easy to implement and well suited for parallel implementations;
- no optimality: although the satisfiability problem for these two logics is EXPSPACE-complete, our tableau systems run in doubly exponential space (logarithmic encoding for the constants).
Results: we developed two original one-pass and tree-shaped tableau systems for the logics TPTL and TPTL$_b$+P.

- easy to implement and well suited for parallel implementations;
- no optimality: although the satisfiability problem for these two logics is EXPSPACE-complete, our tableau systems run in *doubly* exponential space (logarithmic encoding for the constants).
Conclusions - 2

Future developments:

- SAT-based encoding of the tableau for LTL, based on bounded satisfiability;
- SMT-based encoding of the tableau for TPTL, using Difference Logic (DL) as the underlying theory;
- Tableau system for TPTL+P:
  - there is a heavy price to pay for the addition of past modalities to TPTL: the satisfiability problem for TPTL+P is nonelementary;
  - at the moment, there exists no direct procedure for deciding its satisfiability;
  - the main problem to solve is how to recognize a period.
- Extending the tableau systems of [Ber+16] to other linear time temporal logics, e.g., metric LTL.
Future developments:

- SAT-based encoding of the tableau for LTL, based on bounded satisfiability;
- SMT-based encoding of the tableau for TPTL, using Difference Logic (DL) as the underlying theory;
- Tableau system for TPTL+P:
  - there is a heavy price to pay for the addition of past modalities to TPTL: the satisfiability problem for TPTL+P is nonelementary;
  - at the moment, there exists no direct procedure for deciding its satisfiability;
  - the main problem to solve is how to recognize a period.
- Extending the tableau systems of [Ber+16] to other linear time temporal logics, e.g., metric LTL.
Conclusions - 2

Future developments:

- SAT-based encoding of the tableau for LTL, based on bounded satisfiability;
- SMT-based encoding of the tableau for TPTL, using Difference Logic (DL) as the underlying theory;
- Tableau system for TPTL+P:
  - there is a heavy price to pay for the addition of past modalities to TPTL: the satisfiability problem for TPTL+P is nonelementary;
  - at the moment, there exists no direct procedure for deciding its satisfiability;
  - the main problem to solve is how to recognize a period.
- Extending the tableau systems of [Ber+16] to other linear time temporal logics, e.g., metric LTL.
Conclusions - 2

Future developments:

- SAT-based encoding of the tableau for LTL, based on bounded satisfiability;
- SMT-based encoding of the tableau for TPTL, using Difference Logic (DL) as the underlying theory;
- Tableau system for TPTL+P:
  - there is a heavy price to pay for the addition of past modalities to TPTL: the satisfiability problem for TPTL+P is nonelementary;
  - at the moment, there exists no direct procedure for deciding its satisfiability;
    - the main problem to solve is how to recognize a period.
- Extending the tableau systems of [Ber+16] to other linear time temporal logics, e.g., metric LTL.
Future developments:

• SAT-based encoding of the tableau for LTL, based on bounded satisfiability;
• SMT-based encoding of the tableau for TPTL, using Difference Logic (DL) as the underlying theory;
• tableau system for TPTL+P:
  • there is a heavy price to pay for the addition of past modalities to TPTL: the satisfiability problem for TPTL+P is nonelementary;
  • at the moment, there exists no direct procedure for deciding its satisfiability;
  • the main problem to solve is how to recognize a period.
• extending the tableau systems of [Ber+16] to other linear time temporal logics, e.g., metric LTL.
Future developments:

- SAT-based encoding of the tableau for LTL, based on bounded satisfiability;
- SMT-based encoding of the tableau for TPTL, using Difference Logic (DL) as the underlying theory;
- tableau system for TPTL+P:
  - there is a heavy price to pay for the addition of past modalities to TPTL: the satisfiability problem for TPTL+P is nonelementary;
  - at the moment, there exists no direct procedure for deciding its satisfiability;
  - the main problem to solve is how to recognize a period.
- extending the tableau systems of [Ber+16] to other linear time temporal logics, e.g., metric LTL.
Thank you for your attention!