One-Pass and Tree-Shaped Tableau Systems for TPTL and $\mathrm{TPTL_b}$+Past

Luca Geatti, Nicola Gigante, Angelo Montanari

University of Udine, Italy

Mark Reynolds

The University of Western Australia

Abstract

Linear Temporal Logic (LTL) is one of the most commonly used formalisms for representing and reasoning about temporal properties of computations. Its application domains range from formal verification to artificial intelligence. Many *real-time* extensions of LTL have been proposed over the years, including *Timed Propositional Temporal Logic* (TPTL), that makes it possible to constrain the temporal ordering of pairs of events as well as the exact time elapsed between them.

The paper focuses on TPTL and *Bounded* TPTL *with Past* ($\mathsf{TPTL_b}$+P), a bounded variant of TPTL enriched with past operators, which has been recently introduced to formalise a meaningful class of timeline-based planning problems. $\mathsf{TPTL_b}$+P allows one to refer to the past while keeping the computational complexity under control: in contrast to the full TPTL *with Past* (TPTL+P), whose satisfiability problem is *non-elementary*, the satisfiability problem for $\mathsf{TPTL_b}$+P is EXPSPACE-complete.

The paper deals with the satisfiability problem for TPTL and $\mathsf{TPTL_b}$+P by providing an original tableau system for each of them that suitably generalises Reynolds' *one-pass* and *tree-shaped* tableau for LTL. First, we show how to handle *past operators*, by devising a one-pass and tree-shaped tableau system for LTL *with Past* (LTL+P). Then, we adapt it to TPTL and $\mathsf{TPTL_b}$+P, providing full proofs of the soundness and completeness of the resulting systems. In particular, completeness is proved by exploiting a novel model-theoretic argument that, compared to the one originally employed for the LTL system, provides a deeper understanding of the crucial role of the *prune* rule of the system.

1. Introduction

In the field of *formal verification* of software and hardware systems, temporal logics have become the de-facto standard specification languages. In particular, *Linear Temporal Logic* (LTL) [21] is one of the most commonly used logics in the field, together with its branching-time counterpart, the *Computation Tree Logic* (CTL) [7]. Many extensions and variants of these logics have been studied over the years, including, in particular, *real-time* logics such as *Metric Temporal Logic* (MTL) [2] and *Timed Propositional Temporal Logic* (TPTL) [3, 4], which is the main subject of this paper.

Over the years, the *model checking* problem, *i.e.*, deciding if a system satisfies a specification, has been thoroughly studied for these and other logics [8]. However, for the model checking procedures to be effective, the specifications themselves must be written with care. For example, checking a system against a valid specification (*i.e.*, a formula which is trivially true in every structure), is useless at the least, and could be severely harmful at worst: the positive answer from the model checking procedure may convince the designers of the system of its safety while potentially life-threatening bugs may instead be present. For this reason, a growing importance is being given to the *sanity checking* of specifications, which is an application of the *satisfiability checking* problem, that is, deciding whether a model satisfying a given formula exists at all. More generally, the satisfiability problem of a temporal formula plays a central role in *property-based design*, since it can be used to check the consistency of the specifications (in the early phases of the design), or whether a set of requirements implies some other set of desired behaviours [20].

Besides such applications, the complexity of the satisfiability problem, and consequently the complexity of *reasoning*, is among the first theoretical questions that are often answered for a given logic, and indeed the LTL satisfiability problem has been thoroughly explored in the literature. Among the many different kinds of techniques proposed, *tableau methods* were among the first to be investigated [27]. Originally devised for propositional logic [6] and later adapted to many other logics during the last half of the century [9], tableau-based techniques provide a useful theoretical tool to reason about the proof theory of the considered logic, as they are tightly related to cut-free Gentzen-style sequent calculi for the given logic.

Often – this is the case for LTL – tableau methods provide the easiest to understand decision procedures for the logic. In the case of linear-time logics, most tableau methods are *graph-shaped*, as they build a graph structure which is then traversed to look for a model for the formula. This is the case for most known tableau methods for LTL [15] and for TPTL [4]. However, despite being valuable theoretical tools for the study of the logic, these systems are hardly applicable in practice, due to the need of building the huge graph structure, that even for simple formulae can have significant size. To overcome this limitation, some authors have proposed incremental [13, 14] and one-pass [24] alternatives, and, recently, a purely tree-shaped and one-pass tableau method for LTL was provided by Reynolds [23]. Thanks to its simple rule-based tree-structure, Reynolds' tableau proved to be efficiently implementable [5] and easily parallelisable [19].

In many contexts, depending on the specifications that have to be expressed, the use of *past modalities* can be needed or convenient [16, 22]. In the case of LTL, the addition of past modalities allows some specifications to be expressed exponentially more succinctly [18], but it affects neither the expressive power nor the computational complexity of the logic [15]. In contrast, adding past modalities to TPTL causes a great blowup of its computational complexity, since the satisfiability checking problem for TPTL is EXPSPACE-complete, while it becomes *non-elementary* for TPTL+P [3]. For this reason, *Bounded* TPTL *with Past* ($\mathsf{TPTL_b}$+P) was recently introduced [10], with the purpose of giving a logical characterisation of timeline-based planning problems. $\mathsf{TPTL_b}$+P enables the use of past operators, while restricting their syntax in a way that keeps the complexity under control, since its satisfiability checking problem is still EXPSPACE-complete [10]. Because of these features, $\mathsf{TPTL_b}$+P is worth studying independently from its original motivation. As usual, the complexity of the $\mathsf{TPTL_b}$+P satisfiability problem was proved by providing a graph-shaped tableau method.

In this paper, we adapt Reynolds' tableau for LTL to provide the first one-pass tree-shaped tableau system for TPTL of $\mathsf{TPTL_b}$+P. Two features have to be handled with respect to LTL: the support for past operators, and the *freeze quantifier* provided by TPTL and $\mathsf{TPTL_b}$+P to support real-time specifications. We first address the issue of past modalities, by providing the first one-pass tree-shaped tableau for LTL+P. The proofs of soundness and completeness of the system are provided in full details, and are based on a novel model-theoretic argument, based on the concept of *greedy pre-models*, which is simpler and more insightful with respect to the argument employed for the original LTL system [23]. Then, we provide the tableau system for TPTL and $\mathsf{TPTL_b}$+P. To do this, we first describe the common structure of both, specialising it to each logic later. To describe the tableau system for $\mathsf{TPTL_b}$+P, we better characterise the semantics of the logic by identifying it as a purely syntactic fragment of the full TPTL+P logic, and providing a translation that embeds the former into the latter. Full proofs of soundness and completeness for the TPTL and $\mathsf{TPTL_b}$+P systems are provided, exploiting the same novel model-theoretic proof technique used above. In total, we extend the original one-pass and tree-shaped tableau system for LTL in three directions, giving evidence to its great extensibility, and improving such an extensibility by providing an easier-to-adapt proof technique.

The paper is organised as follows. Section 2 recalls syntax and semantics of LTL, LTL+P, TPTL, and $\mathsf{TPTL_b}$+P. Moreover, it briefly recaps the main features of existing graph-shaped tableau systems. Then, Section 3 describes the one-pass and tree-shaped tableau system for LTL+P, extending Reynolds' tableau for LTL. Soundness and completeness proofs are given in Section 4. Next, Section 5 adapts such a tableau system to TPTL and $\mathsf{TPTL_b}$+P, and it proves soundness and completeness of the resulting systems. Finally, Section 6 reports some conclusive remarks and highlights future research directions.

2. Preliminaries

This section basically introduces syntax and semantics of the logics studied throughout the paper. Let us first introduce a few elements of notation. In what follows, we denote as $\mathbb{N}$ and $\mathbb{Z}$ the sets of, respectively, natural numbers and integers. Sequences (either finite or infinite) of elements $x_0, x_1, \ldots$ are denoted as $\overline{x} = \langle x_0, x_1, \ldots \rangle$. Given a sequence $\overline{x} = \langle x_0, x_1, \ldots \rangle$ and an index $k \in \mathbb{N}$, the sequence $\overline{x}_{\geq k} = \langle x_k, x_{k+1}, \ldots \rangle$ is the suffix of $\overline{x}$ starting from the kth element. Analogously we define $\overline{x}_{>k}$, $\overline{x}_{\leq k}$ and $\overline{x}_{<k}$. Given $i, j \in \mathbb{N}$, the subsequence between the ith and jth elements is denoted as $\overline{x}_{[i \ldots j]} = \langle x_i, \ldots, x_j \rangle$. Given two sequences $\overline{x} = \langle x_0, \ldots, x_n \rangle$ and $\overline{y} = \langle y_0, y_1, \ldots \rangle$, with $\overline{x}$ finite and $\overline{y}$ either finite or infinite, we denote as $\overline{xy} = \langle x_0, \ldots, x_k, y_0, y_1, \ldots \rangle$ the juxtaposition of $\overline{x}$ and $\overline{y}$. Let us now start by recalling syntax and semantics of LTL.

2.1. Linear Temporal Logic

Linear Temporal Logic (LTL) is a propositional temporal logic interpreted over infinite, discrete linear orders. LTL *with Past* ($\mathsf{LTL{+}P}$) extends LTL with the addition of temporal operators able to talk about what happened in the *past* with respect to the current time. We will now briefly recall the syntax and semantics of $\mathsf{LTL{+}P}$, which encompasses that of LTL as well.

Syntactically, LTL can be seen as an extension of propositional logic with the addition of the *tomorrow* ($\mathsf{X}\phi$, *i.e.*, at the *next* state ϕ holds) and *until* ($\phi_1 \,\mathcal{U}\, \phi_2$, *i.e.*, ϕ_2 will eventually hold and ϕ_1 will hold *until* then) temporal operators. $\mathsf{LTL{+}P}$ is obtained from LTL by adding the following *past* temporal operators: (i) the *yesterday* operator ($\mathsf{Y}\phi$, *i.e.*, there exists a *previous* state in which ϕ holds); (ii) the Z operator ($\mathsf{Z}\phi$, *i.e.*, either a previous state does not exist or in the previous state ϕ holds); (iii) and the *since* operator ($\phi_1 \mathcal{S}\, \phi_2$, *i.e.*, there was a past state where ϕ_2 held, and ϕ_1 has held *since* then). Formally, given a set Σ of proposition letters, $\mathsf{LTL{+}P}$ formulae over Σ are generated by the following grammar:

$$\begin{aligned} \phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 & \qquad \text{propositional connectives} \\ \mid \mathsf{X}\phi_1 \mid \phi_1 \,\mathcal{U}\, \phi_2 \mid \phi_1 \,\mathcal{R}\, \phi_2 \mid \mathsf{F}\phi_1 \mid \mathsf{G}\phi_1 & \qquad \textit{future} \text{ temporal operators} \\ \mid \mathsf{Y}\phi_1 \mid \phi_1 \,\mathcal{S}\, \phi_2 \mid \phi_1 \,\mathcal{T}\, \phi_2 \mid \mathsf{P}\phi_1 \mid \mathsf{H}\phi_1 \mid \mathsf{Z}\phi_1 & \qquad \textit{past} \text{ temporal operators} \end{aligned}$$

where $p \in \Sigma$ and ϕ_1 and ϕ_2 are $\mathsf{LTL{+}P}$ formulae. Most of the temporal operators of the language can be defined in terms of a small number of basic ones. In particular, the *release* ($\phi_1 \mathcal{R} \phi_2 \equiv \neg(\neg\phi_1 \mathcal{U} \neg\phi_2)$), *eventually* ($\mathsf{F}\phi \equiv \top \mathcal{U} \phi$), and *always* ($\mathsf{G}\phi \equiv \neg\mathsf{F}\neg\phi$) future operators can all be defined in terms of the *until* operator, while the *triggered* ($\phi_1 \,\mathcal{T}\, \phi_2 \equiv \neg(\neg\phi_1 \,\mathcal{S}\, \neg\phi_2)$), *once* ($\mathsf{P}\phi \equiv \top \,\mathcal{S}\, \phi$), and *historically* ($\mathsf{H}\phi \equiv \neg\mathsf{P}\neg\phi$) past operators can all be defined in terms of the *since* operator. However, in our setting, it is useful to consider them as primitive. Let the *nesting degree* $\deg(\psi)$ of ψ be defined inductively as $\deg(p) = \deg(\neg p) = 0$ for $p \in \Sigma$, $\deg(\mathsf{X}\psi) = \deg(\mathsf{Y}\psi) = \deg(\mathsf{Z}\psi) = \deg(\psi){+}1$, and $\deg(\phi_1 \circ \phi_2) = \max(\deg(\psi_1), \deg(\psi_2)) + 1$, with $\circ \in \{\wedge, \vee, \mathcal{U}, \mathcal{S}, \mathcal{R}, \mathcal{T}\}$.

A *model* for an $\mathsf{LTL{+}P}$ formula ϕ is an infinite sequence of sets of proposition letters, *i.e.*, $\overline{\sigma} = \langle \sigma_0, \sigma_1, \ldots \rangle$, where each *state* $\sigma_i \subseteq \Sigma$ represents the proposition letters that hold in the state. Given a model $\overline{\sigma}$, a position $i \geq 0$, and an $\mathsf{LTL{+}P}$ formula ϕ, we inductively define the *satisfaction* of ϕ by $\overline{\sigma}$ at position i, written as $\overline{\sigma}, i \models \phi$, as follows:

1. $\overline{\sigma}, i \models p$ iff $p \in \sigma_i$;

2. $\overline{\sigma}, i \models \neg\phi$ iff $\overline{\sigma}, i \not\models \phi$;
3. $\overline{\sigma}, i \models \phi_1 \vee \phi_2$ iff $\overline{\sigma}, i \models \phi_1$ or $\overline{\sigma}, i \models \phi_2$;
4. $\overline{\sigma}, i \models \phi_1 \wedge \phi_2$ iff $\overline{\sigma}, i \models \phi_1$ and $\overline{\sigma}, i \models \phi_2$;
5. $\overline{\sigma}, i \models \mathsf{X}\phi$ iff $\overline{\sigma}, i+1 \models \phi$;
6. $\overline{\sigma}, i \models \mathsf{Y}\phi$ iff $i > 0$ and $\overline{\sigma}, i-1 \models \phi$;
7. $\overline{\sigma}, i \models \mathsf{Z}\phi$ iff either $i = 0$ or $\overline{\sigma}, i-1 \models \phi$;
8. $\overline{\sigma}, i \models \phi_1 \,\mathcal{U}\, \phi_2$ iff there exists $j \geq i$ such that $\overline{\sigma}, j \models \phi_2$, and $\overline{\sigma}, k \models \phi_1$ for all k, with $i \leq k < j$;
9. $\overline{\sigma}, i \models \phi_1 \,\mathcal{S}\, \phi_2$ iff there exists $j \leq i$ such that $\overline{\sigma}, j \models \phi_2$, and $\overline{\sigma}, k \models \phi_1$ for all k, with $j < k \leq i$;
10. $\overline{\sigma}, i \models \phi_1 \,\mathcal{R}\, \phi_2$ iff either $\overline{\sigma}, j \models \phi_2$ for all $j \geq i$, or there exists $k \geq i$ such that $\overline{\sigma}, k \models \phi_1$ and $\overline{\sigma}, j \models \phi_2$ for all $i \leq j \leq k$;
11. $\overline{\sigma}, i \models \phi_1 \,\mathcal{T}\, \phi_2$ iff either $\overline{\sigma}, j \models \phi_2$ for all $0 \leq j \leq i$, or there exists $k \leq i$ such that $\overline{\sigma}, k \models \phi_1$ and $\overline{\sigma}, j \models \phi_2$ for all $i \geq j \geq k$

We say that $\overline{\sigma}$ *satisfies* ϕ, $\overline{\sigma} \models \phi$, if it satisfies the formula at the first state, *i.e.*, if $\overline{\sigma}, 0 \models \phi$. We say that two formulae ϕ and ψ are *equivalent* ($\phi \equiv \psi$) if and only if they are satisfied by the same set of models.

A *literal* ℓ is a formula of the form p or $\neg p$, where $p \in \Sigma$. Since we defined the *release* and *triggered* operators as primitive instead of defining them in terms of the *until* and *since* operators, any LTL+P formula can be turned into its *negated normal form*: a formula ϕ can be translated into an equivalent one $\mathrm{nnf}(\phi)$ where *negations* appear only in literals, by exploiting the duality between the *until/release* and *since/triggered* pairs of operators, and classic *De Morgan* laws and the *until*/*release* and *since*/*triggered* duality (the *tomorrow* operator is its own dual, while $\neg\mathsf{Y}\phi$ is equivalent to $\mathsf{Z}\neg\phi$). In the whole paper, we will assume *w.l.o.g.* that all the considered formulae are in negated normal form.

The satisfiability problem for LTL and LTL+P are known to be PSPACE-complete [15, 25], *i.e.*, past modalities can be added to LTL at no cost, although they allow us to express some kind of specifications exponentially more succinct [18].

Finally, let us define a concept that comes useful in the definition of the tableau systems that follows. The *closure* of a formula ϕ, defined below, can be thought of as the set of formulas that is sufficient to consider in order to reason about the satisfaction of ϕ.

Definition 1 (Closure of an $\mathsf{LTL{+}P}$ formula). Let ϕ be an $\mathsf{LTL{+}P}$ formula. The *closure* of ϕ is the set $\mathcal{C}(\phi)$ of formulae defined as follows:

1. $\phi \in \mathcal{C}(\phi)$;
2. for every subformula ϕ' of ϕ, $\phi' \in \mathcal{C}(\phi)$;
3. for every $p \in \Sigma$, $p \in \mathcal{C}(\phi)$ if and only if $\neg p \in \mathcal{C}(\phi)$;
4. if $\phi_1 \circ \phi_2 \in \mathcal{C}(\phi)$, for $\circ \in \{\mathcal{U}, \mathcal{R}\}$, then $\mathsf{X}(\phi_1 \circ \phi_2) \in \mathcal{C}(\phi)$;
5. if $\phi_1 \circ \phi_2 \in \mathcal{C}(\phi)$, for $\circ \in \{\mathcal{S}, \mathcal{T}\}$, then $\mathsf{Y}(\phi_1 \circ \phi_2) \in \mathcal{C}(\phi)$.

2.2. Timed Propositional Temporal Logic

Timed Propositional Temporal Logic (TPTL) [3, 4] is a *timed* extension of LTL originally introduced for the formal verification of real-time systems. TPTL augments LTL with the addition of a *freeze quantifier* $x.\phi$, which binds the variable x with the *timestamp* of the current state, and with *timing constraints*, such as $x \leq y + 5$, which allow one to compare the timestamp of different states where the corresponding freeze quantifiers were previously interpreted. As an initial example, one can express the requirement that the response (r) to an alarm (a) has to be triggered at most ten time steps after the alarm, in the following way:

$$\mathsf{G}x.(a \rightarrow \mathsf{F}y.(r \wedge y \leq x + 10))$$

TPTL *with Past* (TPTL+P) is the extension of TPTL with past modalities. As we did for LTL and LTL+P, we give the syntax and semantics of TPTL+P, understanding that TPTL is the fragment of TPTL+P without past operators. Hence, let Σ be a set of *proposition letters* and $\mathcal{V}$ be a set of *variables.* A TPTL+P formula ϕ over Σ and $\mathcal{V}$ is recursively defined as follows:

$$\begin{aligned} \phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 & \qquad \text{Boolean connectives} \\ \mid \mathsf{X}\phi_1 \mid \phi_1 \,\mathcal{U}\, \phi_2 \mid \phi_1 \,\mathcal{R}\, \phi_2 & \qquad \text{future temporal operators} \\ \mid \mathsf{Y}\phi_1 \mid \phi_1 \,\mathcal{S}\, \phi_2 \mid \phi_1 \,\mathcal{T}\, \phi_2 \mid \mathsf{Z}\phi_1 & \qquad \text{past temporal operators} \\ \mid x.\phi_1 \mid x \leq y + c \mid x \leq c & \qquad \text{freeze quantifier and timing constraints} \end{aligned}$$

where $p \in \Sigma$, ϕ_1, ϕ_2 are TPTL+P formulae, $x, y \in \mathcal{V}$, $c \in \mathbb{Z}$. Formulae of the form $x.\phi$ are called *freeze quantifications*, while those of the forms $x \leq y + c$ and $x \leq c$ are called *timing constraints*. A formula ϕ is *closed* if any variable x occurring inside ϕ occurs inside a freeze quantification $x.\psi$. TPTL is the fragment of TPTL+P obtained by removing the past operators.

One can note how TPTL+P syntax directly extends that of LTL+P, with the addition of the freeze quantifier and the timing constraints. Similar to the LTL/LTL+P case, we prefer to consider the *release* and *triggered* operators as primitive so that each formula can have a corresponding *negated normal form*, even though they can be defined in terms of the respective dual operators as $\psi_1 \mathcal{R} \psi_2 \equiv \neg(\neg\psi_1 \mathcal{U} \neg\psi_2)$ and $\psi_1 \mathcal{T} \psi_2 \equiv \neg(\neg\psi_1 \mathcal{S} \neg\psi_2)$. Moreover, similar to LTL+P, standard logical and temporal shortcuts are used, such as $\top$ for $p \vee \neg p$, for some $p \in \Sigma$, $\bot$ for $\neg\top$, $\phi_1 \wedge \phi_2$ for $\neg(\neg\phi_1 \vee \neg\phi_2)$, $\mathsf{F}\phi$ for $\top \mathcal{U} \phi$, $\mathsf{G}\phi$ for $\neg\mathsf{F}\neg\phi$, and $\mathsf{P}\phi$ for $\top \mathcal{S} \phi$, as well as constraint shortcuts, such as, *e.g.*, $x \leq y$ for $x \leq y + 0$, $x > y$ for $\neg(x \leq y)$, $x = y$ for $x \leq y \wedge y \leq x$, and others.

TPTL+P formulae are interpreted over *timed state sequences*, *i.e.*, structures of the form $\rho = (\overline{\sigma}, \overline{\tau})$, where $\overline{\sigma} = \langle \sigma_0, \sigma_1, \ldots \rangle$ is an infinite sequence of states $\sigma_i \in 2^{\Sigma}$, with $i \geq 0$, and $\overline{\tau} = \langle \tau_0, \tau_1, \ldots \rangle$ is an infinite sequence of *timestamps* $\tau_i \in \mathbb{N}$, with $i \geq 0$, such that: (1) $\tau_{i+1} \geq \tau_i$ (*monotonicity*), and (2) for all $t \in \mathbb{N}$, there is some $i \geq 0$ such that $\tau_i \geq t$ (*progress*). Formally, the satisfaction of TPTL+P over timed state sequences is defined as follows. An *environment* is a function $\xi : \mathcal{V} \to \mathbb{N}$ that interprets a given variable as a timestamp. A timed state sequence $\rho = (\overline{\sigma}, \overline{\tau})$ satisfies a TPTL+P formula ϕ at position $i \geq 0$, under the environment ξ, written $\rho, i \models_\xi \phi$, if the following conditions are met:

1. $\rho, i \models_\xi p$ iff $p \in \sigma_i$;
2. $\rho, i \models_\xi \phi_1 \vee \phi_2$ iff either $\rho, i \models_\xi \phi_1$ or $\rho, i \models_\xi \phi_2$;
3. $\rho, i \models_\xi \neg\phi_1$ iff $\rho, i \not\models_\xi \phi_1$;
4. $\rho, i \models_\xi x \leq y + c$ iff $\xi(x) \leq \xi(y) + c$;
5. $\rho, i \models_\xi x \leq c$ iff $\xi(x) \leq c$;
6. $\rho, i \models_\xi x.\phi_1$ iff $\rho, i \models_{\xi'} \phi_1$ where $\xi' = \xi[x \leftarrow \tau_i]$;
7. $\rho, i \models_\xi \mathsf{X}\phi_1$ iff $\rho, i + 1 \models_\xi \phi_1$;
8. $\rho, i \models_\xi \phi_1 \mathcal{U} \phi_2$ iff there exists $j \geq i$ such that (a) $\rho, j \models_\xi \phi_2$, and (b) $\rho, k \models_\xi \phi_1$ for all k such that $i \leq k < j$;

9. $\rho, i \models_\xi \phi_1 \,\mathcal{R}\, \phi_2$ iff either $\rho, j \models_\xi \phi_2$ for all $j \geq i$, or there is a $k \geq i$ with $\rho, k \models_\xi \phi_1$ and $\rho, j \models_\xi \phi_2$ for all $i \leq j \leq k$;
10. $\rho, i \models_\xi \mathsf{Y}\phi_1$ iff $i > 0$ and $\rho, i-1 \models_\xi \phi_1$;
11. $\rho, i \models_\xi \mathsf{Z}\phi_1$ iff either $i = 0$ or $\rho, i-1 \models_\xi \phi_1$;
12. $\rho, i \models_\xi \phi_1 \,\mathcal{S}\, \phi_2$ iff there exists $j \leq i$ such that (a) $\rho, j \models_\xi \phi_2$, and (b) $\rho, k \models_\xi \phi_1$ for all k such that $j < k \leq i$;
13. $\rho, i \models_\xi \phi_1 \,\mathcal{T}\, \phi_2$ iff either $\rho, j \models_\xi \phi_2$ for all $0 \leq j \leq i$, or there is a $k \leq i$ with $\rho, k \models_\xi \phi_1$ and $\rho, j \models_\xi \phi_2$ for all $i \geq j \geq k$.

The satisfiability problem for TPTL is EXPSPACE-complete [4]. However, in contrast to what happens for LTL/LTL+P, where past modalities do not increase the computational complexity, the satisfiability problem for TPTL+P is *non-elementary* [3]. In order to prove this result, Alur and Henzinger first define the logic $\mathsf{TPTL}_\exists$, which augments TPTL with the existential quantifier over time variables, and show that, despite being as expressive as TPTL, it is non-elementarily decidable. The proof reduces the validity problem of the monadic first-order theory of $(\mathbb{N}, \leq)$ (from now on S1S[FO]) to the validity problem $\mathsf{TPTL}_\exists$; the idea is to simulate the structure $(\mathbb{N}, \leq)$ in a time sequence by using the following formula:

$$\phi_{+1} \coloneqq \mathsf{G}x.\mathsf{X}y.(y = x + 1)$$

Now, given a formula ϕ of S1S[FO], we can build the $\mathsf{TPTL}_\exists$ formula $\psi \coloneqq \phi_{+1} \to \phi'$, where ϕ' is obtained from ϕ by replacing every atomic proposition $p(i)$ with $\mathsf{F}x.(x = i \land p)$: intuitively, the existential quantifiers of ϕ remain untouched and become quantifiers over time variables in ϕ', and the atomic propositions $p(i)$ in ϕ require a future operator to place the predicate p at the correct state of the time sequence. It holds that ϕ is valid if and only if ψ is valid. Since the lower bound for the validity problem of S1S[FO] is non-elementary [26], also $\mathsf{TPTL}_\exists$ is non-elementarily decidable. In order to show also the non-elementary complexity of TPTL+P [3], we can proceed as for $\mathsf{TPTL}_\exists$, with the additional step of replacing every subformula of type $\exists x.\phi'$ with $y.\mathsf{F}x.\mathsf{P}z.(y = z \land \phi')$: this mechanism thus requires two capabilities to work properly, that is to move forth and back *arbitrarily far*, and to use the freeze quantifier to refer to the timestamps of states found in that way. As before, we obtain an equisatisfiable formula, having that also TPTL+P is non-elementarily decidable.

2.3. Bounded Timed Propositional Temporal Logic with Past

Bounded Timed Propositional Temporal Logic with Past ($\mathsf{TPTL_b{+}P}$) exploits the above observation to forbid those kinds of constructs. This is done by limiting the use of freeze quantifiers only to formulae that can guarantee a *bound* on the distance between different quantified variables. Intuitively, each temporal operator such as $X\phi$ is replaced by a *bounded* version $\mathsf{X}_w\phi$, with $w \in \mathbb{N} \cup \{+\infty\}$, such that $\mathsf{X}_w\phi$ holds if the *timestamp* of the next state differs by at most w from the current one, and ϕ holds there. Hence, $\mathsf{X}_{+\infty}\phi$ is equivalent to $\mathsf{X}\phi$ from $\mathsf{TPTL{+}P}$, but we impose an additional restriction that w can be infinite only if ϕ is a *closed* formula, *i.e.*, we can look *arbitrarily far* in the model only if the formula is not able to refer to timestamps quantified outside of it.

More formally, a $\mathsf{TPTL_b{+}P}$ formula ϕ is recursively defined as follows:

$$\begin{aligned}
\phi := p \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 & \qquad \text{Boolean connectives}\\
\mid \mathsf{X}_w\phi_1 \mid \widetilde{\mathsf{X}}_w\phi_1 \mid \phi_1 \,\mathcal{U}_w\phi_2 \mid \phi_1 \,\mathcal{R}_w\phi_2 & \qquad \text{future temporal operators}\\
\mid \mathsf{Y}_w\phi_1 \mid \widetilde{\mathsf{Y}}_w\phi_1 \mid \phi_1 \,\mathcal{S}_w\phi_2 \mid \phi_1 \,\mathcal{T}_w\phi_2 & \qquad \text{past temporal operators}\\
\mid x.\phi_1 \mid x \le y + c \mid x \le c & \qquad \text{freeze quantifier and constraints}
\end{aligned}$$

where $w \in \mathbb{N} \cup \{+\infty\}$, and in any temporal operator, if the enclosed formulae are not closed then $w \neq +\infty$. Missing subscripts are understood as $w = +\infty$.

The semantics of $\mathsf{TPTL_b{+}P}$ is defined as follows. A $\mathsf{TPTL_b{+}P}$ formula ϕ is satisfied by a timed state sequence ρ at position i with environment ξ if the following conditions are met:

1. same as $\mathsf{TPTL{+}P}$ for all kind of formulae not explicitly considered;
2. $\rho, i \models_\xi \mathsf{X}_w\phi_1$ iff $\tau_{i+1} - \tau_i \le w$ and $\rho, i+1 \models_\xi \phi_1$;
3. $\rho, i \models_\xi \widetilde{\mathsf{X}}_w\phi_1$ iff $\tau_{i+1} - \tau_i \le w$ implies $\rho, i+1 \models_\xi \phi_1$;
4. $\rho, i \models_\xi \phi_1 \,\mathcal{U}_w\phi_2$ iff there exists $j \ge i$ such that:
   (a) $\tau_j - \tau_i \le w$;
   (b) $\rho, j \models_\xi \phi_2$;
   (c) $\rho, k \models_\xi \phi_1$ for all k such that $i \le k < j$;
5. $\rho, i \models_\xi \phi_1 \,\mathcal{R}_w\phi_2$ iff either (a) $\tau_j - \tau_i \le w$ implies $\rho, j \models_\xi \phi_2$ for all $j \ge i$, or (b) there is a $k \ge i$ such that $\tau_k - \tau_i \le w$ and $\rho, k \models_\xi \phi_1$, and $\rho, j \models_\xi \phi_2$ for all $i \le j \le k$;
6. $\rho, i \models_\xi \mathsf{Y}_w\phi_1$ iff $i > 0$, $\tau_i - \tau_{i-1} \le w$, and $\rho, i-1 \models_\xi \phi_1$;
7. $\rho, i \models_\xi \widetilde{\mathsf{Y}}_w\phi_1$ iff $i > 0$ and $\tau_i - \tau_{i-1} \le w$ imply $\rho, i-1 \models_\xi \phi_1$;

8. $\rho, i \models_\xi \phi_1 \, \mathcal{S}_w \phi_2$ iff there exists $j \leq i$ such that:
(a) $\tau_i - \tau_j \leq w$;
(b) $\rho, j \models_\xi \phi_2$;
(c) $\rho, k \models_\xi \phi_1$ for all k such that $j < k \leq i$;
9. $\rho, i \models_\xi \phi_1 \, \mathcal{T}_w \phi_2$ iff either (a) $\tau_i - \tau_j \leq w$ implies $\rho, j \models_\xi \phi_2$ for all $j \leq i$, or (b) there is a $k \leq i$ such that $\tau_i - \tau_k \leq w$ and $\rho, k \models_\xi \phi_1$, and $\rho, j \models_\xi \phi_2$ for all $i \geq j \geq k$.

Intuitively, each temporal operator from TPTL+P is given a bound, limiting the time distance between the current state and the one where the enclosed formula is interpreted. This is similar to temporal operators found in MTL. In addition to MTL, however, $\mathsf{TPTL_b}$+P supports the freeze quantifier, to relate the timestamps of more than two states together, as in the following example:

$$\mathsf{G}x.(p \rightarrow \mathsf{P}_5 y.(q \wedge \mathsf{F}_{10} z.(r \wedge x \leq y + 5 \wedge z \leq x + 5 \wedge z \leq y + 7)))$$

The above formula constrains each occurrence of p to trigger an occurrence of q and r, with a triangular set of constraints on the time between p and q, q and r and p and r. As required by the syntax, each temporal operator applied to a formula with free variables is given a finite bound. Only the outermost *globally* operator can be used with an infinite bound since the underlying formula is closed. This restriction, as we will see, is what allows us to keep the complexity of the logic under control.

A noteworthy addition with respect to TPTL+P is the *weak* version of the *tomorrow* and *yesterday* operators, $\widetilde{\mathsf{X}}_w\psi$ and $\widetilde{\mathsf{Y}}_w\psi$. While $\mathsf{X}_w\psi$ mandates the next state to satisfy ψ and have a timestamp less than w greater than the current one, the weak version $\widetilde{\mathsf{X}}_w\psi$ states that *if* the timestamp bound is satisfied, then ψ has to hold. The *tomorrow* and *weak tomorrow* operators are negated duals, since $\mathsf{X}_w\psi \equiv \neg\widetilde{\mathsf{X}}_w\neg\psi$, and the same applies to the corresponding past operators, so this allows us to have a *negated normal form* for $\mathsf{TPTL_b}$+P formulae as well.

2.4. A discussion of graph-shaped tableau systems

In order to highlight the contributions of this paper with respect to previous work on the matter, it is useful to recall some basic elements of existing tableau methods for LTL and TPTL. Such a summary will ease the comparison with the methods provided in the following sections. Common tableau methods for LTL, such as those described by Lichtenstein and Pnueli

[15], Wolper [28], and Manna and Pnueli [17], are *graph-shaped*. These systems work by building a graph structure whose paths represent a set of possible models of the formula. An actual model can then be possibly found by looking for a path with particular properties.

More in detail, nodes of the tableau for ϕ are *maximal consistent sets* of formulas Γ taken from the *closure* of ϕ ($\Gamma \subseteq \mathcal{C}(\phi)$). Two nodes Γ and Γ' of the tableau are connected by an edge if and only if the *tomorrow* and *yesterday* formulas are respected, *i.e.*, $\mathsf{X}\psi \in \Gamma$ if and only if $\psi \in \Gamma'$ and $\mathsf{Y}\psi \in \Gamma'$ if and only if $\psi \in \Gamma$. In this way, each node represents the formulas holding in a possible state of the system described by the formula, and paths represent state sequences where each transition is locally consistent with the formula. However, due to formulas like $\alpha \,\mathcal{U}\, \beta$, local consistency of state transitions is not enough to obtain a model for the formula. In addition, the path has to fulfil such requests. In order to find such a *fulfilling* path, the graph can be decomposed into its strongly connected components, looking for a component C, reachable from the initial state, where for all the formulas $\alpha \,\mathcal{U}\, \beta$ contained in some node $\Gamma \in C$, there is a node $\Gamma' \in C$ with $\beta \in \Gamma'$. This is enough to ensure the existence of a fulfilling path.

These systems are often referred to as *two-pass* tableaux [23, 24], since the graph structure has to be built in a first pass, before the search for a model can begin. *Incremental* versions of such systems have been proposed in the literature [11, 14], which only build the portion of the graph reachable from the initial state, producing the nodes as needed. Such an incremental construction can reduce the size of the constructed graph structure, but the analysis of the strongly connected components has still to be performed afterwards to look for a fulfilling path.

In contrast, the tree-shaped tableau method developed by Reynolds [23], that here we extend to $\mathsf{LTL}{+}\mathsf{P}$, TPTL, and $\mathsf{TPTL_b}{+}\mathsf{P}$, builds a tree structure, where each accepted branch corresponds to a model of the formula. The construction of branches proceeds independently from each other, and only the branch currently being constructed has to be kept in memory at each single point in time. Accepting branches are identified directly, without any analysis involving the whole tree, which is never built entirely. Moreover, a model of the formula can be directly extracted from an accepted branch.

Another one-pass tableau method was previously proposed by Schwendimann [24], to avoid the construction of the graph structure. It bears many similarities with Reynolds' system, since it builds a tree structure. However, in such a system the construction of branches cannot proceed independently, and the acceptance condition applies to subtrees, not to single branches. Moreover, if one is interested in the construction of a model besides merely deciding its existence, multiple branches have to be kept in memory.

Even though all these systems still need an exponential amount of space in the worst case, their behaviour differs significantly in practice. Experimental evaluations have been performed on an implementation of Reynolds' tableau [5], comparing it with other implementations of graph-shaped tableau and of the one-pass tableau by Schwendimann [24]. The results shows better performance and lower memory usage in many cases. Most importantly, however, the independent exploration of branches in Reynolds' tableau allows it to be very easily and efficiently *parallelised*, with impressive speedups [19].

The tree-shaped tableau systems for $\mathsf{LTL{+}P}$, TPTL and $\mathsf{TPTL_b{+}P}$ presented here maintain the key features of the original system, and even though a proper experimental evaluation is due, these results suggest that the tree-shaped approach is definitely worth exploring.

Regarding TPTL, the tableau system originally provided by Alur and Henzinger [4] follows the classical graph-shaped structure. In this case, however, the explicit handling of time requires a more involved construction. Nevertheless, their solution handles the *freeze quantifiers* without an explicit variable binding machinery thanks to the concept of *temporal shift operators*, that will be described (and generalised) in Section 5 as it also forms the basic building block of our tree-shaped tableau for TPTL and $\mathsf{TPTL_b{+}P}$. Before the present work, neither incremental nor tree-shaped versions of the graph-shaped tableau for TPTL were proposed in the literature.

3. The one-pass and tree-shaped tableau system for $\mathsf{LTL{+}P}$

As a preliminary step to the development of the one-pass and tree-shaped tableau systems for TPTL and $\mathsf{TPTL_b{+}P}$, we devise a tableau system for $\mathsf{LTL{+}P}$, in order to show how past modalities can be handled in a tree-shaped tableau. As the original description of the system was given only for LTL, this is a novel contribution by itself.[1]

In common graph-shaped tableau systems, adding past operators is just a matter of checking an additional constraint on edges between the nodes

[1] As a matter of fact, a one-pass and tree-shaped tableau system for $\mathsf{LTL{+}P}$ was presented in [12]. However, the formulation of the $\mathsf{YESTERDAY}$ given there was incorrect (and incomplete).

of the graph. On the other hand, it was *a priori* unclear how to handle the past in a tree-shaped tableau, where each branch is committed to its history. It turns out that this addition can be made in a very modular way, with the addition of a small number of new specific rules, without changing the pre-existent structure. The resulting system can be cut back to the original one by simply ignoring these rules, and, furthermore, running our system on a future-only formula leads to exactly the same computation as with the future-only system.

To describe the system, we first recall the original one-pass tree-shaped tableau for LTL [23]. Then, we show the changes required to support past modalities.

3.1. The tableau system for LTL

A tableau for ϕ is a tree T where each node u is labelled by a subset $\Gamma(u)$ of the closure $\mathcal{C}(\phi)$ and the label of the root node u_0 contains only ϕ, *i.e.*, $\Gamma(u_0) = \{\phi\}$. The tableau is built recursively from the root, at each step applying one from a set of *rules* to a leaf of the tree. Each rule can add one or two children to the current node, advancing the construction of the tree, or close the current branch by *accepting* (✓) or *rejecting* (✗) the current node. At the end of the section, the construction is proved to terminate with a *complete* tableau, *i.e.*, one where all the leaves are either ticked or crossed. Once a complete tableau is obtained, the formula is recognised as satisfiable if there is at least one accepted branch. Given two nodes u and v, we write $u \leq v$ to mean that u is an ancestor of v, and $u < v$ to mean that u is a *proper* ancestor of v, *i.e.*, $u \leq v$ and $u \neq v$.

The construction of a branch of the tree can be seen as the search for a model of the formula in a state-by-state way. At each step, *expansion rules* are applied first, building a possible assignment of proposition letters for the current state, which is then checked for the absence of contradictions. Next, the *termination rules* are checked to possibly detect if the construction can be stopped. At the end, information about the current state is used to determine the next one, and the construction proceeds by executing a temporal *step*.

The *expansion rules* look for a specific formula into the label, creating one or two children whose labels are obtained by replacing the target formula with some other simpler ones. Table 1 shows the expansion rules with the following notation: each rule looks for a formula $\phi \in \Gamma(u)$, where u is the current node, and two children u' and u'' of u are created with labels $\Gamma(u') = \Gamma(u) \setminus \{\phi\} \cup \Gamma_1(\phi)$ and $\Gamma(u'') = \Gamma(u) \setminus \{\phi\} \cup \Gamma_2(\phi)$, where $\Gamma_1(\phi)$ and $\Gamma_2(\phi)$ are defined in Table 1. If the corresponding expansion rule has an

Rule	$\phi \in \Gamma$	$\Gamma_1(\phi)$	$\Gamma_2(\phi)$
DISJUNCTION	$\alpha \vee \beta$	$\{\alpha\}$	$\{\beta\}$
UNTIL	$\alpha \mathcal{U} \beta$	$\{\beta\}$	$\{\alpha, \mathsf{X}(\alpha \mathcal{U} \beta)\}$
RELEASE	$\alpha \mathcal{R} \beta$	$\{\alpha, \beta\}$	$\{\beta, \mathsf{X}(\alpha \mathcal{R} \beta)\}$
EVENTUALLY	$\mathsf{F}\alpha$	$\{\alpha\}$	$\{\mathsf{XF}\alpha\}$
CONJUNCTION	$\alpha \wedge \beta$	$\{\alpha, \beta\}$	
ALWAYS	$\mathsf{G}\alpha$	$\{\alpha, \mathsf{XG}\alpha\}$	

Table 1: Tableau expansion rules. When a formula ϕ of one of the types shown in the table is found in the label Γ of a node u, one or two children u' and u'' are created with the same label as u excepting for ϕ which is replaced, respectively, by the formulae from $\Gamma_1(\phi)$ and $\Gamma_2(\phi)$.

empty $\Gamma_2(\phi)$, a single child is added. The rules for the primitive operators such as the *until* operator would be sufficient, but Table 1 shows as well some derived rules for operators such as the *eventually* and *always* operators, for ease of exposition. Note that a common feature of all the expansion rules is that the expanded formulae are entailed by their replacement, *i.e.*, both $\Gamma_1(\phi) \models \phi$ and $\Gamma_2(\phi) \models \phi$.

After a finite number of applications of expansion rules, the construction will eventually reach a node whose label contains only *elementary* formulae, *i.e.*, only formulae of the forms p or $\neg p$, for some $p \in \Sigma$, or $\mathsf{X}\alpha$ for some $\alpha \in \mathcal{C}(\phi)$ (while p and $\neg p$ cannot both belong to the same label, it can be the case that both $\mathsf{X}\alpha$ and $\mathsf{X}\neg\alpha$ belong to it, in which case the contradiction will be detected at a later stage). Such a label is called a *poised label* and the node is called a *poised node.* A *poised branch* $\overline{u} = \langle u_0, \ldots, u_n \rangle$ is a branch of the tableau where u_n is a poised node. Intuitively, a poised node represents a guess for the truth of elementary formulae at the current state. Once a poised node has been obtained, the search can proceed to the next state, by applying the STEP rule:

STEP Given a poised branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$, a child u_{n+1} is added to u_n, with $\Gamma(u_{n+1}) = \{\alpha \mid \mathsf{X}\alpha \in \Gamma(u_n)\}$.

It is worth to make a couple of observations at this point. First of all, note that the expansion rules can be seen as an alternative way to describe the rules used to build the nodes of common graph-shaped tableaux for LTL (*e.g.*, the system by Lichtenstein and Pnueli [15]). At the same time, note that the resulting labels are different from those nodes. In a graph-shaped tableau, nodes may contain a disjunction $\alpha \vee \beta$ and both its disjuncts, while in this system, one child of a node containing $\alpha \vee \beta$ contains α, and the other contains β, but there is no child necessarily containing both (unless both have been expanded by other means). During the construction, adding either formula to the opposite child would still result in locally consistent labels, but would be redundant, since once one of the two disjuncts has been chosen to hold, the truth of the other is irrelevant. A single node of the one-pass tree-shaped system can thus be viewed as a representative of a set of atoms from the graph-shaped tableau that ignores some formulae whose truth is irrelevant in that particular point of the model. Similarly, a branch of the tree can be seen as a set of paths of the graph-shaped tableau which differ only regarding the truth value of these irrelevant formulae.

Since the STEP rule represents an advancement in time, (some of the) poised nodes will be used later to extract a model of ϕ from a successful tableau branch. Before this can happen, the label of each poised node has to be checked for the absence of *contradictions*. If the check succeeds, then the STEP rule can be applied to advance to the next temporal state; otherwise, the branch is crossed.

CONTRADICTION If $\{p, \neg p\} \subseteq \Gamma(u_n)$, for some $p \in \Sigma$, then u_n is *crossed*.

The rules introduced so far allow us to build a tentative model for the formula step-by-step, but we introduced only rules that can reject wrong branches, thus we still need to specify how to recognise good branches corresponding to actual models. The first obvious case in which we have to stop the construction is when there is nothing left to do:

EMPTY Given a branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$, if $\Gamma(u_n) = \varnothing$, then u_n is *ticked*.

Since LTL models are in general infinite, only tableaux for simple formulae will have branches satisfying the EMPTY rule. Others, *e.g.*, $\mathsf{GF}p$, at any state require something to happen in the future. Thus, some criteria are needed to ensure that the construction can find models exhibiting infinitary behaviours, while guaranteeing that the expansion of every branch eventually terminates.

For this reason, the system includes a pair of *termination rules*, the LOOP rule and the PRUNE rule, which are checked at each poised node. Note that the potentially infinite expansion of the branches is caused by the recursive nature of most of the expansion rules. The UNTIL rule (and the derived EVENTUALLY rule) is the most critical one. In the expansion of a formula of the form $\alpha \mathcal{U} \beta$, the rule tries to either fulfil the request for β immediately or to postpone its fulfilment to a later step by adding $\mathsf{X}(\alpha\mathcal{U}\beta)$ to the label. A formula of the form $\mathsf{X}(\alpha\mathcal{U}\beta)$ is called X-*eventuality*. If an X-eventuality appears in a label, it means that some pending request still needs to be fulfilled in the future, and some criterion has to be used to avoid postponing its fulfilment indefinitely. The same arguments hold for formulae of the form $\mathsf{XF}\beta$, but since the EVENTUALLY rule can be derived from the UNTIL rule, we focus on the latter. Given a branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$, an X-eventuality of the form $\mathsf{X}(\alpha\mathcal{U}\beta)$ appearing in $\Gamma(u_i)$ for some $0 \leq i \leq n$ is said to be *requested* in u_i; moreover, we say that it is *fulfilled* in u_k, with $i < k$, if $\beta \in \Gamma(u_k)$ and $\alpha \in \Gamma(u_j)$ for all $i \leq j < k$. Moreover, we say that it is fulfilled in a subsequence $\overline{u}_{[i \ldots j]}$ if it is fulfilled in u_k for some $i < k \leq j$.

The LOOP and PRUNE rules, checked in this order, allow us to respectively handle the case of a branch that is repeating by successfully fulfilling recurrent requests and the case of a branch that is indefinitely postponing the fulfilment of an eventuality that is impossible to fulfil. They are defined as follows.

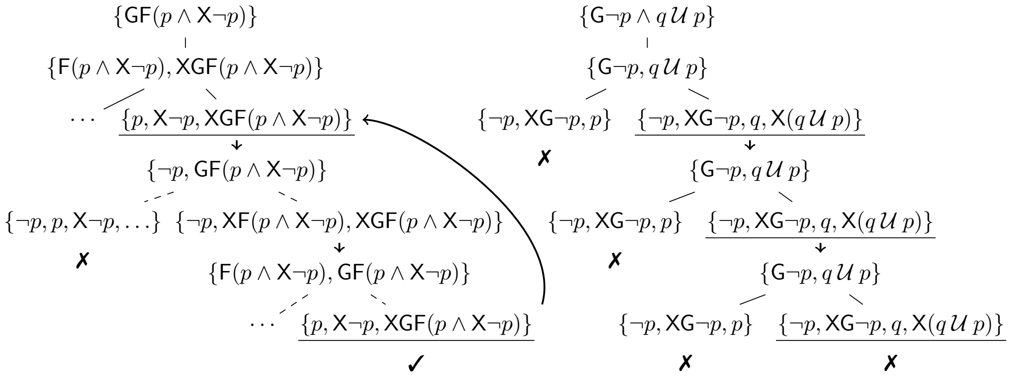
LOOP If there exists a poised node u_i such that $u_i < u_n$, $\Gamma(u_i) = \Gamma(u_n)$, and all the X-eventualities requested in u_i are fulfilled in $\overline{u}_{[i+1 \ldots n]}$, then u_n is *ticked*.

PRUNE If there exists two positions i and j such that $i < j \leq n$, $\Gamma(u_i) = \Gamma(u_j) = \Gamma(u_n)$, and among the X-eventualities requested in these nodes, all those fulfilled in $\overline{u}_{[j+1 \ldots n]}$ are fulfilled in $\overline{u}_{[i+1 \ldots j]}$ as well, then u_n is *crossed*.

Intuitively, the LOOP rule is responsible for recognising when a model for the formula has been found, while the PRUNE rule rejects branches that were doing redundant work without managing to fulfil any new eventuality. Then, if none of these two rules have closed the branch, the current state is ready and the construction can advance in time by applying the STEP rule. The PRUNE rule was the main novelty of this tableau system when introduced by Reynolds [23], and is the main focus of the novel completeness proof described in Section 4.

The rules described above are repeatedly applied to the leaves of any open branch until all branches have been either ticked or crossed. This process is guaranteed to terminate, as will be proved in Section 4.

As for the computational complexity of the procedure, it can be seen that the whole decision procedure runs using at most an exponential amount of


$\{\mathsf{GF}(p \wedge \mathsf{X}\neg p)\}$
$\{\mathsf{F}(p \wedge \mathsf{X}\neg p), \mathsf{XGF}(p \wedge \mathsf{X}\neg p)\}$
$\cdots$
$\{p, \mathsf{X}\neg p, \mathsf{XGF}(p \wedge \mathsf{X}\neg p)\}$
$\{\neg p, \mathsf{GF}(p \wedge \mathsf{X}\neg p)\}$
$\{\neg p, p, \mathsf{X}\neg p, \ldots\}$
$\{\neg p, \mathsf{XF}(p \wedge \mathsf{X}\neg p), \mathsf{XGF}(p \wedge \mathsf{X}\neg p)\}$
✗
$\{\mathsf{F}(p \wedge \mathsf{X}\neg p), \mathsf{GF}(p \wedge \mathsf{X}\neg p)\}$
$\cdots$
$\{p, \mathsf{X}\neg p, \mathsf{XGF}(p \wedge \mathsf{X}\neg p)\}$
✓
$\{\mathsf{G}\neg p \wedge q\,\mathcal{U}\,p\}$
$\{\mathsf{G}\neg p, q\,\mathcal{U}\,p\}$
$\{\neg p, \mathsf{XG}\neg p, p\}$
$\{\neg p, \mathsf{XG}\neg p, q, \mathsf{X}(q\,\mathcal{U}\,p)\}$
✗
$\{\mathsf{G}\neg p, q\,\mathcal{U}\,p\}$
$\{\neg p, \mathsf{XG}\neg p, p\}$
$\{\neg p, \mathsf{XG}\neg p, q, \mathsf{X}(q\,\mathcal{U}\,p)\}$
✗
$\{\mathsf{G}\neg p, q\,\mathcal{U}\,p\}$
$\{\neg p, \mathsf{XG}\neg p, p\}$
$\{\neg p, \mathsf{XG}\neg p, q, \mathsf{X}(q\,\mathcal{U}\,p)\}$
✗
✗

(a) Tableau for $\mathsf{GF}(p \wedge \mathsf{X}\neg p)$, closed by the LOOP rule.

(b) Tableau for $\mathsf{G}\neg p \wedge q\,\mathcal{U}\,p$, closed by the PRUNE rule.

Figure 1: Example tableaux for two formulae, involving the LOOP and PRUNE rules. Dashed edges represent subtrees collapsed to save space, bold arrows represent the application of a STEP rule to a poised label.

space: only a single branch at the time is needed to be kept in memory, and any branch cannot be longer than an exponential upper bound, given by the number of possible different labels that can appear on a branch before triggering either the LOOP or the PRUNE rule. The running time of the procedure is therefore at most doubly exponential in the size of the formula. The procedure is thus not optimal with regards to the complexity of LTL satisfiability, which is PSPACE-complete. However, this is in line with the classic graph-shaped tableau-based decision procedures and with the one-pass system by Schwendimann [24].

To get an intuitive understanding of how the tableau works for typical formulae, take a look at Fig. 1, where two example tableaux are shown. Figure 1a shows part of the tableau for the liveness formula $\mathsf{GF}(p \wedge \mathsf{X}\neg p)$. This formula requires something, *i.e.*, $p \wedge \mathsf{X}\neg p$, to happen infinitely often. As we go down any branch, we can see that the request $\mathsf{XGF}(p \wedge \mathsf{X}\neg p)$ is present at any poised label, propagated by the corresponding expansion rule. Then, any time $\mathsf{F}(p \wedge \mathsf{X}\neg p)$ is added to a label, the branch forks to choose between adding $p \wedge \mathsf{X}\neg p$ immediately or postponing it: precisely, in the nodes at depth 2 and 6 of Figure 1a, the left children postpone the request (*i.e.*, $\mathsf{XF}(p \wedge \mathsf{X}\neg p)$), while the right children fulfil the eventuality $p \wedge \mathsf{X}\neg p$ (the converse happens for the node at depth 4). When the request is fulfilled, p cannot hold twice

$$\begin{aligned}\phi \equiv p \wedge \mathsf{G}(p \longleftrightarrow \mathsf{X}\neg p) \wedge \mathsf{GF}q_1 \wedge \mathsf{GF}q_2 \wedge \\ \mathsf{G}\neg(q_1 \wedge q_2) \wedge \mathsf{G}(q_1 \to \neg p) \wedge \mathsf{G}(q_2 \to \neg p)\end{aligned}$$

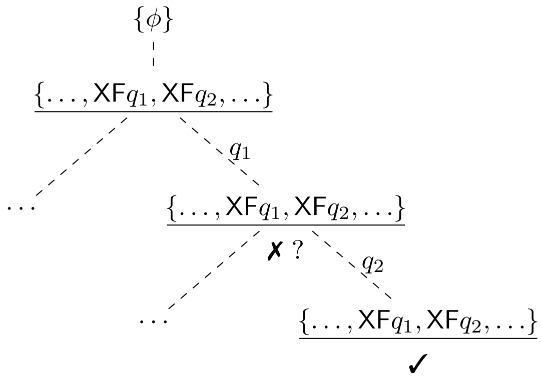


Figure 2: Example of why the PRUNE rule waits for *three* repetitions of the same label.

in a row, and this is handled by the CONTRADICTION rule, that fires when the wrong choice is made. Then, the LOOP rule is triggered by the rightmost branch, which is repeating the same label for the second time. The looping arrow does not represent a real edge, since otherwise it would not be a tree, but it is just a way to highlight to which label the loop is jumping to.

Figure 1b shows an example of application of the PRUNE rule in the tableau for the formula $\mathsf{G}\neg p \wedge q \,\mathcal{U}\, p$. The formula is unsatisfiable, not directly because of a propositional contradiction, but rather because the eventuality requested by $q \,\mathcal{U}\, p$ cannot be realised for the presence of the $\mathsf{G}\neg p$ component. The expansion of the *until* operator will then try to realise p at each step, each time resulting into a propositional contradiction. The rightmost branch would then continue postponing the X-eventuality forever, if not for the PRUNE rule which crosses the branch after the third repetition of the same label (with no X-eventuality fulfilled in between).

One may wonder why the PRUNE rule needs to look for the third occurrence of a label before triggering. An enlightening example is given in Fig. 2. The formula ϕ shown in the figure requires q_1 and q_2 to appear infinitely often, but never at the same time, thus forcing a kind of (not necessarily strict) alternation between the two. Developing the tableau for ϕ, one can see that the requests $\mathsf{XF}q_1$ and $\mathsf{XF}q_2$ will be permanently present in the labels, including, in particular, after realising one of the two. Thus, crossing the branch after the second occurrence of the label would be wrong, since the repetition of the label alone does not imply that the branch is doing wasteful.

Rule	$\phi \in \Gamma$	$\Gamma_1(\phi)$	$\Gamma_2(\phi)$
SINCE	$\alpha\,\mathcal{S}\,\beta$	$\{\beta\}$	$\{\alpha, \mathsf{Y}(\alpha\,\mathcal{S}\,\beta)\}$
TRIGGERED	$\alpha\,\mathcal{T}\,\beta$	$\{\alpha, \beta\}$	$\{\beta, \mathsf{Z}(\alpha\,\mathcal{T}\,\beta)\}$
PAST	$\mathsf{P}\alpha$	$\{\alpha\}$	$\{\mathsf{YP}\alpha\}$
HISTORICALLY	$\mathsf{H}\alpha$	$\{\alpha, \mathsf{ZH}\alpha\}$	

Table 2: Additional expansion rules supporting LTL+P past operators.

Indeed, after the first repetition, the branch can continue making different choices, realising the other request, and can be closed by the LOOP rule after having realised both.

3.2. The tableau system for LTL+P

The one-pass and tree-shaped tableau system for LTL+P extends the one for LTL, with just a few specific rules. Past operators supported by LTL+P are specular to future ones supported by LTL. In particular, as a formula employing the *until* operator $\psi_1\,\mathcal{U}\,\psi_2$ holds if either ψ_2 or $\psi_1 \wedge \mathsf{X}(\psi_1\,\mathcal{U}\,\psi_2)$ hold at the current state, so the *since* operator can be recursively expressed in terms of the *yesterday* operator, as $\psi_1\,\mathcal{S}\,\psi_2$ holds if either ψ_2 or $\psi_1 \wedge \mathsf{Y}(\psi_1\,\mathcal{S}\,\psi_2)$ hold. Hence, as a first step in extending the tableau system to LTL+P, we can extend the set of *expansion rules* of Table 1 to a set of specular rules involving past operators, as shown in Table 2.

Formulae of the form $\mathsf{Y}\psi$ or $\mathsf{Z}\psi$ are considered *elementary*, in the same way as *tomorrow* operators and literals, since they cannot be further expanded. Hence, the *yesterday* and the Z operators need to be handled in a specific way: the YESTERDAY rule and the W-YESTERDAY rule are added for this purpose. Let $\overline{u} = \langle u_0, \ldots, u_n\rangle$ be a poised branch and let $\overline{\pi} = \langle \pi_0, \ldots, \pi_m\rangle$ be the sequence of its poised nodes. Then, given a poised node u_i, let $\Gamma^*(u_i) = \bigcup_{j<k\leq i} \Gamma(u_k)$, where u_j is the child of the closest node preceding u_i where the STEP rule was applied, or the root if the STEP rule was never applied before u_i. In other terms, $\Gamma^*(u_i)$ contains all the formulae that were expanded to build the poised label of u_i.

YESTERDAY If $\mathsf{Y}\alpha \in \Gamma(u_n)$, then let $Y_n = \{\psi \mid \mathsf{Y}\psi \in \Gamma(u_n)\}$, and let u_k be the closest ancestor of u_n where the STEP rule was applied.

Then, the node u_n is *crossed* either if u_k does not exist because there is no application of the STEP rule preceding u_n, or if $Y_n \not\subseteq \Gamma^*(u_k)$.

W-YESTERDAY If $\mathsf{Z}\alpha \in \Gamma(u_n)$, then let $Z_n = \{\psi \mid \mathsf{Z}\psi \in \Gamma(u_n)\}$, and let u_k be the closest ancestor of u_n where the STEP rule was applied, if any.

Then, the node u_n is *crossed* if u_k exists and $Z_n \not\subseteq \Gamma^*(u_k)$.

Intuitively, the YESTERDAY rule (and the W-YESTERDAY rule) checks whether the transitions from a state to the next are consistent with the coming past requests. It is also in this sense specular to the STEP rule, but differs significantly from the latter because instead of actively building the model such that the requests are satisfied, it is forced to retroactively check this consistency.

Whenever a formula $\mathsf{Y}\alpha$ is found in a poised label, the previous state is checked for the presence of α. Note that, of course, the rule could not conceivably test the exact satisfaction of the formula, because that would require testing whether α is a logical consequence of the formulae found in the labels, which would be as hard as the satisfiability problem itself. Instead, the presence of the formula in the labels is taken as a good approximation of whether the formula holds in the state or not.

However, the YESTERDAY rule alone would result into an incomplete system. Given a formula $\mathsf{Y}\alpha$ in a given node, most of the times there are no other reasons for α to be forced to hold at the previous state. Hence, if not explicitly accounted for, the formula may be never expanded and any instance of the YESTERDAY rule would fail. Hence, we need a further rule, running on *poised nodes* before the application of the STEP rule, which *guesses* which formulae must be true at the current state because of yet-unseen past requests coming from future nodes. This guess is made explicit in the tableau by adding a number of children whose label contain the additional formulae. Hence, let $\overline{u} = \langle u_0, \ldots, u_n \rangle$ be a poised branch.

FORECAST Let G_n be the set of all formulae involved in any *yesterday* or *weak yesterday* operator related to the formulae of $\Gamma(u_n)$:

$$G_n = \{\alpha \in \mathcal{C}(\phi) \mid \mathsf{Y}\alpha \in \mathcal{C}(\psi) \text{ or } \mathsf{Z}\alpha \in \mathcal{C}(\psi) \text{ for some } \psi \in \Gamma(u_n)\ \}$$

Then, for each subset $G'_n \subseteq G_n$ (including the empty set), a child u'_n is added to u_n such that $\Gamma(u'_n) = \Gamma(u_n) \cup G'_n$. This is done at most once before each application of the STEP rule.

The added children will not be, in general, poised nodes, hence the expansion continues as usual and their whole subtrees are explored. The

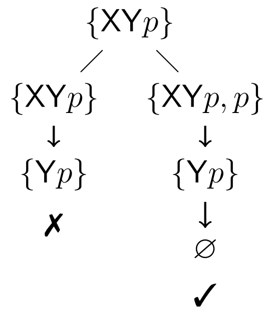


Figure 3: Example tableau for the formula $\mathsf{XY}p$, applying the additional rules.

expansion eventually leads to further poised nodes, which then can be subject to the STEP rule in order to advance in time. In this way, we ensure that for any potentially failing instance of the YESTERDAY or W-YESTERDAY rule, another branch exists where the rule does not fail (if this is possible at all).

Figure 3 shows an example of the use of the new rules on the very simple formula $\mathsf{XY}p$. Before the first application of the STEP rule, the root node u_0, whose label is $\Gamma(u_0) = \{\mathsf{XY}p\}$, triggers the FORECAST rule, which adds to it two children, one with the same label, and one with an additional p. Then, proceeding with the expansion of such children as usual, a STEP rule is performed in each branch and the following instance of the YESTERDAY rule fails in one case but succeeds in the other. In the succeeding case, a second STEP rule then produces an empty label, as no occurrence of *tomorrow* is present, accepting the branch. Note that, in this example, the branch where p is not added might seem useless. Indeed in more complex formulae, some of the added children might be redundant, but in general there is no way to know which formulae will be actually needed without expanding until the failure of the YESTERDAY rule, at which time it would be too late. For this reason, any possible combination is tried beforehand.

4. Soundness and completeness

This section gives the full proofs of soundness and completeness of the one-pass and tree-shaped tableau system for LTL+P shown in Section 3. With respect to the original exposition of the LTL system [23], we employ a different proof based on a novel model-theoretic argument. The new argument provides a deeper understanding of the system, by characterising the behaviour of the

PRUNE rule. Because of the modularity of the new system with regards to the future-only one, the new argument applies to the LTL system as well.

Before going through soundness and completeness proofs, let us first prove that the construction of the tableau as described in the previous section effectively always terminates.

Theorem 1 (Termination)**.** *Given an* LTL+P *formula ϕ, the construction of a (complete) tableau T for ϕ always terminates in a finite number of steps.*

Proof. To start with, we observe that the tree has a finite branching degree: each rule adds at most two children to every node, with the exception of the FORECAST rule that can add more than two nodes; however, by definition, such a rule will never result into two siblings with the same label, and the set of possible labels is finite. Thus, by König's lemma, for the construction to go on forever the tree should contain at least one infinite branch: let $\overline{v} := \langle v_0, v_1, \dots \rangle$ be such a branch. In turn, since the number of possible formulae is finite, there exists at least one label that occurs infinitely often; more precisely, since each expansion rule creates a new label containing formulae whose *nesting degree* (see Section 2.1) is less than or equal to the original label, and since the nesting degree is always an integer number, there are in $\overline{v}$ an infinite number of *step nodes*: let $\overline{\pi} := \langle \pi_0, \pi_1, \dots \rangle$ be the subsequence of all the step nodes in $\overline{v}$. We divide in cases, depending if the LOOP rule can be triggered at some point in $\overline{v}$ or not. If this is the case, then obviously we have a contradiction with the fact that $\overline{v}$ is an infinite branch. Let's consider then the case in which the LOOP rule cannot be triggered. We define:

$$R(i) := \{\beta \in \mathcal{C}(\phi) \mid \text{ either } \alpha \,\mathcal{U}\, \beta \text{ or } \mathsf{F}\beta \text{ belongs to } \Gamma(\pi_i)\}$$

$$F^i(j) := \{\beta \in \mathcal{C}(\phi) \mid \beta \in R(i) \text{ and } \beta \text{ is fulfilled between } \pi_i \text{ and } \pi_j\}$$

We take the first step node of $\overline{v}$, *i.e.*, π_0. Obviously, F^0 is a monotonically increasing function with a finite range (*i.e.*, $\mathcal{C}(\phi)$), and thus there exists a $j_{max} \in \mathbb{N}$ such that $F^0(j_{max}) = F^0(j_{max}+1)$. Since $F^0(j_{max}+1) = F^0(j_{max}) \cup F^{j_{max}}(j_{max}+1)$, we have that $F^{j_{max}}(j_{max}+1) \subseteq F^0(j_{max})$. Therefore, the step nodes π_0, $\pi_{j_{max}}$ and $\pi_{j_{max}+1}$ constitutes an instance of the PRUNE rule. This however is again in contradiction with the fact of having an infinite branch. □

We can now proceed by showing soundness and completeness of the system. The two proofs are independent, but related by the common use of the concept of *pre-model*, which is an abstract representation of one or more models of a formula. The concept of pre-model is not new in the expositions of tableau methods, but here we define it in such a way to cope with the fact, noted above, that the labels of our tableau can avoid to contain formulae that are not relevant for the satisfiability of the formula in a given state. The rest of the section is organised as follows. In Section 4.1, we introduce the concept of pre-model, and of *greedy pre-model*, setting up the necessary machinery used in the subsequent proofs. Then, Sections 4.2 and 4.3 exploit this machinery to prove the system to be sound and complete, respectively.

4.1. Pre-models

Pre-models are an abstract representation of one or more models of a formula that differ in some details that are not relevant to the satisfaction of the formula. Pre-models are made of basic objects called *atoms*.

Definition 2 (Atom). An *atom* for an $\mathsf{LTL{+}P}$ formula ϕ is a set $\Delta \subseteq \mathcal{C}(\phi)$, such that:

1. $\{p, \neg p\} \not\subseteq \Delta$, for any $p \in \Sigma$;
2. if $\psi \in \Delta$, then either $\Gamma_1(\psi) \subseteq \Delta$ or $\Gamma_2(\psi) \neq \varnothing$ and $\Gamma_2(\psi) \subseteq \Delta$, where $\Gamma_1(\psi)$ and $\Gamma_2(\psi)$ are defined as in Tables 1 and 2;
3. for each $\psi, \psi' \in \mathcal{C}(\phi)$, if $\psi \in \Delta$ and $\psi \models \psi'$, then $\psi' \in \Delta$, *i.e.*, Δ is closed by logical deduction (as far as the closure is concerned).

Our notion of atom follows the expansion rules as defined in Tables 1 and 2. In addition to that, however, the set is closed by logical entailment. Note that atoms are theoretical devices that will be useful in our proofs, but are not handled by any effective procedure. As such, there is no circularity in requiring them to be closed by logical entailment. Another difference is that our atoms are *incomplete*, *i.e.*, they do not necessarily specify the truth of all the formulae, but only of those that are required to justify the presence of others. This is in line with how labels of the tableau are constructed, as noted above. Note that the empty set is a valid atom in our definition. Single atoms are not useful by themselves, but have to be arranged in sequences, called *pre-models*, which are an abstract representation of the models of a formula.

Definition 3 (Pre-model). Let ϕ be an $\mathsf{LTL{+}P}$ formula. A *pre-model* of ϕ is an infinite sequence $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ of atoms for ϕ such that, for all $i \geq 0$:

1. $\phi \in \Delta_0$;
2. if $\mathsf{X}\psi \in \Delta_i$, then $\psi \in \Delta_{i+1}$;
3. if $\mathsf{Y}\psi \in \Delta_i$, then $i > 0$ and $\psi \in \Delta_{i-1}$;
4. if $\mathsf{Z}\psi \in \Delta_i$, then either $i = 0$ or $\psi \in \Delta_{i-1}$;
5. if $\psi_1 \mathcal{U} \psi_2 \in \Delta_i$, then there is a $j \geq i$ with $\psi_2 \in \Delta_j$ and $\psi_1 \in \Delta_k$ for all $i \leq k < j$;
6. Δ_i is *minimal*, *i.e.*, there is no $\psi \in \Delta_i$ such that, if ψ is removed, Δ_i is still an atom and the above items are still satisfied.

Pre-models take their name from the fact that they abstractly represent a model for their formula, and thus the existence of a pre-model witnesses the satisfiability of the formula.

Lemma 1 (Extraction of a model from a pre-model)**.** *Let ϕ be an* LTL+P *formula. If ϕ has a pre-model, then ϕ is satisfiable.*

Proof. Let $\overline{\Delta}$ be a pre-model of ϕ and let $\overline{\sigma}$ be a state sequence such that $p \in \Delta_i$ if and only if $\overline{\sigma}, i \models p$. Then, we show that $\overline{\sigma} \models \phi$ and thus the formula is satisfiable. Note that this definition makes the precise choice of setting as *false* all the literals that are missing from the atoms of the pre-model, but any arbitrary choice would work for such literals. For any $\psi \in \mathcal{C}(\phi)$, let $\deg(\psi)$ be the *nesting degree* of ψ, as defined in Section 2.1. We prove by induction on $\deg(\psi)$ that if $\psi \in \Delta_i$, then $\overline{\sigma}, i \models \psi$ for any $\psi \in \mathcal{C}(\phi)$ and any $i \geq 0$. The thesis then follows immediately, since $\phi \in \Delta_0$ by definition.

As for the base case, if $p \in \Delta_i$ or $\neg p \in \Delta_i$, then the thesis follows by the definition of $\overline{\sigma}$. As for the inductive step, we go by cases:

1. if $\psi_1 \vee \psi_2 \in \Delta_i$ (resp., $\psi_1 \wedge \psi_2 \in \Delta_i$), then by definition of atom and by the inductive hypothesis, either $\overline{\sigma}, i \models \psi_1$ or $\overline{\sigma}, i \models \psi_2$ (resp., both), and thus $\overline{\sigma}, i \models \psi_1 \vee \psi_2$ (resp., $\overline{\sigma}, i \models \psi_1 \wedge \psi_2$);
2. if $\mathsf{X}\psi \in \Delta_i$, then, by Item 2 of Definition 3, it holds that $\psi \in \Delta_{i+1}$. Since $\deg(\psi) < \deg(\mathsf{X}\psi)$, by the inductive hypothesis it follows that $\overline{\sigma}, i+1 \models \psi$. Then, by the semantics of the *tomorrow* operator, we have $\overline{\sigma}, i \models \mathsf{X}\psi$;

3. if $\mathsf{Y}\psi \in \Delta_i$, by Item 3 of Definition 3, it holds that $i > 0$ and $\psi \in \Delta_{i-1}$. Since $\deg(\psi) < \deg(\mathsf{Y}\psi)$, by the inductive hypothesis we have $\overline{\sigma}, i-1 \models \psi$. Then, by the semantics of the *yesterday* operator, it follows that $\overline{\sigma}, i \models \mathsf{Y}\psi$.

4. if $\mathsf{Z}\psi \in \Delta_i$, by Item 4 of Definition 3, it holds that either $i = 0$ or $\psi \in \Delta_{i-1}$. In the latter case, since $\deg(\psi) < \deg(\mathsf{Z}\psi)$, by the inductive hypothesis we have $\overline{\sigma}, i-1 \models \psi$. Then, by the semantics of the Z operator, it follows that $\overline{\sigma}, i \models \mathsf{Z}\psi$.

5. if $\psi_1 \mathcal{U} \psi_2 \in \Delta_i$, then, by definition of atom, there exists $j \geq i$ such that $\psi_2 \in \Delta_j$ and $\psi_1 \in \Delta_k$, for all $i \leq k < j$. Then, by the inductive hypothesis, $\overline{\sigma}, j \models \psi_2$ and $\overline{\sigma}, k \models \psi_1$ for all $i \leq k < j$, hence by the semantics of the *until* operator, we have $\overline{\sigma}, i \models \psi_1 \mathcal{U} \psi_2$;

6. the argument is similar to Item 1 when $\psi_1 \mathcal{R} \psi_2 \in \Delta_i$, $\psi_1 \mathcal{S} \psi_2 \in \Delta_i$, or $\psi_1 \mathcal{T} \psi_2 \in \Delta_i$, since they can be broken into disjunctions after the application of the corresponding expansion rules and, in contrast to the *until* operator, they do not require to check any global property on the pre-model. □

Note that, on the contrary, a pre-model can be obtained straightforwardly from any model $\overline{\sigma} \models \phi$ of a formula by simply setting $\phi \in \Delta_0$, and then following the definition obtaining a sequence $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ of atoms where each Δ_i is the *minimal* atom that contains every formula from $\mathcal{C}(\phi)$ that holds at $\overline{\sigma}_i$.

4.2. Soundness

Here we prove that the tableau system is *sound*, that is, if a complete tableau for a formula has a successful branch, then the formula is satisfiable. Moreover, a model for the formula can be effectively obtained from any successful branch. The proof shows how a pre-model for ϕ can be extracted from a complete tableau with a successful branch, which then lets us obtain one or more models of the formula. Intuitively, the expansion of non-poised nodes builds the set of formulae that have to be included in the current atom, and the application of the STEP rule to a poised node marks the advancement to the next one. However, because of the FORECAST rule, the STEP rule is not applied to all poised nodes.

Definition 4 (Step nodes). Consider a branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$ of a complete tableau T. A poised node u_i is said to be a *step node* if either $u_i = u_n$ or u_{i+1} is the child of u_i added by applying the STEP rule.

Only *step nodes* will be considered when looking at a given tableau branch to build the corresponding pre-model. Now we can state how exactly to perform this extraction. Let us first define how single atoms are built from each step node.

Definition 5 (Atom of a tableau node). Let ϕ be an $\mathsf{LTL{+}P}$ formula, and let u_i be a step node of a tableau for ϕ. The *atom* of u_i, written $\Delta(u_i)$, is the closure by logical entailment (within $\mathcal{C}(\phi)$) of $\Gamma(u_i)$.

Let us now define how to extract a pre-model from a successful branch of a complete tableau.

Lemma 2 (Extraction of a pre-model from a successful tableau)**.** *Let ϕ be an* $\mathsf{LTL{+}P}$ *formula and T a complete tableau for ϕ. If T has a successful branch, then there exists a pre-model for ϕ.*

Proof. Let $\overline{u} = \langle u_0, \ldots, u_n \rangle$ be a successful branch of T and let $\overline{\pi} = \langle \pi_0, \ldots, \pi_m \rangle$ be the subsequence of step nodes of $\overline{u}$. Intuitively, a pre-model for ϕ can be obtained from $\overline{u}$ by building the atoms from the labels of the step nodes, and extending them to an infinite sequence. If the branch has been accepted by the LOOP rule, we can identify a position $0 \leq k \leq m$ in $\overline{\pi}$ such that $\Delta(\pi_k) = \Delta(\pi_m)$ and all the X-eventualities requested in π_k are fulfilled in $\pi_{[k\ldots m]}$. If instead $\overline{u}$ has been accepted by the EMPTY rule, then $\Gamma(\pi_m) = \varnothing$, and in particular there are no X-eventualities requested, hence setting $k = m$ we obtain the same effect. Therefore, we can extract from $\overline{\pi}$ the *periodic* sequence of atoms $\overline{\Delta} = \overline{\Delta}_0 \overline{\Delta}_T^{\omega}$, where $\overline{\Delta}_0 = \langle \Delta(\pi_0), \ldots, \Delta(\pi_k) \rangle$, and either $\overline{\Delta}_T = \langle \Delta(\pi_{k+1}), \ldots, \Delta(\pi_m) \rangle$ or $\overline{\Delta}_T = \langle \pi_m \rangle$ depending on which rule accepted the branch, respectively the LOOP or the EMPTY rule. In other words, we build a periodic pre-model that infinitely repeats the fulfilling loop identified by the LOOP rule, or the last empty node otherwise. Then let $K : \mathbb{N} \to \mathbb{N}$ be the map from positions in the pre-model to their original positions in the branch, which is defined as $K(i) = i$ for $0 \leq i < k$, and for $i \geq k$ is defined either as $K(i) = k + ((i - k) \mod T)$, with $T = m - k$ (LOOP rule), or as $K(i) = k$ (EMPTY rule).

We can now show that $\overline{\Delta}$ is indeed a pre-model for ϕ. First, observe that $\phi \in \Delta_0$ because $\phi \in \Gamma(\pi_0)$ by construction, thus Item 1 of Definition 3 is satisfied. Then, we check Items 2 to 5 of Definition 3.

2. Consider any formula $\mathsf{X}\psi \in \Delta_i$. Being an elementary formula, we can observe that $\mathsf{X}\psi \in \Gamma(\pi_{K(i)})$. Two cases have to be considered. If $\pi_{K(i+1)} = \pi_{K(i)+1}$, *i.e.*, the next atom comes from the actual successor of the current one in the tableau branch, then, by the STEP rule, we know $\psi \in \Gamma(\pi_{K(i+1)}) \subseteq \Delta_{i+1}$. Otherwise, $\Delta_i = \Delta_m = \Delta(\pi_m)$ and π_m was ticked by the LOOP rule (because Δ_i is not empty), and thus $\Delta_{i+1} = \Delta(\pi_{k+1})$ for some $k < m$ such that $\Gamma(\pi_k) = \Gamma(\pi_m)$. Hence, $\mathsf{X}\psi \in \Gamma(\pi_k)$ as well, and, by the STEP rule applied to π_k, we know that $\psi \in \Gamma^*(\pi_{k+1}) \subseteq \Delta(\pi_{k+1}) = \Delta_{i+1}$.

3. Consider any formula $\mathsf{Y}\psi \in \Delta_i$ and thus $\mathsf{Y}\psi \in \Gamma(\pi_{K(i)})$. By the YESTERDAY rule, $i > 0$. As in the previous case, either Δ_{i-1} is the atom coming from the previous step node, and thus $\psi \in \Delta_{i-1}$ by the YESTERDAY rule, or $\Delta_i = \Delta(\pi_{k+1})$ for some k that triggered the LOOP rule because $\Gamma(\pi_k) = \Gamma(\pi_m)$, which implies that $\Delta_k = \Delta_m$. Hence, by the YESTERDAY rule, we have $\psi \in \Delta_k = \Delta_m = \Delta_{i-1}$.

4. The case $\mathsf{Z}\psi_1 \in \Delta_i$ is similar to the previous one.

5. The other cases, such as if $\psi_1 \,\mathcal{U}\, \psi_2 \in \Delta_i$, are straightforward in view of the expansion rules definition. □

The above results let us conclude the soundness of the tableau system.

Theorem 2 (Soundness)**.** *Let ϕ be an* LTL+P *formula, and let T be a complete tableau for ϕ. If T has a successful branch, then ϕ is satisfiable.*

Proof. Extract a pre-model for ϕ from the successful branch of T as shown in Lemma 2, and then obtain from it an actual model for the formula as shown by Lemma 1. □

4.3. Completeness

We now prove the completeness of the tableau system, *i.e.*, if a formula ϕ is satisfiable, then any complete tableau T for it has an accepting branch.

The proof uses a pre-model for ϕ, which we know exists if the formula is satisfiable, as a guide to suitably descend through the tableau to look for an accepted branch. Then, we will show how to make sure that this descent must obtain an accepted branch. The descent is performed as follows.

Lemma 3 (Extraction of the branch)**.** *Let $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ be a pre-model for a formula ϕ. Then, any complete tableau for ϕ has a branch $\overline{u}$, with sequence of step nodes $\overline{\pi} = \langle \pi_0, \ldots, \pi_m \rangle$, such that $\Delta(\pi_i) = \Delta_i$ for all $0 \leq i \leq m$.*

Proof. To find $\overline{u}$, we traverse the tree using $\overline{\Delta}$ as a guide, starting from the root u_0, building a sequence of branch prefixes $\overline{u}_i = \langle u_0, \ldots, u_i \rangle$, suitably choosing u_{i+1} at each step among the children of u_i. Meanwhile, we maintain a non-decreasing function $J : \mathbb{N} \to \mathbb{N}$ that maps positions in $\overline{u}_i$ to positions in $\overline{\Delta}$ such that $\Gamma(u_k) \subseteq \Delta_{J(k)}$ for each $0 \leq k \leq i$, starting from $\overline{u}_0 = \langle u_0 \rangle$ and $J(0) = 0$. With this base case the invariant holds since $\Gamma(u_0) = \{\phi\}$ and $\phi \in \Delta_0$ by definition. Then, at each step $i \geq 0$, we choose u_{i+1} among the children of u_i as follows:

1. if u_i is a step node but not a leaf, then it has a single child which is chosen as u_{i+1}, defining $J(i+1) = J(i) + 1$, since we need to advance to the next position in the pre-model as well;

2. if u_i is not a step node and it was expanded by the FORECAST rule, then u_i has a number of children $\{u_i^0, \ldots, u_i^k\}$, such that $\Gamma(u_i) \subseteq \Gamma(u_i^j)$ for all $0 \leq j \leq k$. Then, we set $J(i+1) = J(i)$, and we choose as u_{i+1} the child u_i^j with the *maximal* label such that $\Gamma(u_i^j) \subseteq \Delta_{J(i)}$. Note that at least one such child exists since one among them has the same label as u_i;

3. if u_i is not a step node and it was expanded by an expansion rule, then it has two children u_i' and u_i'' and a formula ϕ such that $\Gamma(u_i') = \Gamma(u_i) \backslash \{\phi\} \cup \Gamma_1(\phi)$ and $\Gamma(u_i'') = \Gamma(u_i) \backslash \{\phi\} \cup \Gamma_2(\phi)$, where $\Gamma_1(\phi)$ and $\Gamma_2(\phi)$ are defined in Tables 1 and 2. Since we maintained that $\Gamma(u_i) \subseteq \Delta_{J(i)}$, and thus $\phi \in \Delta_{J(i)}$, by Definition 2 we know that either $\Gamma_1(\phi) \subseteq \Delta_{J(i)}$ or $\Gamma_2(\phi) \subseteq \Delta_{J(i)}$, hence either $\Gamma_1(u_i') \subseteq \Delta_{J(i)}$ or $\Gamma_2(u_i'') \subseteq \Delta_{J(i)}$, so we can choose u_{i+1} accordingly. Note that both might be suitable choices, in which case, which one is chosen is not important. In any case, we set $J(i+1) = J(i)$, since we do not need to advance in the pre-model.

Now, let $\overline{u} = \langle u_0, \ldots, u_n \rangle$ be the branch found as described above, and let $\overline{\pi} = \langle \pi_0, \ldots, \pi_m \rangle$ be the sequence of its step nodes. Since in the traversal the value of $J(i)$ is incremented only when an application of the STEP rule is traversed, it holds that $\Gamma(\pi_i) \subseteq \Delta_i$. By Definition 5, $\Delta(\pi_i)$ is the closure under logical entailment of $\Gamma(\pi_i)$, hence $\Delta(\pi_i) \subseteq \Delta_i$.

Now, consider the sets of formulae X_i, Y_i, and Z_i, such that $X_0 = \{\phi\}$, and $X_{i+1} = \{\psi \mid \mathsf{X}\psi \in \Delta_i\}$, $Y_i = \{\psi \mid \mathsf{Y}\psi \in \Delta_{i+1}\}$ and $Z_i = \{\psi \mid \mathsf{Z}\psi \in \Delta_{i+1}\}$, for all $i \geq 0$. Note that $X_i, Y_i, Z_i \subseteq \Delta_i$ by Definition 3. By the way the tree has been descended, one can check by induction that $X_i, Y_i, Z_i \subseteq \Delta(\pi_i)$ as well. Moreover, for the minimality of the atoms in the pre-model, required by Definition 3, we know that for any formula $\psi \in \Delta_i$, either $\psi \in X_i \cup Y_i \cup Z_i$, ψ has been obtained by the expansion of $X_i \cup Y_i \cup Z_i$ or by the closure by logical entailment. Similarly, it can be checked that, by construction, for any $\psi \in \Delta(\pi_i)$, either $\psi \in X_i \cup Y_i \cup Z_i$, or ψ has been obtained by the expansion of the three sets. Recall that, by the structure of the expansion rules, we have that $\Gamma(\pi_i) \models X_i \cup Y_i \cup Z_i$. Hence, both $\Delta(\pi_i)$ and Δ_i are closures by logical deduction of $\Gamma_i(\pi_i)$, and we can conclude that $\Delta(\pi_i) = \Delta_i$. □

The particular branch found as described above might, in general, be crossed. However, it is immediate to note that it cannot possibly have been crossed by an application of the CONTRADICTION rule, since this would imply the existence of some $\{p, \neg p\} \subseteq \Delta_i$ for some i, which cannot be the case. Similarly, it cannot have been crossed by the YESTERDAY or W-YESTERDAY rules, since this would imply some $Y\alpha \in \Delta_i$ such that $\alpha \notin \Delta_{i-1}$. Hence, if a crossed leaf is found, it has been crossed by the PRUNE rule. The novelty of the proof presented here is how we can select a proper class of models (and their pre-models) such that the descent described by Lemma 3, applied on one of these particular models, cannot possibly find a node crossed by the PRUNE rule, neither.

The key concept behind our proof is that of *greedy pre-models*. Given a pre-model $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$, an X-eventuality $\psi \equiv \mathsf{X}(\psi_1 \,\mathcal{U}\, \psi_2)$ is *requested* at position $i \geq 0$ if $\psi \in \Delta_i$, and *fulfilled* at $j > i$ if j is the *first* position where $\psi_2 \in \Delta_j$ and $\psi_1 \in \Delta_k$, for all $i < k < j$. Let $\mathcal{U}(\phi) = \{\psi \in \mathcal{C}(\phi) \mid \psi \equiv \mathsf{X}(\psi_1 \,\mathcal{U}\, \psi_2)\}$ be the set of X-eventualities in the closure of ϕ. For each position $i \geq 0$, we define the *delay* at position i as a function $\mathsf{d}_i : \mathcal{U}(\phi) \to \mathbb{N}$ providing a natural number for each eventuality in $\mathcal{U}(\phi)$, as follows:

$$\mathsf{d}_i(\psi) = \begin{cases} 0 & \text{if } \psi \text{ is not requested at position } i \\ n & \text{if } \psi \text{ is requested at } i \text{ and fulfilled at } j \text{ such that } n = j - i \end{cases}$$

Intuitively, $\mathsf{d}_i(\psi)$ is the number of states elapsed between the request and the fulfilment of ψ. Let $\mathbb{D}$ be the set of all possible delays. Then, we can define a partial order $(\mathbb{D}, \preceq)$ between delays by comparing them component-wise, *i.e.*, for any $\mathsf{d}, \mathsf{d}' \in \mathbb{D}$, $\mathsf{d}(\psi) \leq \mathsf{d}'(\psi)$ for each $\psi \in \mathcal{U}(\phi)$. Note that $\mathbb{D}$ is just another way to denote $\mathbb{N}^{|\mathcal{U}(\phi)|}$, and $(\mathbb{D}, \preceq)$ is just $(\mathbb{N}^{|\mathcal{U}(\phi)|}, \leq)$, *i.e.*, tuples of $|\mathcal{U}(\phi)|$ natural numbers ordered component-wise. In particular, this means that $(\mathbb{D}, \preceq)$ is *well-founded*, *i.e.*, there are no infinite descending chains of elements. Given two infinite sequences of delays $\overline{\mathsf{d}} = \langle \mathsf{d}_0, \mathsf{d}_1, \ldots \rangle$ and $\overline{\mathsf{d}}' = \langle \mathsf{d}'_0, \mathsf{d}'_1, \ldots \rangle$, we can compare them lexicographically, hence defining a partial order $(\mathbb{D}^\omega, \preceq_{lex})$ such that $\overline{\mathsf{d}} \preceq_{lex} \overline{\mathsf{d}}'$ iff either $\mathsf{d}_0 \preceq \mathsf{d}'_0$ or $\mathsf{d}_0 = \mathsf{d}_0$ and $\overline{\mathsf{d}}_{\geq 1} \preceq_{lex} \overline{\mathsf{d}}'_{\geq 1}$. Similarly, we write $\overline{\mathsf{d}} \prec_{lex} \overline{\mathsf{d}}'$ iff either $\mathsf{d}_0 \prec \mathsf{d}'_0$ or $\mathsf{d}_0 = \mathsf{d}_0$ and $\overline{\mathsf{d}}_{\geq 1} \prec_{lex} \overline{\mathsf{d}}'_{\geq 1}$.

A preorder can instead be defined over pre-models, by defining $\overline{\Delta} \preceq \overline{\Delta}'$ iff $\overline{\mathsf{d}} \preceq_{lex} \overline{\mathsf{d}}'$, where $\overline{\mathsf{d}}$ and $\overline{\mathsf{d}}'$ are the sequences of delays of $\overline{\Delta}$ and $\overline{\Delta}'$, respectively. We also define the strict variant: $\overline{\Delta} \prec \overline{\Delta}'$ iff $\overline{\mathsf{d}} \prec_{lex} \overline{\mathsf{d}}'$. Note that this is only a preorder because $\overline{\Delta} \preceq \overline{\Delta}'$ and $\overline{\Delta}' \preceq \overline{\Delta}$ do not imply that $\overline{\Delta} = \overline{\Delta}'$, as two different pre-models may have the same delays. Minimal elements in this preorder are called *greedy pre-models*.[2]

Definition 6 (Greedy pre-models). A pre-model $\overline{\Delta}$ for a formula ϕ is *greedy* if there is no pre-model $\overline{\Delta}'$ such that $\overline{\Delta}' \prec \overline{\Delta}$.

Intuitively, in greedy pre-models all the requested X-eventualities are always fulfilled as soon as possible. We will show that starting from one such pre-model ensures we avoid crossed nodes when extracting a branch from the tableau as described in Lemma 3. However, we first need to ensure that a greedy pre-model for a given formula always exists. Note that this preorder is not well-founded, as there can be infinite descending chains. To see this, consider a pre-model $\overline{\Delta}$ for the formula $\mathsf{GF}p$, where there are an infinite number of positions where p holds, as required, but such that $p \in \Delta_i$ iff i is even. In this pre-model there are an infinite number of positions with positive delays, and by decreasing them one by one, by anticipating the position of each fulfilment of p, we can form an infinite descending chain. Nevertheless, a greedy pre-model exists in this case, namely the pre-model $\overline{\Delta}$ where $p \in \Delta_i$ for all $i \geq 0$. Such a pre-model has zero delays everywhere and is comparable with, and less than, all the pre-models of the descending chain defined before. We can generalise this argument to prove that a greedy pre-model always exists for any satisfiable formula.

[2] The term greedy comes from an analogy with greedy algorithms, which always make the best local choice. Recently, the term has been independently used in a similar sense, talking about *greedy paths* in Büchi automata, by Allred and Ultes-Nitsche [1].

Lemma 4 (Limit of a sequence of pre-models)**.** *Let $\overline{\Delta}^1 \succ \overline{\Delta}^2 \succ \overline{\Delta}^3 \succ \dots$ be an infinite descending sequence of pre-models. Then, there exists a pre-model $\overline{\Delta}^\omega$ such that $\overline{\Delta}^\omega \preceq \overline{\Delta}^i$ for all $i \geq 0$.*

Proof. Let $\overline{\Delta}^1 \succ \overline{\Delta}^2 \succ \overline{\Delta}^3 \succ \dots$ be an infinite descending sequence of pre-models. We now prove that for every prefix length $n \in \mathbb{N}$ there is an index $m_n \in \mathbb{N}$ such that the prefix up to position n of pre-models $\overline{\Delta}^{m_n}, \overline{\Delta}^{m_n+1}, \dots$ is the same, that is, the prefix of the pre-models in the sequence stabilises going forward. Once we show that, we can define the *limit* of the sequence as the pre-model $\overline{\Delta}^\omega$ such that $\overline{\Delta}^\omega_{\leq n} = \overline{\Delta}^{m_n}_{\leq n}$. Then, we can observe that $\overline{\Delta}^\omega \preceq \overline{\Delta}^i$ for all $i \geq 0$.

Let us now show how the prefixes of $\overline{\Delta}^i$ stabilise. For $i \geq 1$, let $\mathsf{d}^i = \langle \mathsf{d}^i_0, \mathsf{d}^i_1, \dots \rangle$ be the sequence of delays of $\overline{\Delta}^i$. Let us consider the jth pre-model $\overline{\Delta}^j$, for some $j \geq 1$. By definition of $\succ$, there is a position $n_j \geq 0$ such that $\mathsf{d}^{j+1}_{n_j} < \mathsf{d}^j_{n_j}$, and $\mathsf{d}^{j+1}_m = \mathsf{d}^j_m$, for all $0 \leq m < n_j$. We show that there are finitely many indexes $l > j$ for which there exists a position n_k, with $n_k \leq n_j$, such that $\mathsf{d}^{l+1}_{n_k} < \mathsf{d}^l_{n_k}$, and $\mathsf{d}^{l+1}_m = \mathsf{d}^j_m$, for all $0 \leq m < n_k$. Let $\bar{l}$ be the largest of such indexes l. We prove it by contradiction. Assume that there are infinitely many. Let n_h be the smallest position that comes into play infinitely many times. If $n_h = 0$, then there is an infinite strictly decreasing sequence of delays $\mathsf{d}^{h_1}_0 > \mathsf{d}^{h_2}_0 > \mathsf{d}^{h_3}_0 > \dots$, with $j < h_1 < h_2 < h_3 < \dots$, which cannot be the case since the ordered set $(\mathbb{N}^{|\mathcal{U}(\phi)|}, \leq)$ is well-founded (the definition of temporal shift operators ensures that the closure set of ϕ is finite, and thus $\mathcal{U}(\phi)$ is finite as well). Let $0 < n_h \leq n_j$. Since the positions to the left of n_h are chosen only finitely many times, there exists a tuple $(\mathsf{d}_0, \dots, \mathsf{d}_{n_h-1})$ which is paired with an infinite strictly decreasing sequence of delays $\mathsf{d}^{h_1}_{n_h} > \mathsf{d}^{h_2}_{n_h} > \mathsf{d}^{h_3}_{n_h} > \dots$, with $j < h_1 < h_2 < h_3 < \dots$, which again cannot be the case since the ordered set $(\mathbb{N}^{|\mathcal{U}(\phi)|}, \leq)$ is well-founded. This allows us to conclude that the prefix up to position n_j of all pre-models of index greater than or equal to $\bar{l}$ is the same. □

Lemma 5 (Existence of greedy pre-models)**.** *Let $\overline{\Delta}$ be a pre-model for a formula ϕ. Then, there is a* greedy *pre-model $\overline{\Delta}' \preceq \overline{\Delta}$.*

Proof. We distinguish two cases. If there is a finite sequence $\overline{\Delta}_0 \succ \overline{\Delta}_1 \succ \dots \succ \overline{\Delta}_n$, starting with $\overline{\Delta} = \overline{\Delta}_0$, which is maximal with respect to $\succ$, *i.e.*, it cannot be further extended, then $\overline{\Delta}' = \overline{\Delta}_n$ is a greedy pre-model with $\overline{\Delta}' \preceq \overline{\Delta}$. Otherwise, we can consider an infinite descending sequence of pre-models $\overline{\Delta}^0 \succeq \overline{\Delta}^1 \succ \dots$, with $\overline{\Delta} = \overline{\Delta}^0$ such that each step $\overline{\Delta}^i$, minimises the delays at position i, that is, the following holds: for $i \geq 1$, if $\overline{\Delta}^i = \langle \Delta^i_1, \Delta^i_2, \ldots \rangle$ and $\mathsf{d}^i = \langle \mathsf{d}^i_1, \mathsf{d}^i_2, \ldots \rangle$ is the sequence of delays of $\overline{\Delta}^i$, then there is no pre-model $\overline{\Delta}'$ (with sequence of delays d') such that $\mathsf{d}'_{<i} = \mathsf{d}^i_{<i}$ and $\mathsf{d}'_i < \mathsf{d}^i_i$, *i.e.*, such that the delays at position i causes that $\overline{\Delta}' \prec \overline{\Delta}^i$. Note that minimising the delays at a single position is always possible because $(\mathbb{N}^{|\mathcal{U}(\phi)|})$ is well-founded. Given this sequence (which is infinite because we excluded the existence of finite chains at the beginning), we can find its limit pre-model $\overline{\Delta}^\omega$, as shown in Lemma 4, for which we know that $\overline{\Delta}^\omega \preceq \overline{\Delta}^i$ for all $i \geq 0$. We can now observe that $\overline{\Delta}^\omega$ is *greedy*: if a smaller pre- model $\overline{\Delta}'$ existed, there would be a position i such that $\overline{\Delta}'_{<i} = \overline{\Delta}^\omega_{<i}$, and $\overline{\Delta}'_i < \overline{\Delta}^\omega_i$ but then we would have as well that $\overline{\Delta}'_{<i} = \overline{\Delta}^i_{<i}$ and $\overline{\Delta}'_i < \overline{\Delta}^i_i$, which contradicts how we defined the sequence. Hence, there cannot be a pre-model smaller than $\overline{\Delta}^\omega$, which therefore is a *greedy* pre-model. □

Now, we can introduce the connection between the PRUNE rule and greedy pre-models. To this aim, we define a similar contraction rule on pre-models.

Definition 7 (Redundant segments). Let $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ be a pre-model for ϕ, and let $i < j < k$ be three positions such that $\Delta_i = \Delta_j = \Delta_k$. Then, the subsegment $\overline{\Delta}_{[j+1\ldots k]}$ of $\overline{\Delta}$ is *redundant* if not all the X-eventualities requested in the atoms are fulfilled between in $\overline{\Delta}_{[j+1\ldots k]}$, and all those fulfilled in $\overline{\Delta}_{[j+1\ldots k]}$ are fulfilled in $\overline{\Delta}_{[i+1\ldots j]}$ as well.

Intuitively, a redundant segment is one where the pre-model is doing some useless work, because there are no new X-eventualities fulfilled in the segment that were not already fulfilled before, among those recurrently requested. It can be recognised that the condition of Definition 7 is similar to the one checked by the PRUNE rule on a branch of the tableau, but transferred to pre-models. The important feature of redundant segments is that they can be safely removed obtaining another pre-model which, most importantly, has lower delays than the original.

Lemma 6 (Removal of redundant segments)**.** *Let $\overline{\Delta}$ be a pre-model for a formula ϕ with a redundant segment $\overline{\Delta}_{[j+1\ldots k]}$. Then, the sequence of atoms $\overline{\Delta}' = \overline{\Delta}_{\leq j}\overline{\Delta}_{>k}$ is a pre-model for ϕ, and $\overline{\Delta}' \prec \overline{\Delta}$.*

Proof. First of all, we check that $\overline{\Delta}' = \overline{\Delta}_{\leq j}\overline{\Delta}_{>k}$ is still a pre-model for ϕ. This can be seen by observing that, Δ_j and Δ_k being equal, for any $\mathsf{X}\psi \in \Delta_j$ there is $\psi \in \Delta_{k+1}$ and for any $\mathsf{Y}\psi \in \Delta_{k+1}$ (and $\mathsf{Z}\psi \in \Delta_{k+1}$) there is $\psi \in \Delta_j$.

Now, let $\overline{\mathsf{d}}$ and $\overline{\mathsf{d}}'$ be the sequences of delays of $\overline{\Delta}$ and $\overline{\Delta}'$, respectively. Since $\overline{\Delta}_{[j+1\ldots k]}$ is a redundant segment, there is an atom Δ_i with $i < j$ such that $\Delta_i = \Delta_j (= \Delta_k)$. We proceed by showing that $\mathsf{d}'_i \prec \mathsf{d}_i$, while $\mathsf{d}'_n \preceq \mathsf{d}_n$ for all $n < i$, thus implying that $\overline{\Delta}' \prec \overline{\Delta}$. To this aim we show that there is at least one X-eventuality $\psi \equiv \mathsf{X}(\psi_1 \,\mathcal{U}\, \psi_2)$ for which $\mathsf{d}'_i(\psi) < \mathsf{d}_i(\psi)$ while the other values of the delay vector for the other eventualities at worst do not increase. Now, let $\psi \equiv \mathsf{X}(\psi_1 \,\mathcal{U}\, \psi_2)$ be any X-eventuality requested in Δ_i (and Δ_j and Δ_k), and consider the position $h > i$ where ψ is first fulfilled (such that $\mathsf{d}_i(\psi) = h - i$). Note that it cannot be the case that h appears inside the redundant segment, *i.e.*, that $j < h \leq k$, since ψ, by Definition 7, would have to be fulfilled between Δ_i and Δ_j as well, hence h would not be the point of its first fulfilment. Hence, there are two cases. If $h < j$, then the point of first fulfilment of ψ is not affected by the cut and the delay cannot decrease because of it. Otherwise, if $h > k$, the cut decreases the delay of ψ, hence $\mathsf{d}'_i(\psi) = \mathsf{d}_i(\psi) - (k - j)$. Note that at least one X-eventuality fulfilled after k exists, because they cannot be all fulfilled in $\overline{\Delta}_{[i+1\ldots k]}$ by Definition 7, hence $\mathsf{d}'_i \prec \mathsf{d}_i$.

Now, consider any position $n < i$. In any of those positions, $\mathsf{d}_n(\psi)$, for any X-eventuality $\psi_1 \,\mathcal{U}\, \psi_2$, cannot increase because of the cut, otherwise the first fulfilment of ψ would have been in $\overline{\Delta}_{[j+1\ldots k]}$ (thus postponing its first fulfilment to a later point), which cannot be the case because all the eventualities fulfilled there are fulfilled also in $\overline{\Delta}_{[i+1\ldots j]}$. Hence, $\mathsf{d}'_n \preceq \mathsf{d}_n$ for all $n < i$, thus $\overline{\mathsf{d}}' \prec_{lex} \overline{\mathsf{d}}$, which implies $\overline{\Delta}' \prec \overline{\Delta}$. □

Now we can exploit the results obtained so far to prove the completeness of the tableau system, proving that a complete tableau for a satisfiable formula contains at least an accepted branch.

Theorem 3 (Completeness). *Let ϕ be an* LTL+P *formula and let T be a complete tableau for ϕ. If ϕ is satisfiable, then T contains a successful branch.*

Proof. Let $\overline{\sigma}$ be a model for ϕ. As already noted, it is straightforward to build a pre-model for ϕ from $\overline{\sigma}$. Then, given a pre-model for ϕ, Lemma 5 ensures that a *greedy* pre-model for ϕ exists. We can thus consider $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ to be a greedy pre-model for ϕ. Now, given a complete tableau T for ϕ, thanks to Lemma 3 we can obtain a branch from T, with sequence of step nodes $\overline{\pi} = \langle \pi_0, \ldots, \pi_m \rangle$ such that $\Delta(\pi_k) = \Delta_k$ for all $0 \leq k \leq m$. As already noted, we know that if π_m is crossed, then it has to have been crossed by the PRUNE rule. If this was the case, however, it would mean there are other two step nodes π_i and π_j with $i < j < m$ and $\Gamma(\pi_i) = \Gamma(\pi_j) = \Gamma(\pi_m)$, and such that all the X-eventualities requested in the three nodes and fulfilled between π_{j+1} and π_m are fulfilled between π_{i+1} and π_j as well. Since $\Delta(\pi_k) = \Delta_k$ for all $0 \leq k \leq m$, this fact reflects onto the pre-model, hence $\Delta_i = \Delta_j = \Delta_m$, and all the X-eventualities requested in these atoms and fulfilled in $\overline{\Delta}_{[j+1\ldots m]}$ are fulfilled in $\overline{\Delta}_{[i+1\ldots j]}$ as well. In other words, $\overline{\Delta}_{[j+1\ldots m]}$ is a redundant segment, but this, by Lemma 6 contradicts the assumption that $\overline{\Delta}$ is greedy. □

5. One-pass and tree-shaped tableau systems for the logics TPTL and TPTL$_\mathsf{b}$+P

This section describes the one-pass and tree-shaped tableau systems for TPTL and TPTL$_\mathsf{b}$+P, that respectively extend the one for LTL [23] and the LTL+P system presented in the previous section. Soundness and completeness of the systems are proved by employing the same model-theoretic argument of the proofs shown in Section 4. The two systems are very similar, differing only in specific parts and sharing the vast majority of their workings. Hence, a common skeleton is first described, making some assumptions that will then be fulfilled for the two specific logics.

5.1. The common skeleton

The parts in common between the two tableau systems will be presented as if they were supposed to handle TPTL+P formulae. TPTL is a proper fragment of TPTL+P, and TPTL$_\mathsf{b}$+P, as it will be shown later, can be fully embedded in a proper guarded fragment of TPTL+P. Hence, both tableaux do indeed handle TPTL+P formulae, albeit of a specific kind. We will mention the specific logics when stating results that do not hold for full TPTL+P.

W.l.o.g., we may assume that formulae are given in *negated normal form*, which is guaranteed to exist for formulae of both logics. Moreover, as shown in [4] for TPTL and in [10] for TPTL$_\mathsf{b}$+P, *w.l.o.g.*, we can also restrict ourselves to models with a bound on the maximum *temporal* distance between two subsequent states. A model $\rho = (\sigma, \tau)$ of ϕ is said to be δ*-bounded*, for some $\delta \geq 0$, if $\tau_{i+1} - \tau_i \leq \delta$ for all $i \geq 0$. The following result is known [4, 10].

Proposition 1 (δ-bounded models [4, 10])**.** *Let ϕ be a closed* TPTL *or* TPTL$_\mathsf{b}$+P *formula. Then, a natural number $\delta_\phi \geq 0$ can be computed from ϕ such that ϕ is satisfiable if and only ϕ has a δ_ϕ-bounded model.*

For both logics, the value of δ_ϕ can be computed from the constants appearing in ϕ [4, 10]: roughly, δ_ϕ is the product of all the constants in ϕ. Similarly, we can assume that no *absolute* timing constraints (those of the form $x \leq c$) are used in the formulae (see Lemma 6 in [4]). Finally, *w.l.o.g.*, we can also assume that any variable x is used only in one freeze quantifier in any formula, so that in a formula like $x.\psi$ any occurrence of x in ψ is free. Since freeze quantifiers can be pushed out of Boolean connectives, when talking about closed formulae we will write them as $x.\psi$, with explicit reference to the outermost freeze quantifier.

We start by defining an important building block of the system.

Definition 8 (Temporal shift). Let us denote as wff the set of all the well-formed $\mathsf{TPTL{+}P}$ formulae. The *temporal shift operator* is a function $\cdot^\delta : \mathsf{wff} \times \mathbb{Z} \rightarrow \mathsf{wff}$ such that:

1. for any closed $\mathsf{TPTL{+}P}$ formula $x.\psi$ and any $\delta \in \mathbb{Z}$, timed state sequence ρ, environment ξ, and position $i \geq 0$, it holds that $\rho \models^i_\xi x.\psi^\delta$ if and only if $\rho \models^i_{\xi'} \psi$, where $\xi' = \xi[x \leftarrow \tau_i - \delta]$;

2. there exists $\delta' \in \mathbb{Z}$ such that $x.\psi^{\delta'} = x.\psi^{\delta'+i}$ and $x.\psi^{-\delta'} = x.\psi^{-\delta'-i}$ for all $i \geq 0$.

Note that we write $x.\psi^\delta$ to mean $(x.\psi)^\delta$, and the definition of the temporal shift operator explicitly refers to the outermost quantified variable x. Item 1 of Definition 8 states that the truth value of $x.\psi^\delta$, interpreted at the current state, is the same as that of ψ in the case where x was bound to the timestamp of a previous state located exactly δ time units before. By Item 2, this transformation has to be defined in such a way that it converges to a fixed point after a large enough amount of shifting, so that for a given $x.\psi$, the number of different formulae of the form $x.\psi^\delta$ is finite. It is not known whether such an operator exists for full $\mathsf{TPTL{+}P}$. Later, we will show how to define it in the cases of TPTL and $\mathsf{TPTL_b{+}P}$.

The *closure* of a formula $z.\phi$ contains all the formulae that are relevant to the satisfaction of $z.\phi$.

Definition 9 (Closure of a formula). Let $z.\phi$ be a closed $\mathsf{TPTL{+}P}$ formula and let $\cdot^\delta$ be a temporal shift operator. Then, the *closure* of $z.\phi$ is the set $\mathcal{C}(z.\phi)$ recursively defined as follows:

1. $z.\phi \in \mathcal{C}(z.\phi)$;

2. if $x.(\psi_1 \wedge \psi_2) \in \mathcal{C}(z.\phi)$, then $\{x.\psi_1, x.\psi_2\} \subseteq \mathcal{C}(z.\phi)$;
3. if $x.(\psi_1 \vee \psi_2) \in \mathcal{C}(z.\phi)$, then $\{x.\psi_1, x.\psi_2\} \subseteq \mathcal{C}(z.\phi)$;
4. if $x.\mathsf{X}\psi \in \mathcal{C}(z.\phi)$, then $x.\psi^{\delta} \in \mathcal{C}(z.\phi)$, for all $\delta \geq 0$;
5. if $x.\mathsf{Y}\psi \in \mathcal{C}(z.\phi)$, then $x.\psi^{-\delta} \in \mathcal{C}(z.\phi)$, for all $\delta \geq 0$;
6. if $x.\mathsf{Z}\psi \in \mathcal{C}(z.\phi)$, then $x.\psi^{-\delta} \in \mathcal{C}(z.\phi)$, for all $\delta \geq 0$;
7. if $x.(\psi_1 \circ \psi_2) \in \mathcal{C}(z.\phi)$, for $\circ \in \{\mathcal{U}, \mathcal{R}\}$, then $\{x.\psi_1, x.\psi_2, x.\mathsf{X}(\psi_1 \circ \psi_2)\} \subseteq \mathcal{C}(z.\phi)$;
8. if $x.(\psi_1 \circ \psi_2) \in \mathcal{C}(z.\phi)$, for $\circ \in \{\mathcal{S}, \mathcal{T}\}$, then $\{x.\psi_1, x.\psi_2, x.\mathsf{Y}(\psi_1 \circ \psi_2)\} \subseteq \mathcal{C}(z.\phi)$;
9. if $x.y.\psi \in \mathcal{C}(z.\phi)$, then $x.\psi[y/x] \in \mathcal{C}(z.\phi)$, *i.e.*, y is replaced with x in the formula, collapsing the $y.\phi$ quantifier into x.

Note that, if $\cdot^{\delta}$ is a temporal shift operator, then $\mathcal{C}(z.\phi)$ is a finite set, thanks to Item 2 of Definition 8. Moreover, note that, by construction, every formula in $\mathcal{C}(z.\phi)$ is a *closed* formula.

Now we can effectively start describing the common skeleton of the one-pass and tree-shaped tableau systems for TPTL and $\mathsf{TPTL_b{+}P}$. The basic structure of the system is very similar to the $\mathsf{LTL{+}P}$ tableau. A tableau for a $\mathsf{TPTL{+}P}$ formula $x.\phi$ is a tree where any node u is labelled by a subset $\Gamma(u) \subseteq \mathcal{C}(x.\phi)$ of the closure of $x.\phi$. The tree is built, starting from the root u_0 with $\Gamma(u_0) = \{x.\phi\}$, by applying a set of rules to each leaf. Each rule can potentially create some children, thus creating new leaves from which the process can continue, or close the branch by accepting or rejecting it.

The set of rules is similar to those used in the $\mathsf{LTL{+}P}$ tableau. The *expansion rules* for $\mathsf{TPTL{+}P}$ are shown in Table 3, and are very similar to those for $\mathsf{LTL{+}P}$. They look for a formula $x.\psi$ in $\Gamma(u)$, and create two children u' and u'' such that $\Gamma(u') = \Gamma(u) \setminus \{x.\psi\} \cup \Gamma_1(u)$ and $\Gamma(u'') = \Gamma(u) \setminus \{x.\psi\} \cup \Gamma_2(u)$, or a single child u' if $\Gamma_2(u)$ is empty. Apart from the different syntax of the formulae themselves, it can be seen that the expansions of each formula are mostly unchanged with regards to $\mathsf{LTL{+}P}$.

Atomic formulae such as propositions and timing constraints, and *tomorrow* and *yesterday* operators, are considered *elementary* formulae. Nodes whose labels contain only elementary formulae are called *poised* nodes. The

Rule	$\phi \in \Gamma$	$\Gamma_1(\phi)$	$\Gamma_2(\phi)$
DISJUNCTION	$x.\psi_1 \vee x.\psi_2$	$\{x.\psi_1\}$	$\{x.\psi_2\}$
CONJUNCTION	$x.\psi_1 \wedge x.\psi_2$	$\{x.\psi_1, x.\psi_2\}$	
UNTIL	$x.(\psi_1 \,\mathcal{U}\, \psi_2)$	$\{x.\psi_2\}$	$\{x.\psi_1, x.\mathsf{X}(\psi_1 \,\mathcal{U}\, \psi_2)\}$
RELEASE	$x.(\psi_1 \,\mathcal{R}\, \psi_2)$	$\{x.\psi_1, x.\psi_2\}$	$\{x.\psi_2, x.\mathsf{X}(\psi_1 \,\mathcal{R}\, \psi_2)\}$
EVENTUALLY	$x.\mathsf{F}\psi_1$	$\{x.\psi_1\}$	$\{x.\mathsf{XF}\psi_1\}$
ALWAYS	$x.\mathsf{G}\psi_1$	$\{x.\psi_1, x.\mathsf{XG}\psi_1\}$	
SINCE	$x.(\psi_1 \,\mathcal{S}\, \psi_2)$	$\{x.\psi_2\}$	$\{x.\psi_1, x.\mathsf{Y}(\psi_1 \,\mathcal{S}\, \psi_2)\}$
TRIGGERED	$x.(\psi_1 \,\mathcal{T}\, \psi_2)$	$\{x.\psi_1, x.\psi_2\}$	$\{x.\psi_2, x.\mathsf{Z}(\psi_1 \,\mathcal{T}\, \psi_2)\}$
PAST	$x.\mathsf{P}\psi_1$	$\{x.\psi_1\}$	$\{x.\mathsf{YP}\psi_1\}$
HISTORICALLY	$x.\mathsf{H}\psi_1$	$\{x.\psi_1, x.\mathsf{ZH}\psi_1\}$	

Table 3: Expansion rules for the TPTL+P tableau.

major difference of this system in contrast to the LTL+P one is the STEP rule, which here does not only have to propagate the *tomorrow* requests from a state to the next, but also has to choose how much time has to pass between the two states. This is done by simply creating as many children as are the possible time advancements. Recall that we can assume that the maximum time gap between two states is bounded by $\delta_{\max}$, hence the number of such choices is finite.

STEP Let u be a poised node. Then, $\delta_{\max} + 1$ children nodes $u_0, \ldots, u_{\delta_{\max}}$ are added to u, such that $\Gamma(u_\delta) = \{x.\psi^\delta \mid x.\mathsf{X}\psi \in \Gamma(u)\}$ for all $0 \leq \delta \leq \delta_{\max}$.

Similar to the LTL+P tableau, the STEP rule is not always applied to all poised nodes, but rather the FORECAST rule is applied first to guess which formulae will be needed for future instances of the YESTERDAY or W-YESTERDAY rules. Only when the expansion of such nodes leads again to poised nodes, the STEP rule is applied to them. As a consequence, any poised node has either a number of children added by the FORECAST rule or $\delta_{\max}$ children added by the STEP rule. As in the LTL+P case, given a branch $\overline{u} = \{u_0, \ldots, u_n\}$ of the tableau, we define the *step nodes* as those poised nodes u_i where u_{i+1} is one of the children u_δ, for some $0 \leq \delta \leq \delta_{\max}$, added by the STEP rule. We denote such value of δ as $\delta(u_i)$. For each node u_i in the branch we can thus define a quantity $\mathrm{time}(u_i) = \sum_{0<j\leq i} \delta(u_i)$, which will correspond to the timestamp of the state corresponding to u_i in the model

extracted from the branch. With this notation in place, we can show the remaining rules of the system. The rules are shown in the order they have to be checked on any given poised node. Hence, let $\overline{u} = \langle u_0, \ldots, u_n \rangle$ be a poised branch of the tableau:

EMPTY If $\Gamma(u) = \varnothing$, then, u_n is *ticked.*

CONTRADICTION If $\{x.p, x.\neg p\} \subseteq \Gamma(u_n)$, for some $p \in \Sigma$, then u is *crossed.*

SYNC If either $x.(x \leq x + c) \in \Gamma(u_n)$ or $x.\neg(x \leq x + c) \in \Gamma(u_n)$, but, respectively, $c < 0$ or $c \geq 0$, then u_n is *crossed.*

FORECAST Let G_n be the set of all the formulae involved in any *yesterday* or *weak yesterday* operator related to the formulae of $\Gamma(u_n)$:

$$G_n = \{\alpha \in \mathcal{C}(\phi) \mid \mathsf{Y}\alpha \in \mathcal{C}(\psi) \text{ or } \mathsf{Z}\alpha \in \mathcal{C}(\psi) \text{ for some } \psi \in \Gamma(u_n)\ \}$$

Then, at most once before any application of the STEP rule, for each subset $G'_n \subseteq G_n$ (including the empty set), a child u'_n is added to u_n such that $\Gamma(u'_n) = \Gamma(u_n) \cup G'_n$.

YESTERDAY If $x.\mathsf{Y}\psi \in \Gamma(u_n)$, let u_k be the closest ancestor of u_n where the STEP rule was applied, and let $Y_n = \{x.\psi \mid x.\mathsf{Y}\psi \in \Gamma(u_n)\}$.

Then, the node u_n is *crossed* if u_k does not exist because there is no application of the STEP rule preceding u_n, or if $Y_n \not\subseteq \Gamma^*(u_k)$.

W-YESTERDAY If $x.\mathsf{Z}\psi \in \Gamma(u_n)$, let u_k be the closest ancestor of u_n where the STEP rule was applied, and let $Z_n = \{x.\psi \mid x.\mathsf{Z}\psi \in \Gamma(u_n)\}$.

Then, the node u_n is *crossed* if u_k exists because there is an application of the STEP rule preceding u_n, and $Z_n \not\subseteq \Gamma^*(u_k)$.

LOOP If there exists a step node u_i such that $u_i < u_n$, $\Gamma(u_i) = \Gamma(u_n)$, and all the X-eventualities requested in u_i are fulfilled in $\overline{u}_{[i+1\ldots n]}$, then:

1. if $\text{time}(u_i) = \text{time}(u_n)$, then u_n is *crossed*;
2. if $\text{time}(u_i) < \text{time}(u_n)$, then u_n is *ticked.*

PRUNE If there are two positions i and j such that $i < j \leq n$, $\Gamma(u_i) = \Gamma(u_j) = \Gamma(u_n)$, and among the X-eventualities requested in these nodes, all those fulfilled in $\overline{u}_{[j+1\ldots n]}$ are fulfilled in $\overline{u}_{[i+1\ldots j]}$ as well, then u_n is *crossed.*

In comparison with the rules for the $\mathsf{LTL{+}P}$ tableau, an additional SYNC rule is present which, similarly to the $\mathsf{CONTRADICTION}$ rule, checks for contradictions, but regarding the timing constraints. The STEP rule, thanks to the temporal shift operator, can push freeze quantifiers to the next state, without explicitly keeping track of variable bindings. In such a way, it ensures that nodes are labelled only by closed formulae of the form $x.\psi$, the base case of timing constraints consisting only in formulae of the form $x.(x \sim x + c)$, which involve a single variable. Judging the satisfiability of these timing constraints is then trivial. This mechanism was originally exploited in the graph-shaped tableau for TPTL given in [4]. The LOOP rule handles the case where the branch is cycling through a segment which fulfils all the requests, and thus a model of the formula has been found. However, since timed state sequences must satisfy the *progress* property, the rule has to reject those branches where the loop has not advanced in time (LOOP_1) and to accept a branch only if some progress has been made (LOOP_2).

If we suppose the existence of a temporal shift operator for $\mathsf{TPTL{+}P}$ formulae, the closure set defined in Definition 9 is finite. Moreover, the branching factor of the tree is finite as well thanks to Proposition 1. Hence, an argument totally similar to that employed for Theorem 1 would guarantee that the construction of a tableau for a $\mathsf{TPTL{+}P}$ formula terminates. However, whether a temporal shift operator exists for general $\mathsf{TPTL{+}P}$ formulae is still an open question. The following part of this section shows how to specialise this generic tableau system for TPTL and $\mathsf{TPTL_b{+}P}$. This task, among everything, includes the definition of temporal shift operators for the two logics.

5.2. The tableau system for TPTL

Let us now specialise the above general rules to TPTL formulae. Basically, we need to define a temporal shift operator for the logic. Consider a TPTL formula $x.\psi \in \mathcal{C}(z.\phi)$, and any other variable y appearing in ψ. Since $x.\psi$ is a closed formula, y must be quantified inside ψ, and, since ψ is a future-only formula, it can only be bound to a timestamp greater than or equal to x. Hence, any timing constraint of the form $x \leq y + c$, with $c \geq 0$, always holds regardless of the specific evaluation of the variables. A similar consideration can be made for timing constraints of the form $y \leq x + c$, with $c < 0$, which are always false. This fact, originally exploited by Alur and Henzinger [4], leads to the following definition of the temporal shift operator for TPTL formulae.

Definition 10 (Temporal shift operator for TPTL formulae [4]). Let δ be a natural number and $x.\psi$ be a *closed* TPTL formula. Then, $x.\psi^\delta$ is the formula obtained by applying the following steps:

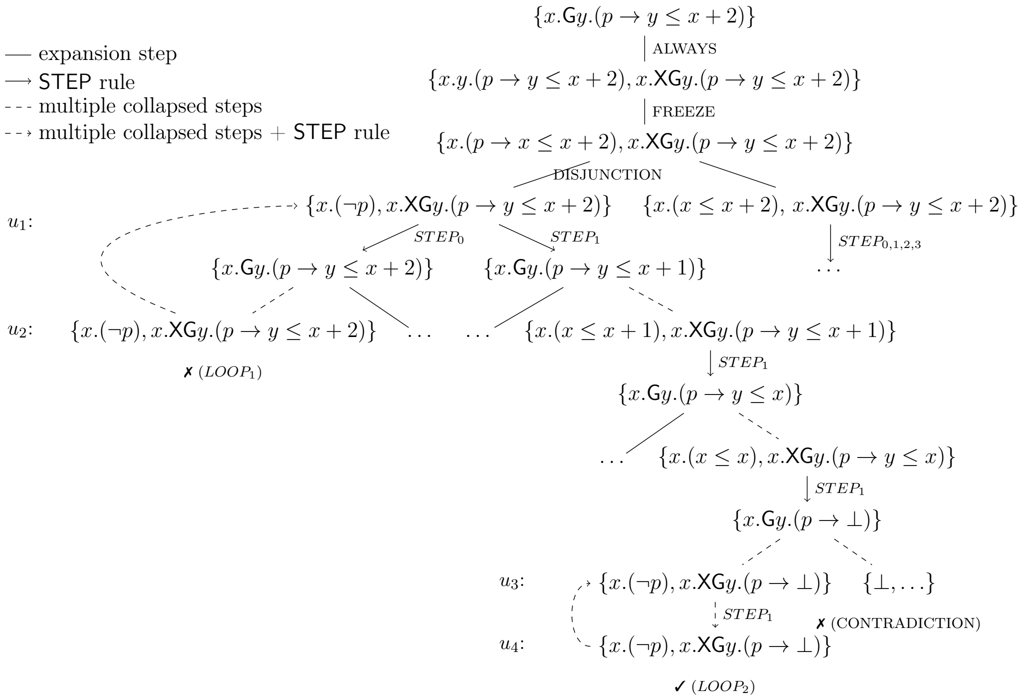
1. replace any timing constraint of the forms $x \leq y + c$ and $y \leq x + c$, for any other variable $y \in \mathcal{V}$, by, respectively, $x \leq y + c'$ and $y \leq x + c''$, where $c' = c + \delta$ and $c'' = c - \delta$; and then

2. replace any timing constraint of the forms $x \leq y + c'$ and $y \leq x + c''$, with $c' \geq 0$ and $c'' < 0$, by, respectively, $\top$ and $\bot$.

We give a brief example to show an application of the temporal shift operator; let $x.\psi$ be the formula $x.\mathsf{G}y.(p \to y \leq x + 2)$; we have that $x.\psi^1$ and $x.\psi^2$ corresponds to $x.\mathsf{G}y.(p \to y \leq x + 1)$ and $x.\mathsf{G}y.(p \to y \leq x)$, respectively; finally, $x.\psi^\delta$ corresponds to $x.\mathsf{G}y.(p \to \bot)$, for all $\delta \geq 3$.

The one-pass and tree-shaped tableau system for TPTL is obtained from the set of rules of Section 5.1 by considering the temporal shift operator of Definition 10. Alur and Henzinger [4] prove that Definition 10 satisfies the requirements of Definition 8 for any non-negative $\delta \geq 0$. Since the $\mathsf{YESTERDAY}$ and $\mathsf{W\text{-}YESTERDAY}$ rules never come into play with TPTL formulae, this is sufficient, as that is the only reason for applying the temporal shift operator with a negative shift amount.

To see the system in action, Fig. 4 shows the tableau for the example TPTL formula $x.\mathsf{G}y.(p \to y \leq x + 2)$. When interpreted at the start of a timed state sequence, the formula expresses the property that p holds only on states with timestamp less than 2. Firstly we focus on node u_2: it is crossed by the LOOP_1 rule because there is another node (*i.e.*, u_1) such that all the conditions of the LOOP rule are satisfied but time does *not* increase between these two nodes. Nevertheless, if we choose to increment by one time unit the candidate model by means of the STEP rule, we eventually reach node u_3, which does not contain any timed constraint, since they all have been simplified by the temporal shift $\cdot^\delta$. Now the rule LOOP_2 can be applied on node u_4, since nodes u_3 and u_4 have the same label, all the X-eventualities (there are none) are fulfilled in between, and the time between u_3 and u_4 *does* increase: thus, we tick u_4 and accept the corresponding branch. This, in turn, corresponds to a correct model of the input formula which starts from the root of the tableau, goes down to u_4 and then cycles between u_3 and u_4.



— expansion step
⟶ STEP rule
- - - multiple collapsed steps
- - → multiple collapsed steps + STEP rule
$\{x.\mathsf{G}y.(p \to y \leq x+2)\}$
ALWAYS
$\{x.y.(p \to y \leq x+2), x.\mathsf{XG}y.(p \to y \leq x+2)\}$
FREEZE
$\{x.(p \to x \leq x+2), x.\mathsf{XG}y.(p \to y \leq x+2)\}$
DISJUNCTION
u_1:
$\{x.(\neg p), x.\mathsf{XG}y.(p \to y \leq x+2)\}$
$\{x.(x \leq x+2), x.\mathsf{XG}y.(p \to y \leq x+2)\}$
$STEP_0$
$STEP_1$
$STEP_{0,1,2,3}$
$\{x.\mathsf{G}y.(p \to y \leq x+2)\}$
$\{x.\mathsf{G}y.(p \to y \leq x+1)\}$
...
u_2:
$\{x.(\neg p), x.\mathsf{XG}y.(p \to y \leq x+2)\}$
...
...
$\{x.(x \leq x+1), x.\mathsf{XG}y.(p \to y \leq x+1)\}$
✗ ($LOOP_1$)
$STEP_1$
$\{x.\mathsf{G}y.(p \to y \leq x)\}$
...
$\{x.(x \leq x), x.\mathsf{XG}y.(p \to y \leq x)\}$
$STEP_1$
$\{x.\mathsf{G}y.(p \to \perp)\}$
u_3:
$\{x.(\neg p), x.\mathsf{XG}y.(p \to \perp)\}$
$\{\perp, \ldots\}$
$STEP_1$
✗ (CONTRADICTION)
u_4:
$\{x.(\neg p), x.\mathsf{XG}y.(p \to \perp)\}$
✓ ($LOOP_2$)

Figure 4: The tableau for the formula $x.\mathsf{G}y.(p \to y \leq x+2)$

5.3. The tableau system for $\mathsf{TPTL_b{+}P}$

Let us now specialise the above set of rules to $\mathsf{TPTL_b{+}P}$. $\mathsf{TPTL_b{+}P}$ is not a proper fragment of $\mathsf{TPTL{+}P}$ *as-is*, and thus it may seem that those rules cannot be directly applied to $\mathsf{TPTL_b{+}P}$ formulae. However, $\mathsf{TPTL_b{+}P}$ can be embedded into a *guarded* fragment of $\mathsf{TPTL{+}P}$, that is, a syntactic fragment of the logic, that we call $\mathsf{G(TPTL{+}P)}$, where each occurrence of any temporal operator is guarded by an additional formula which implements the bounded semantics of $\mathsf{TPTL_b{+}P}$ operators. We now define the $\mathsf{G(TPTL{+}P)}$ fragment and show how $\mathsf{TPTL_b{+}P}$ formulae can be translated into $\mathsf{G(TPTL{+}P)}$. Then, we show how to define a temporal shift operator for $\mathsf{G(TPTL{+}P)}$ formulae, obtaining the one-pass and tree-shaped tableau system for $\mathsf{TPTL_b{+}P}$. $\mathsf{G(TPTL{+}P)}$ syntax is defined as follows:

$$\begin{aligned}\phi := {} & p \mid \neg p \mid \phi_1 \vee \phi_2 \mid x \leq y + c \mid x \leq c \\ & \mid x.\mathsf{X}y.(\gamma_w^{x,y} \wedge \phi_1) \mid x.\mathsf{X}y.(\gamma_w^{x,y} \to \phi_1) \\ & \mid x.\mathsf{Y}y.(\gamma_w^{x,y} \wedge \phi_1) \mid x.\mathsf{Y}y.(\gamma_w^{x,y} \to \phi_1)\end{aligned}$$

$$\begin{aligned}&\mid x.\big(z.(\gamma_w^{x,z} \to \phi_1)\,\mathcal{U}\,y.(\gamma_w^{x,y} \wedge \phi_2)\big) \mid x.\big(z.(\gamma_w^{x,z} \wedge \phi_1)\,\mathcal{R}\,y.(\gamma_w^{x,y} \to \phi_2)\big)\\ &\mid x.\big(z.(\gamma_w^{x,z} \to \phi_1)\,\mathcal{S}\,y.(\gamma_w^{x,y} \wedge \phi_2)\big) \mid x.\big(z.(\gamma_w^{x,z} \wedge \phi_1)\,\mathcal{T}\,y.(\gamma_w^{x,y} \to \phi_2)\big)\end{aligned}$$

where $\gamma_w^{x,y} = y \leq x + w$, if $w \neq +\infty$, and $\gamma_w^{x,y} = \top$ otherwise, with $w \in \mathbb{N} \cup \{+\infty\}$ and x and y fresh in ϕ_1 and ϕ_2. Moreover, as in $\mathsf{TPTL_b{+}P}$, each temporal operator can appear with $w = +\infty$ only if the corresponding formula is closed. All the temporal operators where $w \neq +\infty$ are called *guarded*.

One can check that (i) the *negated normal form* of a $\mathsf{G(TPTL{+}P)}$ formula is still a $\mathsf{G(TPTL{+}P)}$ formula, and (ii) each $\mathsf{TPTL_b{+}P}$ formula can be translated into an equivalent $\mathsf{G(TPTL{+}P)}$ one. A notable example is the translation of the X and $\widetilde{\mathsf{X}}$ operators (and, symmetrically, Y and $\widetilde{\mathsf{Y}}$), that both get translated into a formula using a guarded X operator, but with the guard that, respectively, is conjunct to and implies the target formula, *i.e.*, $\mathsf{X}_w\psi \equiv x.\mathsf{X}y.(y \leq x + w \wedge \psi)$ and $\widetilde{\mathsf{X}}_w\psi \equiv x.\mathsf{X}y.(y \leq x + w \to \psi)$, if $w \neq +\infty$, and simply $\mathsf{X}_{+\infty}\psi \equiv \widetilde{\mathsf{X}}_{+\infty}\psi \equiv \mathsf{X}\psi$ otherwise. The translation provides a semantically faithful embedding of $\mathsf{TPTL_b{+}P}$ into (the $\mathsf{G(TPTL{+}P)}$ syntactic fragment of) $\mathsf{TPTL{+}P}$. Formally, the following result can be shown.

Lemma 7 (Translation of $\mathsf{TPTL_b{+}P}$ formulae into $\mathsf{G(TPTL{+}P)}$)**.** *Let ϕ be a $\mathsf{TPTL_b{+}P}$ formula over the proposition letters Σ and the variables $\mathcal{V}$. Then, ϕ can be effectively turned into a $\mathsf{G(TPTL{+}P)}$ formula ϕ' such that for any timed state sequence ρ, any environment ξ, and any $i \geq 0$, it holds that $\rho, i \models_\xi \phi$ if and only if $\rho, i \models_\xi \phi'$.*

Proof. In order to show the effectiveness of the translation from $\mathsf{TPTL_b{+}P}$ formulae to $\mathsf{G(TPTL{+}P)}$ formulae, we define the function $\mathbb{G}$ from the set of all $\mathsf{TPTL_b{+}P}$ formulae to the set of all $\mathsf{G(TPTL{+}P)}$ formulae, as follows:

- $\mathbb{G}(\varphi) = \varphi$, for any atomic proposition or timing constraint φ;
- $\mathbb{G}(\neg\varphi) = \neg\varphi$, for any atomic proposition or timing constraint φ;
- $\mathbb{G}(z.\phi) = z.\mathbb{G}(\phi)$ for every ϕ in the closure;
- $\mathbb{G}(\mathsf{X}_w\phi_1) = \begin{cases} x'.\mathsf{X}y'.(y' \leq x' + w \wedge \mathbb{G}(\phi_1)) & \text{if } w \in \mathbb{N} \\ \mathsf{X}\mathbb{G}(\phi_1) & \text{if } w = +\infty \end{cases}$
- $\mathbb{G}(\widetilde{\mathsf{X}}_w\phi_1) = \begin{cases} x'.\mathsf{X}y'.(y' \leq x' + w \to \mathbb{G}(\phi_1)) & \text{if } w \in \mathbb{N} \\ \mathsf{X}\mathbb{G}(\phi_1) & \text{if } w = +\infty \end{cases}$

- $\mathbb{G}(\phi_1 \,\mathcal{U}_w \phi_2) = \begin{cases} x'.(\mathbb{G}(\phi_1)\,\mathcal{U}\, y'.(y' \leq x' + w \wedge \mathbb{G}(\phi_2))) & \text{if } w \in \mathbb{N} \\ \mathbb{G}(\phi_1)\,\mathcal{U}\,\mathbb{G}(\phi_2) & \text{if } w = +\infty \end{cases}$

- $\mathbb{G}(\phi_1 \,\mathcal{R}_w \phi_2) = \begin{cases} x'.(\mathbb{G}(\phi_1)\,\mathcal{R}\, y'.(y' \leq x' + w \rightarrow \mathbb{G}(\phi_2))) & \text{if } w \in \mathbb{N} \\ \mathbb{G}(\phi_1)\,\mathcal{U}\,\mathbb{G}(\phi_2) & \text{if } w = +\infty \end{cases}$

- $\mathbb{G}(\mathsf{Y}_w \phi_1) = \begin{cases} x'.\mathsf{Y}y'.(x' \leq y' + w \wedge \mathbb{G}(\phi_1)) & \text{if } w \in \mathbb{N} \\ \mathsf{Y}\mathbb{G}(\phi_1) & \text{if } w = +\infty \end{cases}$

- $\mathbb{G}(\widetilde{\mathsf{Y}}_w \phi_1) = \begin{cases} x'.\mathsf{Z}y'.(x' \leq y' + w \rightarrow \mathbb{G}(\phi_1))) & \text{if } w \in \mathbb{N} \\ \mathsf{Z}\mathbb{G}(\phi_1) & \text{if } w = +\infty \end{cases}$

- $\mathbb{G}(\phi_1 \,\mathcal{S}_w \phi_2) = \begin{cases} x'.(\mathbb{G}(\phi_1)\,\mathcal{S}\, y'.(x' \leq y' + w \wedge \mathbb{G}(\phi_2))) & \text{if } w \in \mathbb{N} \\ \mathbb{G}(\phi_1)\,\mathcal{S}\,\mathbb{G}(\phi_2) & \text{if } w = +\infty \end{cases}$

- $\mathbb{G}(\phi_1 \,\mathcal{T}_w \phi_2) = \begin{cases} x'.(\mathbb{G}(\phi_1)\,\mathcal{T}\, y'.(x' \leq y' + w \rightarrow \mathbb{G}(\phi_2))) & \text{if } w \in \mathbb{N} \\ \mathbb{G}(\phi_1)\,\mathcal{S}\,\mathbb{G}(\phi_2) & \text{if } w = +\infty \end{cases}$

It holds that $\rho, i \models_\xi \phi$ if and only if $\rho, i \models_\xi \mathbb{G}(\phi)$, for all $\mathsf{TPTL_b{+}P}$ formulae ϕ, timed state sequences ρ, environments ε, and positions $i \geq 0$. The proof goes by induction on the complexity of ϕ; the base cases, that is if ϕ is a positive or negative occurrence of an atomic proposition or timing constraint, are trivial; as for the inductive step, we sketch the proof showing the case for the $\mathcal{U}_w$ operator. Suppose therefore that $\rho \models^i_\xi \phi_1 \,\mathcal{U}_w \phi_2$; by the semantics of $\mathcal{U}_w$ (see Section 2.3), it holds that there exists a position $j \geq i$ such that: (i) $\tau_j - \tau_i \leq w$; (ii) $\rho, j \models_\xi \phi_2$ and (iii) $\rho, k \models_\xi \phi_i$ for all k such that $j < k \leq i$. By inductive hypothesis, $\rho, j \models_\xi \mathbb{G}(\phi_2)$; therefore, by points (i) and (ii) we know that $\rho, j \models_{\xi_{[x' \leftarrow \tau_i]}} y'.(y' \leq x' + w \wedge \mathbb{G}(\phi_2))$. Moreover, we know that by inductive hypothesis $\rho, k \models_\xi \mathbb{G}(\phi_1)$ for all $i < k \leq j$. Overall, this means that $\rho, i \models_\xi x'.(\mathbb{G}(\phi_1)\,\mathcal{U}\, y'.(y' \leq x' + w \wedge \mathbb{G}(\phi_2)))$. The opposite direction can be simply proved by applying the semantics of the $\mathcal{U}$ operator. □

Hence, we can apply the general tableau rules to the $\mathsf{G(TPTL{+}P)}$ translation of any $\mathsf{TPTL_b{+}P}$ formula, provided that, similar to the TPTL case, a temporal shift operator can be defined. This can actually be done by exploiting the following observation: thanks to the bounds applied to the $\mathsf{TPTL_b{+}P}$ temporal operators, whose semantics is implemented in $\mathsf{G(TPTL{+}P)}$ formulae by means of the guards, when interpreting a timing constraint like $x \leq y + c$, the distance between variables x and y cannot be greater than an upper bound W that depends on the bounds applied to the temporal operators of the formula. This observation was exploited in [10] to prove decidability and $\mathsf{EXPSPACE}$-completeness of $\mathsf{TPTL_b{+}P}$. Now, given a $\mathsf{G(TPTL{+}P)}$ formula $z.\phi$, let m be the number of *guarded* temporal operators used in $z.\phi$, let $w_0 = \max\{w_1, \ldots, w_m, \delta_{z.\phi}\}$, where $w_1, \ldots, w_m$ are the bounds applied to the respective guarded temporal operators and $\delta_{z.\phi}$ is computed as per Proposition 1, and let $W_{z.\phi} = w_0 \cdot (m + 1)$.

Definition 11 (Temporal shift operator for $\mathsf{G(TPTL{+}P)}$). Let $z.\phi$ be a *closed* $\mathsf{G(TPTL{+}P)}$ formula, $\delta \in \mathbb{N}$, and $x.\psi \in \mathcal{C}(z.\phi)$. Then, $x.\psi^\delta$ is the formula obtained by applying the following steps:

1. replace any timing constraint of the forms $x \leq y + c$ and $y \leq x + c$, for any other variable $y \in V$, by, respectively, $x \leq y + c'$ and $y \leq x + c''$, where $c' = c + \delta$ and $c'' = c - \delta$; and then

2. replace any timing constraint of the forms $x \leq y + c$ and $y \leq x + c$ either by $\top$, if $c \geq W_{z.\phi}$, or by $\bot$, if $c < -W_{z.\phi}$.

We will now show that Definition 11 indeed defines a temporal shift operator as per Definition 8.[3] It can be easily seen that Item 2 of Definition 8 is satisfied: in $x.\phi^\delta$, the coefficients of $x.\phi$ are increased or decreased accordingly. With a large enough δ, all the coefficients in $x.\phi^\delta$ (and $x.\phi^{-\delta}$) hit the $W_{x.\phi}$ limit, turning all the timing constraints into $\top$ or $\bot$. The more involved part is the satisfaction of Item 1 of Definition 8.

Lemma 8 (Temporal shift operator for $\mathsf{G(TPTL{+}P)}$). *Let $x.\phi$ be a closed $\mathsf{G(TPTL{+}P)}$ formula, $\rho = (\overline{\sigma}, \overline{\tau})$ a timed state sequence and ξ an environment. Consider a position $i \geq 0$ and $\delta \in \mathbb{Z}$ such that $\delta \leq \tau_i$.*

Then, for any formula $z.\psi \in \mathcal{C}(x.\phi)$, it holds that:

$$\rho, i \models_\xi z.\psi^\delta \text{ iff } \rho, i \models_{\xi'} \psi,$$

where $\xi' = \xi[x \leftarrow \tau_i - \delta]$.

[3] We stated this fact in [10], but the proof has never been properly published.

Proof. Let us first define some notation. For each $\mathsf{TPTL_b}$+P formula ψ, let $\deg(\psi)$ be the *temporal nesting* of ψ, *i.e.*, the nesting degree defined counting only *bounded* temporal operators. If ψ is a subformula of $x.\phi$, let $\mathrm{d}(\psi) = \deg(x.\phi) - \deg(\psi) + 1$. Observe that $\mathrm{d}(x.\phi) = 1$ and that the value of $\mathrm{d}(\psi)$ is maximal when ψ is atomic (*i.e.*, a literal or a timing constraint). Note, moreover, that since the closure of an *until* or *since* operator can increase the temporal nesting by one, it may be that $\mathrm{d}(z.\psi) = 0$ (*e.g.*, when $z.\psi$ is $x.\mathsf{X}_w\phi$). Recall that $W = w_{\max} \cdot (m+1)$ where m is the number of temporal operators with finite bound that appear in $x.\phi$, and $w_{\max}$ is the maximum one. Thus, since $\deg(z.\psi) \leq m+1$, observe that $W \geq w_{\max} \cdot \deg(z.\psi)$.

We first prove a more general claim, namely that if ψ is a *subformula*, *not necessarily closed*, of some formula $x.\psi' \in \mathcal{C}(x.\phi)$ (including itself), and ξ is an environment such that $|\xi(y) - \tau_i| \leq w_{\max} \cdot \mathrm{d}(\psi)$ for any variable y that is *free* in ψ, then $\rho, i \models_\xi x.\psi^\delta$ if and only if $\rho, i \models_{\xi[x \leftarrow \tau_i - \delta]} \psi$. The thesis then follows as a special case, since all the $x.\psi \in \mathcal{C}(\phi)$ are closed and the above condition on ξ is trivially satisfied if there are no free variables.

The claim is proved by structural induction on $x.\psi$. The first base case $x.p$ for $p \in \Sigma$ is trivial since $x.p^\delta \equiv p$. The interesting base case is when $x.\psi$ is a timing constraint involving x. If $x.\psi \equiv x.(x \leq z + c)$, we must distinguish the following two cases.

1. If $|c+\delta| \leq W$, then $x.\psi^\delta \equiv x.(x \leq z + (c+\delta))$. In this case, $\rho, i \models_\xi x.\psi^\delta$ if and only if $\rho, i \models_{\xi[x \leftarrow \tau_i]} x \leq z + (c+\delta)$, but the constraint $x \leq z + (c+\delta)$ is equivalent to $x - \delta \leq z + c$, and thus $\rho, i \models_{\xi[x \leftarrow \tau_i - \delta]} x \leq z + c \equiv \psi$.

2. If $c > 0$ and $c + \delta > W$, then $x.\psi^\delta \equiv \top$. Thus, we have to show that it cannot be the case that $\rho, i \not\models_{\xi[x \leftarrow \tau_i - \delta]} \psi$. Going by contradiction, if that was the case, it would imply that $\tau_i - \delta > \xi(z) + c$, and, consequently, it would be impossible that $\tau_i > \xi(z) + W$ since we assumed $|\xi(z) - \tau_i| \leq w_{\max} \cdot \mathrm{d}(x.\psi) \leq W$.

If $x.\psi \equiv x.(z \leq x + c)$ the argument is symmetrical. Hence, the base case holds, and we can consider the inductive step.

The cases of Boolean connectives come directly from the inductive hypothesis. Hence, we focus on temporal operators.

If $x.\psi \equiv x.\mathsf{X}_w\psi'$, then $x.\psi^\delta \equiv x.\mathsf{X}_w\psi'^\delta$. Thus, by considering the semantics of the *tomorrow* operator and the inductive hypothesis, we obtain:

$$\rho, i \models_\xi x.\mathsf{X}_w\psi'^\delta$$

$$\begin{aligned}
\rho, i+1 &\models_{\xi[x \leftarrow \tau_i]} {\psi'}^{\delta} && \text{at the next state}\\
\rho, i+1 &\models_{\xi[x \leftarrow \tau_{i+1}-\delta']} {\psi'}^{\delta} && \text{where } \delta' = \tau_{i+1} - \tau_i\\
\rho, i+1 &\models_{\xi} x.{\psi'}^{\delta+\delta'} && \text{by the ind. hyp.}\\
\rho, i+1 &\models_{\xi[x \leftarrow \tau_{i+1}-\delta-\delta']} \psi' && \text{by the ind. hyp.}\\
\rho, i+1 &\models_{\xi[x \leftarrow \tau_i-\delta]} \psi' && \text{since } \tau_{i+1} - \delta' = \tau_i\\
\rho, i &\models_{\xi[x \leftarrow \tau_i-\delta]} \mathsf{X}_w\psi' && \text{back one state}
\end{aligned}$$

We still have to check that the inductive hypothesis was applicable, by showing that $|\xi(y) - \tau_i| \leq w_{\max} \cdot \mathrm{d}(x.\psi)$ implies that $|\xi(y) - \tau_{i+1}| \leq w_{\max} \cdot \mathrm{d}(x.\psi')$ for any variable y that is free in $x.\psi$, and thus free in $x.\psi'$. Observe that if $w = \infty$, then ψ' has to be a closed formula, so there are no free variables y whatsoever, and this condition on ξ is trivially satisfied.

Otherwise, we know that $\delta' \leq w \leq w_{\max}$, and thus we obtain:

$$\begin{aligned}
|\xi(y) - \tau_i| &\leq w_{\max} \cdot \mathrm{d}(\psi)\\
|\xi(y) - \tau_{i+1} - \delta'| &\leq w_{\max} \cdot \mathrm{d}(\psi) && \text{because } \tau_i = \tau_{i+1} - \delta'\\
|\xi(y) - \tau_{i+1}| &\leq w_{\max} \cdot \mathrm{d}(\psi) + \delta' && \text{because } \delta' \geq 0\\
&\leq w_{\max} \cdot \mathrm{d}(\psi) + w_{\max} && \text{because } \delta' \leq w_{\max}\\
&\leq w_{\max} \cdot (\mathrm{d}(\psi) + 1)\\
&\leq w_{\max} \cdot \mathrm{d}(\psi')
\end{aligned}$$

The converse is symmetric, and the argument is similar for the other operators. □

5.4. Soundness and completeness

We conclude the section by proving soundness and completeness of the tableau systems for TPTL and $\mathsf{TPTL_b{+}P}$ shown above. We give the proof for the $\mathsf{TPTL_b{+}P}$ case, the TPTL one being completely similar.

Since the tableau for a $\mathsf{TPTL_b{+}P}$ formula is in fact built on its $\mathsf{G(TPTL{+}P)}$ equivalent, from now on we will directly consider $\mathsf{G(TPTL{+}P)}$ formulae. We will again base our arguments on the notion of *pre-model*, adapted to $\mathsf{G(TPTL{+}P)}$ from Section 4. Here, an atom is thus a set $\Delta \subseteq \mathcal{C}(x.\phi)$ of formulae from the closure of $x.\phi$ that are closed by expansion (by Table 3) and by logical deduction, similarly to Definition 2. Then, pre-models are infinite sequences of such atoms, defined similarly to Definition 3, but suitably adapted to the structure of $\mathsf{G(TPTL{+}P)}$ and of our tableau.

Definition 12 (Pre-models for $\mathsf{G}(\mathsf{TPTL}{+}\mathsf{P})$). Let $x.\phi$ be a $\mathsf{G}(\mathsf{TPTL}{+}\mathsf{P})$ formula. A *pre-model* for $x.\phi$ is a pair $\Pi = (\overline{\Delta}, \overline{\delta})$, where $\overline{\delta} = \langle \delta_0, \delta_1, \ldots \rangle$ is an infinite sequence of non-negative integers $\delta_i \in \mathbb{N}$, and $\overline{\Delta}$ is an infinite sequence $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ of atoms for $x.\phi$ such that, for all $i \geq 0$:

1. $x.\phi \in \Delta_0$;
2. if $x.\mathsf{X}\psi \in \Delta_i$, then $x.\psi^{\delta_{i+1}} \in \Delta_{i+1}$;
3. if $x.\mathsf{Y}\psi \in \Delta_i$, then $i > 0$ and $x.\psi^{-\delta_i} \in \Delta_{i-1}$;
4. if $x.(\psi_1 \mathcal{U} \psi_2) \in \Delta_i$, there is a $j \geq i$ with $x.\psi_2^{\delta_{ij}} \in \Delta_j$ and $x.\psi_1^{\delta_{ik}} \in \Delta_k$ for all $i \leq k < j$, where $\delta_{ij} = \sum_{i<k\leq j} \delta_k$;
5. there are infinitely many positions i such that $\delta_i > 0$;
6. Δ_i is *minimal*, *i.e.*, there is no $x.\psi \in \Delta_i$ such that, if ψ is removed, Δ_i is still an atom and the above items are still satisfied.

With this definition, we can adapt the arguments of Theorems 2 and 3 to prove the following result.

Theorem 4 (The tree-shaped tableau for $\mathsf{TPTL_b}{+}\mathsf{P}$ is sound and complete)**.** *A* $\mathsf{TPTL_b}{+}\mathsf{P}$ *formula is* satisfiable *if and only if the tableau built on its* $\mathsf{G}(\mathsf{TPTL}{+}\mathsf{P})$ *translation has an accepted branch.*

Proof (Soundness). Let $x.\phi$ be the $\mathsf{G}(\mathsf{TPTL}{+}\mathsf{P})$ translation of a $\mathsf{TPTL_b}{+}\mathsf{P}$ formula. From Lemma 7 we know that $x.\phi$ is satisfiable if and only if the original formula is, hence let us focus on the tableau built on $x.\phi$.

To show the soundness of the system, *i.e.*, that if the tableau for ϕ has an accepted branch then the formula is satisfiable, we look at one such accepted branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$ and extract a model for $x.\phi$. Let $\overline{\pi} = \langle \pi_0, \ldots, \pi_m \rangle$ be the sequence of step nodes of $\overline{u}$. A pre-model $\Pi = (\overline{\Delta}, \overline{\delta})$ can be extracted from an accepted branch. A suitable periodic sequence of atoms $\overline{\Delta}$ can be extracted in the same way as in Lemma 2, and $\overline{\delta}$ can be defined such that $\delta_i = \mathrm{time}(\pi_i)$, where π_i is the tableau node corresponding to Δ_i, for the prefix, and consequently in the period. We can check that thanks to the definition of LOOP rule, an accepted branch can only lead to a pre-model satisfying Item 5 of Definition 12. An actual model for $x.\psi$ can then be extracted from Π as in Lemma 1, with arguments totally similar to those used by [10], and suitably computing from $\overline{\delta}$ the absolute timestamps of the timed state sequence, proving the soundness of the system.

(Completeness). To show the completeness, *i.e.*, that the tableau for a satisfiable formula has at least one accepted branch, the argument based on *greedy pre-models* used for LTL and $\mathsf{LTL{+}P}$ in Section 4 can again be adapted to the $\mathsf{G}(\mathsf{TPTL{+}P})$ case. In particular, the definition of *delays* of the requests of X-eventualities is completely similar to that employed in Section 4. Note that these delays still only count the number of atoms from the request of an X-eventuality to its satisfaction, disregarding the timestamps of such atoms. Following the argument used in the proof of Theorem 3, knowing that $x.\phi$ is satisfiable we can suppose to have a *greedy* pre-model $\Pi = (\overline{\Delta}, \overline{\delta})$ for $x.\phi$, and traverse the tree to obtain a branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$, as in Lemma 3. In this traversal, when descending from a step node u_i through the application of the STEP rule, we have to choose the child $u_i^{\delta_{i+1}}$ in the branch, matching the time advancement made by the pre-model. If $\overline{\pi} = \langle \pi_0, \ldots, \pi_m \rangle$ is the sequence of step nodes of $\overline{u}$, then, as in the $\mathsf{LTL{+}P}$ case, by construction we have that $\Delta_i = \Delta(\pi_i)$. In addition to the argument employed in the $\mathsf{LTL{+}P}$ case, we only have to observe that if $\overline{u}$ is not accepting, then u_n cannot have been crossed by the SYNC rule, nor by the LOOP_1 rule, since this would contradict the fact that Π is a pre-model for Π. Then, as in Theorem 3, the node cannot have been crossed by the PRUNE rule either, because it would contradict the assumption that Π is a greedy pre-model. Hence $\overline{u}$ must be an accepted branch, completing the proof. □

6. Conclusions

In this paper, we developed one-pass and tree-shaped tableau systems for $\mathsf{LTL{+}P}$, TPTL, and $\mathsf{TPTL_b{+}P}$, that suitably generalise the tableau system for LTL proposed in [23].

We started with the tableau system for $\mathsf{LTL{+}P}$, which is obtained from that for LTL by adding the YESTERDAY, W-YESTERDAY, and FORECAST rules. We proved its soundness and completeness by using a novel model-theoretic argument, that can be viewed as a semantic counterpart of the PRUNE rule. As a matter of fact, the completeness proof turns out to be much simpler than the one for LTL given in [23]. Then, we developed the tableau systems for TPTL and $\mathsf{TPTL_b{+}P}$, showing that their structure is basically the same except for the associated *temporal shift*, as formulated in Definition 8. Notably, to prove their completeness, we exploited the same model-theoretic argument we used for LTL+P, since the PRUNE rule remains unchanged. Moreover, we showed that they can be viewed as specialisations of a general tableau system for TPTL+P, since TPTL and TPTL$_\mathsf{b}$+P are proper fragments of TPTL+P. Soundness of the tableau system for TPTL+P can be easily proved; however, termination is not guaranteed anymore, as the identification of an appropriate *temporal shift operator* for TPTL+P is still an open question.

There are at least three strengths of the proposed one-pass and tree-shaped tableau systems that are worth pointing out. The first one is *modularity*: all of them are obtained from the original system for LTL by adding a few rules, but leaving the fundamental structure of the system unchanged. The second one is the fact that, in most cases, the construction of the *entire tableau* is not necessary. This is similar to what happens for the incremental version of the two-pass and graph-shaped tableau systems [14]. In contrast to those, however, since the construction of each branch is totally independent from the others, the generation process can be easily *parallelised*.

Possible future developments of this research line include the development of one-pass and tree-shaped tableau systems for other real-time logics, such as, for instance, *Metric Temporal Logic* (MTL) [2]. Moreover, whether a complete and terminating one-pass and tree-shaped tableau system can be provided for the full TPTL+P logic is an important open question.

Acknowledgements

First of all, we would like to thank the reviewers for their very useful comments. We would also like to thank Dario Della Monica, who pointed out a problem with the YESTERDAY rule as defined in [12], and Nicolas Rapin, for his useful feedback on an earlier version of this manuscript. Luca Geatti, Nicola Gigante, and Angelo Montanari acknowledge the support from the GNCS projects “Formal methods for verification and synthesis of discrete and hybrid systems” and “Formal methods for combined verification techniques”, and the PRID project “Efforts in the uNderstanding of Complex interActing SystEms (ENCASE)”.

Bibliography

References

[1] J. D. Allred and U. Ultes-Nitsche. A simple and optimal complementation algorithm for büchi automata. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 46–55, 2018. doi: 10.1145/3209108.3209138.

[2] R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 74–106. Springer, 1991.

[3] R. Alur and T. A. Henzinger. Real-Time Logics: Complexity and Expressiveness. *Information and Computation*, 104(1):35–77, 1993. doi: 10.1006/inco.1993.1025.

[4] R. Alur and T. A. Henzinger. A Really Temporal Logic. *Journal of the ACM*, 41(1):181–204, 1994. doi: 10.1145/174644.174651.

[5] M. Bertello, N. Gigante, A. Montanari, and M. Reynolds. Leviathan: A new LTL satisfiability checking tool based on a one-pass tree-shaped tableau. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 950–956. IJCAI/AAAI Press, 2016. URL `http://www.ijcai.org/Abstract/16/139`.

[6] E. W. Beth. Semantic Entailment and Formal Derivability. *Koninklijke Nederlandse Akademie van Wentenschappen, Proceedings of the Section of Sciences*, 18:309–342, 1955.

[7] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, 1981.

[8] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, editors. *Handbook of Model Checking*. Springer, 2018. ISBN 978-3-319-10574-1. doi: 10.1007/978-3-319-10575-8.

[9] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of Tableau Methods*. Springer, 1999. ISBN 978-0-7923-5627-1. doi: 10.1007/978-94-017-1754-0.

[10] D. Della Monica, N. Gigante, A. Montanari, P. Sala, and G. Sciavicco. Bounded timed propositional temporal logic with past captures timeline-based planning with bounded constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1008–1014, 2017. doi: 10.24963/ijcai.2017/140.

[11] S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science: Finite-State Systems.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016. ISBN 9781107028364. doi: 10.1017/CBO9781139236119.

[12] N. Gigante, A. Montanari, and M. Reynolds. A One-Pass Tree-Shaped Tableau for LTL+Past. In *Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 46 of *EPiC Series in Computing*, pages 456–473. EasyChair, 2017. URL `http://www.easychair.org/publications/paper/340363`.

[13] V. Goranko, A. Kyrilov, and D. Shkatov. Tableau Tool for Testing Satisfiability in LTL: Implementation and Experimental Analysis. *Electronic Notes in Theoretical Computer Science*, 262:113–125, 2010. doi: 10.1016/j.entcs.2010.04.009.

[14] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A Decision Algorithm for Full Propositional Temporal Logic. In *Proceedings of the 5th International Conference on Computer Aided Verification*, volume 697 of *LNCS*, pages 97–109. Springer, 1993. doi: 10.1007/3-540-56922-7_9.

[15] O. Lichtenstein and A. Pnueli. Propositional Temporal Logics: Decidability and Completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000. doi: 10.1093/jigpal/8.1.55.

[16] O. Lichtenstein, A. Pnueli, and L. D. Zuck. The Glory of the Past. In R. Parikh, editor, *Proceedings of the 1st Conference on Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985. doi: 10.1007/3-540-15648-8_16.

[17] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems - Safety.* Springer, 1995. ISBN 978-0-387-94459-3.

[18] N. Markey. Temporal logic with past is exponentially more succinct. *Bulletin of the EATCS*, 79:122–128, 2003.

[19] J. C. McCabe-Dansted and M. Reynolds. A Parallel Linear Temporal Logic Tableau. In *Proceedings of the 8th International Symposium on Games, Automata, Logics and Formal Verification*, volume 256 of *EPTCS*, pages 166–179, 2017. doi: 10.4204/EPTCS.256.12.

[20] I. Pill, S. Semprini, R. Cavada, M. Rovers, R. Bloem, and A. Cimatti. Formal analysis of hardware requirements. In *2006 43rd ACM/IEEE Design Automation Conference*, pages 821–826. IEEE, 2006.

[21] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society, 1977. doi: 10.1109/SFCS.1977.32.

[22] M. Reynolds. More Past Glories. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*, pages 229–240. IEEE Computer Society, 2000. doi: 10.1109/LICS.2000.855772.

[23] M. Reynolds. A New Rule for LTL Tableaux. In *Proc. of the 7^{th} International Symposium on Games, Automata, Logics and Formal Verification*, volume 226 of *EPTCS*, pages 287–301, 2016. doi: 10.4204/EPTCS.226.20.

[24] S. Schwendimann. A New One-Pass Tableau Calculus for PLTL. In *Proceedings of the 7th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1397 of *LNCS*, pages 277–292. Springer, 1998. doi: 10.1007/3-540-69778-0_28.

[25] A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *Journal of the ACM*, 32(3):733–749, 1985. doi: 10.1145/3828.3837.

[26] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

[27] P. Wolper. Temporal Logic Can Be More Expressive. *Information and Control*, 56(1/2):72–99, 1983. doi: 10.1016/S0019-9958(83)80051-5.

[28] P. Wolper. The Tableau Method for Temporal Logic: An Overview. *Logique et Analyse*, 28(110/111):119–136, 1985. URL `http://www.jstor.org/stable/44084125`.