A one-pass tree-shaped tableau for defeasible LTL

Anasse Chafik
CRIL, Univ Artois & CNRS, France

Fahima Cheikh-Alili
CRIL, Univ Artois & CNRS, France

Jean-François Condotta
CRIL, Univ Artois & CNRS, France

Ivan Varzinczak
CRIL, Univ Artois & CNRS, France

Abstract

Defeasible Linear Temporal Logic is a defeasible temporal formalism for representing and verifying exception-tolerant systems. It is based on Linear Temporal Logic (LTL) and builds on the preferential approach of Kraus et al. for non-monotonic reasoning, which allows us to formalize and reason with exceptions. In this paper, we tackle the satisfiability checking problem for defeasible LTL. One of the methods for satisfiability checking in LTL is the one-pass tree shaped analytic tableau proposed by Reynolds. We adapt his tableau to defeasible LTL by integrating the preferential semantics to the method. The novelty of this work is in showing how the preferential semantics works in a tableau method for defeasible linear temporal logic. We introduce a sound and complete tableau method for a fragment that can serve as the basis for further exploring tableau methods for this logic.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases Temporal logic, Non-monotonic reasoning, Tableau Calculi

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Acknowledgements The open access publication of this article was supported by the Alpen-Adria-Universität Klagenfurt, Austria.

1 Introduction

Linear temporal logic (LTL) was introduced by Pnueli [13] as a formal tool for reasoning about programs execution. Many properties that an execution should have can be expressed elegantly using this formalism. The logic LTL is used for systems verification [16]. With advances in technologies, systems became more and more complex, displaying new features and behaviours. One of these behaviours is tolerating exceptions. In more general terms, if an error occurs, within an execution of a program, at certain points of time where it is tolerated, the program can still function properly.

Let us say, for the sake of argument, that there is an execution of a program in which a parameter cannot have a certain value. We notice that, at some given points of time, the execution produces the invalid value in the aforementioned parameter. Nevertheless, we do not mind that the program produces the error at these time points deemed to be harmless. The crucial point is that this behaviour is not present in other, more important, points of time. We want to be sure that the execution still continues and the program functions properly even in the presence of such benign time points.

We want a formalism for verifying properties of executions that can, on one hand, be strictly required at some points of time, and on the other hand, be missing in other points of time. That is why we introduced an extended formalism of LTL, called defeasible linear temporal logic ($LTL^{\sim}$) [6]. It uses the preferential approach of Kraus et al. to non-monotonic reasoning [11] (a.k.a. the KLM approach). The defeasible aspect of $LTL^{\sim}$

adds a new dimension to the verification of a program's execution. We can order time points from the important ones, which we call *normal*, to the lesser and lesser ones. *Normality* in LTL indicates the importance of a time point within an execution compared to others.

We also introduced defeasible versions of the modalities *always* and *eventually*. With these defeasible modalities, we can express properties similar to their classical counterparts, targeting the most normal time points within the execution.

The main goal of this paper is to establish a *satisfiability checking* method for our logic, in particular, for a fragment thereof. In the case of LTL, many tableau methods were proposed in the literature. There are two types of tableau methods: *multi-pass* and *one-pass* tableaux. Multi-pass tableau methods [22, 12, 10] go through an initial phase of building a tree-shaped structure by putting the sentence in the root node and expanding the tableau via a systematic application of a set of rules. The second phase is a *culling* phase, which uses an auxiliary structure built from the tableau, and checks for the satisfiability of the input sentence in this structure. Whereas in *one-pass* tableau methods [17, 14], the construction and the verification are done simultaneously. Reynolds' tableau for LTL [15, 14] is a tree-shaped one-pass tableau where each branch is independent from the others. Moreover, each successful branch by itself is a representation of an interpretation that satisfies the sentence.

As for the KLM approach, tableau methods were developed for the preferential approach of Kraus et al. logic [11] and formalisms extending the preferential approach [9, 4, 5]. In the case of preferential modal logic, Britz and Varzinczak [4] proposed a tree-shaped tableau that builds the ordering relation on worlds at the same time as the tableau is expanded. The tableau method in this paper is based on both the one-pass tableau of Reynolds [14] and the tableau for preferential modal logic by Britz and Varzinczak [4]. The novelty of this paper is in showing how preferential semantics works in a tableau for a fragment of $LTL^{\sim}$.

The plan of this paper goes the following way: We talk briefly about LTL and $LTL^{\sim}$ in Section 2. We then describe a tableau method for a fragment of $LTL^{\sim}$ in Section 3. We show soundness, and completeness of our method in Section 4. Section 5 concludes the paper.

2 Preliminaries

Linear Temporal Logic [1] is a modal logic in which modalities are considered to be temporal operators that describe events happening in different time points over a linearly ordered time-line. Let $\mathscr{P}$ be a finite set of *propositional atoms*. The set of operators in LTL can be split into two parts: the set of *Boolean connectives* $(\neg, \wedge, \vee)$, and that of *temporal operators* $(\Box, \Diamond, \bigcirc)$, where $\Box$ reads as *always*, $\Diamond$ as *eventually*, and $\bigcirc$ as *next*. Let $p \in \mathscr{P}$, sentences in LTL are built up according to the following grammar: $\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Box\alpha \mid \Diamond\alpha \mid \bigcirc\alpha$.

Standard abbreviations are included in LTL, such as: $\top \stackrel{\text{def}}{=} p \vee \neg p$, $\bot \stackrel{\text{def}}{=} p \wedge \neg p$, $\alpha \to \beta \stackrel{\text{def}}{=} \neg\alpha \vee \beta$ and $\alpha \leftrightarrow \beta \stackrel{\text{def}}{=} (\alpha \to \beta) \wedge (\beta \to \alpha)$. There are other temporal operators such as $\mathscr{U}$ (until operator) and $\mathscr{R}$ (release operator) in LTL, but we chose to omit them in this paper.

The temporal semantics structure is a chronological linear succession of time points. We use the set of natural numbers in order to label each of these time points i.e., $(\mathbb{N}, <)$. Hence, a temporal interpretation associates each time point t with a truth assignment of all propositional atoms. A temporal interpretation is defined as follows:

▶ **Definition 1** (Temporal interpretation). *A* temporal interpretation I *is a mapping function* $V : \mathbb{N} \longrightarrow 2^{\mathscr{P}}$ *which associates each time point* $t \in \mathbb{N}$ *with a set of propositional atoms* $V(t)$ *corresponding to the set of propositions that are true in* t. *(Propositions not belonging to* $V(t)$ *are assumed to be false at the given time point.)*

The truth value of a sentence in an interpretation I at a time point $t \in \mathbb{N}$, denoted by $I, t \models \alpha$, is recursively defined as follows:

- $I, t \models p$ if $p \in V(t)$; $I, t \models \neg\alpha$ if $I, t \not\models \alpha$;
- $I, t \models \alpha \wedge \alpha'$ if $I, t \models \alpha$ and $I, t \models \alpha'$; $I, t \models \alpha \vee \alpha'$ if $I, t \models \alpha$ or $I, t \models \alpha'$;
- $I, t \models \Box\alpha$ if $I, t' \models \alpha$ for all $t' \in \mathbb{N}$ s.t. $t' \geq t$; $I, t \models \Diamond\alpha$ if $I, t' \models \alpha$ for some $t' \in \mathbb{N}$ s.t. $t' \geq t$;
- $I, t \models \bigcirc\alpha$ if $I, t+1 \models \alpha$.

In previous work [6], we introduced a new formalism called preferential linear temporal logic. The motivation is to provide a formalism for the specification and verification of systems where exceptions can be tolerated.

Let $p \in \mathscr{P}$, sentences of the logic $LTL^{\sim}$ are built up according to the following grammar:

$$\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Box\alpha \mid \Diamond\alpha \mid \bigcirc\alpha \mid \widetilde{\Box}\alpha \mid \widetilde{\Diamond}\alpha$$

The intuition behind the new temporal operators is the following: $\widetilde{\Box}$ reads as *non-monotonic always* and $\widetilde{\Diamond}$ reads as *non-monotonic eventually*. The set of all well-formed $LTL^{\sim}$ sentences is denoted by $\mathcal{L}^{\sim}$. It is worth to mention that any well-formed sentence α in LTL is a sentence of $\mathcal{L}^{\sim}$.

A sentence such as $\widetilde{\Box}\alpha$ reads as: in all normal future time points, α is true. A sentence of the form $\widetilde{\Diamond}\alpha$ reads as: in some normal future time point, α is true. We can even express properties using a mix of classical and non-monotonic operators. A sentence $\Box\widetilde{\Diamond}\alpha$ reads as: always, there is a normal future time point where α is true.

The preferential component of the interpretation of our language is directly inspired by the preferential semantics proposed by Shoham [19] and used in the KLM approach [11]. The ordering relation, denoted by $\prec$, is a strict partial order on points of time. Following Kraus et al. [11], $t \prec t'$ means that t is more preferred than t'. We use the pair notation $(t, t') \in \prec$ to indicate that t is more normal than t' w.r.t. $\prec$.

▶ **Definition 2 (Minimality w.r.t. $\prec$).** *Let $\prec$ be a strict partial order on a set $\mathbb{N}$ and $N \subseteq \mathbb{N}$. The set of the minimal elements of N w.r.t. $\prec$, denoted by $min_{\prec}(N)$, is defined by $min_{\prec}(N) \stackrel{\text{def}}{=} \{t \in N \mid$ there is no $t' \in N$ such that $(t', t) \in \prec\}$.*

▶ **Definition 3 (Well-founded set).** *Let $\prec$ be a strict partial order on a set $\mathbb{N}$. We say $\mathbb{N}$ is* well-founded *w.r.t. $\prec$ iff $min_{\prec}(N) \neq \emptyset$ for every $\emptyset \neq N \subseteq \mathbb{N}$.*

In what follows, given a relation $\prec$ and a time point $t \in \mathbb{N}$, the set of *preferred time points relative to t* is the set $min_{\prec}([t, \infty[)$ which is denoted in short by $min_{\prec}(t)$.

▶ **Definition 4 (Preferential temporal interpretation).** *An $LTL^{\sim}$ interpretation on a set of propositional atoms $\mathscr{P}$, also called preferential temporal interpretation on $\mathscr{P}$, is a pair $I \stackrel{\text{def}}{=} (V, \prec)$ where V is a mapping function $V : \mathbb{N} \longrightarrow 2^{\mathscr{P}}$, and $\prec \subseteq \mathbb{N} \times \mathbb{N}$ is a strict partial order on $\mathbb{N}$ such that $\mathbb{N}$ is well-founded w.r.t. $\prec$. We denote the set of preferential temporal interpretations by $\mathfrak{I}$.*

Preferential temporal interpretations provide us with an intuitive way of interpreting sentences of $\mathcal{L}^{\sim}$. Let $\alpha \in \mathcal{L}^{\sim}$, let $I = (V, \prec)$ be a preferential temporal interpretation, and let t be a time point in I in $\mathbb{N}$. Satisfaction of α at t in I, denoted $I, t \models \alpha$, is defined as follows:

- The truth values of Boolean connectives and classical modalities are defined as in LTL.
- $I, t \models \widetilde{\Box}\alpha$ if $I, t' \models \alpha$ for *all* $t' \in min_{\prec}(t)$;
- $I, t \models \widetilde{\Diamond}\alpha$ if $I, t' \models \alpha$ for *some* $t' \in min_{\prec}(t)$.

We say $\alpha \in \mathcal{L}^{\sim}$ is *satisfiable* if there is a preferential temporal interpretation I and a time point t in $\mathbb{N}$ such that $I, t \models \alpha$. We can show that $\alpha \in \mathcal{L}^{\sim}$ is *satisfiable* iff there is a preferential temporal interpretation I s.t. $I, 0 \models \alpha$.

3 A one-pass tableau for $LTL^{\sim}$

In this paper, we address the computational task of *satisfiability checking* in $LTL^{\sim}$. That is, given a sentence α in $LTL^{\sim}$, decide whether or not there is an interpretation I that satisfies the sentence α. As mentioned in the Introduction, we propose a one-pass tree-shaped tableau for a fragment of $LTL^{\sim}$ based on Reynolds' tableau [15] and inspired by the semantic rules for defeasible modalities in modal logic proposed by Britz and Varzinczak [4]. This fragment, denoted by $\mathcal{L}_1$, serves as a starting point for showing how the ordering $\prec$ is built for preferential interpretations in $LTL^{\sim}$.

3.1 The fragment $\mathcal{L}_1$

The fragment $\mathcal{L}_1$ considers that sentences are in NNF (negation is only allowed on the level of atomic propositions). On the other hand, the non-monotonic operator $\Box\!\!\!\!\sim$ is omitted from $\mathcal{L}_1$. Furthermore, only Boolean sentences are permitted within the scope of $\Box$ sentences. In what follows, we define formally well formed sentences of $\mathcal{L}_1$. In order to do that, we introduce first the set of Boolean sentences $\mathcal{L}_{bool}$. Let $p \in \mathscr{P}$, sentences $\alpha_{bool} \in \mathcal{L}_{bool}$ are defined recursively as such:

$$\alpha_{bool} ::= p \mid \neg p \mid \alpha_{bool} \wedge \alpha_{bool} \mid \alpha_{bool} \vee \alpha_{bool}$$

Next, let $\alpha_{bool} \in \mathcal{L}_{bool}$, sentences in $\mathcal{L}_1$ are recursively defined as such:

$$\alpha ::= \alpha_{bool} \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Diamond\alpha \mid \Box\alpha_{bool} \mid \bigcirc\alpha \mid \Diamond\!\!\!\!\sim\alpha$$

Sentences of the form $\Diamond\alpha$ are called *eventualities*, because its truth depends on α being true in the future. Similarly, sentences of the form $\Diamond\!\!\!\!\sim\alpha$ are called *non-monotonic eventualities*. Their truth depends not only on α being true in some future, but it depends also on this future being preferred to the other future time points. Sentences of the form $\bigcirc\Diamond\alpha$ are called $\bigcirc$-eventuality.

3.2 Tableau method for $\mathcal{L}_1$

A tableau for $\alpha \in \mathcal{L}_1$ is a tree of nodes. Each node has a positive integer n as a *label*. It has also two sets of sentences: one we denote as Γ and the other as *une* (which stands for unfulfilled non-monotonic eventualities, a notion to be detailed below). The set Γ is a subset of $\mathcal{L}_1$ which contains the sentences in the node. The set *une* is a set of pairs $(n_k, \Diamond\!\!\!\!\sim\alpha_k)$, where n_k is a label and $\Diamond\!\!\!\!\sim\alpha_k$ is a non-monotonic eventuality.

▶ **Definition 5** (Labelled node). *A labelled node is a triple of the form* $n : (\Gamma, une)$ *where* $n \in \mathbb{N}$, $\Gamma \subseteq \mathcal{L}_1$ *and* $une \subseteq [0, n] \times \mathcal{L}_1$.

It is worth to mention that different nodes can have the same label. Intuitively, the nodes labelled by a same integer n represent the set of sentences that are satisfied at the time point associated with n. Hence, these nodes correspond with a given temporal state.

A branch B is a sequence of nodes, we introduce also a strict partial ordering relation $\prec_B$ on the labels of the nodes within the branch. The branch B has also a set of pairs of labels denoted by min_B. The relation $\prec_B$ represents a preference relation on the temporal states of the branch B. On the other hand, the set min_B represents some constraints that the final preference relation issued from B must satisfy. More precisely, each pair (n, n') in min_B indicates that n' represents a preferred temporal state compared to all $n'' \geq n$.

▶ **Definition 6** (Branch). *A branch is a tuple* $B \stackrel{\text{def}}{=} (\langle x_0, x_1, x_2, \ldots \rangle, \prec_B, min_B)$ *where the first element is a sequence of labelled nodes* $x_i := n_i : (\Gamma_i, une_i)$*,* $\prec_B$ *is a strict partial order* ($\prec_B \subseteq \mathbb{N} \times \mathbb{N}$) *on labels within the branch, and* min_B *is a set of pairs of labels* ($min_B \subseteq \mathbb{N} \times \mathbb{N}$)*.*

Let $B := (\langle x_0, x_1, x_2, \ldots \rangle, \prec_B, min_B)$ be a branch, x_n, x_m be two labelled nodes in B. If x_m comes after x_n in the sequence, then x_m is a *successor* of x_n, and x_n is a *predecessor* of x_m. We denote it by $x_n \leq x_m$. Moreover, if x_m is not the same labelled node as x_n, we say that x_m is a proper successor of x_n (same goes for a proper predecessor). We denote it by $x_n < x_m$. The last node of a branch is called a *leaf node*. When a leaf node is ticked with ✓, we say that the branch is a successful branch. On the other hand, when a leaf node is crossed with ✗, we say that the branch is a failed branch.

A tree is a set of branches $\mathcal{T} \stackrel{\text{def}}{=} \{B_0, B_1, B_2, B_3, \ldots, B_k\}$ where $k \geq 0$. A tableau $\mathscr{T}$ for α is the limit of a sequence of trees $\langle \mathcal{T}^0, \mathcal{T}^1, \mathcal{T}^2, \ldots \rangle$ where the initial tree is $\mathcal{T}^0 := \{(\langle 0 : (\alpha, \emptyset) \rangle, \emptyset, \emptyset)\}$ and every $\mathcal{T}^{i+1}$ is obtained from $\mathcal{T}^i$ by applying a rule on one of its branches. We say that a tableau $\mathscr{T}$ for α is *saturated* if no more rules can be applied after a tree $\mathcal{T}$.

We have two types of rules, static and dynamic rules. We introduce static rules first. Let $\mathcal{T}$ be a tree, and let B be a branch of $\mathcal{T}$ that has a leaf $n : (\Gamma, une)$. We say that a static rule (ρ) is applicable at the leaf $n : (\Gamma, une)$ if a sentence in Γ or a pair in une instantiates the pattern ρ. A static rule is a rule of the form:

$$(\rho) \ \frac{n : (\Gamma, une), \prec_B, min_B}{n : (\Gamma_1, une_1), \prec_{B_1}, min_{B_1} \ | \ \ldots \ | \ n : (\Gamma_k, une_k), \prec_{B_k}, min_{B_k}}$$

In a tree $\mathcal{T}^i$, after applying the static rule (ρ), we obtain the tree $\mathcal{T}^{i+1}$ by replacing the branch $B := (\langle x_0, x_1, x_2, \ldots, n : (\Gamma, une) \rangle, \prec_B, min_B)$ by the branches $B_1 := (\langle x_0, x_1, x_2, \ldots, n : (\Gamma, une), n : (\Gamma_1, une_1) \rangle, \prec_{B_1}, min_{B_1})$, $B_2 := (\langle x_0, x_1, x_2, \ldots, n : (\Gamma, une), n : (\Gamma_2, une_2) \rangle, \prec_{B_2}, min_{B_2})$, and so on. The symbol '|' indicates the occurrence of a split in the branch, i.e., a non-deterministic choice of possible outcomes, each of which needs to be explored. It is worth to mention that after applying a static rule on $n : (\Gamma, une)$, the leaf nodes of all the new branches keep the same label n.

In what follows, we show the rules for Boolean and the operators $(\Box, \Diamond)$. We also show two stopping conditions, namely, **Empty** and **Contradiction**. We chose to omit $\prec_B$ and min_B to lighten these rules. The crucial detail to remember is that they do not change after applying the rules below, i.e., $\prec_{B_i} = \prec_B$ and $min_{B_i} = min_B$ for all resulting branches. The symbol $\cup$ is the union of two sets. The symbol $\uplus$ represents the union between disjoint sets.

$$\text{(Contradiction)} \ \frac{n : (\{\alpha, \neg\alpha\} \uplus \Sigma\}, une)}{(✗)} \qquad \text{(Empty)} \ \frac{n : (\emptyset, \emptyset)}{(✓)}$$

$$(\wedge) \ \frac{n : (\{\alpha_1 \wedge \alpha_2\} \uplus \Sigma, une)}{n : (\{\alpha_1, \alpha_2\} \cup \Sigma, une)} \qquad (\vee) \ \frac{n : (\{\alpha_1 \vee \alpha_2\} \uplus \Sigma, une)}{n : (\{\alpha_1\} \cup \Sigma, une) \ | \ n : (\{\alpha_2\} \cup \Sigma, une)}$$

$$(\Box) \ \frac{n : (\{\Box\alpha_1\} \uplus \Sigma, une)}{n : (\{\alpha_1, \bigcirc\Box\alpha_1\} \cup \Sigma, une)} \qquad (\Diamond) \ \frac{n : (\{\Diamond\alpha_1\} \uplus \Sigma, une)}{n : (\{\alpha_1\} \cup \Sigma\}, une) \ | \ n : (\{\bigcirc\Diamond\alpha_1\} \cup \Sigma\}, une)}$$

Before introducing the rule for the non-monotonic operator $\hat{\Diamond}$, we discuss firsthand the notion of *fulfillment* for classical and non-monotonic eventualities. Following Reynolds' tableau, let an eventuality $\Diamond\alpha$ be in a node with a label n. If the sentence α appears in a proper successor node x with the label $m \geq n$, we say that $\Diamond\beta$ at the position n is *fulfilled* in m. In a similar fashion, we define the fulfillment for non-monotonic eventualities as follows:

▶ **Definition 7** (Fulfillment of non-monotonic eventualities). *Let a non-monotonic eventuality $\diamondminus\alpha$ be in a node with a label n in a branch B. If α appears in a proper successor node x with a label $m \geq n$, and $(n, m) \in min_B$, we say $\diamondminus\alpha$ at the position n is fulfilled in m.*

The truth value $\diamondminus\alpha$ in a temporal state n depends on α being true on a future temporal state m and m being minimal to all temporal states that come after n w.r.t. $\prec_B$. We say m is minimal to n as shorter way to say that m is minimal to all temporal states that come after n. Unfulfilled non-monotonic eventualities in a node x with the label n are represented by the set $une \stackrel{\text{def}}{=} \{(n_1, \diamondminus\alpha_1), (n_2, \diamondminus\alpha_2), \dots\}$, each pair $(n_k, \diamondminus\alpha_k)$ represents a non-monotonic eventuality $\diamondminus\alpha_k$ at a position n_k that needs to be fulfilled. Therefore each node x has three components: n is a label indicating the temporal state, Γ is the set of sentences within the node and une is the set of non-monotonic eventualities at x that need to be fulfilled. With all of our notions introduced, here is the rule for the $\diamondminus$ operator:

$$(\diamondminus)\quad \frac{n : (\{\diamondminus\alpha_1\} \uplus \Sigma, une), \prec_B, min_B}{n : (\{\alpha_1\} \cup \Sigma, une), \prec_B, min_B \cup \{(n, n)\} \ \mid\ n : (\Sigma, une \cup \{(n, \diamondminus\alpha_1)\}), \prec_B, min_B}$$

For the rule **($\diamondminus$)**, we explore two outcomes. The first outcome is when the non-monotonic eventuality $\diamondminus\alpha_1$ at n is fulfilled in n. We then add α_1 to the set of sentences Γ of the leaf node and add $(n, n) \in min$ of the branch. The second outcome is when $\diamondminus\alpha_1$ is not fulfilled in n, then we add the pair to $(n, \diamondminus\alpha_1)$ to une of the leaf node as a non-monotonic eventuality that needs to be fulfilled. Example 8 shows the application of $[\diamondminus]$ rule.

▶ **Example 8.** Let a branch B have $\prec_B$, min_B and a leaf node $5 : (\{p, q, \Box(p \wedge q), \diamondminus r\}, \emptyset)$. After applying $(\diamondminus)$ rule on $\diamondminus r$, we have two new branches B_1 and B_2. The branch B_1 has a leaf node where the sentence r is in Γ of the leaf node and $(5, 5) \in min_{B_1}$. The branch B_2 has $(5, \diamondminus r)$ in une of the leaf node.

$$5 : (\{p, q, \Box(p \wedge q), \diamondminus r\}, \emptyset), \prec_B, min_B$$

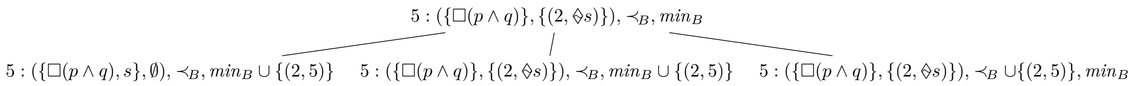
$$5 : (\{p, q, \Box(p \wedge q), r\}, \emptyset), \prec_B, min_B \cup \{(5, 5)\} \qquad 5 : (\{p, q, \Box(p \wedge q)\}, \{(5, \diamondminus r)\}), \prec_B, min_B$$

The next static rule we discuss is the rule **(une)**. Let n, n' be two labels such that $n' < n$, for each label n and a pair $(n', \diamondminus\alpha_1)$, the rule ($une$) is applied one and only one time. The rule goes as follows:

$$(une)\quad \frac{n : (\Gamma, \{(n', \diamondminus\alpha_1)\} \uplus U), \prec_B, min_B}{\begin{array}{c} n : (\{\alpha_1\} \cup \Gamma, U), \prec_B, min_B \cup \{(n', n)\} \mid \\ \hline n : (\Gamma, \{(n', \diamondminus\alpha_1)\} \cup U), \prec_B, min_B \cup \{(n', n)\} \mid \\ \hline n : (\Gamma, \{(n', \diamondminus\alpha_1)\} \cup U), \prec_B \cup \{(n', n)\}, min_B \end{array}}$$

For the rule **(une)**, we explore three outcomes. The first outcome is when $\diamondminus\alpha_1$ at the position n' is fulfilled at n. We remove $(n', \diamondminus\alpha_1)$ from une, then we add α in Γ of the leaf node and (n', n) in min of the branch. In the second and third branches, we explore the outcome of $\diamondminus\alpha_1$ not being fulfilled yet in n, we keep the pair $(n', \diamondminus\alpha_1)$ on the leaves of two branches. The second branch explore the outcome of n being minimal to n' w.r.t. to $\prec$ of the branch. We then add (n', n) to the *min* of the branch. In the third branch, we explore the outcome of n not being minimal to n' w.r.t. $\prec$ of the branch. It means that there exists a temporal state m' that come after n' where m' is preferred to n w.r.t. to $\prec$ of the branch, we add the pair (n', n) in $\prec$ of the branch to represent this case. It is worth to mention that the rule (*une*) does not apply when the label of the node n is the same as $(n, \Diamond\alpha_1)$. The reason behind this is that we have already explored the case when the eventuality is fulfilled in n thanks to ($\Diamond$) rule. Example 9 shows the application of (*une*) rule.

▶ **Example 9.** Let a branch B have $\prec_B$, min_B and a leaf node $5 : (\{\Box(p \wedge q)\}, \{(2, \Diamond s)\})$. After the application of *une* on $(2, \Diamond s)$, we have three branches B_1, B_2 and B_3. B_1 has the sentence s in Γ of its leaf node, it has also $(2, 5)$ in min_{B_1}. B_2 keeps $(2, \Diamond s)$ in the *une* of its leaf node, with $(2, 5) \in min_{B_2}$. B_3 keeps also $(2, \Diamond s)$ in *une* of its leaf node, with $(2, 5) \in \prec_{B_3}$.

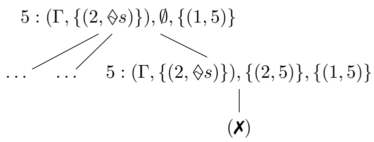


With the (*une*) and ($\Diamond$) introduced, we need to check the consistency of $\prec$ of all the new branches. We apply this check each time we apply (*une*) or ($\Diamond$) rule. Let $B := (\langle x_0, x_1, x_2, \ldots \rangle, \prec_B, min_B)$ be a branch, the rule goes as follows:

- [**$\prec$-inconsistency**] If $(n, n') \in min_B$ and there exists $n'' \geq n$ s.t. $(n'', n') \in \prec_B$, then the branch is crossed (✗).

In a branch B, if $(n, n') \in min_B$, then we are currently exploring a branch where n' is minimal to n w.r.t. $\prec_B$. Therefore there should be no $n'' \geq n$ where $(n'', n) \in \prec_B$. Each time we explore a branch where this inconsistency arises, we close the branch.

▶ **Example 10.** Let B be a branch where $\prec_B$ is empty, min_B has $(1, 5)$ in it, and a leaf node $5 : (\Gamma, \{(2, \Diamond s)\})$. After applying *une* rule on $(2, \Diamond s)$, we have three branches B_1, B_2 and B_3. The relation $\prec_{B_1}$ is empty, and min_{B_1} has the pairs $(1, 5)$ and $(2, 5)$. In this case, there is no inconsistency w.r.t. $\prec_{B_1}$ so far. The same goes for B_2. However, we add $(2, 5)$ to $\prec_{B_3}$. Since we already have $(1, 5) \in min_{B_3}$, we then cannot have $(2, 5) \in \prec_{B_3}$. We close B_3.



In a branch B of a tree $\mathcal{T}$ with a leaf node x_i, after applying every static rule aforementioned (the order of application these rules is non-deterministic) that can be applied, all leaf nodes of the generated branches contain only sentences of the form $p, \neg p$ or $\bigcirc\alpha$ in their Γ. When no more static rules can be applied in a node, this node is called a *state-labelled node*. State-labelled nodes mark the full expansion of all sentences that hold in a state n.

Once we are in a state-labelled node, in order to go from a temporal state to the next, we need a transition rule (a rule to go from a temporal state n to the next $n + 1$). In a branch B with a leaf state-labelled node, the rule **transition** goes the following way:

$$\text{(Transition)} \quad \frac{n : (\{\bigcirc\alpha_1, \bigcirc\alpha_2, \bigcirc\alpha_3, \ldots, \bigcirc\alpha_k\} \uplus \Sigma, une), \prec_B, min_B}{n + 1 : (\{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k\}, une), \prec_B, min_B}$$

After the **transition** rule is applied to a state-labelled node $n : (\Gamma, une)$, we add a node with the label $n+1$. It marks the start of a new temporal state $n+1$. We carry over to $n+1$ only sentences within the scope of $\bigcirc\alpha_i$ sentences. The set une gets transferred as well to the next temporal state. Any pair $(n', \diamondsuit\alpha_1) \in une$ remaining in the state node with the label n indicates that the rule (une) was applied on the temporal state n and the current branch explores an outcome where $\diamondsuit\alpha_1$ is not yet fulfilled in n. Therefore, these non-monotonic eventualities need to be fulfilled in $n'' \geq n+1$.

Before applying the **transition** rule, we need to add a set of checks to prevent branches from expanding indefinitely. These checks are called **loop** and **prune** rules. These rules, together with the **transition** rule, are called *dynamic rules*.

Let $B := (\langle x_0, x_1, x_2, \ldots, v\rangle, \prec_B, min_B)$ be a branch where v is a state-labelled node $n : (\Gamma_v, une_v)$. Let u be the last state-labelled node $n-1 : (\Gamma_u, une_u)$ that comes before v in the branch B. Before applying the transition rule on v, we check for these rules:

- **[Loop]** Let v be a state-labelled node such that it has at least one sentence of the form $\bigcirc\Box\alpha_{bool}$ in Γ_v but has no $\bigcirc\alpha_{bool}, \bigcirc\Diamond\beta, \bigcirc\diamondsuit\beta$ in Γ_v and $une_v = \emptyset$. If for all $\bigcirc\Box\alpha_{bool}$ in Γ_v, there exists $u < s \leq v$ such that $\Box\alpha_{bool} \in \Gamma_s$, then the branch B is ticked (✓).

Notice that once an eventuality is fulfilled, it does not appear any longer in the successors of the node. In this case, we say that the sentence is *consumed*. On the other hand, sentences of the form $\Box\alpha_{bool}$ never get consumed and get replicated indefinitely. Once a branch has no eventuality left, $\Box\alpha_{bool}$ sentences give rise to an infinite tableau with repetitive nodes. Nevertheless, we can represent this by looping nodes of the last temporal state. We can, in this case, stop the branch from ever going infinite. The **loop** rule states that when the leaf state node v has no eventualities (classical or non-monotonic), has only $\bigcirc\Box\alpha_{bool}$ as sentences with the pattern $\bigcirc$, and each $\bigcirc\Box\alpha_{bool}$ is a result from applying the $\Box$ rule to a node in B with label n, the branch is ticked and marked as a successful branch.

- **[Prune]** Let $u < v$ be two consecutive state-labelled nodes s.t. $\Gamma_v = \Gamma_u$ and $une_v = une_u$ and that there is at least one eventuality in x_u (either $\bigcirc\Diamond\beta \in \Gamma_u$ or $(n', \diamondsuit\beta) \in une_u$), then the branch is crossed (✗).

The **prune** rule states that when the last two state nodes u and v have the same set of classical and non-monotonic eventualities that need to be fulfilled, and there is at least one eventuality in u, the branch is then crossed and marked as an unsuccessful branch. Any branch that does not fulfill at least one eventuality between the current and the last temporal state is closed, to prioritize the exploration of branches that fulfill one or more eventuality of the last temporal state. If neither **prune** or **loop** apply on v, we apply the **transition** rule on the node v. Note that the **loop** and **prune** rules are fundamentally different from the ones proposed in Reynolds' tableau [14]. These rules are tailored to the restrictions of the fragment $\mathcal{L}_1$, in particular, the restriction of not allowing temporal sentences inside the $\Box$ operator. We argue in this paper that when eventualities (either classical or non-monotonic) are not infinitely replicated inside *globally* operators, we only need to check the current state node with the last one that comes beforehand. It is the reason why we also omit also the operator $\mathcal{U}$, since the right part of a $\mathcal{U}$-sentence can also replicate eventualities.

Once we are in a state-labelled node, we check for the loop and prune within the branch before applying the transition rule. If the transition rule is applied on a state node with a label n, we obtain a new node with the label $n+1$. We can then expand the tree from this node by applying static rules until we find ticked branches (thanks to the **empty** rule), closed branches (thanks to the **contradiction** or **$\prec$-inconsistency** rules), or branches with a state node that has the label $n+1$. We then repeat the cycle between static and dynamic rules. We can see that the tableau method does not go indefinitely. Thanks to prune rule, we close any branch (✗) that does not fulfill any eventuality in the current temporal state. Anytime we apply a transition rule (from n to $n+1$), we need to fulfill at least one eventuality in n. Therefore, as long as a branch is not closed with prune rule, eventuality sentences (either classical or non-monotonic) get consumed one by one over the execution of the method. Thus any branch that is not closed with prune has no eventualities left to fulfill. Note that if a branch contains at least one sentence of the form $\Box\alpha_{bool}$, it is then ticked thanks to the loop rule ($\Box\alpha_{bool}$ sentences do not get consumed). Otherwise, it is ticked thanks to the empty rule. Therefore any tableau $\mathscr{T}$ for a sentence in $\mathcal{L}_1$ is a *saturated* tableau.

4 Soundness and completeness

4.1 Soundness

Here we prove that the tableau method is sound, that is, when a tableau $\mathscr{T}$ of a sentence $\alpha \in \mathcal{L}_1$ has a successful branch, then α is satisfiable. As a first step, we show that we can extract an interpretation $I \in \mathfrak{I}$ from the successful branch. Let $B := (\langle x_0, x_1, x_2, \ldots, x_n, (✓)\rangle, \prec_B, min_B)$ be a successful branch of a tableau $\mathscr{T}$ for α, the sequence of nodes contains normal and state-labelled nodes. Each state-labelled node, denoted by x_{j_i}, within this sequence has a distinct label i. Figure 1 shows an example of the branch B.

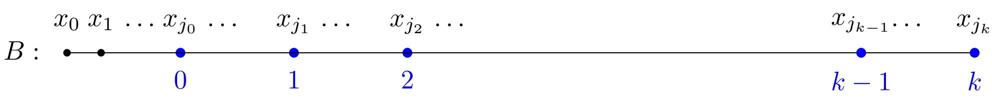


Figure 1 Illustration of the branch B.

From the aforementioned branch B, we can build an interpretation $I_B = (V, \prec)$. In this section, k denotes the label of the last state node. The function V is defined as follows:

$$V(i) := \begin{cases} \{p \in \mathscr{P} \mid p \in \Gamma_{x_{j_i}}\}, & \text{if } 0 \leq i \leq k; \\ V(k), & \text{otherwise.} \end{cases}$$

The ordering relation $\prec$ is defined as follows $\prec := \{(n, n') \mid (n, n') \in \prec_B\}$. We can see that $\prec$ is irreflexive, since there is no $(n, n) \in \prec_B$. The relation $\prec$ is also transitive, since for all (n_1, n_2) and (n_2, n_3) in $\prec_B$, there is no $(n_3, n_1) \in \prec_B$. Finally, since $\prec_B$ has no infinitely descending chains, then we can conclude that $\prec$ preserves the well-foundness condition over $\mathbb{N}$. Therefore the interpretation $I_B \in \mathfrak{I}$.

With the model construction introduced, we can move on to the second part of the proof of soundness. We need to show that the model I satisfies the sentence α. In order to do so, we introduce a mapping function, denoted by Δ_B, that links each time point $i \in \mathbb{N}$ to a set of sentences that are true in said i. These sentences come from the branch B. Depending on how the branch is ticked, the function Δ_B is defined in the following way.

If the branch was ticked with the empty rule:

$$\Delta_B(i) := \begin{cases} \bigcup_{x_0 \leq x \leq x_{j_0}} \Gamma_x, & \text{if } i = 0; \\ \bigcup_{x_{j_{i-1}} < x \leq x_{j_i}} \Gamma_x, & \text{if } 1 \leq i \leq k-1; \\ \{\}, & \text{otherwise.} \end{cases}$$

If the branch was ticked with the loop rule:

$$\Delta_B(i) := \begin{cases} \bigcup_{x_0 \le x \le x_{j_0}} \Gamma_x, & \text{if } i = 0; \\ \bigcup_{x_{j_{i-1}} < x \le x_{j_i}} \Gamma_x, & \text{if } 1 \le i \le k; \\ \Delta_B(k), & \text{otherwise.} \end{cases}$$

For a time point $0 \le i \le k$, $\Delta_B(i)$ contains the set of all sentences in Γ of the node between the two consecutive state nodes $x_{j_{i-1}}$ and x_{j_i}, $x_{j_{i-1}}$ not included. If B is ticked thanks to the empty rule, then $\Delta_B(i)$ is empty for all $i \ge k$. If B is ticked thanks to the loop rule, then $\Delta_B(i)$ has the same set of sentences as $\Delta_B(k)$ for all $i \ge k$. We can show next that if a sentence α_1 is in $\Delta_B(i)$, then $I_B, i \models \alpha_1$. In what follows, let B be a successful branch of a tableau $\mathscr{T}$, let k be the label of the last state node in B, and let I_B, Δ_B be the interpretation and the mapping function of sentences extracted from B.

▶ **Lemma 11.** *Let B be a successful branch, and $i \in \mathbb{N}$. If $\bigcirc\alpha_1 \in \Delta_B(i)$, then $\alpha_1 \in \Delta_B(i+1)$.*

▶ **Lemma 12.** *Let B be a successful branch, and $i \in \mathbb{N}$. If $\Box\alpha_1 \in \Delta_B(i)$, then for all $f \ge i$, we have $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(f)$.*

▶ **Lemma 13.** *Let B be a successful branch, and $i \in \mathbb{N}$. If $\Diamond\alpha_1 \in \Delta_B(i)$, then there exists $d \ge i$ s.t. $\alpha_1 \in \Delta_B(d)$ and for all $i \le f < d$, we have $\{\Diamond\alpha_1, \bigcirc\Diamond\alpha_1\} \subseteq \Delta_B(f)$.*

Lemma 11 to 13 are analogous to Reynolds' method [14]. Their proof are in Appendix A.

▶ **Proposition 14.** *Let B be a successful branch. If $(i, i') \in min_B$, then there is no $i \le i''$ where $(i'', i') \in \prec_B$.*

Proof. Let B be a successful branch s.t. $(i, i') \in min_B$. Since the branch is successful, then it is not closed with $\prec$-inconsistency and therefore there is no $i \le i''$ where $(i'', i') \in \prec_B$. ◀

▶ **Lemma 15.** *Let B be a successful branch and $0 \le i \le k$. If $\lozenge\alpha_1 \in \Delta_B(i)$, then there exists $d \ge i$ s.t. $(i, d) \in min_B$ and $\alpha_1 \in \Delta_B(d)$.*

Proof. Let B be a ticked branch of the tableau, k be the label of the last state node and $i \in \mathbb{N}$. We discuss two possibilities:

- When the branch B is ticked with empty rule, whenever $\lozenge\alpha_1 \in \Delta_B(i)$, then we have $0 \le i \le k-1$. Since $\lozenge\alpha_1 \in \Delta_B(i)$, then $\lozenge\alpha_1 \in \Gamma_x$ where $x_{j_{i-1}} < x \le x_{j_i}$. Let x be the node where we apply the rule $(\lozenge)$ on $\lozenge\alpha_1$, then we either have α_1 in Γ of the next node with $(i, i) \in min_B$ or we have $(i, \lozenge\alpha_1) \in une$ of the next node. If α_1 is in Γ of the next node, then the lemma holds. If $(i, \lozenge\alpha_1) \in une$ of the next node, then we find $(i, \lozenge\alpha_1) \in une_{x_{j_i}}$. Thanks to the transition rule, we have $(i, \lozenge\alpha_1) \in une_{x_{j_i+1}}$. By applying the rule une on a node with the label $i+1$,then we either have α_1 in Γ of the next node with $(i, i+1) \in min_B$ or we have $(i, \lozenge\alpha_1) \in une$ (the two remaining branches) of the next node. In a similar way as in i, we can conclude that either $\alpha_1 \in \Delta_B(i+1)$ with $(i, i+1) \in min_B$ (the lemma holds) or $(i, \lozenge\alpha_1) \in une_{x_{j_{i+1}}}$. Without loss of generality, $(i, \lozenge\alpha_1)$ is in $une_{x_{j_f}}$ for $i \le f \le k-1$ unless we find $i \le d \le f$ with $\alpha_1 \in \Delta_B(d)$ and $(i, d) \in min_B$. Since the branch is closed thanks to the empty rule, it means that $(i, \lozenge\alpha_1) \notin une_{x_{j_{k-1}}}$. Therefore, there is a state $i \le d \le k-1$ where $\alpha_1 \in \Delta_B(d)$ with $(i, d) \in min_B$.
- When the branch B is ticked with loop rule, the proof is analogous to the case of the empty rule (notice that we also have $(i, \lozenge\alpha_1) \notin une_{x_{j_k}}$).

◀

▶ **Theorem 16.** *Let B be a ticked branch from a saturated tableau, and $I_B = (V, \prec)$ be the model built from the branch B. For all $\alpha \in \mathcal{L}_1$, for all $i \geq 0$, if $\alpha \in \Delta_B(i)$ then $I_B, i \models \alpha$.*

Proof. We prove this lemma using structural induction on the size of the sentence α. Let B be a successful branch for a tableau $\mathscr{T}$, and $I_B = (V, \prec)$ be the model built from B.

- $\alpha = p$. Let $p \in \Delta_B(i)$. By construction of the model I_B, we have $p \in V(i)$. Therefore, we have $I_B, i \models p$.
- $\alpha = \neg p$. Let $\neg p \in \Delta_B(i)$. Since B is a ticked branch, then it was not closed with the contradiction rule, therefore we have $p \notin V(i)$. Therefore, we have $I_B, i \models \neg p$.
- $\alpha = \alpha_1 \wedge \alpha_2$. Let $\alpha_1 \wedge \alpha_2 \in \Delta_B(i)$. By $\wedge$-rule, we have $\alpha_1, \alpha_2 \in \Delta_B(i)$. By induction hypothesis on α_1, α_2, we have $I_B, i \models \alpha_1$ and $I_B, i \models \alpha_2$. Thus, we have $I_B, i \models \alpha_1 \wedge \alpha_2$.
- $\alpha = \alpha_1 \vee \alpha_2$. Let $\alpha_1 \vee \alpha_2 \in \Delta_B(i)$. By $\vee$-rule, we either have α_1 or α_2 in $\Delta_B(i)$. Suppose that $\alpha_1 \in \Delta_B(i)$, by induction hypothesis on α_1, we have $I_B, i \models \alpha_1$. Therefore, we have $I_B, i \models \alpha_1 \vee \alpha_2$. Same reasoning applies when $\alpha_2 \in \Delta_B(i)$.
- $\alpha = \bigcirc\alpha_1$. Let $\bigcirc\alpha_1 \in \Delta_B(i)$. Thanks to Lemma 11, we have $\alpha_1 \in \Delta_B(i+1)$. By induction hypothesis on α_1, we have $I_B, i+1 \models \alpha_1$. Therefore, we have $I_B, i \models \bigcirc\alpha_1$.
- $\alpha = \Box\alpha_1$. Let $\Box\alpha_1 \in \Delta_B(i)$. Thanks to Lemma 12, we have $\alpha_1 \in \Delta_B(f)$ for all $f \geq i$. By induction hypothesis on α_1, we have $I_B, f \models \alpha_1$ for all $f \geq i$. Therefore, we have $I_B, i \models \Box\alpha_1$.
- $\alpha = \Diamond\alpha_1$. Let $\Diamond\alpha_1 \in \Delta_B(i)$. Thanks to Lemma 13, we have $\alpha_1 \in \Delta_B(d)$ for some $d \geq i$. By induction hypothesis on α_1, we have $I_B, d \models \alpha_1$. Therefore, we have $I_B, i \models \Diamond\alpha_1$.
- $\alpha = \tilde{\Diamond}\alpha_1$. Let $\tilde{\Diamond}\alpha_1 \in \Delta_B(i)$. Depending on where i is, we have two cases:
  - In the case of $i > k$, since $\tilde{\Diamond}\alpha_1 \in \Delta_B(i)$, then we have $\tilde{\Diamond}\alpha_1 \in \Delta_B(k)$. Furthermore, since the branch is ticked with loop rule, we know that $(i, \tilde{\Diamond}\alpha_1) \notin une_{x_{j_k}}$. Therefore $\alpha_1 \in \Delta_B(k)$, thus $\alpha_1 \in \Delta_B(i)$. Furthermore, since $\prec := \prec_B$, and there is no $f \geq i$ such $(f, i) \in \prec_B$, then $i \in min_{\prec}(i)$, and therefore, $I_B, i \models \tilde{\Diamond}\alpha_1$.
  - $0 \leq i \leq k$. Thanks to Lemma 15, there exists $d \geq i$ s.t. $\alpha_1 \in \Delta_B(d)$ and $(i, d) \in min_B$. By induction hypothesis on α_1, we have $I_B, d \models \alpha_1$. Thanks to Proposition 14, there is no $i \leq f \leq k$ where $(f, d) \in \prec_B$ and therefore there is no $i \leq f \leq k$ where $(f, d) \in \prec$. Furthermore, by the construction of the model I_B, there is no $f \geq k$ where $(f, d) \in \prec$. Therefore, we have $d \in \min_{\prec}(i)$. Thus, we have $I_B, i \models \tilde{\Diamond}\alpha_1$.

◀

Let $\alpha \in \mathcal{L}_1$, B be a ticked branch from a saturated tableau for α, $I_B = (V, \prec)$ be a model built from B. Since we have $\alpha \in \Delta_B(0)$, then we have $I_B, 0 \models \alpha$.

4.2 Completeness

We conclude this paper by proving the completeness of the tableau method for sentences in $\mathcal{L}_1$ i.e., if a sentence α is satisfiable, then any tableau for α has a successful branch, no matter the order of applying the rules. We use a model I for α to find a ticked node.

▶ **Theorem 17.** *Let $\alpha \in \mathcal{L}_1$ be a satisfiable sentence of $LTL^{\sim}$. Then any tableau for α has a successful branch.*

The idea behind this proof is to have an intermediate sequence s that serves as a link between an interpretation I that satisfies the sentence α and a tableau $\mathscr{T}$ for α. The sequence s is a tuple $s := (\langle x_0, x_1, x_2, \ldots \rangle, \prec_s, min_s)$ where each x_i is a pair (Γ, une), $\prec_s, min_s$ are the set of constraints that the sequence s must follow in order to be coherent with $\prec$ of the interpretation. The set $\prec_s$ is not an ordering relation, it records instances of points of time not being minimal to other points of time w.r.t. the ordering relation $\prec$. Remember that each when we apply the *une* rule, we add a pair (n', n) to $\prec$ in order to symbolize the outcome of n not being minimal to n'. The set of min_s records the instances of points of points of time being minimal to other points of time w.r.t. the ordering relation $\prec$.

We link each node of the sequence x_i to a time point $J(x_i)$ of the interpretation I and a labelled node $f(x_i)$ of the tableau $\mathscr{T}$. Depending on I, we can build the sequence s using the tableau, we then show the sequence s ends up with a tick (✓). We make sure that for each node x_i with the index time point $J(x_i)$ of the sequence, we have the following invariant:

$$Inv(x_i, J(x_i)) \begin{cases} \text{For each } \alpha \in \Gamma_{x_i}, \text{ we have } I, J(x_i) \models \alpha; \\ \text{For each } (J_1, \Diamond\!\!\!\!\sim\alpha_1) \in une_{x_i}, \text{ there exists } J_2 \geq J(x_i) \text{ where} \\ J_2 \in min_{\prec}(J_1) \text{ and } I, J_2 \models \alpha_1; \\ \text{For each } (J_1, J_2) \in min_s, \text{ we have } J_2 \in min_{\prec}(J_1); \\ \text{For each } (J_1, J_2) \in \prec_s, \text{ there exists } J_3 \geq J_1 \text{ s.t. } (J_3, J_2) \in \prec \\ \text{(in other words } J_2 \notin min_{\prec}(J_1)). \end{cases}$$

We start by putting the root node $0 : (\{\alpha\}, \emptyset)$ with the index time point $J(x_0) := 0$ at the start of the sequence. For the first node x_0 with the index time point 0 (since there is no rule applied before the root node, the sets min_s and $\prec_s$ are empty at the start), we have $I, 0 \models \alpha$. Therefore the invariant $Inv(x_0, 0)$ holds. Suppose that the invariant holds up to x_i, and a rule was applied to x_i, we then add a new node x_{i+1} to the sequence depending on which outcome of the rule represents the interpretation I. We then move to the outcome node in the tableau, and see which rule is applied to it, and so on and so forth. Each time we add a new node x_{i+1} to the sequence s, we need to make sure that the invariant $Inv(x_{i+1}, J(x_{i+1}))$ holds. In general, the sequence will head from the parent node to a child node but it might occasionally jump backwards (only in the case of the parent being a prune node, more on that later). It is worth to point out that since we might be jumping back and forth between nodes of $\mathscr{T}$, each time we are add a new node x_{i+1} to the sequence s, we are going to rename labels within the sets une_x, $\prec_B$ and min_B by their respective indexed time points J. The function f links each node x_i of the sequence s to a labelled node $f(x_i)$ of the tableau $\mathscr{T}$. It is worth to mention that, since we are only renaming labels of other sets, then we have $\Gamma_{x_i} = \Gamma_{f(x_i)}$. In Appendix B, we discuss the case of each rule that is applied to x_i.

5 Conclusion

We introduced the basis for a tableau method for $LTL^{\sim}$. We showed how preferential semantics work in a one-pass tree-shaped tableau. We also established semantic rules for the $\Diamond\!\!\!\!\sim$ operator. We showed how to handle non-monotonic eventualities using *une*, $\prec_B$ and min_B. In the end, we proved that our method is sound and complete. The loop/prune checkers proposed in this paper are specific to $\mathcal{L}_1$, and work well under these restrictions.

With the foundation laid in this work, the next step is to establish semantic rules for the $\boxtimes$ operator. The next fragment of $LTL^{\sim}$ that we are investigating is the sub-language that allows only Boolean sentences within $\Box$ and $\boxtimes$. We conjecture that the satisfiability of this fragment is decidable and has an upper bound model property similar to one that we published in [6].

References

1. M. Ben-Ari. *Mathematical Logic for Computer Science, third edition*. Springer, 2012.
2. K. Britz, T. Meyer, and I. Varzinczak. Preferential reasoning for modal logics. *Electronic Notes in Theoretical Computer Science*, 278:55 – 69, 2011. doi:https://doi.org/10.1016/j.entcs.2011.10.006.
3. K. Britz, T. Meyer, and I. Varzinczak. Semantic foundation for preferential description logics. In *AI 2011: Advances in Artificial Intelligence*, pages 491–500, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
4. K. Britz and I. Varzinczak. From KLM-style conditionals to defeasible modalities, and back. *Journal of Applied Non-Classical Logics*, 28(1):92–121, 2018. doi:10.1080/11663081.2017.1397325.
5. K. Britz and I. Varzinczak. Preferential tableaux for contextual defeasible $\mathcal{ALC}$. In *Proceedings of the 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, number 11714 in LNCS, pages 39–57, 2019.
6. A. Chafik, F. Cheikh Alili, J.-F. Condotta, and I. Varzinczak. On the decidability of a fragment of preferential LTL. In *27th International Symposium on Temporal Representation and Reasoning, TIME 2020*, volume 178 of *LIPIcs*, pages 19:1–19:19, 2020. doi:10.4230/LIPIcs.TIME.2020.19.
7. D. M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Temporal Logic in Specification, Altrincham, UK, April 8-10, 1987, Proceedings*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer, 1987. doi:10.1007/3-540-51803-7_36.
8. L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. Preferential description logics. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, number 4790 in LNAI, pages 257–272. Springer, 2007.
9. L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. Analytic tableaux calculi for KLM logics of nonmonotonic reasoning. *ACM Transactions on Computational Logic*, 10(3):18:1–18:47, 2009.
10. Y. Kesten, Z. Manna, H. Mcguire, and A. Pnueli. A decision algorithm for full propositional temporal logic. *LNCS*, 697, 04 1999.
11. S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.
12. Z. Manna and A. Pnueli. *Temporal verification of reactive systems - safety*. Springer, 1995.
13. A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57, Oct 1977. doi:10.1109/SFCS.1977.32.
14. M. Reynolds. A new rule for LTL tableaux. *Electronic Proceedings in Theoretical Computer Science*, 226:287–301, Sep 2016. URL: http://dx.doi.org/10.4204/EPTCS.226.20, doi:10.4204/eptcs.226.20.
15. M. Reynolds. A traditional tree-style tableau for LTL, 2016. arXiv:1604.03962.
16. K. Y. Rozier and M. Y. Vardi. LTL satisfiability checking. In *Model Checking Software*, pages 149–167, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
17. S. Schwendimann. A new one-pass tableau calculus for PLTL. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 277–291. Springer Berlin Heidelberg, 1998.
18. Y. Shoham. *A Semantical Approach to Nonmonotonic Logics*, page 227–250. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.
19. Y. Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, 1988.
20. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985. doi:10.1145/3828.3837.
21. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1):72 – 99, 1983. doi:https://doi.org/10.1016/S0019-9958(83)80051-5.
22. P. Wolper. The tableau method for temporal logic. *Logique Et Analyse*, 28:110–111, 01 1985.

A Soundness proof

▶ **Lemma 11.** *Let B be a successful branch, and $i \in \mathbb{N}$. If $\bigcirc\alpha_1 \in \Delta_B(i)$, then $\alpha_1 \in \Delta_B(i+1)$.*

Proof. Let B be a ticked branch of the tableau, k be the label of the last node and $i \in \mathbb{N}$. We discuss two possibilities:

- When the branch B is ticked with empty rule. We can see that when $i \geq k$, $\Delta_B(i) = \{\}$ and therefore $\bigcirc\alpha_1 \notin \Delta_B(i)$. We also know that since $\Delta_B(k) = \{\}$, then there is no $\bigcirc\alpha_1 \in \Gamma_{x_{j_{k-1}}}$. Furthermore, there is no static rule that removes $\bigcirc\alpha_1$, we can conclude that there is no $\bigcirc\alpha_1 \in \Delta_B(k-1)$.
  Otherwise, in the case of $0 \leq i < k-1$, if $\bigcirc\alpha_1 \in \Delta_B(i)$, then $\bigcirc\alpha_1 \in \Gamma_x$ where $x_{j_{i-1}} < x \leq x_{j_i}$. Since there is no static rule that removes $\bigcirc\alpha_1$, we have $\bigcirc\alpha_1 \in \Gamma_{x_{j_i}}$. Furthermore, after applying the transition rule on the node x_{j_i}, we have $\alpha_1 \in \Gamma_{x_{j_i+1}}$. Thus, we have $\alpha_1 \in \Delta_B(i+1)$.
- When the branch B is ticked with loop rule. In the case of $0 \leq i < k$, the proof is analogous to the case of empty rule. When $i = k$, if $\bigcirc\alpha_1 \in \Delta_B(k)$, then $\bigcirc\alpha_1$ is subsequently in $\Gamma_{x_{j_k}}$. Since B is ticked with loop, then α_1 is a sentence of the form $\Box\alpha_{bool}$ and $\Box\alpha_{bool} \in \Gamma_x$ $(x_{j_{k-1}} < x \leq x_{j_k})$ and therefore $\Box\alpha_{bool} \in \Delta_B(k)$. Moreover, we have $\Delta_B(k) = \Delta_B(k+1)$. Therefore, we have $\Box\alpha_{bool} \in \Delta_B(k+1)$ and thus $\alpha_1 \in \Delta_B(k+1)$.
  In the case where $i \geq k$. If $\bigcirc\alpha_1 \in \Delta_B(i)$, then $\bigcirc\alpha_1 \in \Delta_B(k-1)$. As mentioned before, since $\bigcirc\alpha_1 \in \Delta_B(k-1)$, then α_1 is $\Box\alpha_2$ and $\Box\alpha_2 \in \Delta_B(k-1)$. Since $\Box\alpha_2 \in \Delta_B(k-1)$, then $\Box\alpha_2 \in \Delta_B(i+1)$ and therefore $\alpha_1 \in \Delta_B(i+1)$.

◀

▶ **Lemma 12.** *Let B be a successful branch, and $i \in \mathbb{N}$. If $\Box\alpha_1 \in \Delta_B(i)$, then for all $f \geq i$, we have $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(f)$.*

Proof. Let B be a ticked branch of the tableau, k be the label of the last node and $i \in \mathbb{N}$.

For all $0 \leq i \leq k$, whenever $\Box\alpha_1 \in \Delta_B(i)$, then both α_1 and $\bigcirc\Box\alpha_1$ is in $\Delta_B(i)$. By Lemma 11, since $\bigcirc\Box\alpha_1 \in \Delta_B(i)$, then we have $\Box\alpha_1 \in \Delta_B(i+1)$. By successive applications of Lemma 11, we have $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(f)$ for all $i \leq f \leq k$. Note that in the case of a branch ticked with empty rule, since $\Delta_B(k) = \{\}$, $\Box\alpha_1$ cannot be in any $\Delta_B(i)$ where $0 \leq i \leq k$. In other words, if a branch contains $\Box\alpha_1$, it can only be ticked with loop rule.

Since $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(k)$, and for all $f \geq k$, we have $\Delta_B(f) = \Delta_B(k)$, then $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(f)$. Thus, the lemma holds when $0 \leq i \leq k$.

In the case of $i > k$, since $\Box\alpha_1 \in \Delta_B(i)$ and $\Delta_B(i) = \Delta_B(k-1)$. Thanks to $\Box$-rule, $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(k-1)$. Thus, we have $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(f)$ for all $f \geq k$ and subsequently $\{\alpha_1, \Box\alpha_1, \bigcirc\Box\alpha_1\} \subseteq \Delta_B(f)$ for all $f \geq i$. ◀

▶ **Lemma 13.** *Let B be a successful branch, and $i \in \mathbb{N}$. If $\Diamond\alpha_1 \in \Delta_B(i)$, then there exists $d \geq i$ s.t. $\alpha_1 \in \Delta_B(d)$ and for all $i \leq f < d$, we have $\{\Diamond\alpha_1, \bigcirc\Diamond\alpha_1\} \subseteq \Delta_B(f)$.*

Proof. Let B be a ticked branch of the tableau, k be the label of the last node and $i \in \mathbb{N}$. We discuss two possibilities:

- When the branch B is ticked with empty rule. In the case of $0 \leq i \leq k-1$, whenever $\Diamond\alpha_1 \in \Delta_B(i)$, then either α_1 is in $\Delta_B(i)$ or $\bigcirc\Diamond\alpha_1$ is in $\Delta_B(i)$. If $\alpha_1 \in \Delta_B(i)$, the lemma holds. Otherwise, by Lemma 11, if $\bigcirc\Diamond\alpha_1 \in \Delta_B(i)$ then $\Diamond\alpha_1 \in \Delta_B(i+1)$. By successive applications of Lemma 11, $\{\Diamond\alpha_1, \bigcirc\Diamond\alpha_1\}$ is in $\Delta_B(f)$ for $i \leq f \leq k-1$, unless we find $i \leq d \leq f$ with $\alpha_1 \in \Delta_B(d)$.

It remains to show that there is a time point d where $\alpha_1 \in \Delta_B(d)$. Since the branch is closed thanks to the empty rule, it means that $\bigcirc\Diamond\alpha_1 \notin \Delta_B(k-1)$. Therefore, there is a state $i \leq d \leq k-1$ where $\alpha_1 \in \Delta_B(d)$.

- When the branch B is ticked with loop rule and in the case of $0 \leq i \leq k$, the proof is analogous to the case of empty rule (notice that $\bigcirc\Diamond\alpha_1 \notin \Delta_B(k)$ also in the case of branches ticked with loop).
  In the case of $i > k$, since $\Diamond\alpha_1 \in \Delta_B(i)$, then we have $\Diamond\alpha_1 \in \Delta_B(k-1)$. Furthermore, since the branch is ticked with loop rule, we know that $\bigcirc\Diamond\alpha_1 \notin \Delta_B(k)$. Therefore $\alpha_1 \in \Delta_B(k)$, thus $\alpha_1 \in \Delta_B(i)$.

◀

B Completeness proof

Proof. In this section, suppose that we build the sequence s up to x_i and the invariant holds for all the nodes in the sequence.

[Empty, Loop]: If we end up with a ticked node in the sequence s, the theorem holds.

[Contradiction]: If the sequence s is closed, then we have p and $\neg p$ in Γ_{x_i}. Since we have $Inv(x_i, J(x_i))$, then we $I, J(x_i) \models p$ and $I, J(x_i) \models \neg p$. This cannot happen in a interpretation $I \in \mathfrak{I}$.

[$\wedge$]: Suppose that the rule $\wedge$ is applied to the sentence $\alpha_1 \wedge \alpha_2$ on the node $f(x_i)$ of the tableau $\mathscr{T}$. Let y be the child node of the node $f(x_i)$ in the branch. We have $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\alpha_1 \wedge \alpha_2\}) \cup \{\alpha_1, \alpha_2\}$. We define the next node in the sequence x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_y$, $une_{x_{i+1}} = une_{x_i}$, and the sets min_s, $\prec_s$ remain unchanged. Since we have $Inv(x_i, J(x_i))$ and $\alpha_1 \wedge \alpha_2 \in \Gamma_{x_i}$, then $I, J(x_i) \models \alpha_1$ and $I, J(x_i) \models \alpha_2$. For the node x_{i+1}, we have $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\alpha_1 \wedge \alpha_2\}) \cup \{\alpha_1, \alpha_2\}$ and $une_{x_{i+1}} = une_{x_i}$. Therefore the first and second conditions of $Inv(x_{i+1}, J(x_i))$ are met. Moreover, since min_s, $\prec_s$ remain unchanged and we have $Inv(x_i, J(x_i))$, then the third and forth conditions of $Inv(x_{i+1}, J(x_i))$ are met. Consider that $J(x_{i+1}) = J(x_i)$, the invariant $Inv(x_{i+1}, J(x_i))$ holds.

We can see that by applying a static rule of the from $(\wedge, \vee, \Box, \Diamond)$ on the node $f(x_i)$, we do not add in either une, $\prec_B$ or min_B while applying these rules nor add a new non-monotonic eventuality to be fulfilled in the outcome nodes. In order to lighten the proof, we skip the check for the second, third and fourth conditions of Inv up until $\mathrel{\Diamond\!\!\!\!\cdot}$ and une rules.

[$\vee$]: Suppose that the rule $\vee$ is applied to the sentence $\alpha_1 \vee \alpha_2$ on the node $f(x_i)$ of the tableau $\mathscr{T}$. We obtain two children nodes y and z of $f(x_i)$. We have $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_1\}$ and $\Gamma_z = (\Gamma_{f(x_i)} \setminus \{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_2\}$. Since we have $Inv(x_i, J(x_i))$, and $\alpha_1 \vee \alpha_2 \in \Gamma_{x_i}$, then we either have $I, J(x_i) \models \alpha_1$ or $I, J(x_i) \models \alpha_2$. Consider that $J(x_{i+1}) = J(x_i)$, we discuss two cases:

- Case 1: If $I, J(x_i) \models \alpha_1$, then we define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_y$ and $une_{x_{i+1}} = une_{x_i}$. We know that $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_1\}$. Therefore for all $\gamma \in \Gamma_{x_{i+1}}$, we have $I, J(x_i) \models \gamma$. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.
- Case 2: Otherwise, when $I, J(x_i) \models \alpha_2$, then we define the node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_z$ and $une_{x_{i+1}} = une_{x_i}$. We know that $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_2\}$. Therefore for all $\gamma \in \Gamma_{x_{i+1}}$, we have $I, J(x_i) \models \gamma$. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.

[$\Diamond$]: Suppose that the rule $\Diamond$ is applied to the sentence $\Diamond\alpha_1$ on the node $f(x_i)$ of the tableau $\mathscr{T}$. We obtain two children nodes y and z of $f(x_i)$. We have $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\Diamond\alpha_1\}) \cup \{\alpha_1\}$ and $\Gamma_z = (\Gamma_{f(x_i)} \setminus \{\Diamond\alpha_1\}) \cup \{\bigcirc\Diamond\alpha_1\}$. Since we have $Inv(x_i, J(x_i))$, and $I, J(x_i) \models \Diamond\alpha_1$, then we have $I, J(x_i) \models \alpha_1 \vee \bigcirc\Diamond\alpha_1$. Therefore, we either have $I, J(x_i) \models \alpha_1$ or $I, J(x_i) \models \bigcirc\Diamond\alpha_1$. Consider that $J(x_{i+1}) = J(x_i)$, we discuss two cases:

- Case 1: If $I, J(x_i) \models \alpha_1$, then we define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_y$ and $une_{x_{i+1}} = une_{x_i}$. We know that $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Diamond\alpha_1\}) \cup \{\alpha_1\}$. Therefore for all $\gamma \in \Gamma_{x_{i+1}}$, we have $I, J(x_i) \models \gamma$. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.
- Case 2: When $I, J(x_i) \models \bigcirc\Diamond\alpha_1$, then we define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_z$ and $une_{x_{i+1}} = une_{x_i}$. We know that $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Diamond\alpha_1\}) \cup \{\bigcirc\Diamond\alpha_1\}$. Therefore for all $\gamma \in \Gamma_{x_{i+1}}$, we have $I, J(x_i) \models \gamma$. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.

[$\Box$]: Suppose that the rule $\Box$ is applied to the sentence $\Box\alpha_1$ on the node $f(x_i)$ of the tableau $\mathscr{T}$. Let y be the child node of the node $f(x_i)$ in the branch. We have $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\Box\alpha_1\}) \cup \{\alpha_1, \bigcirc\Box\alpha_1\}$. We define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_y$ and $une_{x_{i+1}} = une_{x_i}$ and $I, J(x_i) \models \Box\alpha_1$, then we have $I, J(x_i) \models \alpha_1 \wedge \bigcirc\Box\alpha_1$. Therefore, we have $I, J(x_i) \models \alpha_1$ and $I, J(x_i) \models \bigcirc\Box\alpha_1$. We know that $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Box\alpha_1\}) \cup \{\alpha_1, \bigcirc\Box\alpha_1\}$. Therefore for all $\gamma \in \Gamma_{x_{i+1}}$, we have $I, J(x_i) \models \gamma$. Consider that $J(x_{i+1}) = J(x_i)$, the invariant $Inv(x_{i+1}, J(x_i))$ holds.

[$\diamondsuit$]: When the rule [$\diamondsuit$] is applied to $\diamondsuit\alpha_1$ on the node $f(x_i)$ of $\mathscr{T}$, we explore two outcomes. Let n be the label of the node $f(x_i)$ in the branch. In the first outcome, we have a child y with $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\diamondsuit\alpha_1\}) \cup \{\alpha_1\}$ and (n, n) in *min* of the branch. In the second outcome, we have a child node z with $\Gamma_z = (\Gamma_{f(x_i)} \setminus \{\diamondsuit\alpha_1\})$ and $une_z = une_{f(x_i)} \cup (n, \diamondsuit\alpha_1)$. Since we have $Inv(x_i, J(x_i))$, and $\diamondsuit\alpha_1 \in \Gamma_{x_i}$, then we have $I, J(x_i) \models \diamondsuit\alpha_1$. It means that there exists $J_1 \geq J(x_i)$ s.t. $J_1 \in min_{\prec}(J(x_i))$ and $I, J_1 \models \alpha_1$. Consider that $J(x_{i+1}) = J(x_i)$, we discuss two cases:

- Case 1: If $J_1 = J(x_i)$, then we have $I, J(x_i) \models \alpha_1$ and $J(x_i) \in min_{\prec}(J(x_i))$. We then define the next node x_{i+1} of the sequence with $\Gamma_{x_{i+1}} = \Gamma_y$, $une_{x_{i+1}} = une_{x_i}$ and add the pair $(J(x_i), J(x_i))$ to min_s. Notice that we swap the labels of nodes with the position of their indexed time point $J(x_i)$, we will be using indexed time point J instead of labels throughout this proof. We know that $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\diamondsuit\alpha_1\}) \cup \{\alpha_1\}$ with $I, J(x_i) \models \alpha_1$. Additionally, we have $min_s := min_s \cup \{(J(x_i), J(x_i))\}$ with $J(x_i) \in min_{\prec}(J(x_i))$. The sets $une_{x_{i+1}}, \prec_s$ remains unchanged. Therefore, the invariant $Inv(x_{i+1}, J(x_i))$ holds.
- Case 2: when $J_1 > J(x_i)$, then we define the next node x_{i+1} of the sequence with $\Gamma_{x_{i+1}} = \Gamma_z$, $une_{x_{i+1}} = une_{x_i} \cup \{(J(x_i), \diamondsuit\alpha_1)\}$. We also know that $J_1 > J(x_i)$ and $J_1 \in min_{\prec}(J(x_i))$ and $I, J_1 \models \alpha_1$. Therefore, the second condition of $Inv(x_{i+1}, J(x_i))$ holds on the pair $(J(x_i), \diamondsuit\alpha_1)$. The sets min_s and $\prec_s$ remain unchanged. The invariant $Inv(x_{i+1}, J(x_i))$ holds.

[*une*]: When the rule [*une*] is applied on a pair $(n_1, \diamondsuit\alpha_1)$ in *une* of $f(x_i)$. Let n be the label of the node $f(x_i)$. Let x be the predecessor of x_i in s where the rule [$\diamondsuit$] was applied on $\diamondsuit\alpha_1$, let $J(x)$ be the indexed time point of x. Note that the label of $f(x)$ is n_1. In the first outcome, we have a child y where $\Gamma_y = \Gamma_{f(x_i)} \cup \{\alpha_1\}$, $une_y = une_{f(x_i)} \setminus \{(n_1, \diamondsuit\alpha_1)\}$ and (n_1, n) in *min* of the branch. In the second outcome, we have a child z where $\Gamma_z = \Gamma_{f(x_i)}$, $une_z = une_{f(x_i)}$ and (n_1, n) in *min* of the branch. In the third outcome, we have a child v where $\Gamma_v = \Gamma_{f(x_i)}$, $une_v = une_{f(x_i)}$ and (n_1, n) in $\prec$ of the branch.

On the other hand, since x is a predecessor of x_i in s, then we have $Inv(x, J(x))$. Furthermore, since we have $(n_1, \diamondsuit\alpha_1) \in une_{f(x_i)}$, it means that when the rule [$\diamondsuit$] is applied on the node $f(x)$, the branch where $(n_1, \diamondsuit\alpha_1) \in une_{f(x+1)}$ is the path that corresponds with the interpretation I. By [$\diamondsuit$] rule, since we have $Inv(x+1, J(x+1))$, $(n_1, \diamondsuit\alpha_1) \in une_{f(x+1)}$ and we know that $J(x+1) = J(x)$, then we have $(J(x), \diamondsuit\alpha_1) \in une_{x+1}$. Furthermore, since no rule application consumed $(n_1, \diamondsuit\alpha_1)$ up to $f(x_i)$, then the pair $(J(x), \diamondsuit\alpha_1)$ remains also in une_{x_i}. Also, we have $Inv(x_i, J(x_i))$, then there is $J' \geq J(x_i)$ where $J' \in min_{\prec}(J(x))$ and $I, J' \models \alpha_1$. Consider that $J(x_{i+1}) = J(x_i)$, we discuss all possibilities below:

- Case 1: If $J' = J(x_i)$, then we have $J(x_i) \in min_{\prec}(J(x))$ and $I, J(x_i) \models \alpha_1$. We define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_y$, $une_{x_{i+1}} = une_{x_i} \setminus \{(J(x), \diamondsuit\alpha_1)\}$ and add $(J(x), J(x_i))$ to min_s. We have $\Gamma_{x_{i+1}} = \Gamma_{x_i} \cup \{\alpha_1\}$ with $I, J(x_i) \models \alpha_1$. Additionally, we have $(J(x), J(x_i)) \in min_s$ with $J(x_i) \in min_{\prec}(J(x))$. The set $\prec_s$ remains unchanged. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.
- Case 2: when $J' > J(x_i)$, we have two possibilities:
  - Case 2.1: If $J(x_i) \in min_{\prec}(J(x))$, then we define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_z$, $une_{x_{i+1}} = une_{x_i}$ and add $(J(x), J(x_i))$ to min_s. We have $(J(x), J(x_i)) \in min_s$ with $J(x_i) \in min_{\prec}(J(x))$. The sets $\Gamma_{x_{i+1}}$, $une_{x_{i+1}}$ and $\prec_s$ remain unchanged. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.
  - Case 2.2: If $J(x_i) \notin min_{\prec}(n_1)$, then there exists $J'' \geq J(x)$ s.t. $(J'', J(x_i)) \in \prec$. We define the next node x_{i+1} with $\Gamma_{x_{i+1}} = \Gamma_v$, $une_{x_{i+1}} = une_{x_i}$ and add $(J(x), J(x_i))$ to $\prec_s$. We have $(J(x), J(x_i)) \in \prec_s$ with $J(x_i) \notin min_{\prec}(n_1)$. The sets $\Gamma_{x_{i+1}}$, $une_{x_{i+1}}$ and min_s remain unchanged. Thus, the invariant $Inv(x_{i+1}, J(x_i))$ holds.

[Transition]: Suppose that the transition rule is applied on the state node $f(x_i)$. Let y be the child node of the node x_i in the branch. We have $\Gamma_y = \{\alpha_1 \mid \bigcirc\alpha_1 \in \Gamma_{f(x_i)}\}$ and $une_y = une_{f(x_i)}$. We define the next node x_{i+1} in s with $\Gamma_{x_{i+1}} = \Gamma_y$ and $une_{x_{i+1}} = une_{x_i}$. We consider that $J(x_{i+1}) = J(x_i) + 1$.

Since we have $Inv(x_i, J(x_i))$, then for all $\bigcirc\alpha_1 \in \Gamma_{x_i}$, we have $I, J(x_i) \models \bigcirc\alpha_1$ and therefore $I, J(x_i) + 1 \models \alpha_1$. The first condition of the invariant $Inv(x_{i+1}, J(x_i) + 1)$ is met.

Secondly, since x_i is a state node, then for each remaining $(n_1, \diamondsuit\alpha_1) \in une_{f(x_i)}$, either the rule $[\diamondsuit]$ or $[une]$ was applied to a node $f(x_i')$ with the index $J(x_i') = J(x_i)$ and $(n_1, \diamondsuit\alpha_1)$ was carried over to $f(x_i)$. In both rules, for each $(n_1, \diamondsuit\alpha_1) \in une_{f(x_i)}$, we have $(J(x_1), \diamondsuit\alpha_1) \in une_{x_i}$ s.t. $f(x_1)$ is the node where the rule $[\diamondsuit]$ was applied to $\diamondsuit\alpha_1$ (see Case 2 for $[\diamondsuit]$ and $[une]$ rules). Furthermore, since we have $Inv(x_i, J(x_i))$ and $f(x_i)$ is a state node, then for each $(J(x_1), \diamondsuit\alpha_1) \in une_{x_i}$, there exists $J_2 > J(x_i)$ where $J_2 \in min_{\prec}(J(x_1))$ and $I, J_2 \models \alpha_1$. Without loss of generality, there exists $J_2 \geq J(x_i)+1$ where $J_2 \in min_{\prec}(J(x_1))$ and $I, J_2 \models \alpha_1$. The second condition of the invariant $Inv(x_{i+1}, J(x_i) + 1)$ is met. Since min_s and $\prec_s$ remain unchanged, the invariant $Inv(x_{i+1}, J(x_i) + 1)$ holds.

[$\prec$-inconsistency]: Suppose that the $\prec$-inconsistency rise on the node $f(x_i)$, and let n be the label of the $f(x_i)$ on the branch B. If this inconsistency rises, we have (n_1, n) in min_B and (n_2, n) in $\prec_B$ where $n_1 \leq n_2 \leq n$. These two pairs come from applying $[\diamondsuit]$ or $[une]$ rule on two predecessors $f(x), f(x')$ of $f(x_i)$ with the same label n and the same indexed time point $J(x) = J(x') = J(x_i)$.

Let J_1 be the time point corresponding to the node $f(x_1)$ with the label n_1, and let J_2 be the time point corresponding to the node $f(x_2)$ with the label n_2. It is worth to mention that $J_1 \leq J_2 \leq J(x_i)$. Since x, x' are predecessors of x, we have $Inv(x, J(x))$, $Inv(x', J(x'))$ and $Inv(x_i, J(x_i))$. Therefore, we the rules are applied on x and x', we end up with $(J_1, J(x_i)) \in min_s$ and $(J_2, J(x_i)) \in \prec_s$. Since $(J_1, J(x_i)) \in min_s$, then we have $J(x_i) \in min_{\prec}(J_1)$. On the other hand, since $(J_2, J(x_i)) \in \prec_s$, then there exists $J_3 \geq J_2$ s.t. $(J_3, J(x_i)) \in \prec$. Moreover, we have $J_1 \leq J_2$, this entails that there exists $J_3 \geq J_1$ s.t. $(J_3, J(x_i)) \in \prec$. This contradicts Definition 4 of minimality w.r.t. to the relation $\prec$. Therefore this cannot happen in a interpretation $I \in \mathfrak{I}$.

[Prune]: Let $f(x_i)$ be a state node where the prune condition is met. There is a sequence within s that goes the following way, $x_h = u, x_{h+1}, x_{h+2}, \ldots, v = x_i$. The node u or x_h is the state node that comes before x_i and the node v is the current state node. Since v is a prune node, we have $\Gamma_v = \Gamma_u$ and $une_u = une_v$. We can see that if we apply the transition rule to the node x_i, we will have $\Gamma_{x_{i+1}} = \Gamma_{x_{h+1}}$ and $une_{x_{i+1}} = une_{x_{h+1}}$. Therefore, we can proceed with the construction of s as if x_i was linked to $f(u)$ instead of $f(v)$. Thanks to the transition, since we have $Inv(x_u, J(x_u))$, then we have $Inv(x_{i+1}, J(x_i) + 1)$.

Each time we find a pair (u, v) in the sequence s, we call it a *jump*. These jumps may occur once or many times (and it may go infinite) in s. In a sequence s, if a pair (u, v) jumps repeatedly in succession, we call the pair a *recurring jump*. It is worth to point out that, each time we jump backwards because of a node closed with prune, we return to the state labelled node that comes before. In general, the sequence s explores one branch B of $\mathscr{T}$, and it deviates sometime to a prune node and goes back to B. Furthermore, since no eventuality is fulfilled within a prune loop, eventualities and their fulfillment are in the same branch B.

What we showed so far is that for an interpretation I and its corresponding sequence s, we have $Inv(x_i, J(x_i))$ for each $i \geq 0$. Going back to the start of the proof, we need to prove that the sequence finishes with a ticked node (such is the case when we end up in [loop] or [empty] node). We can see that if the sequence s is on a [prune] node, we jump back to the state node that comes before it. Theoretically, this jump can recur infinitely many times. This means that sequence goes infinite on this case (and never find a ticked node). We need to prove that this case cannot happen in the sequence s of I. Suppose that is the case, that means the last jump (u_k, v_k) in the sequence s is a recurring jump that goes infinitely many times. The jumps (u_j, v_j) that come before may recur many times but not infinitely many times (otherwise, (u_k, v_k) would not be the last jump). In the recurring jump (u_k, v_k), no eventuality is fulfilled (whether it is classical or non-monotonic). This entails that when we are in a parent node $u_k < x_l < v_k$ that applies either [$\Diamond$] or [*une*] rule, we move to the child node that delays the propagation of the eventuality (we are in Case 2 for both rules).

It is worth to point out that we have at least one eventuality in u_k. Let us take $\bigcirc\Diamond\alpha_1 \in \Gamma_{u_k}$ for example, since we have $Inv(u_k, J(u_k))$, that means that $I, J(u_k) \models \bigcirc\Diamond\alpha_1$. Thus, we take the *first* time point $J_{\alpha_1} > J(u_k)$ s.t. $I, J_{\alpha_1} \models \alpha_1$. We also have $I, J_{\alpha_1} \models \Diamond\alpha_1$. On the other hand, for all $J(u_k) < J < J_{\alpha_1}$, we have $I, J \models \Diamond\alpha_1$ $I, J \models \bigcirc\Diamond\alpha_1$. In other words, each time we encounter $\Diamond\alpha_1 \in \Gamma_{x_{l-1}}$ within our jumps (keep in mind we have $Inv(x_{l-1}, J)$), we pick the node in Case 2 of the [$\Diamond$] rule i.e., $\bigcirc\Diamond\alpha_1 \in \Gamma_{x_l}$. However, in the node indexed with J_{α_1}, when we encounter $\Diamond\alpha_1 \in \Gamma_{x_{l'-1}}$ (keep in mind we have $Inv(x_{l'-1}, J_{\alpha_1})$), we pick the node in Case 1 of the [$\Diamond$] rule i.e., $\alpha_1 \in \Gamma_{x'_l}$. This raises a contradiction, because the node $x_{l'}$ is not present within the jump (u_k, v_k). Thus breaking the infinite recurring jump (u_k, v_k).

If the eventuality is a non-monotonic one, namely $(J_1, \Diamond\!\!\!\!\sim\alpha_1) \in une_{u_k}$. Since we have $Inv(u_k, J(u_k))$ with u_k being a state node, there exists $J' > J(u_k)$ s.t. $J' \in min_{\prec}(J_1)$ and $I, J' \models \alpha_1$. Let J_{α_1} be the first time point that met these criteria. For all $J(u_k) < J < J_{\alpha_1}$, each time we encounter $(J_1, \Diamond\!\!\!\!\sim\alpha_1) \in une_{x_{l-1}}$ with the index J, we have $J_{\alpha_1} > J$, $J_{\alpha_1} \in min_{\prec}(J_1)$ and $I, J_{\alpha_1} \models \alpha_1$. Therefore, we pick Case 2 of [*une*] rule i.e., $(J_1, \Diamond\!\!\!\!\sim\alpha_1) \in une_{x_l}$. However, when we encounter $(J_1, \Diamond\!\!\!\!\sim\alpha_1) \in une_{x_{l'-1}}$ with the index J_{α_1}, we have $J_{\alpha_1} \in min_{\prec}(J_1)$ and $I, J_{\alpha_1} \models \alpha_1$, then we pick the node in Case 1 of [*une*] rule i.e., $\alpha_1 \in x_{l'}$. This raises a contradiction, because the node $x_{l'}$ is not present within the jump (u_k, v_k).

We proved that since $I, 0 \models \alpha$, then the corresponding sequence s cannot finish on a contradiction, $\prec$-inconsistency or a prune jump. Therefore it must finish with a ticked node. Hence, the tableau $\mathscr{T}$ of α has a ticked node and therefore a successful branch.

◀