



**HAL**  
open science

## A One-Pass Tree-Shaped Tableau for Defeasible LTL

Anasse Chafik, Fahima Cheikh-Alili, Jean-François Condotta, Ivan Varzinczak

► **To cite this version:**

Anasse Chafik, Fahima Cheikh-Alili, Jean-François Condotta, Ivan Varzinczak. A One-Pass Tree-Shaped Tableau for Defeasible LTL. 28th International Symposium on Temporal Representation and Reasoning (TIME 2021), Alpen-Adria-Universität Klagenfurt, Sep 2021, Klagenfurt, Austria. pp.16:1-16:18, 10.4230/LIPIcs.TIME.2021.16 . hal-03699248

**HAL Id: hal-03699248**

**<https://hal.science/hal-03699248>**

Submitted on 4 Apr 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 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 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*<sup>~</sup>) [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*<sup>~</sup>



© Anasse Chafik, Fahima Cheikh Alili, Jean-François Condotta, Ivan Varzinczak; licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

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

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

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

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

## 72 2 Preliminaries

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

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

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

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

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

- 92 ■  $I, t \models p$  if  $p \in V(t)$ ;  $I, t \models \neg\alpha$  if  $I, t \not\models \alpha$ ;
- 93 ■  $I, t \models \alpha \wedge \alpha'$  if  $I, t \models \alpha$  and  $I, t \models \alpha'$ ;  $I, t \models \alpha \vee \alpha'$  if  $I, t \models \alpha$  or  $I, t \models \alpha'$ ;
- 94 ■  $I, t \models \Box\alpha$  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.  
95  $t' \geq t$ ;
- 96 ■  $I, t \models \bigcirc\alpha$  if  $I, t + 1 \models \alpha$ .

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

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

$$101 \quad \alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \Box\alpha \mid \Diamond\alpha \mid \bigcirc\alpha \mid \boxminus\alpha \mid \diamond\alpha$$

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

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

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

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

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

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

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

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

- 130 ■ The truth values of Boolean connectives and classical modalities are defined as in  $LTL$ .
- 131 ■  $I, t \models \boxminus\alpha$  if  $I, t' \models \alpha$  for all  $t' \in \min_{\prec}(t)$ ;
- 132 ■  $I, t \models \diamond\alpha$  if  $I, t' \models \alpha$  for some  $t' \in \min_{\prec}(t)$ .

133 We say  $\alpha \in \mathcal{L}^\sim$  is *satisfiable* if there is a preferential temporal interpretation  $I$  and a  
 134 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  
 135 preferential temporal interpretation  $I$  s.t.  $I, 0 \models \alpha$ .

### 136 **3 A one-pass tableau for $LTL^\sim$**

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

#### 144 **3.1 The fragment $\mathcal{L}_1$**

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

$$151 \quad \alpha_{bool} ::= p \mid \neg p \mid \alpha_{bool} \wedge \alpha_{bool} \mid \alpha_{bool} \vee \alpha_{bool}$$

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

$$153 \quad \alpha ::= \alpha_{bool} \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \diamond \alpha \mid \square \alpha_{bool} \mid \circ \alpha \mid \diamond \alpha$$

154 Sentences of the form  $\diamond \alpha$  are called *eventualities*, because its truth depends on  $\alpha$  being  
 155 true in the future. Similarly, sentences of the form  $\diamond \alpha$  are called *non-monotonic eventualities*.  
 156 Their truth depends not only on  $\alpha$  being true in some future, but it depends also on this  
 157 future being preferred to the other future time points. Sentences of the form  $\circ \diamond \alpha$  are called  
 158  $\circ$ -eventuality.

#### 159 **3.2 Tableau method for $\mathcal{L}_1$**

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

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

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

170 A branch  $B$  is a sequence of nodes, we introduce also a strict partial ordering relation  
 171  $\prec_B$  on the labels of the nodes within the branch. The branch  $B$  has also a set of pairs of

172 labels denoted by  $\text{min}_B$ . The relation  $\prec_B$  represents a preference relation on the temporal  
 173 states of the branch  $B$ . On the other hand, the set  $\text{min}_B$  represents some constraints that  
 174 the final preference relation issued from  $B$  must satisfy. More precisely, each pair  $(n, n')$  in  
 175  $\text{min}_B$  indicates that  $n'$  represents a preferred temporal state compared to all  $n'' \geq n$ .

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

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

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

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

$$194 \quad (\rho) \frac{n : (\Gamma, \text{une}), \prec_B, \text{min}_B}{n : (\Gamma_1, \text{une}_1), \prec_{B_1}, \text{min}_{B_1} \mid \dots \mid n : (\Gamma_k, \text{une}_k), \prec_{B_k}, \text{min}_{B_k}}$$

195 In a tree  $\mathcal{T}^i$ , after applying the static rule ( $\rho$ ), we obtain the tree  $\mathcal{T}^{i+1}$  by repla-  
 196 cing the branch  $B := (\langle x_0, x_1, x_2, \dots, n : (\Gamma, \text{une}) \rangle, \prec_B, \text{min}_B)$  by the branches  $B_1 :=$   
 197  $(\langle x_0, x_1, x_2, \dots, n : (\Gamma, \text{une}), n : (\Gamma_1, \text{une}_1) \rangle, \prec_{B_1}, \text{min}_{B_1})$ ,  $B_2 := (\langle x_0, x_1, x_2, \dots, n : (\Gamma, \text{une}), n :$   
 198  $(\Gamma_2, \text{une}_2) \rangle, \prec_{B_2}, \text{min}_{B_2})$ , and so on. The symbol ‘|’ indicates the occurrence of a split in  
 199 the branch, i.e., a non-deterministic choice of possible outcomes, each of which needs to be  
 200 explored. It is worth to mention that after applying a static rule on  $n : (\Gamma, \text{une})$ , the leaf  
 201 nodes of all the new branches keep the same label  $n$ .

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

$$\begin{array}{l} \text{(Contradiction)} \quad \frac{n : (\{\alpha, \neg\alpha\} \uplus \Sigma, \text{une})}{(\text{✗})} \qquad \qquad \qquad \text{(Empty)} \quad \frac{n : (\emptyset, \emptyset)}{(\text{✓})} \\ \\ \text{(}\wedge\text{)} \quad \frac{n : (\{\alpha_1 \wedge \alpha_2\} \uplus \Sigma, \text{une})}{n : (\{\alpha_1, \alpha_2\} \cup \Sigma, \text{une})} \qquad \qquad \qquad \text{(}\vee\text{)} \quad \frac{n : (\{\alpha_1 \vee \alpha_2\} \uplus \Sigma, \text{une})}{n : (\{\alpha_1\} \cup \Sigma, \text{une}) \mid n : (\{\alpha_2\} \cup \Sigma, \text{une})} \\ \\ \text{(}\square\text{)} \quad \frac{n : (\{\square\alpha_1\} \uplus \Sigma, \text{une})}{n : (\{\alpha_1, \square\alpha_1\} \cup \Sigma, \text{une})} \qquad \qquad \qquad \text{(}\diamond\text{)} \quad \frac{n : (\{\diamond\alpha_1\} \uplus \Sigma, \text{une})}{n : (\{\alpha_1\} \cup \Sigma, \text{une}) \mid n : (\{\square\diamond\alpha_1\} \cup \Sigma, \text{une})} \end{array}$$

208 Before introducing the rule for the non-monotonic operator  $\diamond$ , we discuss firsthand the  
 209 notion of *fulfillment* for classical and non-monotonic eventualities. Following Reynolds’  
 210 tableau, let an eventuality  $\diamond\alpha$  be in a node with a label  $n$ . If the sentence  $\alpha$  appears in a

211 proper successor node  $x$  with the label  $m \geq n$ , we say that  $\diamond\beta$  at the position  $n$  is *fulfilled* in  
 212  $m$ . In a similar fashion, we define the fulfillment for non-monotonic eventualities as follows:

213 ► **Definition 7** (Fulfillment of non-monotonic eventualities). *Let a non-monotonic eventuality*  
 214  *$\diamond\alpha$  be in a node with a label  $n$  in a branch  $B$ . If  $\alpha$  appears in a proper successor node  $x$  with*  
 215 *a label  $m \geq n$ , and  $(n, m) \in \text{min}_B$ , we say  $\diamond\alpha$  at the position  $n$  is fulfilled in  $m$ .*

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

$$225 \quad (\diamond) \frac{n : (\{\diamond\alpha_1\} \uplus \Sigma, \text{une}), \prec_B, \text{min}_B}{n : (\{\alpha_1\} \cup \Sigma, \text{une}), \prec_B, \text{min}_B \cup \{(n, n)\} \mid n : (\Sigma, \text{une} \cup \{(n, \diamond\alpha_1)\}), \prec_B, \text{min}_B}$$

226 For the rule  $(\diamond)$ , we explore two outcomes. The first outcome is when the non-monotonic  
 227 eventuality  $\diamond\alpha_1$  at  $n$  is fulfilled in  $n$ . We then add  $\alpha_1$  to the set of sentences  $\Gamma$  of the leaf  
 228 node and add  $(n, n) \in \text{min}$  of the branch. The second outcome is when  $\diamond\alpha_1$  is not fulfilled in  
 229  $n$ , then we add the pair to  $(n, \diamond\alpha_1)$  to  $\text{une}$  of the leaf node as a non-monotonic eventuality  
 230 that needs to be fulfilled. Example 8 shows the application of  $[\diamond]$  rule.

231 ► **Example 8.** Let a branch  $B$  have  $\prec_B, \text{min}_B$  and a leaf node  $5 : (\{p, q, \square(p \wedge q), \diamond r\}, \emptyset)$ .  
 232 After applying  $(\diamond)$  rule on  $\diamond r$ , we have two new branches  $B_1$  and  $B_2$ . The branch  $B_1$  has a  
 233 leaf node where the sentence  $r$  is in  $\Gamma$  of the leaf node and  $(5, 5) \in \text{min}_{B_1}$ . The branch  $B_2$   
 has  $(5, \diamond r)$  in  $\text{une}$  of the leaf node.

$$\begin{array}{c} 5 : (\{p, q, \square(p \wedge q), \diamond r\}, \emptyset), \prec_B, \text{min}_B \\ \swarrow \quad \searrow \\ 5 : (\{p, q, \square(p \wedge q), r\}, \emptyset), \prec_B, \text{min}_B \cup \{(5, 5)\} \quad 5 : (\{p, q, \square(p \wedge q)\}, \{(5, \diamond r)\}), \prec_B, \text{min}_B \end{array}$$

234

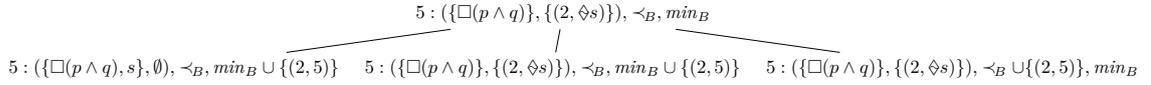
235 The next static rule we discuss is the rule  $(\text{une})$ . Let  $n, n'$  be two labels such that  $n' < n$ ,  
 236 for each label  $n$  and a pair  $(n', \diamond\alpha_1)$ , the rule  $(\text{une})$  is applied one and only one time. The  
 237 rule goes as follows:

$$238 \quad (\text{une}) \frac{n : (\Gamma, \{(n', \diamond\alpha_1)\} \uplus U), \prec_B, \text{min}_B}{n : (\{\alpha_1\} \cup \Gamma, U), \prec_B, \text{min}_B \cup \{(n', n)\} \mid} \\ \frac{n : (\Gamma, \{(n', \diamond\alpha_1)\} \cup U), \prec_B, \text{min}_B \cup \{(n', n)\} \mid}{n : (\Gamma, \{(n', \diamond\alpha_1)\} \cup U), \prec_B \cup \{(n', n)\}, \text{min}_B}$$

239 For the rule  $(\text{une})$ , we explore three outcomes. The first outcome is when  $\diamond\alpha_1$  at the  
 240 position  $n'$  is fulfilled at  $n$ . We remove  $(n', \diamond\alpha_1)$  from  $\text{une}$ , then we add  $\alpha$  in  $\Gamma$  of the leaf  
 241 node and  $(n', n)$  in  $\text{min}$  of the branch. In the second and third branches, we explore the  
 242 outcome of  $\diamond\alpha_1$  not being fulfilled yet in  $n$ , we keep the pair  $(n', \diamond\alpha_1)$  on the leaves of two  
 243 branches. The second branch explore the outcome of  $n$  being minimal to  $n'$  w.r.t. to  $\prec$  of

244 the branch. We then add  $(n', n)$  to the  $min$  of the branch. In the third branch, we explore  
 245 the outcome of  $n$  not being minimal to  $n'$  w.r.t.  $\prec$  of the branch. It means that there exists  
 246 a temporal state  $m'$  that come after  $n'$  where  $m'$  is preferred to  $n$  w.r.t. to  $\prec$  of the branch,  
 247 we add the pair  $(n', n)$  in  $\prec$  of the branch to represent this case. It is worth to mention that  
 248 the rule  $(une)$  does not apply when the label of the node  $n$  is the same as  $(n, \diamond\alpha_1)$ . The  
 249 reason behind this is that we have already explored the case when the eventuality is fulfilled  
 250 in  $n$  thanks to  $(\diamond)$  rule. Example 9 shows the application of  $(une)$  rule.

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



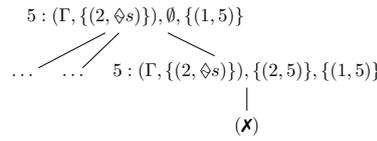
255

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

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

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

264 ► **Example 10.** Let  $B$  be a branch where  $\prec_B$  is empty,  $min_B$  has  $(1, 5)$  in it, and a leaf node  
 265  $5 : (\Gamma, \{(2, \diamond s)\})$ . After applying  $une$  rule on  $(2, \diamond s)$ , we have three branches  $B_1$ ,  $B_2$  and  $B_3$ .  
 266 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  
 267 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  $B_3$ .



268

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

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

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

277

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

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

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

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

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

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

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

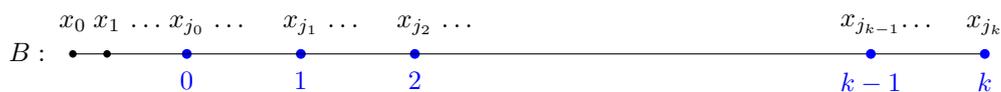
319 Once we are in a state-labelled node, we check for the loop and prune within the branch  
 320 before applying the transition rule. If the transition rule is applied on a state node with a  
 321 label  $n$ , we obtain a new node with the label  $n + 1$ . We can then expand the tree from this  
 322 node by applying static rules until we find ticked branches (thanks to the **empty** rule), closed  
 323 branches (thanks to the **contradiction** or  **$\prec$ -inconsistency** rules), or branches with a state

324 node that has the label  $n + 1$ . We then repeat the cycle between static and dynamic rules.  
325 We can see that the tableau method does not go indefinitely. Thanks to prune rule, we close  
326 any branch ( $\times$ ) that does not fulfill any eventuality in the current temporal state. Anytime  
327 we apply a transition rule (from  $n$  to  $n + 1$ ), we need to fulfill at least one eventuality in  $n$ .  
328 Therefore, as long as a branch is not closed with prune rule, eventuality sentences (either  
329 classical or non-monotonic) get consumed one by one over the execution of the method. Thus  
330 any branch that is not closed with prune has no eventualities left to fulfill. Note that if a  
331 branch contains at least one sentence of the form  $\Box\alpha_{bool}$ , it is then ticked thanks to the loop  
332 rule ( $\Box\alpha_{bool}$  sentences do not get consumed). Otherwise, it is ticked thanks to the empty  
333 rule. Therefore any tableau  $\mathcal{T}$  for a sentence in  $\mathcal{L}_1$  is a *saturated* tableau.

## 4 Soundness and completeness

### 4.1 Soundness

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



■ **Figure 1** Illustration of the branch  $B$ .

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

$$344 \quad V(i) := \begin{cases} \{p \in \mathcal{P} \mid p \in \Gamma_{x_{j_i}}\}, & \text{if } 0 \leq i \leq k; \\ V(k), & \text{otherwise.} \end{cases}$$

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

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

355 If the branch was ticked with the empty rule:

$$356 \quad \Delta_B(i) := \begin{cases} \bigcup_{x_0 \leq x \leq x_{j_0}} \Gamma_x, & \text{if } i = 0; \\ \bigcup_{x_{j_{i-1}} < x \leq x_{j_i}} \Gamma_x, & \text{if } 1 \leq i \leq k - 1; \\ \{\}, & \text{otherwise.} \end{cases}$$

357 If the branch was ticked with the loop rule:

$$358 \quad \Delta_B(i) := \begin{cases} \bigcup_{x_0 \leq x \leq x_{j_0}} \Gamma_x, & \text{if } i = 0; \\ \bigcup_{x_{j_{i-1}} < x \leq x_{j_i}} \Gamma_x, & \text{if } 1 \leq i \leq k; \\ \Delta_B(k), & \text{otherwise.} \end{cases}$$

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

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

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

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

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

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

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

376 ► **Lemma 15.** *Let  $B$  be a successful branch and  $0 \leq i \leq k$ . If  $\diamond\alpha_1 \in \Delta_B(i)$ , then there exists  
377  $d \geq i$  s.t.  $(i, d) \in \min_B$  and  $\alpha_1 \in \Delta_B(d)$ .*

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

380 ■ When the branch  $B$  is ticked with empty rule, whenever  $\diamond\alpha_1 \in \Delta_B(i)$ , then we have  
381  $0 \leq i \leq k - 1$ . Since  $\diamond\alpha_1 \in \Delta_B(i)$ , then  $\diamond\alpha_1 \in \Gamma_x$  where  $x_{j_{i-1}} < x \leq x_{j_i}$ . Let  $x$  be  
382 the node where we apply the rule ( $\diamond$ ) on  $\diamond\alpha_1$ , then we either have  $\alpha_1$  in  $\Gamma$  of the next  
383 node with  $(i, i) \in \min_B$  or we have  $(i, \diamond\alpha_1) \in \text{une}$  of the next node. If  $\alpha_1$  is in  $\Gamma$  of  
384 the next node, then the lemma holds. If  $(i, \diamond\alpha_1) \in \text{une}$  of the next node, then we find  
385  $(i, \diamond\alpha_1) \in \text{une}_{x_{j_i}}$ . Thanks to the transition rule, we have  $(i, \diamond\alpha_1) \in \text{une}_{x_{j_{i+1}}}$ . By applying  
386 the rule *une* on a node with the label  $i + 1$ , then we either have  $\alpha_1$  in  $\Gamma$  of the next node  
387 with  $(i, i + 1) \in \min_B$  or we have  $(i, \diamond\alpha_1) \in \text{une}$  (the two remaining branches) of the  
388 next node. In a similar way as in  $i$ , we can conclude that either  $\alpha_1 \in \Delta_B(i + 1)$  with  
389  $(i, i + 1) \in \min_B$  (the lemma holds) or  $(i, \diamond\alpha_1) \in \text{une}_{x_{j_{i+1}}}$ . Without loss of generality,  
390  $(i, \diamond\alpha_1)$  is in  $\text{une}_{x_{j_f}}$  for  $i \leq f \leq k - 1$  unless we find  $i \leq d \leq f$  with  $\alpha_1 \in \Delta_B(d)$   
391 and  $(i, d) \in \min_B$ . Since the branch is closed thanks to the empty rule, it means that  
392  $(i, \diamond\alpha_1) \notin \text{une}_{x_{j_{k-1}}}$ . Therefore, there is a state  $i \leq d \leq k - 1$  where  $\alpha_1 \in \Delta_B(d)$  with  
393  $(i, d) \in \min_B$ .

394 ■ When the branch  $B$  is ticked with loop rule, the proof is analogous to the case of the  
395 empty rule (notice that we also have  $(i, \diamond\alpha_1) \notin \text{une}_{x_{j_k}}$ ).

396

397 ► **Theorem 16.** *Let  $B$  be a ticked branch from a saturated tableau, and  $I_B = (V, \prec)$  be the*  
 398 *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  $I_B, i \models \alpha$ .*

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

- 401 ■  $\alpha = p$ . Let  $p \in \Delta_B(i)$ . By construction of the model  $I_B$ , we have  $p \in V(i)$ . Therefore,  
 402 we have  $I_B, i \models p$ .
- 403 ■  $\alpha = \neg p$ . Let  $\neg p \in \Delta_B(i)$ . Since  $B$  is a ticked branch, then it was not closed with the  
 404 contradiction rule, therefore we have  $p \notin V(i)$ . Therefore, we have  $I_B, i \models \neg p$ .
- 405 ■  $\alpha = \alpha_1 \wedge \alpha_2$ . 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  
 406 hypothesis on  $\alpha_1, \alpha_2$ , we have  $I_B, i \models \alpha_1$  and  $I_B, i \models \alpha_2$ . Thus, we have  $I_B, i \models \alpha_1 \wedge \alpha_2$ .
- 407 ■  $\alpha = \alpha_1 \vee \alpha_2$ . Let  $\alpha_1 \vee \alpha_2 \in \Delta_B(i)$ . By  $\vee$ -rule, we either have  $\alpha_1$  or  $\alpha_2$  in  $\Delta_B(i)$ . Suppose  
 408 that  $\alpha_1 \in \Delta_B(i)$ , by induction hypothesis on  $\alpha_1$ , we have  $I_B, i \models \alpha_1$ . Therefore, we have  
 409  $I_B, i \models \alpha_1 \vee \alpha_2$ . Same reasoning applies when  $\alpha_2 \in \Delta_B(i)$ .
- 410 ■  $\alpha = \bigcirc \alpha_1$ . Let  $\bigcirc \alpha_1 \in \Delta_B(i)$ . Thanks to Lemma 11, we have  $\alpha_1 \in \Delta_B(i+1)$ . By induction  
 411 hypothesis on  $\alpha_1$ , we have  $I_B, i+1 \models \alpha_1$ . Therefore, we have  $I_B, i \models \bigcirc \alpha_1$ .
- 412 ■  $\alpha = \square \alpha_1$ . Let  $\square \alpha_1 \in \Delta_B(i)$ . Thanks to Lemma 12, we have  $\alpha_1 \in \Delta_B(f)$  for all  $f \geq i$ .  
 413 By induction hypothesis on  $\alpha_1$ , we have  $I_B, f \models \alpha_1$  for all  $f \geq i$ . Therefore, we have  
 414  $I_B, i \models \square \alpha_1$ .
- 415 ■  $\alpha = \diamond \alpha_1$ . Let  $\diamond \alpha_1 \in \Delta_B(i)$ . Thanks to Lemma 13, we have  $\alpha_1 \in \Delta_B(d)$  for some  $d \geq i$ .  
 416 By induction hypothesis on  $\alpha_1$ , we have  $I_B, d \models \alpha_1$ . Therefore, we have  $I_B, i \models \diamond \alpha_1$ .
- 417 ■  $\alpha = \heartsuit \alpha_1$ . Let  $\heartsuit \alpha_1 \in \Delta_B(i)$ . Depending on where  $i$  is, we have two cases:  
 418 ■ In the case of  $i > k$ , since  $\heartsuit \alpha_1 \in \Delta_B(i)$ , then we have  $\heartsuit \alpha_1 \in \Delta_B(k)$ . Furthermore,  
 419 since the branch is ticked with loop rule, we know that  $(i, \heartsuit \alpha_1) \notin \text{une}_{x_{j_k}}$ . Therefore  
 420  $\alpha_1 \in \Delta_B(k)$ , thus  $\alpha_1 \in \Delta_B(i)$ . Furthermore, since  $\prec := \prec_B$ , and there is no  $f \geq i$  such  
 421  $(f, i) \in \prec_B$ , then  $i \in \min_{\prec}(i)$ , and therefore,  $I_B, i \models \heartsuit \alpha_1$ .
- 422 ■  $0 \leq i \leq k$ . Thanks to Lemma 15, there exists  $d \geq i$  s.t.  $\alpha_1 \in \Delta_B(d)$  and  $(i, d) \in \min_B$ .  
 423 By induction hypothesis on  $\alpha_1$ , we have  $I_B, d \models \alpha_1$ . Thanks to Proposition 14, there  
 424 is no  $i \leq f \leq k$  where  $(f, d) \in \prec_B$  and therefore there is no  $i \leq f \leq k$  where  $(f, d) \in \prec$ .  
 425 Furthermore, by the construction of the model  $I_B$ , there is no  $f \geq k$  where  $(f, d) \in \prec$ .  
 426 Therefore, we have  $d \in \min_{\prec}(i)$ . Thus, we have  $I_B, i \models \heartsuit \alpha_1$ .

427

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

## 430 4.2 Completeness

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

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

436 The idea behind this proof is to have an intermediate sequence  $s$  that serves as a link  
 437 between an interpretation  $I$  that satisfies the sentence  $\alpha$  and a tableau  $\mathcal{T}$  for  $\alpha$ . The sequence  
 438  $s$  is a tuple  $s := (\langle x_0, x_1, x_2, \dots \rangle, \prec_s, \min_s)$  where each  $x_i$  is a pair  $(\Gamma, \text{une})$ ,  $\prec_s, \min_s$  are  
 439 the set of constraints that the sequence  $s$  must follow in order to be coherent with  $\prec$  of

440 the interpretation. The set  $\prec_s$  is not an ordering relation, it records instances of points of  
 441 time not being minimal to other points of time w.r.t. the ordering relation  $\prec$ . Remember  
 442 that each when we apply the *une* rule, we add a pair  $(n', n)$  to  $\prec$  in order to symbolize the  
 443 outcome of  $n$  not being minimal to  $n'$ . The set of  $min_s$  records the instances of points of  
 444 points of time being minimal to other points of time w.r.t. the ordering relation  $\prec$ .

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

$$449 \quad Inv(x_i, J(x_i)) \left\{ \begin{array}{l} \text{For each } \alpha \in \Gamma_{x_i}, \text{ we have } I, J(x_i) \models \alpha; \\ \text{For each } (J_1, \diamond\alpha_1) \in une_{x_i}, \text{ there exists } J_2 \geq J(x_i) \text{ where} \\ \quad J_2 \in min_{\prec}(J_1) \text{ and } I, J_2 \models \alpha_1; \\ \text{For each } (J_1, J_2) \in min_s, \text{ we have } J_2 \in min_{\prec}(J_1); \\ \text{For each } (J_1, J_2) \in \prec_s, \text{ there exists } J_3 \geq J_1 \text{ s.t. } (J_3, J_2) \in \prec \\ \quad (\text{in other words } J_2 \notin min_{\prec}(J_1)). \end{array} \right.$$

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

## 466 5 Conclusion

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

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

477 ——— **References** ———

- 478 1 M. Ben-Ari. *Mathematical Logic for Computer Science, third edition*. Springer, 2012.
- 479 2 K. Britz, T. Meyer, and I. Varzinczak. Preferential reasoning for modal logics. *Electronic*  
480 *Notes in Theoretical Computer Science*, 278:55 – 69, 2011. doi:[https://doi.org/10.1016/j.](https://doi.org/10.1016/j.entcs.2011.10.006)  
481 [entcs.2011.10.006](https://doi.org/10.1016/j.entcs.2011.10.006).
- 482 3 K. Britz, T. Meyer, and I. Varzinczak. Semantic foundation for preferential description logics.  
483 In *AI 2011: Advances in Artificial Intelligence*, pages 491–500, Berlin, Heidelberg, 2011.  
484 Springer Berlin Heidelberg.
- 485 4 K. Britz and I. Varzinczak. From KLM-style conditionals to defeasible modalities, and back.  
486 *Journal of Applied Non-Classical Logics*, 28(1):92–121, 2018. doi:[10.1080/11663081.2017.](https://doi.org/10.1080/11663081.2017.1397325)  
487 [1397325](https://doi.org/10.1080/11663081.2017.1397325).
- 488 5 K. Britz and I. Varzinczak. Preferential tableaux for contextual defeasible  $\mathcal{ALC}$ . In *Proceedings*  
489 *of the 28th International Conference on Automated Reasoning with Analytic Tableaux and*  
490 *Related Methods (TABLEAUX)*, number 11714 in LNCS, pages 39–57, 2019.
- 491 6 A. Chafik, F. Cheikh Alili, J.-F. Condotta, and I. Varzinczak. On the decidability of a  
492 fragment of preferential LTL. In *27th International Symposium on Temporal Representation*  
493 *and Reasoning, TIME 2020*, volume 178 of *LIPICs*, pages 19:1–19:19, 2020. doi:[10.4230/](https://doi.org/10.4230/LIPICs.TIME.2020.19)  
494 [LIPICs.TIME.2020.19](https://doi.org/10.4230/LIPICs.TIME.2020.19).
- 495 7 D. M. Gabbay. The declarative past and imperative future: Executable temporal logic for  
496 interactive systems. In *Temporal Logic in Specification, Altrincham, UK, April 8-10, 1987,*  
497 *Proceedings*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer, 1987.  
498 doi:[10.1007/3-540-51803-7\\_36](https://doi.org/10.1007/3-540-51803-7_36).
- 499 8 L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. Preferential description logics. In  
500 *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, number 4790 in LNAI,  
501 pages 257–272. Springer, 2007.
- 502 9 L. Giordano, V. Gliozzi, N. Olivetti, and G.L. Pozzato. Analytic tableaux calculi for KLM logics  
503 of nonmonotonic reasoning. *ACM Transactions on Computational Logic*, 10(3):18:1–18:47,  
504 2009.
- 505 10 Y. Kesten, Z. Manna, H. Mcguire, and A. Pnueli. A decision algorithm for full propositional  
506 temporal logic. *LNCS*, 697, 04 1999.
- 507 11 S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and  
508 cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.
- 509 12 Z. Manna and A. Pnueli. *Temporal verification of reactive systems - safety*. Springer, 1995.
- 510 13 A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of*  
511 *Computer Science (SFCS 1977)*, pages 46–57, Oct 1977. doi:[10.1109/SFCS.1977.32](https://doi.org/10.1109/SFCS.1977.32).
- 512 14 M. Reynolds. A new rule for LTL tableaux. *Electronic Proceedings in Theoretical Computer*  
513 *Science*, 226:287–301, Sep 2016. URL: <http://dx.doi.org/10.4204/EPTCS.226.20>, doi:  
514 [10.4204/eptcs.226.20](https://doi.org/10.4204/eptcs.226.20).
- 515 15 M. Reynolds. A traditional tree-style tableau for LTL, 2016. arXiv:[1604.03962](https://arxiv.org/abs/1604.03962).
- 516 16 K. Y. Rozier and M. Y. Vardi. LTL satisfiability checking. In *Model Checking Software*, pages  
517 149–167, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- 518 17 S. Schwendimann. A new one-pass tableau calculus for PLTL. In *Automated Reasoning with*  
519 *Analytic Tableaux and Related Methods*, pages 277–291. Springer Berlin Heidelberg, 1998.
- 520 18 Y. Shoham. *A Semantical Approach to Nonmonotonic Logics*, page 227–250. Morgan Kaufmann  
521 Publishers Inc., San Francisco, CA, USA, 1987.
- 522 19 Y. Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial*  
523 *Intelligence*. MIT Press, 1988.
- 524 20 A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*,  
525 32(3):733–749, July 1985. doi:[10.1145/3828.3837](https://doi.org/10.1145/3828.3837).
- 526 21 P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1):72 – 99,  
527 1983. doi:[https://doi.org/10.1016/S0019-9958\(83\)80051-5](https://doi.org/10.1016/S0019-9958(83)80051-5).
- 528 22 P. Wolper. The tableau method for temporal logic. *Logique Et Analyse*, 28:110–111, 01 1985.

## 529 **A** Soundness proof

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

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

533 ■ When the branch  $B$  is ticked with empty rule. We can see that when  $i \geq k$ ,  $\Delta_B(i) = \{\}$   
534 and therefore  $\circ\alpha_1 \notin \Delta_B(i)$ . We also know that since  $\Delta_B(k) = \{\}$ , then there is no  
535  $\circ\alpha_1 \in \Gamma_{x_{j_{k-1}}}$ . Furthermore, there is no static rule that removes  $\circ\alpha_1$ , we can conclude  
536 that there is no  $\circ\alpha_1 \in \Delta_B(k-1)$ .

537 Otherwise, in the case of  $0 \leq i < k-1$ , if  $\circ\alpha_1 \in \Delta_B(i)$ , then  $\circ\alpha_1 \in \Gamma_x$  where  
538  $x_{j_{i-1}} < x \leq x_{j_i}$ . Since there is no static rule that removes  $\circ\alpha_1$ , we have  $\circ\alpha_1 \in \Gamma_{x_{j_i}}$ .  
539 Furthermore, after applying the transition rule on the node  $x_{j_i}$ , we have  $\alpha_1 \in \Gamma_{x_{j_{i+1}}}$ .  
540 Thus, we have  $\alpha_1 \in \Delta_B(i+1)$ .

541 ■ When the branch  $B$  is ticked with loop rule. In the case of  $0 \leq i < k$ , the proof is analogous  
542 to the case of empty rule. When  $i = k$ , if  $\circ\alpha_1 \in \Delta_B(k)$ , then  $\circ\alpha_1$  is subsequently in  $\Gamma_{x_{j_k}}$ .  
543 Since  $B$  is ticked with loop, then  $\alpha_1$  is a sentence of the form  $\Box\alpha_{bool}$  and  $\Box\alpha_{bool} \in \Gamma_x$   
544 ( $x_{j_{k-1}} < x \leq x_{j_k}$ ) and therefore  $\Box\alpha_{bool} \in \Delta_B(k)$ . Moreover, we have  $\Delta_B(k) = \Delta_B(k+1)$ .  
545 Therefore, we have  $\Box\alpha_{bool} \in \Delta_B(k+1)$  and thus  $\alpha_1 \in \Delta_B(k+1)$ .

546 In the case where  $i \geq k$ . If  $\circ\alpha_1 \in \Delta_B(i)$ , then  $\circ\alpha_1 \in \Delta_B(k-1)$ . As mentioned before,  
547 since  $\circ\alpha_1 \in \Delta_B(k-1)$ , then  $\alpha_1$  is  $\Box\alpha_2$  and  $\Box\alpha_2 \in \Delta_B(k-1)$ . Since  $\Box\alpha_2 \in \Delta_B(k-1)$ ,  
548 then  $\Box\alpha_2 \in \Delta_B(i+1)$  and therefore  $\alpha_1 \in \Delta_B(i+1)$ .

549 ◀

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

552 **Proof.** Let  $B$  be a ticked branch of the tableau,  $k$  be the label of the last node and  $i \in \mathbb{N}$ .  
553 For all  $0 \leq i \leq k$ , whenever  $\Box\alpha_1 \in \Delta_B(i)$ , then both  $\alpha_1$  and  $\circ\Box\alpha_1$  is in  $\Delta_B(i)$ . By  
554 Lemma 11, since  $\circ\Box\alpha_1 \in \Delta_B(i)$ , then we have  $\Box\alpha_1 \in \Delta_B(i+1)$ . By successive applications  
555 of Lemma 11, we have  $\{\alpha_1, \Box\alpha_1, \circ\Box\alpha_1\} \subseteq \Delta_B(f)$  for all  $i \leq f \leq k$ . Note that in the case  
556 of a branch ticked with empty rule, since  $\Delta_B(k) = \{\}$ ,  $\Box\alpha_1$  cannot be in any  $\Delta_B(i)$  where  
557  $0 \leq i \leq k$ . In other words, if a branch contains  $\Box\alpha_1$ , it can only be ticked with loop rule.

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

560 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,  
561  $\{\alpha_1, \Box\alpha_1, \circ\Box\alpha_1\} \subseteq \Delta_B(k-1)$ . Thus, we have  $\{\alpha_1, \Box\alpha_1, \circ\Box\alpha_1\} \subseteq \Delta_B(f)$  for all  $f \geq k$  and  
562 subsequently  $\{\alpha_1, \Box\alpha_1, \circ\Box\alpha_1\} \subseteq \Delta_B(f)$  for all  $f \geq i$ . ◀

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

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

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

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

575 ■ When the branch  $B$  is ticked with loop rule and in the case of  $0 \leq i \leq k$ , the proof  
 576 is analogous to the case of empty rule (notice that  $\circ\Diamond\alpha_1 \notin \Delta_B(k)$  also in the case of  
 577 branches ticked with loop).

578 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,  
 579 since the branch is ticked with loop rule, we know that  $\circ\Diamond\alpha_1 \notin \Delta_B(k)$ . Therefore  
 580  $\alpha_1 \in \Delta_B(k)$ , thus  $\alpha_1 \in \Delta_B(i)$ .

581



## 582 B Completeness proof

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

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

586 **[Contradiction]:** If the sequence  $s$  is closed, then we have  $p$  and  $\neg p$  in  $\Gamma_{x_i}$ . Since we  
 587 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  
 588 interpretation  $I \in \mathcal{I}$ .

589 **[ $\wedge$ ]:** Suppose that the rule  $\wedge$  is applied to the sentence  $\alpha_1 \wedge \alpha_2$  on the node  $f(x_i)$   
 590 of the tableau  $\mathcal{T}$ . Let  $y$  be the child node of the node  $f(x_i)$  in the branch. We have  
 591  $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\alpha_1 \wedge \alpha_2\}) \cup \{\alpha_1, \alpha_2\}$ . We define the next node in the sequence  $x_{i+1}$  with  
 592  $\Gamma_{x_{i+1}} = \Gamma_y$ ,  $une_{x_{i+1}} = une_{x_i}$ , and the sets  $min_s, \prec_s$  remain unchanged. Since we have  
 593  $Inv(x_i, J(x_i))$  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  $x_{i+1}$ ,  
 594 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  
 595 second conditions of  $Inv(x_{i+1}, J(x_i))$  are met. Moreover, since  $min_s, \prec_s$  remain unchanged  
 596 and we have  $Inv(x_i, J(x_i))$ , then the third and fourth conditions of  $Inv(x_{i+1}, J(x_i))$  are met.  
 597 Consider that  $J(x_{i+1}) = J(x_i)$ , the invariant  $Inv(x_{i+1}, J(x_i))$  holds.

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

602 **[ $\vee$ ]:** Suppose that the rule  $\vee$  is applied to the sentence  $\alpha_1 \vee \alpha_2$  on the node  $f(x_i)$  of  
 603 the tableau  $\mathcal{T}$ . We obtain two children nodes  $y$  and  $z$  of  $f(x_i)$ . We have  $\Gamma_y = (\Gamma_{f(x_i)} \setminus$   
 604  $\{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_1\}$  and  $\Gamma_z = (\Gamma_{f(x_i)} \setminus \{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_2\}$ . Since we have  $Inv(x_i, J(x_i))$ ,  
 605 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  
 606  $J(x_{i+1}) = J(x_i)$ , we discuss two cases:

607 ■ 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  
 608  $une_{x_{i+1}} = une_{x_i}$ . We know that  $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\alpha_1 \vee \alpha_2\}) \cup \{\alpha_1\}$ . Therefore for all  
 609  $\gamma \in \Gamma_{x_{i+1}}$ , we have  $I, J(x_i) \models \gamma$ . Thus, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.

610 ■ Case 2: Otherwise, when  $I, J(x_i) \models \alpha_2$ , then we define the node  $x_{i+1}$  with  $\Gamma_{x_{i+1}} = \Gamma_z$   
 611 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  
 612  $\gamma \in \Gamma_{x_{i+1}}$ , we have  $I, J(x_i) \models \gamma$ . Thus, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.

613 **[ $\diamond$ ]:** Suppose that the rule  $\diamond$  is applied to the sentence  $\Diamond\alpha_1$  on the node  $f(x_i)$  of the tableau  
 614  $\mathcal{T}$ . We obtain two children nodes  $y$  and  $z$  of  $f(x_i)$ . We have  $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\Diamond\alpha_1\}) \cup \{\alpha_1\}$   
 615 and  $\Gamma_z = (\Gamma_{f(x_i)} \setminus \{\Diamond\alpha_1\}) \cup \{\circ\Diamond\alpha_1\}$ . Since we have  $Inv(x_i, J(x_i))$ , and  $I, J(x_i) \models \Diamond\alpha_1$ , then

616 we have  $I, J(x_i) \models \alpha_1 \vee \circ\Diamond\alpha_1$ . Therefore, we either have  $I, J(x_i) \models \alpha_1$  or  $I, J(x_i) \models \circ\Diamond\alpha_1$ .  
 617 Consider that  $J(x_{i+1}) = J(x_i)$ , we discuss two cases:

- 618 ■ 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  
 619  $une_{x_{i+1}} = une_{x_i}$ . We know that  $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Diamond\alpha_1\}) \cup \{\alpha_1\}$ . Therefore for all  $\gamma \in \Gamma_{x_{i+1}}$ ,  
 620 we have  $I, J(x_i) \models \gamma$ . Thus, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.
- 621 ■ Case 2: When  $I, J(x_i) \models \circ\Diamond\alpha_1$ , then we define the next node  $x_{i+1}$  with  $\Gamma_{x_{i+1}} = \Gamma_z$   
 622 and  $une_{x_{i+1}} = une_{x_i}$ . We know that  $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Diamond\alpha_1\}) \cup \{\circ\Diamond\alpha_1\}$ . Therefore for all  
 623  $\gamma \in \Gamma_{x_{i+1}}$ , we have  $I, J(x_i) \models \gamma$ . Thus, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.

624  $[\Box]$ : Suppose that the rule  $\Box$  is applied to the sentence  $\Box\alpha_1$  on the node  $f(x_i)$  of the  
 625 tableau  $\mathcal{T}$ . Let  $y$  be the child node of the node  $f(x_i)$  in the branch. We have  $\Gamma_y = (\Gamma_{f(x_i)} \setminus$   
 626  $\{\Box\alpha_1\}) \cup \{\alpha_1, \circ\Box\alpha_1\}$ . We define the next node  $x_{i+1}$  with  $\Gamma_{x_{i+1}} = \Gamma_y$  and  $une_{x_{i+1}} = une_{x_i}$   
 627 and  $I, J(x_i) \models \Box\alpha_1$ , then we have  $I, J(x_i) \models \alpha_1 \wedge \circ\Box\alpha_1$ . Therefore, we have  $I, J(x_i) \models \alpha_1$   
 628 and  $I, J(x_i) \models \circ\Box\alpha_1$ . We know that  $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Box\alpha_1\}) \cup \{\alpha_1, \circ\Box\alpha_1\}$ . Therefore  
 629 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  
 630  $Inv(x_{i+1}, J(x_i))$  holds.

631  $[\Diamond]$ : When the rule  $[\Diamond]$  is applied to  $\Diamond\alpha_1$  on the node  $f(x_i)$  of  $\mathcal{T}$ , we explore two outcomes.  
 632 Let  $n$  be the label of the node  $f(x_i)$  in the branch. In the first outcome, we have a child  $y$   
 633 with  $\Gamma_y = (\Gamma_{f(x_i)} \setminus \{\Diamond\alpha_1\}) \cup \{\alpha_1\}$  and  $(n, n)$  in  $min$  of the branch. In the second outcome,  
 634 we have a child node  $z$  with  $\Gamma_z = (\Gamma_{f(x_i)} \setminus \{\Diamond\alpha_1\})$  and  $une_z = une_{f(x_i)} \cup (n, \Diamond\alpha_1)$ . Since  
 635 we have  $Inv(x_i, J(x_i))$ , and  $\Diamond\alpha_1 \in \Gamma_{x_i}$ , then we have  $I, J(x_i) \models \Diamond\alpha_1$ . It means that there  
 636 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  
 637 discuss two cases:

- 638 ■ 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  
 639 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  
 640 pair  $(J(x_i), J(x_i))$  to  $min_s$ . Notice that we swap the labels of nodes with the position of  
 641 their indexed time point  $J(x_i)$ , we will be using indexed time point  $J$  instead of labels  
 642 throughout this proof. We know that  $\Gamma_{x_{i+1}} = (\Gamma_{x_i} \setminus \{\Diamond\alpha_1\}) \cup \{\alpha_1\}$  with  $I, J(x_i) \models \alpha_1$ .  
 643 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  
 644 sets  $une_{x_{i+1}}, \prec_s$  remains unchanged. Therefore, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.
- 645 ■ Case 2: when  $J_1 > J(x_i)$ , then we define the next node  $x_{i+1}$  of the sequence with  
 646  $\Gamma_{x_{i+1}} = \Gamma_z$ ,  $une_{x_{i+1}} = une_{x_i} \cup \{(J(x_i), \Diamond\alpha_1)\}$ . We also know that  $J_1 > J(x_i)$  and  
 647  $J_1 \in min_{\prec}(J(x_i))$  and  $I, J_1 \models \alpha_1$ . Therefore, the second condition of  $Inv(x_{i+1}, J(x_i))$   
 648 holds on the pair  $(J(x_i), \Diamond\alpha_1)$ . The sets  $min_s$  and  $\prec_s$  remain unchanged. The invariant  
 649  $Inv(x_{i+1}, J(x_i))$  holds.

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

657 On the other hand, since  $x$  is a predecessor of  $x_i$  in  $s$ , then we have  $Inv(x, J(x))$ .  
 658 Furthermore, since we have  $(n_1, \Diamond\alpha_1) \in une_{f(x_i)}$ , it means that when the rule  $[\Diamond]$  is applied  
 659 on the node  $f(x)$ , the branch where  $(n_1, \Diamond\alpha_1) \in une_{f(x+1)}$  is the path that corresponds with  
 660 the interpretation  $I$ . By  $[\Diamond]$  rule, since we have  $Inv(x+1, J(x+1))$ ,  $(n_1, \Diamond\alpha_1) \in une_{f(x+1)}$   
 661 and we know that  $J(x+1) = J(x)$ , then we have  $(J(x), \Diamond\alpha_1) \in une_{x+1}$ . Furthermore, since  
 662 no rule application consumed  $(n_1, \Diamond\alpha_1)$  up to  $f(x_i)$ , then the pair  $(J(x), \Diamond\alpha_1)$  remains also

663 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  
 664  $I, J' \models \alpha_1$ . Consider that  $J(x_{i+1}) = J(x_i)$ , we discuss all possibilities below:

- 665 ■ 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  
 666 define the next node  $x_{i+1}$  with  $\Gamma_{x_{i+1}} = \Gamma_y$ ,  $une_{x_{i+1}} = une_{x_i} \setminus \{(J(x), \diamond\alpha_1)\}$  and add  
 667  $(J(x), J(x_i))$  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  
 668 have  $(J(x), J(x_i)) \in \min_s$  with  $J(x_i) \in \min_{\prec}(J(x))$ . The set  $\prec_s$  remains unchanged.  
 669 Thus, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.
- 670 ■ Case 2: when  $J' > J(x_i)$ , we have two possibilities:
  - 671 ■ Case 2.1: If  $J(x_i) \in \min_{\prec}(J(x))$ , then we define the next node  $x_{i+1}$  with  $\Gamma_{x_{i+1}} = \Gamma_z$ ,  
 672  $une_{x_{i+1}} = une_{x_i}$  and add  $(J(x), J(x_i))$  to  $\min_s$ . We have  $(J(x), J(x_i)) \in \min_s$  with  
 673  $J(x_i) \in \min_{\prec}(J(x))$ . The sets  $\Gamma_{x_{i+1}}$ ,  $une_{x_{i+1}}$  and  $\prec_s$  remain unchanged. Thus, the  
 674 invariant  $Inv(x_{i+1}, J(x_i))$  holds.
  - 675 ■ 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  
 676 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  
 677  $\prec_s$ . 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  
 678  $\min_s$  remain unchanged. Thus, the invariant  $Inv(x_{i+1}, J(x_i))$  holds.

679 **[Transition]:** Suppose that the transition rule is applied on the state node  $f(x_i)$ . Let  
 680  $y$  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  
 681  $une_y = une_{f(x_i)}$ . We define the next node  $x_{i+1}$  in  $s$  with  $\Gamma_{x_{i+1}} = \Gamma_y$  and  $une_{x_{i+1}} = une_{x_i}$ .  
 682 We consider that  $J(x_{i+1}) = J(x_i) + 1$ .

683 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  
 684 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.

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

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

699 Let  $J_1$  be the time point corresponding to the node  $f(x_1)$  with the label  $n_1$ , and let  
 700  $J_2$  be the time point corresponding to the node  $f(x_2)$  with the label  $n_2$ . It is worth to  
 701 mention that  $J_1 \leq J_2 \leq J(x_i)$ . Since  $x, x'$  are predecessors of  $x$ , we have  $Inv(x, J(x))$ ,  
 702  $Inv(x', J(x'))$  and  $Inv(x_i, J(x_i))$ . Therefore, we the rules are applied on  $x$  and  $x'$ , we end  
 703 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  
 704  $J(x_i) \in \min_{\prec}(J_1)$ . On the other hand, since  $(J_2, J(x_i)) \in \prec_s$ , then there exists  $J_3 \geq J_2$   
 705 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.  
 706  $(J_3, J(x_i)) \in \prec$ . This contradicts Definition 4 of minimality w.r.t. to the relation  $\prec$ . Therefore  
 707 this cannot happen in a interpretation  $I \in \mathcal{I}$ .

708 **[Prune]:** Let  $f(x_i)$  be a state node where the prune condition is met. There is a sequence  
 709 within  $s$  that goes the following way,  $x_h = u, x_{h+1}, x_{h+2}, \dots, v = x_i$ . The node  $u$  or  $x_h$  is

710 the state node that comes before  $x_i$  and the node  $v$  is the current state node. Since  $v$  is a  
 711 prune node, we have  $\Gamma_v = \Gamma_u$  and  $une_u = une_v$ . We can see that if we apply the transition  
 712 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  
 713 proceed with the construction of  $s$  as if  $x_i$  was linked to  $f(u)$  instead of  $f(v)$ . Thanks to the  
 714 transition, since we have  $Inv(x_u, J(x_u))$ , then we have  $Inv(x_{i+1}, J(x_i) + 1)$ .

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

722 What we showed so far is that for an interpretation  $I$  and its corresponding sequence  $s$ ,  
 723 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  
 724 that the sequence finishes with a ticked node (such is the case when we end up in [loop] or  
 725 [empty] node). We can see that if the sequence  $s$  is on a [prune] node, we jump back to the  
 726 state node that comes before it. Theoretically, this jump can recur infinitely many times.  
 727 This means that sequence goes infinite on this case (and never find a ticked node). We need  
 728 to prove that this case cannot happen in the sequence  $s$  of  $I$ . Suppose that is the case, that  
 729 means the last jump  $(u_k, v_k)$  in the sequence  $s$  is a recurring jump that goes infinitely many  
 730 times. The jumps  $(u_j, v_j)$  that come before may recur many times but not infinitely many  
 731 times (otherwise,  $(u_k, v_k)$  would not be the last jump). In the recurring jump  $(u_k, v_k)$ , no  
 732 eventuality is fulfilled (whether it is classical or non-monotonic). This entails that when we  
 733 are in a parent node  $u_k < x_l < v_k$  that applies either  $[\diamond]$  or  $[une]$  rule, we move to the child  
 734 node that delays the propagation of the eventuality (we are in Case 2 for both rules).

735 It is worth to point out that we have at least one eventuality in  $u_k$ . Let us take  $\circ\Diamond\alpha_1 \in \Gamma_{u_k}$   
 736 for example, since we have  $Inv(u_k, J(u_k))$ , that means that  $I, J(u_k) \models \circ\Diamond\alpha_1$ . Thus, we take  
 737 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  
 738 hand, for all  $J(u_k) < J < J_{\alpha_1}$ , we have  $I, J \models \Diamond\alpha_1$   $I, J \models \circ\Diamond\alpha_1$ . In other words, each time  
 739 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  
 740 the node in Case 2 of the  $[\diamond]$  rule i.e.,  $\circ\Diamond\alpha_1 \in \Gamma_{x_l}$ . However, in the node indexed with  $J_{\alpha_1}$ ,  
 741 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  
 742 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  
 743 not present within the jump  $(u_k, v_k)$ . Thus breaking the infinite recurring jump  $(u_k, v_k)$ .

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

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