The Journal of Logic and Algebraic Programming 78 (2009) 701–722

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

Dual systems of tableaux and sequents for PLTL[☆]

Jose Gaintzarain[a,*], Montserrat Hermo[b], Paqui Lucio[b], Marisa Navarro[b], Fernando Orejas[c]

[a] Departamento de Lenguajes y Sistemas Informáticos, Universidad del País Vasco, Bilbao, Spain
[b] Departamento de Lenguajes y Sistemas Informáticos, Universidad del País Vasco, San Sebastián, Spain
[c] Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Catalunya, Barcelona, Spain

A R T I C L E I N F O

Article history:
Received 1 August 2008
Accepted 1 May 2009
Available online 9 May 2009

Keywords:
Temporal logic
Tableaux
Invariant-free
Sequent calculi
Cut-free

A B S T R A C T

On one hand, traditional tableau systems for temporal logic (TL) generate an auxiliary graph that must be checked and (possibly) pruned in a second phase of the refutation procedure. On the other hand, traditional sequent calculi for TL make use of a kind of inference rules (mainly, invariant-based rules or infinitary rules) that complicates their automatization. A remarkable consequence of using auxiliary graphs in the tableaux framework and invariants or infinitary rules in the sequents framework is that TL fails to carry out the classical correspondence between tableaux and sequents. In this paper, we first provide a tableau method TTM that does not require auxiliary graphs to decide whether a set of PLTL-formulas is satisfiable. This tableau method TTM is directly associated to a one-sided sequent calculus called TTC. Since TTM is free from all the structural rules that hinder the mechanization of deduction, e.g. weakening and contraction, then the resulting sequent calculus TTC is also free from this kind of structural rules. In particular, TTC is free of any kind of cut, including invariant-based cut. From the deduction system TTC, we obtain a two-sided sequent calculus GTC that preserves all these good freeness properties and is finitary, sound and complete for PLTL. Therefore, we show that the classical correspondence between tableaux and sequent calculi can be extended to TL. Every deduction system is proved to be complete. In addition, we provide illustrative examples of deductions in the different systems.

1. Introduction

Temporal logic (TL) plays a significant role in computer science, since it is an ideal tool for specifying object behaviour, cooperative protocols, reactive systems, digital circuits, concurrent programs and, in general, for reasoning about dynamic systems whose states change along the time.

Tableau systems are refutational proof methods that play a prominent role in the development of automated reasoning for TL (and many other logics). The first detailed description of a tableau method for TL was presented in [24]. Since then, several authors (e.g. [13,2,15]) have proposed and studied tableau methods for different temporal logics, sometimes in the more general frame of modal logic. The interested reader is referred to [12] for a good survey. Traditional tableau methods for TL generate auxiliary graphs that are checked and (possibly) pruned in a second phase of the procedure. Both, the auxiliary graph and the second phase, prevent the association of a sequent calculus proof to each tableau refutation.

Sequent calculi provide a general deductive setting that uniformly embeds refutational methods and other deduction techniques such as goal-directed proofs or natural deduction. Traditional sequent calculi for TL (e.g. [16,17,22]) usually include some inference rules that complicate the automatization of temporal deduction. In particular, temporal sequent calculi either need some form of cut (classical cut or invariant-based cut) or they include infinitary rules. Cut rules imply the

☆ This work has been partially supported by Spanish Project TIN2007-66523 and the Basque Project LoRea GIU07/35.
* Corresponding author. Tel.: +34 94 601 4398; fax: +34 94 601 4300.
E-mail address: jose.gaintzarain@ehu.es (J. Gaintzarain).

1567-8326/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2009.05.001

"invention" of lemmata, called cut formulas, for their application. Invariants are particular cut formulas for proving temporal eventualities. In [16,22], two sequent calculi for TL with invariant-based rules are presented. In fact, in both approaches, a system that includes also a cut rule is presented and then a cut elimination proof is provided. However, invariant-based rules for temporal connectives cannot be avoided. In [17] various sequent calculi are presented for TL without the until operator (this means that the logic considered has a limited expressive power). In that work completeness and cut-elimination proofs, together with various interesting reductions among various calculi are provided. However, every calculus includes either some infinitary rule or some invariant-based rule.

A remarkable consequence of using auxiliary graphs in the tableau framework and invariants or infinitary rules in the sequent framework is that TL fails to carry out the classical correspondence between tableaux and sequents. In classical logic, and even in some non-classical logics (e.g. many-valued logics), each step in a tableau construction corresponds to an inference in the sequent calculus. Therefore, there is an easy, useful and well known correspondence that associates to each tableau a sequent proof, which is a refutation.

In this paper, we introduce a tableau system together with a dual cut-free, invariant-free finitary sequent calculus for Propositional Linear Temporal Logic (PLTL). We first provide a Temporal Tableau Method TTM which does not require auxiliary graphs to decide if a set of PLTL-formulas is satisfiable. The tableau method TTM is directly associated to a one-sided (or Tait style) sequent calculus that we call TTC (from Tait-style Temporal Calculus). Since TTM is free from all the structural rules that hinder the mechanization of deduction, e.g. weakening and contraction, then the resulting sequent calculus TTC is also free from this kind of structural rules. In particular, TTC is free of any kind of cut, including invariant-based cut. From the deduction system TTC, we obtain a two-sided sequent calculus GTC (from Gentzen-style Temporal Calculus) that preserves all these good freeness properties and is finitary, sound and complete for PLTL. Therefore, we show that the classical correspondence between tableaux and sequent calculi can be extended to TL. Such correspondence is mainly enabled by a new style of inference rule for eventualities which introduces a new kind of temporal deduction.

This paper extends and improves the work introduced in two previous papers (cf. [9,8]). In addition to all the work on deductive methods for TL mentioned above, there are two approaches whose results are closely related to ours. On the one hand, in [20] a one-phase tableau calculus is introduced which, unlike our method, is based in checking, on the fly and branch-by-branch, the fulfillment of the so-called eventuality formulas. On the other hand, at the time of the publication of [9], to our knowledge the first published finitary invariant-free sequent calculus for PLTL, we learned the work of Brünnler and Lange (see [5]) which provides an interesting alternative approach to the proof theory of PLTL. The calculus presented in [5] has the analytic superformula property. Actually, in [5], the strategy that leads to prove completeness of the sequent system –which lies in fairly distinguishing exactly one eventuality and sticking to it until it is fulfilled– is incorporated in the sequent system by means of the so called annotated formulas (which do not belong to the logic language). The completeness proof of our system is also based on the mentioned strategy but such a strategy is not incorporated in the system. In this way different strategies can be used. We differentiate between the systematic derivation (which guarantees completeness) and the many other derivations that usually are feasible.

Other proof-theoretic approaches for PLTL include its first axiomatization à la Hilbert presented in [7] and, also, the resolution-based approach started in [6]. See [18] for a good survey about theorem-proving in PLTL and its extensions.

2. Sequent-based deduction systems and tableaux

Sequent calculus, first introduced by Gentzen [10], is the most elegant and flexible system for writing proofs. Each line of a sequent calculus proof is a sequent. A *sequent* was (originally) formed by two sequences of formulas separated by some kind of arrow. The intended meaning of a sequent $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi_1, \psi_2, \ldots, \psi_m$ is the formula

$$\bigwedge_{i=1}^{n} \varphi_i \to \bigvee_{i=1}^{m} \psi_i$$

where $\to$ is the classical connective of implication. The sequence $\varphi_1, \varphi_2, \ldots, \varphi_n$ is called the *antecedent* of the above sequent and $\psi_1, \psi_2, \ldots, \psi_m$ is called its *consequent* (or succedent). Since the seminal work of Gentzen, many variations of the notion of sequent have been explored to provide different sequent-based deduction systems. A sequent calculus is a proof system given by a set of rules that indicates that a sequent may be inferred from a set of sequents. That is, a (finitary) rule consists of a *numerator* formed by a (finite) set of sequents $S_1, \ldots, S_n$ and a *denominator* S separated by a horizontal line, next to which is the name of the rule:[1]

$$(r)\ \frac{S_1, \ldots, S_n}{S}$$

[1] Sometimes, due to space reasons, the rule is formatted as follows:

$$\frac{\begin{array}{c} S_1 \\ \vdots \\ S_n \end{array}}{S}$$

In a rule (r) as above, each sequent S_i is called a *premise* and S is the *conclusion*. Traditionally, a sequent calculus consists of structural rules and connectives rules. The conclusion of a connective rule has a *principal formula* that is affected by the inference. For example

$$(\wedge L)\ \frac{\Delta, \varphi, \psi \vdash \chi}{\Delta, \varphi \wedge \psi \vdash \chi}$$

is a rule for conjunction $(\wedge)$ whose principal formula is $\varphi \wedge \psi$. However, in structural rules, the inference is guided by the whole conclusion. An example of structural rule is classical weakeaning

$$(Wk)\ \frac{\Delta \vdash \chi}{\Delta, \Delta' \vdash \chi}$$

There are many variations of sequents. The simplest one is obtained by allowing the antecedent and consequent to be a (multi)set instead of a sequence. This choice (of sequences, multisets or sets) is directly related to the classical structural rules of exchange and contraction. In particular, the exchange rule only makes sense in sequence-based sequent calculi, whereas the contraction rule, which is well-founded for sequences and multisets, leads to some confusion when sets are considered. More precisely, the classical contraction rule (on the left):

$$\frac{\Delta, \varphi, \varphi \vdash \chi}{\Delta, \varphi \vdash \chi}$$

makes no sense when the antecedent is a set, however some legal application of connectives rules could hide a contraction. For example, the inference

$$\frac{\varphi \wedge \psi, \varphi, \psi \vdash \chi}{\varphi \wedge \psi \vdash \chi}$$

could result from a legal application of the above rule $(\wedge L)$ for $\Delta = \{\varphi \wedge \psi\}$. In classical logic this kind of hidden use of the contraction does not harm, however in temporal logic[2] we must be more careful on this matter. The sequent systems we are going to introduce are based on sets. The notation Δ, φ stands for $\Delta \cup \{\varphi\}$ where $\varphi \notin \Delta$. This convention clearly disallows hidden contraction. In particular, it disallows the above inference that uses the rule $(\wedge L)$ for $\Delta = \{\varphi \wedge \psi\}$.

Another simple variation of sequent is related to the cardinality of the consequent. That is, sequents can be either multiple-conclusioned or single-conclusioned, or even one-sided, respectively depending on whether the consequent is a set, a singleton or empty.[3] One-sided sequents were first used by Schütte [19] with multisets and by Tait [23] with sets, hence when a new system is presented it is usual to point out whether it is a Gentzen-Schütte style calculus or whether it is a Tait style calculus. There are really two kinds of one-sided sequents: left-handed (empty consequent) and right-handed (empty antecedent). In this paper, we will use left-handed sequents because they are very close to tableau systems. In fact, we will give a tableau system TTM that is directly related to the left-handed sequent calculus TTC. Besides, the established results for the calculus TTC can be easily extended to the two-sided sequent calculus GTC. We have preferred to formulate the calculus GTC by means of single-conclusioned sequents, instead of multiple-conclusioned sequents, because in our opinion single-conclusioned sequents are closer to natural deduction and capture better our intuition in logical reasoning. A multiple-conclusioned system can be easily obtained from GTC.

3. PLTL : language and model theory

A PLTL-formula is built using the constant proposition $\mathbf{F}$, propositional variables (denoted by lowercase letters $p, q, \ldots$) from a set Prop, the classical connectives $\neg$ and $\wedge$, and the temporal connectives $\circ$ and $\mathcal{U}$. A lowercase Greek letter $(\varphi, \psi, \chi, \gamma, \ldots)$ denotes a formula and an uppercase one $(\Phi, \Delta, \Gamma, \Psi, \Omega, \ldots)$ denotes a finite set of PLTL-formulas. PLTL-formulas of the form p and $\neg p$, where $p \in$ Prop, are called *literals*. As usual other connectives can be defined in terms of the previous ones: $\mathbf{T} \equiv \neg \mathbf{F}$, $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \mathcal{R} \psi \equiv \neg(\neg\varphi \mathcal{U} \neg\psi)$, $\Diamond\varphi \equiv \mathbf{T} \mathcal{U} \varphi$, $\Box\varphi \equiv \neg\Diamond\neg\varphi$. Note that $\Box\varphi \equiv \mathbf{F} \mathcal{R} \varphi$. The connectives $\mathbf{T}, \vee, \mathcal{R}$ and $\Box$ are the duals of $\mathbf{F}, \wedge, \mathcal{U}$ and $\Diamond$ respectively. The connective $\circ$ is its own dual. The defined connectives will be used as abbreviations for readability. PLTL-formulas of the form $\varphi \mathcal{U} \psi$ and $\Diamond\varphi$ are called *eventualities*. Eventualities of the form $\varphi \mathcal{U} \psi$ are also called *until formulas*. Literals and PLTL-formulas of the form $\mathbf{F}$, $\mathbf{T}$, $\neg\mathbf{F}$, $\neg\mathbf{T}$ and $\circ\varphi$ are called *elementary*, also sets of elementary formulas are called elementary. In the rest of this paper, *formula* means PLTL-formula.

The operator unnext obtains from any (possibly empty) set of formulas another set of formulas as follows:

$$\text{unnext}(\Phi) = \{\gamma \mid \circ\gamma \in \Phi\}$$

Note that, unnext(Φ) could be the empty set, which we denote by $\{\ \}$.

A logic is said to be compact when it verifies that, given any set of formulas Φ, if every finite subset of Φ is satisfiable then Φ is satisfiable. It is well known that PLTL is a non-compact logic. For example, the infinite set of formulas $\{\circ^i p \mid i \in N\} \cup \{\Diamond\neg p\}$ is not satisfiable but every finite subset of it is satisfiable. As a consequence of the fact that PLTL is a non-compact logic, any strongly complete proof system should be infinitary, i.e., its deduction rules may require infinitely many premises. Our

[2] In general, in modal logic.

[3] There are more sophisticated variants of sequents that are obtained, for example, by adding structure or labels into sequents, but they are out of the scope of this paper.

$$\begin{tikzcd}
s_0 \arrow[r, "R"] & s_1 \arrow[r, "R"] & s_2 \arrow[r, "R"] & \cdots \arrow[r, "R"] & s_j \arrow[r, "R"] & \cdots \arrow[r, "R"] & s_k \arrow[ll, bend right, "R"']
\end{tikzcd}$$

Fig. 1. Cyclic sequence of states.

calculus is finitary, hence, as usual (see, e.g. [7,16,22]), our completeness result is in this sense, weak. Therefore, along this paper, every set of formulas is assumed to be finite.

Given a set $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ we will use $\Phi^{\neg}$ to denote the formula $\neg(\varphi_1 \wedge \cdots \wedge \varphi_n)$ and $\bigwedge \Phi$ denotes $\varphi_1 \wedge \cdots \wedge \varphi_n$. In particular, when Φ is empty, $\Phi^{\neg}$ and $\bigwedge \Phi$ are the constants $\mathbf{F}$ and $\mathbf{T}$, respectively.

Formally, a PLTL-*structure* $\mathcal{M}$ is a pair $(S_{\mathcal{M}}, V_{\mathcal{M}})$ such that $S_{\mathcal{M}}$ is a denumerable sequence of states $s_0, s_1, s_2, \ldots$ and $V_{\mathcal{M}}$ is a map $V_{\mathcal{M}} : S_{\mathcal{M}} \to 2^{\mathsf{Prop}}$. Intuitively, $V_{\mathcal{M}}(s)$ specifies which atomic propositions are (necessarily) true in the state s.

The formal semantics of PLTL-formulas is given by the truth of a formula φ in the state s_j of a PLTL-structure $\mathcal{M}$, which is denoted by $\langle \mathcal{M}, s_j \rangle \models \varphi$, which is inductively defined as follows:

- $\langle \mathcal{M}, s_j \rangle \not\models \mathbf{F}$
- $\langle \mathcal{M}, s_j \rangle \models p$ iff $p \in V_{\mathcal{M}}(s_j)$ for $p \in \mathsf{Prop}$
- $\langle \mathcal{M}, s_j \rangle \models \neg\varphi$ iff $\langle \mathcal{M}, s_j \rangle \not\models \varphi$
- $\langle \mathcal{M}, s_j \rangle \models \varphi \wedge \psi$ iff $\langle \mathcal{M}, s_j \rangle \models \varphi$ and $\langle \mathcal{M}, s_j \rangle \models \psi$
- $\langle \mathcal{M}, s_j \rangle \models \circ\varphi$ iff $\langle \mathcal{M}, s_{j+1} \rangle \models \varphi$
- $\langle \mathcal{M}, s_j \rangle \models \varphi \mathcal{U} \psi$ iff there exists $k \geq j$ such that $\langle \mathcal{M}, s_k \rangle \models \psi$ and for every $j \leq i < k$ it holds $\langle \mathcal{M}, s_i \rangle \models \varphi$.

The extension of the above formal semantics to the defined connectives yields:

- $\langle \mathcal{M}, s_j \rangle \models \mathbf{T}$
- $\langle \mathcal{M}, s_j \rangle \models \varphi \vee \psi$ iff $\langle \mathcal{M}, s_j \rangle \models \varphi$ or $\langle \mathcal{M}, s_j \rangle \models \psi$
- $\langle \mathcal{M}, s_j \rangle \models \varphi \mathcal{R} \psi$ iff for every $k \geq j$ it holds either $\langle \mathcal{M}, s_k \rangle \models \psi$ or $\langle \mathcal{M}, s_i \rangle \models \varphi$ for some $j \leq i < k$
- $\langle \mathcal{M}, s_j \rangle \models \diamond\varphi$ iff $\langle \mathcal{M}, s_k \rangle \models \varphi$ for some $k \geq j$
- $\langle \mathcal{M}, s_j \rangle \models \Box\varphi$ iff $\langle \mathcal{M}, s_k \rangle \models \varphi$ for every $k \geq j$.

The semantics is extended from formulas to sets of formulas in the usual way: $\langle \mathcal{M}, s_j \rangle \models \Phi$ iff $\langle \mathcal{M}, s_j \rangle \models \gamma$ for all $\gamma \in \Phi$. We say that $\mathcal{M}$ is a model of Φ, in symbols $\mathcal{M} \models \Phi$, iff $\langle \mathcal{M}, s_0 \rangle \models \Phi$. A satisfiable set of formulas has at least one model, otherwise it is unsatisfiable. The *logical consequence* relation between a set of formulas Φ and a formula χ, denoted as $\Phi \models \chi$, is defined in the following way:

$$\begin{aligned}\Phi \models \chi \quad & \text{iff for every PLTL-structure } \mathcal{M} \text{ and every } s_j \in S_{\mathcal{M}}: \\ & \qquad \text{if } \langle \mathcal{M}, s_j \rangle \models \Phi \text{ then } \langle \mathcal{M}, s_j \rangle \models \chi\end{aligned}$$

The notion of logical consequence above is usually called *local logical consequence*. There is a weaker notion called *global logical consequence* which demands χ to be true at all states in $\mathcal{M}$ if Φ is true at al states in $\mathcal{M}$. This latter notion is also interesting for many applications.

In order to construct models for satisfiable sets of formulas we use *cyclic* PLTL-structures that we define in terms of paths over cycling sequences.

Any infinite sequence $e_0, e_1, \ldots, e_k, \ldots$ involves an implicit successor relation, namely R, such that $(e_i, e_{i+1}) \in R$ for all $i \in N$. When convenient, we will write nRn' to denote $(n, n') \in R$. A finite sequence gives also a corresponding implicit successor relation with a pair for each element except for the last one. A finite sequence $S = e_0, e_1, \ldots, e_k$ is said to be *cyclic* iff its successor relation extends the implicit R with a pair (e_k, e_j) for some $0 \leq j \leq k$ (see Fig. 1). Then, $e_j, \ldots, e_k$ is called the *loop* of S, e_j is called the *cycling element* of S, and the *path* over S is the infinite sequence

$$\mathsf{path}(S) = e_0, e_1, \ldots, e_{j-1} \cdot \langle e_j, e_{j+1}, \ldots, e_k \rangle^{\omega}$$

where $\_ \cdot \_$ is the infix operator of concatenation of sequences and U^{ω} denotes the infinite sequence that results by concatenation of the sequence U infinitely many times. Naturally, for any non-cyclic finite sequence S we consider that $\mathsf{path}(S) = S$.

A PLTL-structure $\mathcal{M}$ is *cyclic* if its (infinite) sequence of states $S_{\mathcal{M}}$ is a path over a cyclic sequence of states.

4. The tableau method TTM

In this section we present a tableau system, called TTM, for PLTL. In TTM, tableaux are essentially trees but branches can end in a leave that represents a loop into another node in its branch. Our tableaux are one-pass in the sense of [20], that is, they do not require a second pass to check an auxiliary graph of states in order to determine if every eventuality is satisfied. As a consequence, temporal stages are represented inside the branches of the tableaux instead of in an auxiliary graph. The contents of this section are divided into four subsections. In Section 4.1 we introduce concepts related to the tableau structure. In Sections 4.2 and 4.3 we present the rules for constructing tableaux and the notion of tableau itself. Finally, in Section 4.4 we provide some detailed examples of tableaux.

4.1. Pre-tableaux

A tableau $\mathcal{T}_\Phi$ for a finite set of formulas Φ is a tree-like structure where each node n is labelled with a set of formulas $L(n)$. The root is labelled with the set Φ whose satisfiability we wish to check. The children of a node n are obtained by applying one of the rules to one of the formulas in $L(n)$. Nodes are organized in branches, so that the rules serve to either enlarge the branch (with one new child) or split the branch with two new children. In order to formalize the notion of branch we recall the concept of strongly generated set.

Definition 4.1. Let Nodes be a finite non-empty set of nodes, n a node in Nodes and Nodes$^+$ the set of all non-empty sequences of elements in Nodes. A non-empty set $B \subseteq \text{Nodes}^+$ is *strongly generated* with respect to Nodes and n iff it verifies the following conditions:

1. If $n_0, n_1, \ldots, n_k \in B$, then $n_i \neq n_j$ for all $0 \le i < j \le k$
2. If $n_0, n_1, \ldots, n_k \in B$, then $n_0 = n$
3. If $n_0, n_1, \ldots, n_k \in B$, then $n_0, n_1, \ldots, n_i \in B$ for any $0 \le i < k$
4. For every node $m \in \text{Nodes}$ there is a unique sequence $n_0, n_1, \ldots, n_k \in B$ such that $n_k = m$.

We denote by trees(Nodes, n) the collection of all subsets of Nodes$^+$ that are strongly generated with respect to Nodes and n. Let $B \in \text{trees}(\text{Nodes}, n)$, each sequence $b \in B$ is called a branch. A branch $b' = n_0, n_1, \ldots, n_i$ is a *prefix* of another branch $b = n_0, n_1, \ldots, n_k$ if $0 \le i \le k$. If, besides, $i \neq k$, we say that b' is a *proper prefix* of b. A branch $b \in B$ is *maximal* whenever b is not proper prefix of any other branch in B.

Note that, in the above Definition 4.1, condition 1 means that a node cannot appear more than once in a branch, condition 2 means that the first element in every branch is the node n, condition 3 means that a strongly generated set is closed with respect to non-empty prefixes and condition 4 states that every node must belong to at least one branch. Note also that trees(Nodes, n) is finite and every sequence $b \in B$ is finite for any $B \in \text{trees}(\text{Nodes}, n)$.

Now we define the concept of pre-tableau for a set of formulas.

Definition 4.2 (*Pre-tableau*)**.** A *pre-tableau* for a finite set of formulas Φ is a tuple $\mathcal{T}_\Phi = (\text{Nodes}, n_\Phi, L, B, R)$ such that:

1. Nodes is a finite non-empty set of nodes
2. n_Φ is a node in Nodes, called initial node
3. $L : \text{Nodes} \to 2^\Gamma$ is the labelling function where Γ is a set of formulas that contains Φ such that the the initial node is labelled by Φ, that is $L(n_\Phi) = \Phi$
4. B is a strongly generated set in trees(Nodes, n_Φ), called the set of branches
5. R is the successor relation over Nodes. R should be coherent with B in the sense that for all $n, n' \in \text{Nodes}$, $(n, n') \in R$ iff there exists $n_0, n_1, \ldots, n_k \in B$ such that $n = n_i$ and $n' = n_{i+1}$ for some $0 \le i < k$.

As usual, R^+ and R^* respectively denote the transitive closure and the reflexive-transitive closure of any binary relation R.

4.2. Tableau rules

A tableau rule is applied to a set of formulas $L(n)$ labelling a node n (which is the last node of a branch). Each rule application requires a previous selection of a formula from $L(n)$. We call the set $L(n) \setminus \{\varphi\}$, where φ is the selected formula, the *context* and it is denoted by Δ.

Rule	α	$A(\alpha)$
$(\neg\neg)$	$\neg\neg\varphi$	$\{\varphi\}$
$(\wedge)$	$\varphi \wedge \psi$	$\{\varphi, \psi\}$
$(\neg\circ)$	$\neg\circ\varphi$	$\{\circ\neg\varphi\}$

Rule	β	$B_1(\beta)$	$B_2(\beta)$
$(\neg\wedge)$	$\neg(\varphi \wedge \psi)$	$\{\neg\varphi\}$	$\{\neg\psi\}$
$(\neg\mathcal{U})$	$\neg(\varphi\,\mathcal{U}\,\psi)$	$\{\neg\varphi, \neg\psi\}$	$\{\varphi, \neg\psi, \neg\circ(\varphi\,\mathcal{U}\,\psi)\}$
$(\mathcal{U})_1$	$\varphi\,\mathcal{U}\,\psi$	$\{\psi\}$	$\{\varphi, \neg\psi, \circ(\varphi\,\mathcal{U}\,\psi)\}$

Rule	β	$B_1(\beta)$	$B_2(\beta, \Delta)$
$(\mathcal{U})_2$	$\varphi\,\mathcal{U}\,\psi$	$\{\psi\}$	$\{\varphi, \neg\psi, \circ((\varphi \wedge \Delta^\neg)\,\mathcal{U}\,\psi)\}$
			where Δ stands for the context

Fig. 2. Primitive TTM-rules.

Rule	α	$A(\alpha)$
$(\Box)$	$\Box\varphi$	$\{\varphi, \circ\Box\varphi\}$

Rule	β	$B_1(\beta)$	$B_2(\beta)$
$(\mathcal{R})$	$\varphi\,\mathcal{R}\,\psi$	$\{\varphi, \psi\}$	$\{\neg\varphi, \psi, \circ(\varphi\,\mathcal{R}\,\psi)\}$
$(\Diamond)_1$	$\Diamond\varphi$	$\{\varphi\}$	$\{\neg\varphi, \circ\Diamond\varphi\}$

Rule	β	$B_1(\beta)$	$B_2(\beta, \Delta)$
$(\Diamond)_2$	$\Diamond\varphi$	$\{\varphi\}$	$\{\neg\varphi, \circ(\Delta^{\neg}\,\mathcal{U}\,\varphi)\}$
			where Δ stands for the context

Fig. 3. Some derived TTM-rules.

As usual, the TTM-rules are based in a classification of the formulas into conjunctive and disjunctive, which are respectively named as α-formulas and β-formulas. In Fig. 2, any α-formula α is decomposed in a unique set, called $A(\alpha)$, and any β-formula β is decomposed into two constituent sets B_1 and B_2. The set B_1 depends on the considered formula β, whereas B_2 can also depend on the context Δ.[4]

This classification gives raise to the tableau rules whose names are also given in Fig. 2. Every rule, except $(\mathcal{U})_2$, is well known in the literature. It is worth to note that $(\mathcal{U})_1$ and $(\mathcal{U})_2$ affect the same β formula, but not in the same way. The rule $(\mathcal{U})_2$ can be considered quite peculiar, since $B_2(\beta, \Delta)$ includes a formula which depends on the whole set of formulas in the node. Moreover, $(\mathcal{U})_2$ leads to a new tableau construction style that allows us to dispense with the auxiliary graph. This rule is based on the fact that if a formula $\varphi\,\mathcal{U}\,\psi$ is satisfiable in a given context Δ, it is because there exists a model that is minimal in the sense that the sequence of states along which ψ is not true should be an ever-changing sequence. Consequently, no context can be repeated from the state where $\varphi\,\mathcal{U}\,\psi$ is true until the state where ψ is true. This property does not allow to postpone indefinitely the truth of ψ, provided that the number of possible contexts is finite. In the proof of the Lemma 5.1, we show in detail that the rule $(\mathcal{U})_2$ is correct. We believe that this correctness proof reflects the intuition behind the rule $(\mathcal{U})_2$. One may wonder whether the rule $(\mathcal{U})_1$ is essential for completeness. Our completeness proof uses it, but it is an open problem whether there exists an alternative proof disregarding the rule $(\mathcal{U})_1$. However, we conjecture that $(\mathcal{U})_1$ is essential for completeness. Anyway, from a practical point of view it is better that the system includes the rule $(\mathcal{U})_1$, since $(\mathcal{U})_2$ is costly to use.

As well as the above primitive TTM-rules, the method TTM also uses the operator unnext (see Section 3) to convert the labelling set $L(n)$ of a node n into another set $\mathsf{unnext}(L(n))$ that will label a new node and that intuitively represents the jump from one instant to the next one.

From the primitive TTM-rules we can derive rules for the defined connectives like the ones in Fig. 3. There are also dual rules for $\neg\Box$, $\neg\Diamond$ and $\neg\,\mathcal{R}$ that are left to the reader.

Tableaux are constructed with the aim of refuting the initial set of formulas.

Definition 4.3. A node n is consistent iff $\mathbf{F} \notin L(n)$ and there is no φ such that $\{\varphi, \neg\varphi\} \subseteq L(n)$. Otherwise, n is *inconsistent*.

Note that, in Definition 4.3, the formula φ is not required to be an atom. Indeed, by demanding φ to be atomic the completeness of TTM would be lost. For example, the set of formulas $\Psi = \{p\,\mathcal{U}\,q, \neg(p\,\mathcal{U}\,q)\}$ would not be refutable, if the label of an inconsistent node should contain $\mathbf{F}$ or $\{p, \neg p\}$ for some $p \in \mathsf{Prop}$. In fact, using the tableau rules there is no way to achieve such atomic inconsistency. However, Ψ must be inconsistent in order to achieve completeness. It is also worthy to note that a node labelled by $\Psi' = \{p\,\mathcal{U}\,q, (\neg p)\,\mathcal{R}\,(\neg q)\}$ (which is equivalent to Ψ) is not inconsistent (in the sense of Definition 4.3). The node Ψ' can be refuted by our tableau method, but using the (non-atomic) inconsistency of $\{\circ((\neg p)\,\mathcal{R}\,(\neg q)), \neg\circ((\neg p)\,\mathcal{R}\,(\neg q))\}$.

When a branch b contains an inconsistent node we say that b is *closed*. Any closed branch is trivially unsatisfiable. Branches that are not closed are said to be *open*. However, open branches are not necessarily satisfiable. In particular, an open branch could be a prefix of a closed one.

4.3. Semantic tableaux

The tableau rules given in Section 4.2, together with the notion of consistent node, allow us to determine when a pre-tableau is a tableau. Along this subsection $\mathcal{T}_\Phi$ stands for a pre-tableau for Φ given by a tuple $(\mathsf{Nodes}, n_\Phi, L, B, R)$.

Definition 4.4. A pre-tableau $\mathcal{T}_\Phi$ is *coherent* if and only if every node n in a non-maximal branch in B is consistent and exactly one of the following items holds for every $b = n_0, n_1, \ldots, n_i, n_{i+1}, \ldots, n_k \in B$ and every $0 \leq i < k$:
(1) $L(n_{i+1}) = A(\alpha) \cup L(n_i) \setminus \{\alpha\}$ for some $\alpha \in L(n_i)$
(2) There exists exactly one node $n' \in N \setminus \{n_{i+1}\}$ and one branch $b' = n_0, n_1, \ldots, n_i, n' \in B$ such that for some $\beta \in L(n_i)$ either

[4] Remember that Δ is always assumed to be a finite set and that $\Delta^{\neg}$ is $\mathbf{F}$ whenever Δ is empty.

- $L(n_{i+1}) = B_1(\beta) \cup L(n_i) \setminus \{\beta\}$ and $L(n') = C(\beta, L(n_i)) \cup L(n_i) \setminus \{\beta\}$ or
- $L(n_{i+1}) = C(\beta, L(n_i)) \cup L(n_i) \setminus \{\beta\}$ and $L(n') = B_1(\beta) \cup L(n_i) \setminus \{\beta\}$

where $C(\beta, L(n_i))$ is $B_2(\beta)$ or $B_2(\beta, L(n_i) \setminus \{\beta\})$.

(3) $L(n_{i+1}) = \mathsf{unnext}(L(n_i))$.

In (1) and (3), every branch in B with proper prefix $n_0, n_1, \ldots, n_i$ must also have prefix $n_0, n_1, \ldots, n_i, n_{i+1}$, whereas in (2) every branch in B with proper prefix $n_0, n_1, \ldots, n_i$ has also prefix $n_0, n_1, \ldots, n_i, n_{i+1}$ or prefix $n_0, n_1, \ldots, n_i, n'$.

In a coherent pre-tableau branches whose last node is inconsistent do not accept more enlargements or splittings. Every enlargement or splitting of a branch corresponds to the application of a TTM-rule or the unnext operator to its last node. The application of an α-rule enlarges a branch $n_0, \ldots, n_i$ with a new node n_{i+1} that includes, in the label, the constituents of the treated formula α, but not α itself. Whereas the application of a β-rule splits a branch $n_0, \ldots, n_i$ with two new nodes n_{i+1} and n' that respectively include the constituents in $B_1(\beta)$ and $B_2(\beta)$ (alternatively $B_2(\beta, \Delta)$), but not the treated formula β.

In order to ensure when an open branch describes a model, we deal with the notions of stage, cyclic branch, saturated set and fulfilling branch.

In a coherent pre-tableau $\mathcal{T}_\Phi$ there exist only a finite number of different labels. Consequently, any infinite branch must contain infinitely many different nodes with the same label. In particular, when a repetition arises in an open branch $n_0, n_1, \ldots, n_{j-1}, n_j, \ldots, n_k$ (i.e., when $L(n_k) = L(n_{j-1})$ for some $0 < j \le k$), then an infinite branch of the form $n_0, n_1, \ldots, n_{j-1}, n_j, \ldots, n_k, n_j, \ldots, n_k, \ldots$ can be obtained. In fact, this will be a cyclic branch that will be finitely represented.

Definition 4.5. If $b = n_0, n_1, \ldots, n_k$ is an open branch such that $L(n_k) = L(n_{j-1})$ for some $0 < j \le k$, then b is cyclic and we define

$$\mathsf{path}(b) = n_0, n_1, \ldots, n_{j-1} \cdot \langle n_j, n_{j+1}, \ldots, n_k \rangle^\omega$$

In other words, we consider that the implicit successor relation on b is extended with $n_k R n_j$.

Every branch (cyclic or not) of a coherent pre-tableau can be seen as divided into *stages* according to the applications of the operator unnext. In other words, a stage is a sequence of consecutive nodes between two consecutive applications of the unnext operator.

Definition 4.6. Given a branch b, every maximal subsequence $n_i, n_{i+1}, \ldots, n_j$ of $\mathsf{path}(b)$ such that $L(n_\ell) \neq \mathsf{unnext} L(n_{\ell-1})$ for every $i < \ell \le j$, is called a *stage*. We denote by $\mathsf{stages}(b)$ the *sequence of all stages* of a branch b. The successor relation on $\mathsf{stages}(b)$ is induced by the successor relation on $\mathsf{path}(b)$. That is, if s and s' are respectively stages $n_0, \ldots n_i$ and $n'_0, \ldots, n'_j$ in $\mathsf{path}(b)$ then sRs' whenever $n_i R n'_0$. Hence, if b is a cyclic sequence of nodes, then $\mathsf{stages}(b)$ is a cyclic sequence of stages.

Example 4.7. Consider a cyclic branch $b = n_1, n_2, n_3, n_4, n_5$ such that $L(n_5) = L(n_3)$. Then, $\mathsf{path}(b) = n_1, n_2, n_3 \cdot \langle n_4, n_5 \rangle^\omega$. Let us suppose that $L(n_2) = \mathsf{unnext}(L(n_1))$ and $L(n_5) = \mathsf{unnext}(L(n_4))$. Then, $\mathsf{stages}(b)$ is formed by three stages: $s_1 = \langle n_1 \rangle$, $s_2 = \langle n_2, n_3, n_4 \rangle$ and $s_3 = \langle n_5, n_4 \rangle$. Therefore, the induced relation R on $\mathsf{stages}(b)$ is given by $s_1 R s_2$, $s_2 R s_3$ and $s_3 R s_3$. Hence, $\mathsf{path}(\mathsf{stages}(b)) = s_1, s_2 \cdot \langle s_3 \rangle^\omega$.

With a slight abuse of notation, the labelling function L is extended from nodes to stages in the natural way. That is, for any stage s:

$$L(s) = \bigcup_{n \in s} L(n)$$

Definition 4.8. Let S be a sequence of stages, $s \in S$ and $\varphi \,\mathcal{U}\, \psi \in L(s)$, we say that $\varphi \,\mathcal{U}\, \psi$ is fulfilled in S iff there exists s' such that sR^*s' and $\psi \in L(s')$. A sequence S of stages is *fulfilling* iff for all $s \in S$ every $\varphi \,\mathcal{U}\, \psi \in L(s)$ is fulfilled in S. A branch b is fulfilling iff the sequence $\mathsf{path}(\mathsf{stages}(b))$ is fulfilling.

The concept of fulfilling branch together with the following concept of $\alpha\beta$-saturated stage is crucial for determining when branches are able to describe a model.

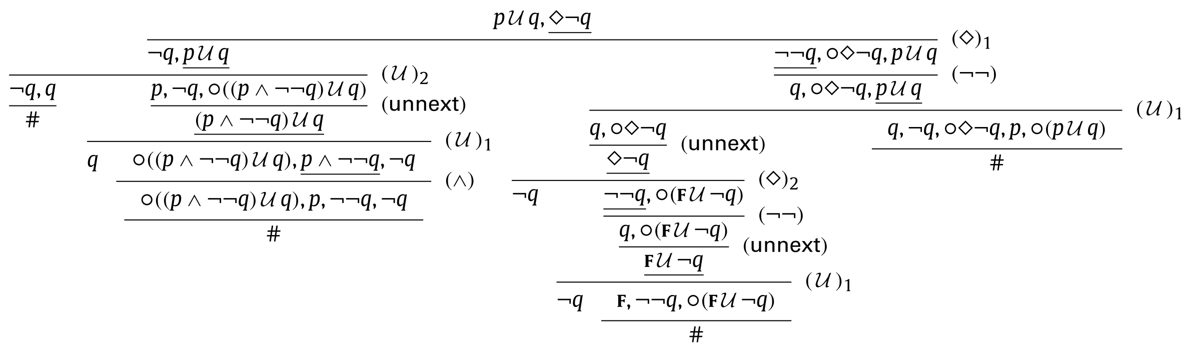
Definition 4.9. A stage s is *$\alpha\beta$-saturated* if and only if for every $\varphi \in L(s)$:

1. If φ is an α-formula then $A(\varphi) \subseteq L(s)$
2. If φ is a β-formula then $B_1(\varphi) \subseteq L(s)$ or $B_2(\varphi) \subseteq L(s)$ or $B_2(\varphi, \Delta) \subseteq L(s)$, where $\Delta = L(n_i) \setminus \{\varphi\}$ for some $n_i \in s$ such that $\varphi = \delta \,\mathcal{U}\, \gamma \in L(n_i)$.

Now, we give a sufficient condition to consider that a branch is (sufficiently) expanded. That is, it is able to describe a collection of models. This condition can be syntactically checked. For the construction of systematic tableaux (see Section 5.2) we will refine this syntactic condition to a simpler one.

Definition 4.10. An open branch b is *expanded* if and only if b is fulfilling and each stage $s \in \mathsf{stages}(b)$ is $\alpha\beta$-saturated.

For example, an expanded branch of a coherent pre-tableau for $\{r \,\mathcal{U}\, p\}$ can be formed by a unique stage s_0 such that $L(s_0) = \{r \,\mathcal{U}\, p, p\}$. Actually that branch is fulfilling and $\alpha\beta$-saturated, hence it is expanded. But also the sequence of stages



$p\mathcal{U}q, \Diamond\neg q$
$(\Diamond)_1$
$\neg q, p\mathcal{U}q$
$\neg\neg q, \circ\Diamond\neg q, p\mathcal{U}q$
$(\neg\neg)$
$q, \circ\Diamond\neg q, p\mathcal{U}q$
$(\mathcal{U})_2$
$\neg q, q$
$\#$
$p, \neg q, \circ((p \wedge \neg\neg q)\mathcal{U}q)$
(unnext)
$(p \wedge \neg\neg q)\mathcal{U}q$
$(\mathcal{U})_1$
q
$\circ((p \wedge \neg\neg q)\mathcal{U}q), p \wedge \neg\neg q, \neg q$
$(\wedge)$
$\circ((p \wedge \neg\neg q)\mathcal{U}q), p, \neg\neg q, \neg q$
$\#$
$(\mathcal{U})_1$
$q, \circ\Diamond\neg q$
(unnext)
$\Diamond\neg q$
$q, \neg q, \circ\Diamond\neg q, p, \circ(p\mathcal{U}q)$
$\#$
$(\Diamond)_2$
$\neg q$
$\neg\neg q, \circ(\mathbf{F}\mathcal{U}\neg q)$
$(\neg\neg)$
$q, \circ(\mathbf{F}\mathcal{U}\neg q)$
(unnext)
$\mathbf{F}\mathcal{U}\neg q$
$(\mathcal{U})_1$
$\neg q$
$\mathbf{F}, \neg\neg q, \circ(\mathbf{F}\mathcal{U}\neg q)$
$\#$

Fig. 4. An example of open tableau.

s_0, s_1, s_2 where $L(s_0)$ is as above and $L(s_1) = L(s_2) = \{\,\}$ is an expanded branch satisfying coherence, since $\text{unnext}(\{r\mathcal{U}p, p\}) = \{\,\}$ and $\text{unnext}(\{\,\}) = \{\,\}$. Unlike the former, the latter branch is cyclic and the loop consists of the last stage. Actually, the path on stages is $s_0, s_1 \cdot \langle s_2\rangle^\omega$. It is easy to see that, in general, any non-cyclic expanded branch b such that $\text{unnext}(L(n)) = \{\,\}$ where n is the last node of b, can be made cyclic by extending it with two empty nodes (each one is an stage). This idea is used in Section 5.2 for the systematic construction of tableaux. We intentionally add two empty sets instead of one because this repetition is what we will use (in the systematic tableau) for detecting the loop.

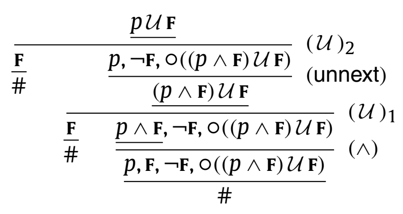
When constructing a tableau, only the non-expanded open branches are enlarged. When all the maximal branches are closed or expanded, the pre-tableau cannot be further expanded. Moreover, a completely expanded tableau is constructed for deciding if the original set of formulas is satisfiable or not, respectively depending on whether there is at least one expanded open branch or all its branches are closed.

Definition 4.11 (*Tableau*)**.** A *tableau* for a set of formulas Φ is a coherent pre-tableau for Φ such that every expanded or closed branch is maximal. An *expanded tableau* is a tableau where every maximal branch is either expanded or closed. An expanded tableau is open if it has at least one open maximal branch,[5] otherwise it is closed.

4.4. Examples of tableaux

Now, we give some examples of expanded tableaux. For readability, we underline the formula which the TTM-rule is applied to. Here and in the following, branches with the mark # are closed branches. Note that, when a formula is treated at one node, this formula does not appear in any successor of the node, although it remains belonging to the whole stage. Hence, already treated formulas cannot be expanded again (at the same stage).

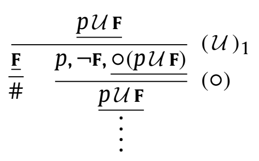
Example 4.12. The following is a closed expanded tableau for the set of formulas $\{p\mathcal{U}\mathbf{F}\}$:



$p\mathcal{U}\mathbf{F}$
$(\mathcal{U})_2$
$\mathbf{F}$
$\#$
$p, \neg\mathbf{F}, \circ((p \wedge \mathbf{F})\mathcal{U}\mathbf{F})$
(unnext)
$(p \wedge \mathbf{F})\mathcal{U}\mathbf{F}$
$(\mathcal{U})_1$
$\mathbf{F}$
$\#$
$p \wedge \mathbf{F}, \neg\mathbf{F}, \circ((p \wedge \mathbf{F})\mathcal{U}\mathbf{F})$
$(\wedge)$
$p, \mathbf{F}, \neg\mathbf{F}, \circ((p \wedge \mathbf{F})\mathcal{U}\mathbf{F})$
$\#$

Note that the rightmost branch consists of two stages, the first one is formed by the two higher nodes. The remaining three nodes form the second stage of the branch.

It is worth to note that using only the rule $(\mathcal{U})_1$ the fulfillment of an eventuality can be indefinitely delayed. It is easy to realize that the above set of formulas $\{p\mathcal{U}\mathbf{F}\}$ cannot be TTM-refuted without using the rule $(\mathcal{U})_2$:



[5] Which is expanded.

Example 4.13. The following is an open tableau for $\{p, \circ\neg p, \neg\mathbf{F}\,\mathcal{U}\,\neg p\}$:

$$\dfrac{p, \circ\neg p, \underline{\neg\mathbf{F}\,\mathcal{U}\,\neg p}}{\dfrac{p,\circ\neg p,\neg p}{\#} \qquad \dfrac{p,\circ\neg p,\neg\neg p,\neg\mathbf{F},\circ(\neg\mathbf{F}\,\mathcal{U}\,\neg p)}{\dfrac{\neg p, \underline{\neg\mathbf{F}\,\mathcal{U}\,\neg p}}{\neg p \qquad \dfrac{\neg p,\neg\mathbf{F},\neg\neg p,\circ(\neg\mathbf{F}\,\mathcal{U}\,p)}{\#}}\;(\mathcal{U})_1}\;(\text{unnext})}\;(\mathcal{U})_1$$

The tableau has two closed branches and one open branch, which is the central one. This open branch describes a collection of models. The first state of those models should satisfy the formulas labelling the first stage of the branch which is formed by the first two nodes. In particular, p should be satisfied at the first state. The second stage is given by the third and fourth nodes of the branch, in particular $\neg p$ should be satisfied in the second state of the models. In fact, any sequence of states prefixed by these two states is a model of the root of the tableau.

Example 4.14. An example of open tableau for the set of formulas $\{p\,\mathcal{U}\,q, \diamond\neg q\}$ can be found in Fig. 4. Note that it makes use of the derived rules $(\diamond)_1$ and $(\diamond)_2$ that are shown in Fig. 3. This tableau has three open branches describing three different collections of models. The leftmost open branch represents the class of models with a first state where p and $\neg q$ are true and a second state where q is true. In the first state of the models represented by the central open branch q is true, whereas in the second one $\neg q$ holds. Finally, the rightmost open branch gives three states which respectively make true the literals q, q and $\neg q$.

5. Soundness and completeness of TTM

A tableau method is *sound* if, whenever a closed tableau exists for Φ, then Φ is unsatisfiable. And a tableau method is *complete* if, whenever Φ is unsatisfiable, a closed tableau for Φ can be constructed. Therefore, a sound and complete tableau method is suitable for deciding in a finite amount of time whether a set of formulas is unsatisfiable. However, the above concept of completeness does not guarantee that the satisfiability of a set of formulas is decidable. For that reason, the above notion of completeness is often called *refutational completeness*, whereas completeness stands for the case when both satisfiability and unsatisfiability are decidable.

In this section, we prove that the tableau system TTM is sound, refutationally complete and also complete. The first subsection is devoted to soundness. In Section 5.2 we introduce the construction of systematic tableaux together with the concepts and results that the algorithm and its correctness give rise to. In particular, we discuss about the analytic superformula property and present our notion of closure. In Section 5.3 we give some examples of systematic tableaux. In Section 5.4 we prove the completeness of TTM, by proving, as a first step, its refutational completeness. In Section 5.5 we provide a practical improvement of the rule $(\mathcal{U})_2$.

5.1. Soundness

First, we show that the TTM-rules preserve equi-satisfiability and that the unnext operator preserves satisfiability. Then, soundness is proved in Theorem 5.2.

Lemma 5.1. *For every set of formulas Φ, any α-formula φ and any β-formula ψ :*

1. *$\Phi \cup \{\varphi\}$ is satisfiable iff $\Phi \cup A(\varphi)$ is satisfiable*
2. *$\Phi \cup \{\psi\}$ is satisfiable iff $\Phi \cup B_1(\psi)$ or $\Phi \cup B_2(\psi)$ or $\Phi \cup B_2(\psi, \Phi)$ is satisfiable.*
3. *If Φ is satisfiable then* unnext(Φ) *is satisfiable.*

Proof. Every right-to-left implication is trivial. For the left-to-right implications, the only difficult case is the rule $(\mathcal{U})_2$. We will show that, if we assume that $\Phi \cup \{\varphi\,\mathcal{U}\,\psi\}$ is satisfiable, then we would build a model for at least one of the two sets: $\Phi \cup \{\psi\}$ and $\Phi \cup \{\varphi, \neg\psi, \circ((\varphi \wedge \Phi^{\neg})\,\mathcal{U}\,\psi)\}$. Let $\langle\mathcal{M}, s_i\rangle \models \Phi \cup \{\varphi\,\mathcal{U}\,\psi\}$ and z the least $j \geq i$ such that $\langle\mathcal{M}, s_j\rangle \models \psi$. If $z = i$ then $\langle\mathcal{M}, s_i\rangle$ yields a model of $\Phi \cup \{\psi\}$. Otherwise, if $z > i$, let y be the greatest j such that $i \leq j < z$ and $\langle\mathcal{M}, s_j\rangle \models \Phi \cup \{\varphi\,\mathcal{U}\,\psi\}$. As a consequence of the choice of z and y, it holds that $\langle\mathcal{M}, s_y\rangle \models \circ((\varphi \wedge \Phi^{\neg})\,\mathcal{U}\,\psi)$. Then, $\langle\mathcal{M}, s_y\rangle$ yields a model of $\Phi \cup \{\varphi, \neg\psi, \circ((\varphi \wedge \Phi^{\neg})\,\mathcal{U}\,\psi)\}$. □

Hence, soundness can be proved.

Theorem 5.2. *If there exists a closed expanded tableau for Φ then Φ is unsatisfiable.*

Proof. Let $\mathcal{T}_\Phi$ be a closed expanded tableau for Φ. The set of formulas labelling each leaf is inconsistent and therefore unsatisfiable. Then, by the Lemma 5.1, each node in $\mathcal{T}_\Phi$ is labelled with an unsatisfiable set of formulas, in particular the root. Therefore Φ is unsatisfiable. □

5.2. Systematic tableaux

In this subsection we provide an algorithm that, given a set of formulas Φ, constructs an expanded tableau for Φ that we will denote by $\mathcal{T}_\Phi$. We also study the main properties that our systematic tableau satisfies.

The construction of $\mathcal{T}_\Phi$ consists in a systematic extension of branches using the TTM-rules for decomposing the α- and β-formulas into its constituents. The application of a β-rule splits the extended branch into two. When no rule can be applied, the operator unnext is used to jump to a new stage.

Classical (propositional) tableaux satisfy the *subformula property* (SP):

> For every formula ψ used in the construction of any tableau for Φ, there exists some formula $\gamma \in \Phi$ such that ψ is a (possibly negated) subformula of γ.

This property ensures the termination of the construction of any tableau for a (finite) set of formulas. Most tableau systems for modal and temporal logics, fail to satisfy the SP, since some of their rules introduce formulas that are not subformulas of the principal formula of the rule. Hence, termination of modal/temporal tableaux is not obvious. However, most tableau systems for modal and temporal logics, satisfy the *analytic superformula property* (ASP):

> For every finite set of formulas Φ, there exists a finite set that contains all the formulas that may occur in any tableau for Φ.

Such set is usually called the closure of Φ. The ASP also ensures the non-existence of infinite branches where all the nodes have different labels. Hence, by controlling loops, the finiteness of proof search can be ensured. In our case, as a consequence of the rule $(\mathcal{U})_2$, the tableau system TTM fails to satisfy the ASP. However, TTM satisfies a slightly weaker variant that is enough for ensuring completeness and that we call the *weak analytic superformula property* (WASP):

> For every finite set of formulas Φ, there exists a finite set $\mathsf{clo}(\Phi)$ (closure of Φ) that contains all the formulas that may occur in any *systematic tableau* for Φ.

Hence we give an algorithm that constructs, for any Φ, a systematic tableau $\mathcal{T}_\Phi$ such that TTM satisfies the WASP (see Fig. 5). This is achieved by keeping at most one distinguished formula to which the rule $(\mathcal{U})_2$ can be applied. In this way, the notion of closure –that we define below– captures the superformulas produced by the rule $(\mathcal{U})_2$. For handling distinguished formulas the algorithm uses a function *d*. Along the construction of the systematic tableau, the function *d* associates to every node *n* one of the following three possible sets of formulas:

1. the empty set
2. a non-elementary singleton of the form $\{\varphi \mathcal{U} \psi\}$
3. an elementary singleton of the form $\{\circ(\varphi \mathcal{U} \psi)\}$.

The case 1 means that no until formula is distinguished. In 2, *d* yields the set containing the distinguished formula to which $(\mathcal{U})_2$ will be applied. The case 3 results after the application of $(\mathcal{U})_2$ to the distinguished formula. At the begining, *d* associates the empty set to the initial node.

Our algorithm for constructing $\mathcal{T}_\Phi$ nondeterministically selects, at each step, a maximal branch to be extended. Actually, the algorithm ends when every maximal branch is either closed or expanded, so that there is no branch that can be extended. Maximal branches that achieve one of these two status are consequently marked. The procedure *unmarked_branches* yields the branches that can be further extended. For extending the selected branch, the algorithm uses three procedures. First, a procedure *non-dist_expand* that applies the corresponding TTM-rule, excepting $(\mathcal{U})_2$, to a formula that has been nondeterministically selected from the set of non-distinguished formulas in the last node of the branch. Second, when the TTM-rules other than $(\mathcal{U})_2$ cannot be further applied, the procedure *dist_expand* applies the rule $(\mathcal{U})_2$ to the until formula that is *distinguished* by the function *d*, if there is some. The procedure *fairly_dist* updates the function *d* using a fair strategy. Third,

```
Input: A finite set of formulas $\Phi$
Output: An expanded tableau $\mathcal{T}_\Phi = (\mathsf{Nodes}, n_\Phi, L, B, R)$ for $\Phi$

Nodes := $\{n_\Phi\}$; $L := \{(n_\Phi, \Phi)\}$; $B := \{n_\Phi\}$; $R := \{\,\}$; $d := \{(n_\Phi, \{\,\})\}$
while $unmarked\_branches(B) \neq \{\,\}$ loop
    choose $b \in unmarked\_branches(B)$
    $n_k := last\_node(b)$;
    if $d(n_k) = \{\,\}$ then $fairly\_dist(\mathcal{T}_\Phi, d)$ end if
    if $L(n_k) \setminus d(n_k)$ is not elementary
      then choose $\gamma \in L(n_k) \setminus d(n_k)$
            $non\text{-}dist\_expand(\gamma, \mathcal{T}_\Phi, d)$
      elsif $d(n_k)$ is neither empty nor elementary then $dist\_expand(\mathcal{T}_\Phi, d)$
      else $\{L(n_k)$ is elementary$\}$ $unnext\_expand(\mathcal{T}_\Phi, d)$
    end if
end loop
```

Fig. 5. Systematic tableau algorithm.

when the node is labelled by an elementary set, then the operator unnext is applied using the procedure *unnext_expand*. Let us give a more detailed explanation of all the procedures used by the algorithm.

last_node(b) gives the last node added to a given branch b.

non-dist_expand$(\gamma, \mathcal{T}_\Phi, d)$ applies to the branch b the α- or β-rule (excepting $(\mathcal{U})_2$) that corresponds to the formula γ. In both cases, the node distinguished by the function d is preserved. That is, for $n_k = \mathit{last\_node}(b)$:

- If γ is an α-formula, create a new node n and a new branch $b' = b \cdot n$ according to the corresponding α-rule such that $L(n) = (L(n_k) \setminus \{\gamma\}) \cup A(\gamma)$ and extend d and R to be $d(n) = d(n_k)$ and $n_k R n$.
- If γ is a (non-distinguished) β-formula, create two new nodes n' and n'' and two new branches $b' = b \cdot n'$ and $b'' = b \cdot n''$ according to the corresponding β-rule such that $L(n') = (L(n_k) \setminus \{\gamma\}) \cup B_1(\gamma)$ and $L(n'') = (L(n_k) \setminus \{\gamma\}) \cup B_2(\gamma)$. Extend d and R to be $d(n') = d(n_k), d(n'') = d(n_k)$ and $n_k R n', n_k R n''$.

dist_expand$(\mathcal{T}_\Phi, d)$ applies the rule $(\mathcal{U})_2$ to an until formula $\varphi \mathcal{U} \psi$ that is distinguished by the function d. The function d yields empty for the new node that contains ψ since the until formula has been fulfilled. In the other branch, the new distinguished formula is $\circ((\varphi \wedge \Delta^{\neg}) \mathcal{U} \psi)$. That is, for $n_k = \mathit{last\_node}(b)$:

Let $d(n_k) = \{\varphi \mathcal{U} \psi\}$. Create two new nodes n' and n'' and two new branches $b' = b \cdot n'$ and $b'' = b \cdot n''$ such that $L(n') = (L(n_k) \setminus \{\varphi \mathcal{U} \psi\}) \cup \{\psi\}$ and $L(n'') = (L(n_k) \setminus \{\varphi \mathcal{U} \psi\}) \cup \{\varphi, \neg\psi, \circ((\varphi \wedge \Delta^{\neg}) \mathcal{U} \psi)\}$ where $\Delta = L(n_k) \setminus \{\varphi \mathcal{U} \psi\}$. Extend d and R to be $d(n') = \{\,\}, d(n'') = \{\circ((\varphi \wedge \Delta^{\neg}) \mathcal{U} \psi)\}$ and $n_k R n', n_k R n''$.

unnext_expand$(\mathcal{T}_\Phi, d)$ creates a new node n and a new branch $b' = b \cdot n$ such that $L(n) = \mathsf{unnext}(L(n_k))$ and extends d and R to be $d(n) = \mathsf{unnext}(d(n_k))$ and $n_k R n$ where $n_k = \mathit{last\_node}(b)$.

unmarked_branches(B) returns the set of unmarked maximal branches in a given set of branches B.

fairly_dist$(\mathcal{T}_\Phi, d)$ distinguishes an until formula, if there is some. That is, for $n_k = \mathit{last\_node}(b)$, whenever $d(n_k) = \{\,\}$ and $L(n_k)$ contains at least one until formula, it updates $d(n_k)$ with a singleton $\{\varphi \mathcal{U} \psi\}$ such that $\varphi \mathcal{U} \psi \in L(n_k)$. Otherwise, $d(n_k)$ remains the empty set. If the node contains more than one until formula, the selection performed by *fairly_dist* on $L(n_k)$ should be *fair*, in the sense that no until formula could remain non-distinguished indefinitely.

The systematic tableau algorithm is depicted by a while-program in Fig. 5. The systematic tableau construction provides a proof search procedure for automated deduction.

Let us give some useful results about the systematic tableau $\mathcal{T}_\Phi$ that this algorithm constructs for any set of formulas Φ.

Proposition 5.3. *If $\{\varphi, \neg\varphi\} \subseteq L(s)$ for some stage s in a branch b of $\mathcal{T}_\Phi$, then every maximal branch of $\mathcal{T}_\Phi$ prefixed by b is closed.*

Proof. By structural induction on φ. It is easy to see that the application of TTM-rules to two complementary formulas that belong to the same stage, but not necessarily to the same node, should generate complementary constituents until they occur in the same node or, at most, they become elementary. □

In the next proposition we show that non-satisfied undistinguished eventualities are kept in branches at least until they are fulfilled or they become distinguished.

Proposition 5.4. *Let b be a branch*[6] *of $\mathcal{T}_\Phi$, and $s_0, s_1, s_2, \dots, s_k$ be any initial subsequence of $\mathsf{path}(\mathsf{stages}(b))$. If $\varphi \mathcal{U} \psi \in L(s_i)$ for some $0 \le i \le k$, $\varphi \mathcal{U} \psi$ is not distinguished in $s_i, \dots, s_k$ and $\psi \notin L(s_i) \cup \dots \cup L(s_k)$, then $\{\varphi, \neg\psi, \circ(\varphi \mathcal{U} \psi)\} \subseteq L(s_j)$ for all $i \le j \le k$.*

Proof. By the construction of $\mathcal{T}_\Phi$, since non-distinguished eventualities are handled by procedure *non-dist_expand* using the rule $(\mathcal{U})_1$. □

Next, we give a more detailed description of the syntactic form of the formulas appearing in sequences of stages where a distinguished eventuality remains unfulfilled. Under that proviso, at each stage, there is exactly one distinguished eventuality and exactly one node to which the procedure *dist_expand* is applied. We also call this node the *distinguished node* of that stage. That is a crucial fact for defining the notion of closure with respect to which TTM satisfies the WASP. We first define some auxiliary sets of sub- and super-formulas of a given set of formulas Φ. Let $\mathsf{sf}(\Phi)$ denote the set of all the subformulas of the formulas in Φ and their negations. Then, the preclosure of Φ, $\mathsf{preclo}(\Phi)$, is the set of formulas that extends $\mathsf{sf}(\Phi)$ with all the superformulas that are generated from $\mathsf{sf}(\Phi)$ by means of all the TTM-rules with the exception of the rule $(\mathcal{U})_2$. That is

$$\begin{aligned}\mathsf{preclo}(\Phi) = \mathsf{sf}(\Phi) \quad &\cup \quad \{\circ(\varphi \mathcal{U} \psi), \neg\circ(\varphi \mathcal{U} \psi), \circ\neg(\varphi \mathcal{U} \psi) \mid \varphi \mathcal{U} \psi \in \mathsf{sf}(\Phi)\} \\ &\cup \quad \{\circ\neg\varphi \mid \neg\circ\varphi \in \mathsf{sf}(\Phi)\}\end{aligned}$$

Note that $\mathsf{preclo}(\Phi)$ cannot be used as closure only because it does not capture the superformulas generated by the application of the rule $(\mathcal{U})_2$. In order to capture these superformulas, we define the following set of conjunctions of negated contexts:

$$\mathsf{conj}(\Phi) = \left\{ \bigwedge \Gamma \mid \Gamma \subseteq \{\varphi \mid \varphi \mathcal{U} \psi \in \mathsf{sf}(\Phi)\} \cup \mathsf{negctx}(\Phi) \right\}$$
$$\text{where } \mathsf{negctx}(\Phi) = \{\Delta^{\neg} \mid \Delta \subseteq \mathsf{preclo}(\Phi)\}$$

[6] The branch b could be cyclic or not, so that $\mathsf{path}(\mathsf{stages}(b))$ could respectively be infinite or finite.

That is, $\mathsf{negctx}(\Phi)$ is the set of all possible *negated contexts* and $\mathsf{conj}(\Phi)$ is formed by all the possible conjunctions of formulas in $\mathsf{negctx}(\Phi)$ and the left-hand side subformulas of all the until formulas in $\mathsf{sf}(\Phi)$. In particular, $\mathbf{F} \in \mathsf{negctx}(\Phi)$ and $\mathbf{F}, \neg\mathbf{F} \in \mathsf{conj}(\Phi)$, since $\mathbf{F}$ and $\neg\mathbf{F}$ are respectively the disjunction and the conjunction of the empty set of formulas. Note also that, by definition, in the conjunctions of $\mathsf{conj}(\Phi)$ every element of $\mathsf{negctx}(\Phi)$ occurs at most once.

Proposition 5.5. *Let b be a branch*[6] *of $\mathcal{T}_\Phi$, let $s_0, s_1, s_2, \dots, s_k$ be any initial subsequence of $\mathsf{path}(\mathsf{stages}(b))$ and $\varphi\,\mathcal{U}\,\psi \in \mathsf{sf}(\Phi)$ such that i is the least natural number such that $d(n) = \{\varphi\,\mathcal{U}\,\psi\}$ for some $n \in s_i$. Then, if $\psi \notin L(s_i) \cup \dots \cup L(s_k)$ then for all $0 \le \ell \le k - i$:*

$$\{\delta_\ell, \neg\psi, \circ(\delta_{\ell+1}\,\mathcal{U}\,\psi)\} \subseteq L(s_{i+\ell})$$

where $\delta_0 = \varphi$ and $\delta_{\ell+1} = \delta_\ell \wedge \chi$ for some $\chi \in \mathsf{negctx}(\Phi)$. Moreover, if $\delta_\ell = \bigwedge \Gamma$ for some Γ such that $\chi \in \Gamma$ then every maximal branch of $\mathcal{T}_\Phi$ prefixed by $s_0, \dots, s_{i+\ell}$ is closed.

Proof. On one hand, by construction of $\mathcal{T}_\Phi$ and induction on ℓ, the procedure *dist_expand* yields two branches such that each branch either contains $\{\delta_\ell, \neg\psi, \circ(\delta_{\ell+1}\,\mathcal{U}\,\psi)\}$ or contains ψ. Note that if $d(n) = \{\circ(\delta_{\ell+1}\,\mathcal{U}\,\psi)\}$ for some $n \in s_{i+\ell}$, then $d(n') = \{\delta_{\ell+1}\,\mathcal{U}\,\psi\}$ for the first node $n' \in s_{i+\ell+1}$. Therefore, $\delta_0 = \varphi$ and for all $j > 0$: $\delta_j = \delta_{j-1} \wedge \Delta^{\neg}_{j-1}$ where $\Delta^{\neg}_{j-1} \in \mathsf{negctx}(\Phi)$ and Δ_{j-1} is the context $L(n) \setminus d(n)$ of the distinguished node n of the stage s_{i+j-1}. Hence, by induction on ℓ, $\delta_\ell \in \mathsf{conj}(\Phi)$ holds for all $0 \le \ell \le k - i$.

On the other hand, since χ is the negation of the context of the distinguished node $n \in s_{i+\ell}$, if $\delta_{\ell+1} = \delta_\ell \wedge \chi$ and $\delta_\ell = \bigwedge \Gamma$ for some Γ such that $\chi \in \Gamma$, then every branch prefixed by $s_0, \dots, s_{i+\ell}$ contains at the same stage (possibly at different nodes) $\{\gamma, \neg\gamma\}$ for some formula γ. Hence, by Proposition 5.3, every maximal branch prefixed by $s_0, \dots, s_{i+\ell}$ is closed. □

Corollary 5.6. *Every distinguished eventuality in a cyclic branch of $\mathcal{T}_\Phi$ is fulfilled.*

Proof. By Proposition 5.5 since, whenever there is an unfulfilled distinguished eventuality in a branch, the presence of the formulas δ_ℓ makes impossible the existence of a loop. □

It is trivial, by construction, that every stage in an open branch of $\mathcal{T}_\Phi$ is $\alpha\beta$-saturated. Hence, by Proposition 5.4 and Corollary 5.6, we can refine the sufficient conditions for being an expanded branch of $\mathcal{T}_\Phi$ as follows:

Proposition 5.7. *Let b be an open branch of $\mathcal{T}_\Phi$, if b satisfies the following two conditions:*
(**i**) *b is cyclic*
(**ii**) *for every eventuality $\gamma \in \mathsf{preclo}(\Phi)$ such that $\gamma \in L(n)$ for some $n \in b$, there exists some $n' \in b$ such that $d(n') = \{\gamma\}$*
then b is an expanded branch.

Proof. By Proposition 5.4, non-distinguished unfulfilled eventualities are preserved from one stage to its successor. In addition, by Corollary 5.6, every distinguished eventuality in a cyclic branch is fulfilled. Hence, by condition (ii), every eventuality from $\mathsf{preclo}(\Phi)$ that occurs in b should be distinguished once and, hence, should be fulfilled. □

Consequently, we use Proposition 5.7 to refine the implementation of the procedure *unmarked_branches*

Remark 5.8. Whenever a branch b satisfies the conditions (i) and (ii) of Proposition 5.7, the procedure *unmarked_branches* considers b to be marked as expanded.

As a consequence, every expanded branch of $\mathcal{T}_\Phi$ is cyclic by construction.

Hence, by Corollary 5.6 and Remark 5.8, TTM satisfies the WASP with respect to the following notion of closure:

$$\begin{aligned}\mathsf{clo}(\Phi) = \;& \mathsf{preclo}(\Phi) \cup \mathsf{conj}(\Phi)\\ & \cup \{(\gamma_1 \wedge \gamma_2)\,\mathcal{U}\,\psi, \circ((\gamma_1 \wedge \gamma_2)\,\mathcal{U}\,\psi) \mid \varphi\,\mathcal{U}\,\psi \in \mathsf{sf}(\Phi) \text{ and } \gamma_1, \gamma_2 \in \mathsf{conj}(\Phi)\}\end{aligned}$$

Note that, γ_1 and γ_2 are enough to represent the unique possible repetition of a negated context. In other words, $L(n) \subseteq \mathsf{clo}(\Phi)$ holds for all node n in $\mathcal{T}_\Phi$, by Corollary 5.6 and Remark 5.8. In addition, the closure set of a finite set of formulas is finite.

Proposition 5.9. *If Φ is a finite set of formulas, then $\mathsf{clo}(\Phi)$ is also finite.*

Proof. It is easy to see that, if $|\mathsf{preclo}(\Phi)| = n$ then $|\mathsf{negctx}(\Phi)| \in O(2^n)$. As a consequence $|\mathsf{conj}(\Phi)|, |\mathsf{clo}(\Phi)| \in O(2^{O(2^n)})$. □

The above results jointly with the fairness of *fairly_dist*, allow us to ensure that the algorithm in Fig. 5 finitely computes an expanded tableau $\mathcal{T}_\Phi$ for any input Φ.

Lemma 5.10. *The algorithm in Fig. 5, for any input Φ, stops leaving in $\mathcal{T}_\Phi$ an expanded tableau.*

Proof. By König's lemma, the only possibility for infinite iteration would be the infinite expansion of (at least) one branch, namely b. By Propositions 5.5, 5.7 and 5.9, the branch b should contain an eventuality that is never distinguished, which contradicts the fairness of the *fairly_dist* procedure. □

Note that the use of a fair strategy for distinguishing the eventualities in each branch of the tableau is essential for proving that the algorithm in Fig. 5 finishes.

We would like to remark that previous tableau methods for PLTL, excepting the one-pass proposal of [21], for obtaining a model of a satisfiable set of formulas (when deciding satisfiability) should generate the whole graph of possible states and all the successive tableaux required for constructing this graph. However, we can use a depth-first strategy and, as soon as a branch is marked expanded, the algorithm could stop providing a model for the original set of formulas.

5.3. Examples of systematic tableaux

In this subsection, we give the systematic expanded tableaux that correspond to the two examples in Section 4.4. For readability, the distinguished formulas are in black boxes. Besides, since open expanded branches are cyclic, we have marked the internal repeated node with a symbol $\lightning_{i_1,\ldots,i_n}$ where $i_1,\ldots,i_n$ denote the number of all maximal branches (from left to right in the whole tableau) whose last node coincides with the marked node.

Example 5.11. The following is the systematic tableau for $\{p\,\mathcal{U}\,\mathbf{F}\}$, which is closed.

$$\dfrac{\boxed{p\,\mathcal{U}\,\mathbf{F}}}{\begin{array}{cc} \dfrac{\mathbf{F}}{\#} & \dfrac{p,\neg\mathbf{F},\boxed{\circ((p\wedge\mathbf{F})\,\mathcal{U}\,\mathbf{F})}}{\dfrac{\boxed{(p\wedge\mathbf{F})\,\mathcal{U}\,\mathbf{F}}}{\begin{array}{cc} \dfrac{\mathbf{F}}{\#} & \dfrac{\underline{p\wedge\mathbf{F}},\neg\mathbf{F},\boxed{\circ((p\wedge\mathbf{F}\wedge\mathbf{F})\,\mathcal{U}\,\mathbf{F})}}{\dfrac{p,\mathbf{F},\neg\mathbf{F},\boxed{\circ((p\wedge\mathbf{F}\wedge\mathbf{F})\,\mathcal{U}\,\mathbf{F})}}{\#}}\,(\wedge) \end{array}}\,(\mathcal{U})_2}\,(\text{unnext}) \end{array}}\,(\mathcal{U})_2$$

Example 5.12. In the following systematic tableau for $\{p,\circ\neg p,\neg\mathbf{F}\,\mathcal{U}\,\neg p\}$, the formula φ stands for $\neg\mathbf{F}\wedge\neg(p\wedge\circ\neg p)$:

$$\dfrac{p,\circ\neg p,\boxed{\neg\mathbf{F}\,\mathcal{U}\,\neg p}}{\begin{array}{cc} \dfrac{p,\neg p,\circ\neg p}{\#} & \dfrac{p,\circ\neg p,\neg\mathbf{F},\underline{\neg\neg p},\boxed{\circ(\varphi\,\mathcal{U}\,\neg p)}}{\dfrac{p,\circ\neg p,\neg\mathbf{F},\boxed{\circ(\varphi\,\mathcal{U}\,\neg p)}}{\dfrac{\neg p,\boxed{\varphi\,\mathcal{U}\,\neg p}}{\begin{array}{cc} \dfrac{\neg p}{\dfrac{\lightning_2\quad\{\,\}}{\{\,\}}\,(\text{unnext})}\,(\text{unnext}) & \dfrac{\neg p,\varphi,\neg\neg p,\boxed{\circ((\varphi\wedge\neg\neg p)\,\mathcal{U}\,p)}}{\#} \end{array}}\,(\mathcal{U})_2}\,(\text{unnext})}\,(\neg\neg) \end{array}}\,(\mathcal{U})_2$$

The central branch represents the collection of models explained in Example 4.13.

5.4. Completeness

In this subsection we prove the completeness of TTM by showing that if Φ is satisfiable then we can associate to any expanded branch b of the systematic tableau for Φ a cyclic PLTL-structure $\mathcal{G}_b$ that yields a model of Φ.

Definition 5.13. For any expanded branch b, we define the PLTL-structure $\mathcal{G}_b = (S_{\mathcal{G}_b}, V_{\mathcal{G}_b})$ such that $S_{\mathcal{G}_b} = \text{path}(\text{stages}(b))$ and $V_{\mathcal{G}_b}(s) = \{p \mid p \in L(s) \text{ and } p \in \text{Prop}\}$.

Note that termination of the systematic tableau construction is guaranteed by the finiteness of the closure (see Proposition 5.9) together with the fairness in distinguishing until formulas. Consequently, since every maximal branch of $\mathcal{T}_\Phi$ is closed or expanded, then any expanded branch must have two nodes with the same label (see Remark 5.8) which necessarily belong to two different stages, since one stage cannot contain two identical nodes. Summarizing, any expanded branch of $\mathcal{T}_\Phi$ has at least two nodes, at least two stages, and is cyclic. In the rest of this subsection we will assume that $b = n_0,\ldots,n_k$ is an expanded branch of $\mathcal{T}_\Phi$, hence b is cyclic, and that $\mathcal{G}_b$ is the cyclic PLTL-structure associated to b.

In the previous Section 5.2 we prove some properties about the behaviour of eventualities along the branches of $\mathcal{T}_\Phi$, that obviously can be applied to $\mathcal{G}_b$. The next proposition shows the behaviour of negated eventualities in $\mathcal{G}_b$.

Proposition 5.14. *Let $s_j \in S_{\mathcal{G}_b}$ such that $\neg(\varphi\,\mathcal{U}\,\psi) \in L(s_j)$. Then, every finite subsequence $\pi = s_j, s_{j+1}, \ldots, s_k$ of $S_{\mathcal{G}_b}$ satisfies one of the two following properties:*

(**a**) $\{\varphi, \neg\psi, \neg\circ(\varphi\,\mathcal{U}\,\psi)\} \subseteq L(s_i)$ *for any* $j \le i \le k$.
(**b**) *There exists* $j \le i \le k$ *such that* $\{\neg\varphi, \neg\psi\} \in L(s_i)$ *and* $\{\varphi, \neg\psi, \neg\circ(\varphi\,\mathcal{U}\,\psi)\} \subseteq L(s_\ell)$ *for all* $j \le \ell \le i-1$.

Proof. By induction on $k-j$. The case $k=j$ is trivial. For $k-j \ge 1$, the induction hypothesis guarantees that $\pi' = s_j, s_1, \dots, s_{k-1}$ satisfies one of the properties (a) or (b). If π' satisfies (b), so does π. If π' satisfies (a) then, by $\alpha\beta$-saturation, we have $\{\varphi, \neg\psi, \neg(\varphi\,\mathcal{U}\,\psi)\} \subseteq L(s_k)$ or $\{\neg\varphi, \neg\psi\} \subseteq L(s_k)$. Hence, π verifies (a) or (b), respectively. □

Therefore, we can prove that each state of $\mathcal{G}_b$ satisfies its labels, that is the set of formulas labelling all nodes that constitute the concerned stage.

Lemma 5.15. *For every* $s \in S_{\mathcal{G}_b}$, *if* $\varphi \in L(s)$ *then* $\langle \mathcal{G}_b, s\rangle \models \varphi$.

Proof. By structural induction on φ. The case of literals is trivial by definition of $\mathcal{G}_b$.
For formulas of the form $\neg\neg\varphi, \varphi \wedge \psi, \neg(\varphi \wedge \psi), \circ\varphi$ and $\neg\circ\varphi$ the property holds because every stage in $S_{\mathcal{G}_b}$ is $\alpha\beta$-saturated and the induction hypothesis on $\{\varphi\}$, $\{\varphi,\psi\}$, $\{\neg\varphi,\neg\psi\}$, $\{\varphi\}$ and $\{\neg\varphi\}$, respectively.
For $\varphi\,\mathcal{U}\,\psi$, by the above Propositions 5.4 and 5.5, there should exist a finite subsequence $s_0, s_1, \dots, s_n$ of $S_{\mathcal{G}_b}$ such that $s_0 = s, \psi \in s_n$ and $\varphi \in s_i$ for every $0 \le i < n$. By the induction hypothesis, $\langle \mathcal{G}_b, s_n\rangle \models \psi$ and $\langle \mathcal{G}_b, s_i\rangle \models \varphi$ for every $0 \le i < n$ and consequently $\langle \mathcal{G}_b, s\rangle \models \varphi\,\mathcal{U}\,\psi$.
For $\neg(\varphi\,\mathcal{U}\,\psi)$ formulas, by the above Propositions 5.3 and 5.14 and the induction hypothesis, there does not exist any finite path $s_0, s_1, \dots, s_n$ in $S_{\mathcal{G}_b}$ such that $s_0 = s, \langle \mathcal{G}_b, s_n\rangle \models \psi$ and $\langle \mathcal{G}_b, s_i\rangle \models \varphi$ for every $0 \le i < n$. Consequently $\langle \mathcal{G}_b, s\rangle \not\models \varphi\,\mathcal{U}\,\psi$ and hence $\langle \mathcal{G}_b, s\rangle \models \neg(\varphi\,\mathcal{U}\,\psi)$. □

Corollary 5.16. $\mathcal{G}_b \models \Phi$.

Proof. Immediate consequence of Lemma 5.15. □

By means of the collection of results proved in this section, we provide an alternative proof of the result that states that “every satisfiable set of PLTL-formulas has a cyclic model" (see Theorem 7.1 in [24] and Theorem 1 in [3]). Our proof is constructive in the sense that it gives a tableau-based procedure that constructs the cyclic model $\mathcal{G}_b$ for any satisfiable Φ.

Now, we prove the refutational completeness of the tableau system TTM.

Theorem 5.17. *If* Φ *is unsatisfiable then there exists a closed tableau for* Φ.

Proof. Suppose that it does not exist any closed TTM-tableau for Φ. Then the systematic tableau $\mathcal{T}_\Phi$ would be open and there would be at least one expanded branch b of $\mathcal{T}_\Phi$. By Corollary 5.16, $\mathcal{G}_b \models \Phi$. Consequently Φ would be satisfiable. □

Moreover, the tableau method TTM is also complete.

Theorem 5.18. *If* Φ *is satisfiable then there exists a (finite) open expanded tableau for* Φ.

Proof. The systematic tableau $\mathcal{T}_\Phi$ suffices to prove this fact. □

Hence, the system TTM can be used as a satisfiability decision procedure for PLTL.

5.5. Improving eventualities handling

The application of the rule $(\mathcal{U})_2$ builds up complex formulas that involve the whole context. Hence, for practical purposes, it is interesting to simplify these formulas as much as possible. In this subsection we are going to show some ideas for avoiding redundant formulas in the negated context produced by application of the rule $(\mathcal{U})_2$. That is, we introduce a new rule $(\mathcal{U})_3$ that is an improvement of $(\mathcal{U})_2$ that prevents two kinds of redundancy:

1. disjuncts stating that the next stage fails to satisfy a formula which the context ensures forever
2. duplication of formulas

Roughly speaking, the first kind of redundancy is related to the logical equivalence of $\Box\delta_1 \wedge \circ((\varphi \wedge \neg(\delta_1 \wedge \delta_2))\,\mathcal{U}\,\psi)$ and $\Box\delta_1 \wedge \circ((\varphi \wedge \neg\delta_2)\,\mathcal{U}\,\psi)$, whereas the second one corresponds to the equivalence of $\varphi \wedge \varphi \wedge \psi$ and $\varphi \wedge \psi$ or equivalently $\varphi \wedge \varphi$ and φ.

At the end of this subsection, we analyze the gain of the new rule with respect to the older one.

In order to deal with the first kind of redundancy, we introduce the following notion of persistence.

Definition 5.19. A formula φ is called *persistent* iff for all $\mathcal{M}$ and all $s_j \in S_{\mathcal{M}}$, if $\langle \mathcal{M}, s_j\rangle \models \varphi$ then $\langle \mathcal{M}, s_k\rangle \models \varphi$ for all $k > j$.

When decomposing formulas in a systematic derivation process some syntactical patterns may be used to detect persistent formulas. That is the case of the formulas of the form $\Box\varphi$ and $\circ\Box\varphi$. Taking also into account that

$$\Box\varphi \equiv \neg\Diamond\varphi \equiv \neg(\mathbf{T}\,\mathcal{U}\,\varphi) \equiv \neg(\neg\mathbf{F}\,\mathcal{U}\,\varphi) \equiv \mathbf{F}\,\mathcal{R}\,\varphi \equiv \neg\mathbf{T}\,\mathcal{R}\,\varphi$$

it is easy to prove the following result which constitutes a syntactical characterization of a subset of persistent formulas.

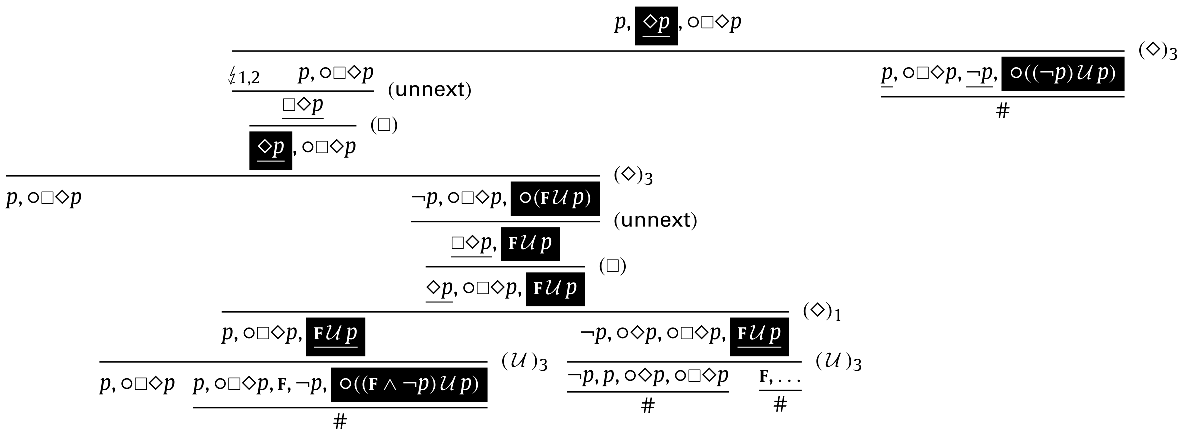


Fig. 6. An example of open systematic tableau.

Rule	β	B_1	B_2
$(\mathcal{U})_3$	$\varphi\,\mathcal{U}\,\psi$	$\{\psi\}$	$\{\varphi, \neg\psi, \circ((\varphi \Cap \widetilde{\Delta})\,\mathcal{U}\,\psi)\}$
$(\Diamond)_3$	$\Diamond\varphi$	$\{\varphi\}$	$\{\neg\varphi, \circ(\widetilde{\Delta}\,\mathcal{U}\,\varphi)\}$

Fig. 7. The rule $(\mathcal{U})_3$.

Proposition 5.20. *Every formula that matches one of the following patterns*:

$$\circ^i\Box\varphi, \circ^i\neg\Diamond\varphi, \neg\circ^i\Diamond\varphi, \circ^i\neg(\mathbf{T}\,\mathcal{U}\,\varphi), \neg\circ^i(\mathbf{T}\,\mathcal{U}\,\varphi),$$
$$\circ^i\neg(\neg\mathbf{F}\,\mathcal{U}\,\varphi), \neg\circ^i(\neg\mathbf{F}\,\mathcal{U}\,\varphi), \circ^i(\mathbf{F}\,\mathcal{R}\,\varphi), \mathbf{T}, \neg\mathbf{F}$$

is persistent. For any set of formulas Φ, *we write* persist_ch(Φ) *to denote the set of all* $\gamma \in \Phi$ *such that* γ *fits one of the above forms.*

Note that we have characterized a proper subset of the set of all persistent formulas. For example, $\neg((\neg(\varphi \wedge \neg\varphi))\,\mathcal{U}\,\psi)$ is a persistent formula which does not match any of the above syntactic patterns.

On one hand, in order to avoid including (in the disjuncts of the negated context) the negation of persistent formulas of the context, we define the following operator:

$$\widetilde{\Delta} = (\Delta \setminus \text{persist\_ch}(\Delta))^{\neg}$$

Therefore, to get rid of the above first kind of redundancy, $(\mathcal{U})_3$ applies this new operator $\simeq$ instead of the previous operator $(\_)^{\neg}$ to the context.

On the other hand, we define an operator $\Cap$ in order to prevent duplication of formulas. First, we need to extract all the negative conjuncts of a formula. The set cnjts(φ) consists of all the conjuncts of φ and is recursively defined as follows:

$$\text{cnjts}(\varphi) = \begin{cases} \text{cnjts}(\varphi_1) \cup \text{cnjts}(\varphi_2) & \text{if } \varphi \text{ is } \varphi_1 \wedge \varphi_2 \\ \{\varphi\} & \text{otherwise} \end{cases}$$

Then, the set of all negative conjuncts of φ is

$$\text{negcnjts}(\varphi) = \{\psi \mid \psi \in \text{cnjts}(\varphi) \text{ and } \psi \text{ is } \mathbf{F} \text{ or } \neg\gamma\}$$

Consequently, the operator $\Cap$ is defined as follows:

$$\varphi \Cap \widetilde{\Delta} = \begin{cases} \mathbf{F} & \text{if } \Delta = \{\ \} \text{ and } \mathbf{F} \in \text{negcnjts}(\varphi) \\ \mathbf{F} & \text{if } \Delta \in \{\text{cnjts}(\psi) \mid \neg\psi \in \text{negcnjts}(\varphi)\} \\ \varphi \wedge \widetilde{\Delta} & \text{otherwise} \end{cases}$$

It is easy to see that the following two sets of formulas are logically equivalent:

$$\Delta \cup \{\circ((\varphi \cap \widetilde{\Delta})\,\mathcal{U}\,\psi)\} \text{ and } \Delta \cup \{\circ((\varphi \wedge \Delta^{\neg})\,\mathcal{U}\,\psi)\}$$

The rule $(\mathcal{U})_3$ of Fig. 7 refines the rule $(\mathcal{U})_2$ of Fig. 2 since the second premise $\circ((\varphi \wedge \Delta^{\neg})\,\mathcal{U}\,\psi)$ of the rule $(\mathcal{U})_2$ is substituted by $\circ((\varphi \Cap \widetilde{\Delta})\,\mathcal{U}\,\psi)$ in the rule $(\mathcal{U})_3$. It is easy to derive, from the new rule $(\mathcal{U})_3$, the corresponding rule $(\Diamond)_3$ for the defined connective $\Diamond$.

Classical Connectives Rules

$$(\neg\neg)\ \frac{\Delta, \varphi \vdash}{\Delta, \neg\neg\varphi \vdash} \qquad (\wedge)\frac{\Delta, \varphi, \psi \vdash}{\Delta, \varphi \wedge \psi \vdash} \qquad (\neg\wedge)\frac{\Delta, \neg\varphi \vdash \qquad \Delta, \neg\psi \vdash}{\Delta, \neg(\varphi \wedge \psi) \vdash}$$

Temporal Connectives Rules

$$(\circ)\ \frac{\mathsf{unnext}(\Delta) \vdash}{\Delta \vdash} \qquad (\mathcal{U})_1\ \frac{\Delta, \psi \vdash \qquad \Delta, \varphi, \neg\psi, \circ(\varphi\,\mathcal{U}\,\psi) \vdash}{\Delta, \varphi\,\mathcal{U}\,\psi \vdash}$$

$$(\neg\circ)\ \frac{\Delta, \circ\neg\varphi \vdash}{\Delta, \neg\circ\varphi \vdash} \qquad (\mathcal{U})_2\ \frac{\Delta, \psi \vdash \qquad \Delta, \varphi, \neg\psi, \circ((\varphi \wedge \Delta^{\neg})\,\mathcal{U}\,\psi) \vdash}{\Delta, \varphi\,\mathcal{U}\,\psi \vdash}$$

$$(\neg\mathcal{U})\ \frac{\Delta, \neg\varphi, \neg\psi \vdash \qquad \Delta, \varphi, \neg\psi, \neg\circ(\varphi\,\mathcal{U}\,\psi) \vdash}{\Delta, \neg(\varphi\,\mathcal{U}\,\psi) \vdash}$$

Structural Rules

$$(Cd_1)\ \frac{}{\Delta, \varphi, \neg\varphi \vdash} \qquad (Cd_2)\ \frac{}{\Delta, \mathbf{F} \vdash}$$

Fig. 8. The sequent calculus TTC.

Now, let us give an example that makes use of these two new rules (Fig. 7) and the derived rules in Fig. 3.

Example 5.21. In Fig. 6 we depict a systematic tableau for $\{p, \Diamond p, \circ\Box\Diamond p\}$. As expected from the satisfiability of the root set, the tableau is open. Concretely, there are two cyclic (expanded) branches with a common repeated node. Recall that the distinguished formulas are in black boxes and the internal repeated node is marked with the symbol $\lightning_{1,2}$ for indicating that this node coincides with the last node of the first and the second branch.

Finally, we formally analyze the gain of using rule $(\mathcal{U})_3$ instead of $(\mathcal{U})_2$. This analysis yields a small difference between both worst cases, although the improvement is very useful for practical implementation.

We reformulate the notion of closure for the system $(\text{TTM} \setminus \{(\mathcal{U})_2\}) \cup \{(\mathcal{U})_3\}$. To this end, we also need to redefine some other previously defined sets of formulas. However, other auxiliary sets, e.g. preclosure, remain defined as before. In order to stress what sets are redefined, we use the prefix new_. The new definitions for the sets of negated contexts and conjunctions are:

$$\mathsf{new\_negctx}(\Phi) = \{\Delta^{\neg} \mid \Delta \subseteq (\mathsf{preclo}(\Phi) \setminus \mathsf{persist\_ch}(\mathsf{preclo}(\Phi)))\}$$

$$\mathsf{new\_conj}(\Phi) = \left\{ \bigwedge_{\delta\in\Gamma} \delta \mid \Gamma \subseteq \mathsf{new\_negctx}(\Phi) \text{ and } \Gamma \text{ is adequate} \right\}$$

where we say that $\Gamma \subseteq \mathsf{new\_negctx}(\Phi)$ is *adequate* iff

$$\mathsf{cnjts}(\delta) \neq \mathsf{cnjts}(\delta') \text{ for every pair } (\neg\delta, \neg\delta') \in \Gamma \times \Gamma \text{ such that } \delta \neq \delta'$$

Now, the closure of Φ can be redefined as follows:

$$\begin{aligned}\mathsf{new\_clo}(\Phi) = \quad & \mathsf{preclo}(\Phi) \cup \mathsf{new\_conj}(\Phi) \cup \\ & \{(\varphi \wedge \gamma)\,\mathcal{U}\,\psi, \circ((\varphi \wedge \gamma)\,\mathcal{U}\,\psi) \mid \varphi\,\mathcal{U}\,\psi \in \mathsf{sf}(\Phi) \text{ and } \gamma \in \mathsf{new\_conj}(\Phi)\}\end{aligned}$$

Hence, the cardinality of this closure is a bit smaller than stated in Proposition 5.9. Actually, if $|\mathsf{preclo}(\Phi)| = n$ then $|\mathsf{new\_negctx}(\Phi)| \in O(2^n)$. Therefore

$$|\mathsf{new\_conj}(\Phi)|, |\mathsf{new\_clo}(\Phi)| \in O(2^{2^n}).$$

Recall that $|\mathsf{clo}(\Phi)| \in O(2^{O(2^n)})$.

6. The sequent calculus TTC

In this section we introduce the sequent calculus TTC that directly corresponds to the previously introduced tableau system TTM. It is a reformulation of TTM as a one-sided sequent calculus that serves as a bridge from TTM to the two-sided sequent calculus GTC that we will introduce in the following section.

The sequent calculus TTC follows the left-handed one-sided approach (also known as Tait-style, [23]), where sequents are formed by a set of formulas. We write $\Delta \vdash$ to represent a sequent whose set of formulas is Δ and whose intended meaning is $\bigwedge \Delta \to \mathbf{F}$.

The rules of TTC (see Fig. 8) are obtained essentially from the TTM-rules writing them upside down with the difference that in TTC we have left-handed sequents and in TTM we have simply sets of formulas. The only exception is the rule $(\circ)$ that corresponds to the application of the operator unnext in TTM. This direct relation between both systems makes possible to obtain a TTC-proof from any closed TTM-tableau in a straightforward manner.

The strong similarity between tableau refutations and left-handed sequent proofs that are cut-free, contraction-free and weakening-free is evident. As a consequence, TTC is cut-free, invariant-free, weakening-free and contraction-free.

We have split the primitive rules of TTC into three packages. Two of them consist of rules for classical and temporal connectives, respectively. These rules follow the traditional style of introduction of the connective and its negation in the sequent. In addition, we need two structural rules which form the third package.

As TTC is sound and complete (Theorems 6.1 and 6.3), given a set of formulas Δ, it holds that Δ is unsatisfiable if and only if there is a TTC-proof for $\Delta \vdash$.

A TTC-derivation is a possibly infinite tree labelled with sequents and built according to the inference rules in TTC. A TTC-proof is a finite derivation where the sequent to be proved labels its root and the leaves are labelled with axioms (which are rules without premises).

A set of formulas Γ is TTC-*consistent* if and only if there is no any TTC-proof for the sequent $\Gamma \vdash$.

The soundness of TTC means that every TTC-provable sequent, namely $\Gamma \vdash$, is correct regarding to satisfiability. In particular, every satisfiable set of formulas Γ is TTC-consistent.

In the TTC sequent calculus all the non-structural rules are *invertible* except for the $(\circ)$ rule. A rule is invertible when it holds that if the conclusion is provable, so are the premises.

Theorem 6.1 (Soundness). *For any set of formulas Γ, if Γ is not* TTC*-consistent, i.e., if there exists a* TTC*-proof, then Γ is unsatisfiable.*

Proof. By induction on the length of the TTC-proof, it suffices to prove that every primitive rule of TTC (see Fig. 8) is correct in the sense that if the set of formulas of each premise is unsatisfiable then the set of formulas of the conclusion is unsatisfiable. The only difficult case is the case of the rule $(\mathcal{U})_2$. The justification for that case is already given in Theorem 5.2. □

Next, we prove that TTC is a complete calculus relating its completeness to the completeness of TTM.

Proposition 6.2. *For any set of formulas Φ, if $\mathcal{T}_\Phi$ is a closed expanded tableau for Φ then there exists a* TTC*-proof for the sequent $\Phi \vdash$.*

Proof. Since each TTM-rule has its corresponding TTC-rule, the TTC-proof is directly obtained from the closed TTM-tableau for Φ. □

Theorem 6.3 (Completeness). *For any set of formulas Φ, if Φ is unsatisfiable, then there exists a* TTC*-proof for Φ.*

Proof. If Φ is unsatisfiable then there exists a closed TTM-tableau for Φ. Hence, by Proposition 6.2 there exists a TTC-proof for Φ. □

The exhaustive application of the rules in the calculus TTC, without any additional restriction or strategy, does not yield a decision procedure for TTC. The reason is that TTC, by itself, does not satisfy the weak analytic superformula property (WASP) (see Section 5.2). Remember that the systematic tableau algorithm of Section 5.2 incorporates an strategy for the application of $(\mathcal{U})_2$ which contributes to the satisfaction of the WASP.

The admissible rules are new sound rules that cannot be derived from the primitive rules of TTC, but do not add deductive power to the system. That is, a set Φ is consistent with respect to TTC if and only if Φ is consistent with respect to TTC plus the admissible rules. In other words, for every TTC-proof that includes the use of some admissible rules there exists another TTC-proof that does not use any admissible rule.

The derived rules can be used as a shortcut for several lines of proofs that are built using only primitive and admissible rules.

Among the admissible rules the most outstanding ones are the following classical structural rules of Weakening and Cut:

$$(Wk)\ \frac{\Delta \vdash}{\Delta, \Delta' \vdash} \qquad\qquad (Cut)\ \frac{\Delta, \varphi \vdash \qquad \Delta, \neg\varphi \vdash}{\Delta \vdash}$$

The sequent calculus TTC is cut-free since we have already proved its soundness and completeness and the cut rule is omitted in TTC. Since TTC is complete without the cut rule, the cut rule is admissible in TTC. However, the classical syntactical techniques for cut elimination cannot be applied here because of the context used in the rule $(\mathcal{U})_2$. Hence, we have been unable to give a syntactic proof of cut elimination. However, we are aware of the work of K. Brünnler, who introduced the notion of *deep sequent* and gave a cut-elimination procedure for modal logic [4]. It seems feasible that the same technique applied to our calculi (extended with the cut rule) could yield a syntactic cut-elimination procedure for PLTL.

$$(\vee)\ \frac{\begin{array}{c}\Delta,\varphi\vdash\\ \Delta,\psi\vdash\end{array}}{\Delta,\varphi\vee\psi\vdash}\qquad(\neg\vee)\ \frac{\Delta,\neg\varphi,\neg\psi\vdash}{\Delta,\neg(\varphi\vee\psi\vdash)}\qquad(\mathcal{R})\ \frac{\begin{array}{c}\Delta,\varphi,\psi\vdash\\ \Delta,\neg\varphi,\psi,\circ(\varphi\mathcal{R}\psi)\vdash\end{array}}{\Delta,\varphi\mathcal{R}\psi\vdash}$$

$$(\neg\mathcal{R})_1\ \frac{\begin{array}{c}\Delta,\neg\psi\vdash\\ \Delta,\neg\varphi,\psi,\circ(\neg\varphi\mathcal{U}\neg\psi)\vdash\end{array}}{\Delta,\neg(\varphi\mathcal{R}\psi)\vdash}\qquad(\neg\mathcal{R})_2\ \frac{\begin{array}{c}\Delta,\neg\psi\vdash\\ \Delta,\neg\varphi,\psi,\circ((\neg\varphi\wedge\Delta^{\neg})\mathcal{U}\neg\psi)\vdash\end{array}}{\Delta,\neg(\varphi\mathcal{R}\psi)\vdash}$$

$$(\Diamond)_1\ \frac{\begin{array}{c}\Delta,\varphi\vdash\\ \Delta,\neg\varphi,\circ(\mathbf{T}\mathcal{U}\varphi)\vdash\end{array}}{\Delta,\Diamond\varphi\vdash}\qquad(\Diamond)_2\ \frac{\begin{array}{c}\Delta,\varphi\vdash\\ \Delta,\neg\varphi,\circ(\Delta^{\neg}\mathcal{U}\varphi)\vdash\end{array}}{\Delta,\Diamond\varphi\vdash}\qquad(\neg\Diamond)\ \frac{\Delta,\neg\varphi,\neg\circ\Diamond\varphi\vdash}{\Delta,\neg\Diamond\varphi\vdash}$$

$$(\Box)\ \frac{\Delta,\varphi,\circ\Box\varphi\vdash}{\Delta,\Box\varphi\vdash}\qquad(\neg\Box)_1\ \frac{\begin{array}{c}\Delta,\neg\varphi\vdash\\ \Delta,\varphi,\circ(\mathbf{T}\mathcal{U}\neg\varphi)\vdash\end{array}}{\Delta,\neg\Box\varphi\vdash}\qquad(\neg\Box)_2\ \frac{\begin{array}{c}\Delta,\neg\varphi\vdash\\ \Delta,\varphi,\circ(\Delta^{\neg}\mathcal{U}\neg\varphi)\vdash\end{array}}{\Delta,\neg\Box\varphi\vdash}$$

Fig. 9. Derived rules for TTC.

The weakening rule (Wk) is non-invertible so it must be used carefully. The rules $(\mathbf{T})$ and $(\neg\mathbf{F})$, that appear below, are particular cases of the rule (Wk) but they are invertible. So they can be used to eliminate the formulas $\mathbf{T}$ and $\neg\mathbf{F}$ knowing that the equivalence with respect to the TTC-consistency is preserved:

$$(\mathbf{T})\ \frac{\Delta\vdash}{\Delta,\mathbf{T}\vdash}\qquad(\neg\mathbf{F})\ \frac{\Delta\vdash}{\Delta,\neg\mathbf{F}\vdash}$$

Since TTC is also contraction-free, admissible rules could be obtained by associating to every non-structural rule (R) the rule (RC) that produces an (implicit) contraction in (R). For example, the rule below $(\wedge C)$ is the admissible rule that corresponds to the primitive rule $(\wedge)$.

$$(\wedge C)\ \frac{\Delta,\varphi\wedge\psi,\varphi,\psi\vdash}{\Delta,\varphi\wedge\psi\vdash}$$

Regarding derived rules, first we use the usual abbreviations of defined connectives in order to derive the rules in Fig. 9. It is easy to check that $(\vee)$ is derived from $(\neg\wedge)$ and $(\neg\neg)$; $(\neg\vee)$ from $(\neg\neg)$ and $(\wedge)$; $(\mathcal{R})$ from $(\neg\mathcal{U})$ and $(\neg\neg)$; for $i\in\{1,2\}$: $(\neg\mathcal{R})_i$ is derived from $(\neg\neg)$ and $(\mathcal{U})_i$; for $i\in\{1,2\}$: $(\Diamond)_i$ is derived from $(\mathcal{U})_i$ and $(\mathbf{T})$; $(\neg\Diamond)$ is derived from $(\neg\mathcal{U})$, $(\mathbf{T})$, $(\neg\neg)$ and $(Cd)_2$; $(\Box)$ from $(\neg\Diamond)$, $(\neg\neg)$, $(\mathbf{T})$ and $(\neg\circ)$; and for $i\in\{1,2\}$: $(\neg\Box)_i$ from $(\neg\neg)$, $(\Diamond)_i$ and $(\mathbf{T})$.

The soundness and invertibility of these derived rules is guaranteed by the fact that they have been obtained using only sound and invertible rules. Note that if the (Wk) rule is used instead of $(\mathbf{T})$ for deriving the previous rules their invertibility could not be directly guaranteed.

It is well known that the until operator, $\mathcal{U}$, is not expressible in temporal logic with only $\circ$, $\Box$, and $\Diamond$ as temporal operators (cf. [14,7]). As a consequence, a complete calculus for the sublogic that uses $\Diamond$ instead of $\mathcal{U}$ cannot be derived (by abbreviation) from TTC, since the rule $(\Diamond)_2$ needs the until operator for expressing its second premise.

Finally, let us recall the respective refinements $(\Diamond)_3$ and $(\mathcal{U})_3$ of the rules $(\Diamond)_2$ and $(\mathcal{U})_2$ that allow us to avoid including persistent formulas and duplications in the negation of the context (see Section 5.5):

$$(\Diamond)_3\ \frac{\begin{array}{c}\Delta,\varphi\vdash\\ \Delta,\neg\varphi,\circ(\widetilde{\Delta}\mathcal{U}\varphi)\vdash\end{array}}{\Delta,\Diamond\varphi\vdash}\qquad(\mathcal{U})_3\ \frac{\begin{array}{c}\Delta,\psi\vdash\\ \Delta,\varphi,\neg\psi,\circ((\varphi\Cap\widetilde{\Delta})\mathcal{U}\psi)\vdash\end{array}}{\Delta,\varphi\mathcal{U}\psi\vdash}$$

7. The sequent calculus GTC

In this section we present a sequent calculus GTC (see Fig. 10) that is two-sided and one-conclusioned (or asymmetric). We prove the soundness of GTC and, then, we discuss about admissible and derived rules. Afterwards, we prove the completeness of GTC with the help of some previously derived rules. Finally, we give three examples of GTC-proofs.

The calculus GTC (see Fig. 10) is straightforwardly obtained from the previous calculus TTC. Actually, almost each primitive rule of TTC has a counterpart in GTC that results from adding a conclusion χ to each sequent in the rule. The only exception are the rules where the context is combined with the principal formula to produce the sequents in the numerator, where χ (or better $\neg\chi$) behaves as part of the context. Moreover, admissible or derived rules in GTC are the same kind of counterparts of TTC rules as the primitive ones.

Classical Connectives Rules

$$(\neg L)\ \frac{\Delta \vdash \varphi}{\Delta, \neg\varphi \vdash \chi} \qquad (R\neg)\ \frac{\Delta, \varphi \vdash \mathbf{F}}{\Delta \vdash \neg\varphi} \qquad (\wedge L)\ \frac{\Delta, \varphi, \psi \vdash \chi}{\Delta, \varphi \wedge \psi \vdash \chi} \qquad (R\wedge)\ \frac{\Delta \vdash \varphi \quad \Delta \vdash \psi}{\Delta \vdash \varphi \wedge \psi}$$

Temporal Connectives Rules

$$(\neg\circ L)\ \frac{\Delta, \circ\neg\varphi \vdash \chi}{\Delta, \neg\circ\varphi \vdash \chi} \qquad (R\circ L)\frac{\mathsf{unnext}(\Delta) \vdash \varphi}{\Delta \vdash \circ\varphi} \qquad (R\circ\neg)\ \frac{\Delta \vdash \neg\circ\varphi}{\Delta \vdash \circ\neg\varphi}$$

$$(\mathcal{U} L)_1\ \frac{\begin{array}{l}\Delta, \psi \vdash \chi \\ \Delta, \varphi, \neg\psi, \circ(\varphi\mathcal{U}\psi) \vdash \chi\end{array}}{\Delta, \varphi\mathcal{U}\psi \vdash \chi} \qquad (R\mathcal{U})\ \frac{\begin{array}{l}\Delta, \neg\varphi \vdash \psi \\ \Delta, \varphi, \neg\circ(\varphi\mathcal{U}\psi) \vdash \psi\end{array}}{\Delta \vdash \varphi\mathcal{U}\psi}$$

$$(\mathcal{U} L)_2\ \frac{\begin{array}{l}\Delta, \psi \vdash \chi \\ \Delta, \varphi, \neg\psi, \circ((\varphi\wedge(\Delta\cup\{\neg\chi\})^{\neg})\mathcal{U}\psi) \vdash \chi\end{array}}{\Delta, \varphi\mathcal{U}\psi \vdash \chi}$$

Structural Rules

$$(As)\ \frac{}{\Delta, \varphi \vdash \varphi} \qquad (Cd)\ \frac{\Delta, \neg\varphi \vdash \mathbf{F}}{\Delta \vdash \varphi} \qquad (\circ\mathbf{F})\ \frac{\Delta \vdash \circ\mathbf{F}}{\Delta \vdash \chi}$$

Fig. 10. The sequent calculus GTC.

$$(\mathbf{F}L)\ \frac{}{\Delta, \mathbf{F} \vdash \chi} \qquad (CdL)\ \frac{}{\Delta, \varphi, \neg\varphi \vdash \chi} \qquad (\neg\wedge L)\ \frac{\begin{array}{l}\Delta, \neg\varphi \vdash \chi \\ \Delta, \neg\psi \vdash \chi\end{array}}{\Delta, \neg(\varphi \wedge \psi) \vdash \chi}$$

$$(\neg\neg L)\ \frac{\Delta, \varphi \vdash \chi}{\Delta, \neg\neg\varphi \vdash \chi} \qquad (\circ L)\ \frac{\mathsf{unnext}(\Delta) \vdash \mathbf{F}}{\Delta \vdash \chi} \qquad (\neg\,\mathcal{U} L)\ \frac{\begin{array}{l}\Delta, \neg\varphi, \neg\psi \vdash \chi \\ \Delta, \varphi, \neg\psi, \neg\circ(\varphi\mathcal{U}\psi) \vdash \chi\end{array}}{\Delta, \neg(\varphi\mathcal{U}\psi) \vdash \chi}$$

Fig. 11. Derived GTC-rules.

The soundness of GTC means that every GTC-provable sequent, namely $\Gamma \vdash \chi$, is correct regarding to logical consequence. In particular, every satisfiable set of formulas is GTC-consistent.

Theorem 7.1 (Soundness). *For any set of formulas $\Gamma \cup \{\chi\}$, if $\Gamma \vdash \chi$ is* GTC*-provable then $\Gamma \models \chi$.*

Proof. By induction on the length of the GTC-proof, it suffices to prove that every primitive rule of GTC (see Fig. 10) is correct in the sense of preserving the logical consequence relation between the antecedent and the consequent.

Now, the correctness proof of most rules is just routine. Actually, the only correctness proof that poses some difficulties is the proof of the rule $(\mathcal{U}L)_2$. Hence, we only give the details for this rule, by mimicking the proof of Lemma 5.1.

Let us assume that $\Delta \cup \{\varphi\mathcal{U}\psi, \neg\chi\}$ is satisfiable, then we can build a countermodel for some of the two premises of the rule $(\mathcal{U}L)_2$. Let $\langle\mathcal{M}, s_i\rangle \models \Delta \cup \{\varphi\mathcal{U}\psi, \neg\chi\}$ and z the least $j \geq i$ such that $\langle\mathcal{M}, s_j\rangle \models \psi$. If $z = i$ then $\langle\mathcal{M}, s_z\rangle$ serves as countermodel for the first premise. Otherwise, if $z > i$, let y be the greatest j such that $i \leq j < z$ and $\langle\mathcal{M}, s_j\rangle \models \Delta \cup \{\varphi\mathcal{U}\psi, \neg\chi\}$. As a consequence of the choice of z and y, it holds that $\langle\mathcal{M}, s_y\rangle \models \{\varphi, \neg\psi, \circ((\varphi \wedge (\Delta \cup \{\neg\chi\})^{\neg})\mathcal{U}\psi)\}$. Then, $\langle\mathcal{M}, s_y\rangle$ yields a countermodel for the second premise. □

The calculus GTC is more versatile than TTC, in particular GTC allows not only refutation proofs, but also goal-directed proofs or, in general, the consequent can directly be used as principal formula in GTC-proofs. As a consequence, in GTC, we can derive rules that have no sense in one-sided systems. For example, the contraposition rules:

$$(Cp1)\ \frac{\Delta, \neg\varphi \vdash \psi}{\Delta, \neg\psi \vdash \varphi} \qquad (Cp2)\ \frac{\Delta, \varphi \vdash \psi}{\Delta, \neg\psi \vdash \neg\varphi}$$

which can be derived in the usual way from the classical connectives primitive rules in GTC.

The derived rules in Fig. 11 are useful for proving the completeness of GTC. They are easily derived with the help of the above rules $(Cp1)$ and $(Cp2)$. It is easy to check that $(\mathbf{F}L)$ is derived from (Cd) and (As); (CdL) from $(\neg L)$ and (As); $(\circ L)$ from $(\circ\mathbf{F})$ and $(R\circ L)$; $(\neg\neg L)$ from $(Cp1)$ and $(Cp2)$; $(\neg\wedge L)$ from $(Cp1)$ and $(R\wedge)$; and $(\neg\,\mathcal{U}L)$ from $(Cp1)$ and $(R\mathcal{U})$.

Now, we can associate to each TTC-proof a GTC-proof.

Proposition 7.2. *If $\Phi \vdash$ is* TTC*-provable then $\Phi \vdash \mathbf{F}$ is* GTC*-provable.*

Proof. Suppose that $\Phi \vdash$ is TTC-provable. Then, by admissibility of the rule $(\neg\mathbf{F})$ (see Section 6), $\Phi, \neg\mathbf{F} \vdash$ is also TTC-provable.
It is easy to see that for each TTC-rule there is a closely related (primitive or derived) GTC-rule. In particular, TTC-rules are GTC-derived rules or single instances of GTC-rules. More precisely, the TTC-rules $(\neg\neg)$, $(\vee)$, $(\neg\vee)$, $(\circ)$, $(\mathcal{U})_1$, $(\mathcal{U})_2$, $(\neg\circ)$, $(\neg\mathcal{U})$, (Cd_1) and (Cd_2), respectively correspond to $(\neg\neg L)$, $(\vee L)$, $(\neg\vee L)$, $(\circ L)$, $(\mathcal{U}L)_1$, $(\mathcal{U}L)_2$, $(\neg\circ L)$, $(\neg\mathcal{U}L)$, (CdL) and $(\mathbf{F}L)$. As a consequence, we can construct a GTC-proof of the two-sided sequent $\Phi, \neg\mathbf{F} \vdash \mathbf{F}$. Therefore, using the GTC-rule (Cd), the sequent $\Phi \vdash \mathbf{F}$, is also GTC-provable. □

Theorem 7.3 (Completeness)**.** *For any set of formulas $\Gamma \cup \{\chi\}$, if $\Gamma \models \chi$ then $\Gamma \vdash \chi$ is* GTC*-provable.*

Proof. If $\Gamma \vdash \chi$ is not GTC-provable, then by rule (Cd) the sequent $\Gamma \cup \{\neg\chi\} \vdash \mathbf{F}$ is not GTC-provable. By Proposition 7.2, $\Gamma \cup \{\neg\chi\} \vdash$ is not TTC-provable, which is a contradiction by Theorem 6.3. □

Using the abbreviations $\Diamond\varphi$ and $\Box\varphi$ for $\mathbf{T}\,\mathcal{U}\,\varphi$ and $\neg\Diamond\neg\varphi$, respectively, we are also able to derive the following useful rules:

$$(\Diamond L)_1\ \dfrac{\begin{array}{l}\Delta, \varphi \vdash \chi\\ \Delta, \neg\varphi, \circ(\mathbf{T}\,\mathcal{U}\,\varphi) \vdash \chi\end{array}}{\Delta, \Diamond\varphi \vdash \chi} \qquad (\Diamond L)_2\ \dfrac{\begin{array}{l}\Delta, \varphi \vdash \chi\\ \Delta, \neg\varphi, \circ((\Delta \cup \{\neg\chi\})^{\neg}\,\mathcal{U}\,\varphi) \vdash \chi\end{array}}{\Delta, \Diamond\varphi \vdash \chi}$$

$$(R\Diamond)\ \dfrac{\Delta, \neg\circ\Diamond\varphi \vdash \varphi}{\Delta \vdash \Diamond\varphi} \qquad (\Box L)\ \dfrac{\Delta, \varphi, \circ\Box\varphi \vdash \chi}{\Delta, \Box\varphi \vdash \chi}$$

$$(R\Box)_1\ \dfrac{\begin{array}{l}\Delta \vdash \varphi\\ \Delta, \circ(\mathbf{T}\,\mathcal{U}\,\neg\varphi) \vdash \neg\varphi\end{array}}{\Delta \vdash \Box\varphi} \qquad (R\Box)_2\ \dfrac{\begin{array}{l}\Delta \vdash \varphi\\ \Delta, \circ(\Delta^{\neg}\,\mathcal{U}\,\neg\varphi) \vdash \neg\varphi\end{array}}{\Delta \vdash \Box\varphi}$$

In addition, the TTC-rules $(\mathcal{U})_3$ and $(\Diamond)_3$ produce the corresponding GTC-rules where $\Delta' = \Delta \cup \{\neg\chi\}$:

$$(\mathcal{U}L)_3\ \dfrac{\begin{array}{l}\Delta, \psi \vdash \chi\\ \Delta, \varphi, \neg\psi, \circ((\varphi \Cap \widetilde{\Delta'})\,\mathcal{U}\,\psi) \vdash \chi\end{array}}{\Delta, \varphi\,\mathcal{U}\,\psi \vdash \chi} \qquad (\Diamond L)_3\ \dfrac{\begin{array}{l}\Delta, \varphi \vdash \chi\\ \Delta, \neg\varphi, \circ(\widetilde{\Delta'}\,\mathcal{U}\,\varphi) \vdash \chi\end{array}}{\Delta, \Diamond\varphi \vdash \chi}$$

and it is easy to derive the following rule $(R\Box)_3$ for the defined connective $\Box$:

$$(R\Box)_3\ \dfrac{\begin{array}{l}\Delta \vdash \varphi\\ \Delta, \circ(\widetilde{\Delta}\,\mathcal{U}\,\neg\varphi) \vdash \neg\varphi\end{array}}{\Delta \vdash \Box\varphi}$$

Note that, by $(\Box L)$ and (CdL), the following contradiction rule is also derivable:

$$(Cd\Box)\ \dfrac{}{\Delta, \Box\varphi, \neg\circ\Box\varphi \vdash \chi}$$

Let us now illustrate the GTC-style of reasoning by means of some examples of GTC-proofs. In order to enhance readability, we have underlined, at each step, the principal formula. Both primitive and derived rules are used in the derivations.

Example 7.4. The following GTC-proof shows that the formula q is a logical consequence of the set of formulas $\{p\,\mathcal{U}\,q, \neg\circ\Diamond q\}$.

$$\begin{prooftree}
\AxiomC{}
\RightLabel{(As)}
\UnaryInfC{$\underline{q}, \circ\neg\diamond q \vdash \underline{q}$}
\AxiomC{}
\RightLabel{(As)}
\UnaryInfC{$\underline{q}, \neg\circ\diamond q \vdash \underline{q}$}
\AxiomC{}
\RightLabel{(CdL)}
\UnaryInfC{$p, \underline{\neg\neg q}, \underline{\neg q}, \circ((p \wedge \neg\neg q)\,\mathcal{U}\,q), \circ\neg\diamond q \vdash q$}
\RightLabel{$(\wedge)$}
\UnaryInfC{$\underline{p \wedge \neg\neg q}, \neg q, \circ((p \wedge \neg\neg q)\,\mathcal{U}\,q), \circ\neg\diamond q \vdash q$}
\RightLabel{$(\mathcal{U}L)_3$}
\BinaryInfC{$\underline{(p \wedge \neg\neg q)\,\mathcal{U}\,q}, \circ\neg\diamond q \vdash q$}
\doubleLine
\RightLabel{$(\neg\circ L)$}
\UnaryInfC{$(p \wedge \neg\neg q)\,\mathcal{U}\,q, \underline{\neg\circ\diamond q} \vdash q$}
\RightLabel{$(R\diamond)$}
\UnaryInfC{$(p \wedge \neg\neg q)\,\mathcal{U}\,q \vdash \underline{\diamond q}$}
\RightLabel{$(\neg L)$}
\UnaryInfC{$(p \wedge \neg\neg q)\,\mathcal{U}\,q, \underline{\neg\diamond q} \vdash \mathbf{F}$}
\RightLabel{$(R\circ L)$}
\UnaryInfC{$p, \neg q, \circ((p \wedge \neg\neg q)\,\mathcal{U}\,q), \circ\neg\diamond q \vdash \circ\mathbf{F}$}
\RightLabel{$(\circ\mathbf{F})$}
\UnaryInfC{$p, \neg q, \circ((p \wedge \neg\neg q)\,\mathcal{U}\,q), \circ\neg\diamond q \vdash q$}
\RightLabel{$(\mathcal{U}L)_3$}
\BinaryInfC{$\underline{p\,\mathcal{U}\,q}, \circ\neg\diamond q \vdash q$}
\doubleLine
\RightLabel{$(\neg\circ L)$}
\UnaryInfC{$p\,\mathcal{U}\,q, \underline{\neg\circ\diamond q} \vdash q$}
\end{prooftree}$$

Note that by using $(\mathcal{U}L)_3$ we avoid to consider the persistent formula $\circ\neg\Diamond q$ and also the repetition of $\neg\neg q$.

Example 7.5. The following GTC-proof shows that the formula $\neg\Diamond p$ is a logical consequence of the set of formulas $\{\Box\neg p\}$.

$$
\dfrac{
\dfrac{\dfrac{}{\underline{\neg p}, \circ\Box\neg p, \underline{p} \vdash \mathbf{F}}\,(CdL)}{\underline{\Box\neg p}, p \vdash \mathbf{F}}\,(\Box L)
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\underline{\neg p}, \circ\Box\neg p, \underline{p} \vdash \mathbf{F}}\,(CdL)}{\underline{\Box\neg p}, p \vdash \mathbf{F}}\,(\Box L)
\quad
\dfrac{}{\Box\neg p, \underline{\mathbf{F}}, \neg p, \circ(\mathbf{F}\,\mathcal{U}\,p) \vdash \mathbf{F}}\,(\mathbf{F}L)}{\Box\neg p, \underline{\mathbf{F}\,\mathcal{U}\,p} \vdash \mathbf{F}}\,(\mathcal{U}\,L)_3}{\circ\Box\neg p, \neg p, \circ(\mathbf{F}\,\mathcal{U}\,p) \vdash \circ\mathbf{F}}\,(R\circ L)}{\circ\Box\neg p, \neg p, \circ(\mathbf{F}\,\mathcal{U}\,p) \vdash \mathbf{F}}\,(\circ\mathbf{F})}{\underline{\Box\neg p}, \neg p, \circ(\mathbf{F}\,\mathcal{U}\,p) \vdash \mathbf{F}}\,(\Box L)
}{
\dfrac{\Box\neg p, \underline{\diamond p} \vdash \mathbf{F}}{\Box\neg p \vdash \underline{\neg\diamond p}}\,(R\neg)
}\,(\diamond L)_3
$$

Note that, when applying the rule $(\Diamond L)_3$ and $(\mathcal{U}\,L)_3$, the persistent formulas $\Box\neg p$ and $\neg\mathbf{F}$ are left out.

Example 7.6. The following GTC-proof shows that the formula $\Box p$ is a logical consequence of $\{p, \Box(\neg p \vee \circ p)\}$. This is a typical property of *induction on time*. In the GTC-proof below φ is an abbreviation for $\neg p \vee \circ p$ and ψ for $\neg p\,\mathcal{U}\,\neg p$.

$$
\dfrac{
\dfrac{}{\underline{p}, \Box\varphi \vdash \underline{p}}\,(As)
\qquad
\dfrac{\dfrac{\dfrac{}{p, \underline{\neg p}, \circ\Box\varphi, \circ\psi \vdash \underline{\neg p}}\,(As)
\quad
\dfrac{\dfrac{\dfrac{}{\underline{p}, \Box\varphi, \underline{\neg p} \vdash \mathbf{F}}\,(CdL)
\quad
\dfrac{}{p, \Box\varphi, \underline{\neg p, \neg\neg p}, \circ\psi \vdash \mathbf{F}}\,(CdL)}{p, \Box\varphi, \underline{\psi} \vdash \mathbf{F}}\,(\mathcal{U}\,)_1}{p, \underline{\circ p, \circ\Box\varphi, \circ\psi} \vdash \neg p}\,(\circ L)}{p, \underline{\varphi}, \circ\Box\varphi, \circ\psi \vdash \neg p}\,(\vee L)}{p, \underline{\Box\varphi}, \circ\psi \vdash \neg p}\,(\Box L)
}{p, \Box\varphi \vdash \underline{\Box p}}\,(R\Box)_3
$$

8. Concluding remarks

We have introduced a tableau system and two sequent calculi for the logic PLTL that can be seen as dual systems. The former system TTM differs from traditional temporal tableaux in that it does not require auxiliary graphs for checking the fulfilling property for eventualities. The latter calculus GTC is finitary and completely cut-free, in particular invariant-free. In addition, as a consequence of the duality with the tableau system, the sequent calculus is also weakening- and contraction-free. A sequent calculus called $\mathcal{FC}$, that is very similar to GTC, was presented in [9]. There, in order to prove completeness, the weakening rule (Wk), as well as a hidden contraction, were needed. In this sense, GTC is an improvement of $\mathcal{FC}$ that has been achieved using its duality with the tableau system TTM.

We have contributed new ideas to the proof-theory of PLTL. In particular, we believe that automated reasoning in temporal logic can take benefit from the systems introduced in this paper. We are working for improving the existing methods of temporal resolution (see [6]). Concretely, using the presented ideas we are able not only to avoid the construction of invariants in the clausal resolution setting, but also to keep the classical form of clausal normal forms. However, as a consequence of the double exponential size of the closure set (see Proposition 5.9), our decision procedure has a poor worst case performance. Actually, the decision problem for PLTL is known to be PSPACE-complete (see e.g. [21]) and traditional (graph-based) methods require exponential time, whereas our method requires double-exponential time (in the worst case). Hence our decision procedure is, in the worst case, suboptimal. However, we are convinced that a practical implementation that incorporates the simplifications explained in Section 5.9 may compete with traditional methods in several cases –e.g. when most of the formulas (in the context) are always-formulas– and even be faster in others, e.g. when satisfiability can be detected without constructing a graph. Of course, much more experimental work still needs to be done in order to precisely compare the performance of both decision procedures.

Another important point that can be addressed is the extension of these results to other temporal or, in general, modal logics. Actually, PLTL can be seen as the core logic of many logics. Hence the ideas introduced in this paper could serve as a basis for extensions to the first-order case (in spite of its incompleteness), or its complete fragments (e.g. the monodic fragment), or other kinds of extensions of PLTL such as the modal μ-calculus, and so on. The extension to the branching case has already been studied in [5] and [1]. In [5] the authors extend their cut-free sequent calculus to CTL. In [1] they present an analogue of the Schwendimann's tableau method for CTL.

Acknowledgements

The authors would like to thank two anonymous referees for their helpful suggestions that have contributed to improve the final version of this paper.

References

[1] P. Abate, R. Goré, F. Widman, One-pass tableaux for computation tree logic, in: N. Dershowitz, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15–19, 2007, Proceedings, Lecturer Notes in Computer Science, vol. 4790, Springer, 2007, pp. 32–47.
[2] B. Banieqbal, H. Barringer, Temporal logic with fixed points, in: B. Banieqbal, H. Barringer, A. Pnueli (Eds.), Temporal Logic in Specification, Altrincham, UK, April 8–10, 1987, Proceedings, Lecture Notes in Computer Science, vol. 398, Springer, 1987, pp. 62–74.
[3] A. Biere, A. Cimatti, E. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Highly Dependable Software, Advances in Computers, vol. 58, Academic Press, 2003.
[4] K. Brünnler, Deep sequent systems for modal logic, in: G. Governatori, I. Hodkinson, Y. Venema (Eds.), Advances in Modal Logic, Lecture Notes in Computer Science, vol. 6, College Publications, 2006, pp. 107–119.
[5] K. Brünnler, M. Lange, Cut-free sequent systems for temporal logic, J. Algebraic Logic Program 76 (2) (2008) 216–225.
[6] M. Fisher, A resolution method for temporal logic, in: IJCAI, 1991, pp. 99–104.
[7] D. Gabbay, A. Pnueli, S. Shelah, J. Stavi, On the temporal analysis of fairness, POPL '80: Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press, New York, NY, USA, 1980, pp. 163–173.
[8] J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, Systematic semantic tableaux for PLTL, in: E. Pimentel (Ed.), Proceedings of the Seventh Spanish Conference on Programming and Computer Languages (PROLE 2007), Selected Papers, Electronic Notes in Theoretical Computer Science, vol. 206, 2008, pp. 59–73.
[9] J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, F. Orejas, A cut-free and invariant-free sequent calculus for PLTL, in: J. Duparc, T.A. Henzinger (Eds.), Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11–15, 2007, Proceedings, Lecture Notes in Computer Science, vol. 4646, Springer, 2007, pp. 481–495.
[10] G. Gentzen, Untersuchungen über das Logische Schliessen, Mathematische Zeitschrift, 39, 176–210 and 405–431, 1934. English translation in [11], pp. 68–131.
[11] G. Gentzen, The collected papers of Gerhard Gentzen, in: M.E. Szabo (Ed.), Studies in Logic and the Foundations of Mathematics, North-Holland, 1969.
[12] R. Goré, Tableau Methods for Modal and Temporal Logics, Kluwer Academic Publishers, 1999.
[13] G.D. Gough, Decision Procedures for Temporal Logic. Master's thesis. Department of Computer Science, University of Manchester, England, 1984.
[14] J.A.W. Kamp, Tense Logic and the Theory of Linear Order. Ph.D. Thesis, University of California, Los Angeles, 1968.
[15] O. Lichtenstein, A. Pnueli, Propositional temporal logics: decidability and completeness, Logic J. IGPL 8 (1) (2000).
[16] B. Paech, Gentzen-systems for propositional temporal logics, in: CSL, 1988, pp. 240–253.
[17] R. Pliuskevicius, Investigation of finitary calculus for a discrete linear time logic by means of infinitary calculus, in: Baltic Computer Science, 1991, pp. 504–528.
[18] M. Reynolds, C. Dixon, Theorem-proving for discrete temporal logic, Handbook of Temporal Reasoning in Artificial Intelligence, Elsevier, 2005, pp. 279–314.
[19] K. Schutte, Schluweisen-kalkule der pradikatenlogik, Math. Ann. 122 (1950) 47–65.
[20] S. Schwendimann, A new one-pass tableau calculus for PLTL, in: Analytic Tableaux and Related Methods, Lecture Notes in Computer Science, vol. 1397, 1998, pp. 277–292.
[21] A.P. Sistla, E.M. Clarke, The complexity of propositional linear temporal logics, J. ACM 32 (3) (1985) 733–749.
[22] A. Szalas, Temporal logic of programs: a standard approach, Time and Logic. A Computational Approach, UCL Press Ltd., 1995, pp. 1–50.
[23] W. Tait, Normal derivability in classical logic, The Syntax and Semantics of Infinitary Languages, Springer Verlag, 1968, pp. 204–236.
[24] P. Wolper, Temporal logic can be more expressive, Inform. and Control 56 (1–2) (1983) 72–99.