Gaintzarain, J., Hermo, M., Lucio, P. et al. Invariant-Free Clausal Temporal Resolution. **J Autom Reasoning** 50, 1–49 (2013), This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s10817-011-9241-2

# Invariant-Free Clausal Temporal Resolution

**Jose Gaintzarain · Montserrat Hermo · Paqui Lucio · Marisa Navarro · Fernando Orejas**

Received: date / Accepted: date

**Abstract** Resolution is a well-known proof method for classical logics that is well suited for mechanization. The most fruitful approach in the literature on temporal logic, which was started with the seminal paper of M. Fisher, deals with Propositional Linear-time Temporal Logic (PLTL) and requires to generate invariants for performing resolution on eventualities. The methods and techniques developed in that approach have also been successfully adapted in order to obtain a clausal resolution method for Computation Tree Logic (CTL), but invariant handling seems to be a handicap for further extension to more general branching temporal logics. In this paper, we present a new approach to applying resolution to PLTL. The main novelty of our approach is that we do not generate invariants for performing resolution on eventualities. Hence, we say that the approach presented in this paper is invariant-free. Our method is based on the dual methods of tableaux and sequents for PLTL that we presented in a previous paper. Our resolution method involves translation into a clausal normal form that is a direct extension of classical CNF. We first show that any PLTL-formula can be transformed into this clausal normal form. Then, we present our temporal resolution method, called TRS-resolution, that extends classical propositional resolution. Finally, we

This work has been partially supported by the Spanish Project TIN2007-66523 and the Basque Project LoRea GIU07/35.

Jose Gaintzarain
The University of the Basque Country, 48012-Bilbao, Spain.
E-mail: jose.gaintzarain@ehu.es

Montserrat Hermo
The University of the Basque Country, 20080-San Sebastián, Spain.
E-mail: montserrat.hermo@ehu.es

Paqui Lucio
The University of the Basque Country, 20080-San Sebastián, Spain.
E-mail: paqui.lucio@ehu.es

Marisa Navarro
The University of the Basque Country, 20080-San Sebastián, Spain.
E-mail: marisa.navarro@ehu.es

Fernando Orejas
Technical University of Catalonia, 08034-Barcelona, Spain.
E-mail: orejas@lsi.upc.edu

prove that TRS-resolution is sound and complete. In fact, it finishes for any input formula deciding its satisfiability, hence it gives rise to a new decision procedure for PLTL.

**Keywords** Propositional Linear-time Temporal Logic · Resolution · Invariant-free · Clausal Normal Form

## 1 Introduction

Temporal logic 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 over time. In particular, several concepts which are useful for the specification of properties of dynamic systems –such as fairness, non-starvation, liveness, safety, mutual exclusion, etc– can be formally stated in temporal logic using very concise and readable formulas. Several different temporal logics have been devised –as formalisms for representing dynamic systems– that mainly differ in their underlying model of time and in their expressiveness. Regarding time modeling there are linear vs. branching, discrete vs. dense, future vs. past-and-future, finite vs infinite, etc. Regarding expressiveness, they involve different temporal connectives and logical constructions (such as, quantifiers, variables, fixpoint operators). Propositional Linear-time Temporal Logic (PLTL) is one of the most widely used temporal logics. This logic has, as the intended model for time, the standard model of natural numbers. Different contributions in the literature on temporal logic show its usefulness in computer science and other related areas. For a recent and extensive monograph on PLTL techniques and tools, we refer to [13], where the reader can find sample applications along with references to specific work that uses this temporal formalism to represent dynamic entities in a wide variety of fields. The minimal language for PLTL adds to classical propositional connectives two basic temporal connectives $\circ$ ("next") and $\mathcal{U}$ ("until") such that $\circ p$ is interpreted as "the next state makes p true" and $p\,\mathcal{U}\,q$ is interpreted as "p is true from now until q eventually becomes true". Many other useful temporal connectives can be defined as derived connectives, e.g. $\diamond$ ("eventually"), $\square$ ("always") and $\mathcal{R}$ ("release"). From the extensive literature on technical aspects of PLTL we mention here [15, 16, 29, 31] where more references can be found.

Automated reasoning for temporal logic is a quite recent trend. In temporal logics, as well as in the more general framework of modal logic, different proof methods are starting to be designed, implemented, compared, and improved. The interested reader is referred to [31] for a good survey about theorem-proving in PLTL and its extensions. The proof theory for temporal logics is mainly based on three kinds of proposals: automata, tableau and resolution. The most developed approach is model checking, which is automata-based. In fact, model checking of temporal formulas is traditionally carried out by a conversion to Büchi automata (see e.g. [35]), and there is a large body of research in this area. However, the automata approach is not well suited for automated deduction, in the sense that it cannot be used to generate proofs or deductions of a conclusion from a set of premises.

Automated reasoning for PLTL, and related logics, is mainly based on tableaux and resolution. Indeed, there is recently published work comparing implementations of the different tableau and resolution procedures for PLTL and similar logics (see e.g. [20, 26]).
The first tableau method for PLTL was introduced by P. Wolper in [37] and it is a *two-pass method*. In the first pass, it generates an auxiliary graph. This graph is checked and (possibly) pruned in a second phase that analyzes whether the so-called *eventualities* are fulfilled. An eventuality is a formula that asserts that something does eventually hold. For

example, to fulfill the formula $\diamond\varphi$ or the formula $\chi\,\mathcal{U}\,\varphi$ the formula $\varphi$ must eventually be satisfied. Hence, any path in the graph that includes $\diamond\varphi$ or $\chi\,\mathcal{U}\,\varphi$, but does not include $\varphi$, is pruned. At the end, an empty graph means unsatisfiability. Since Wolper's seminal paper [37], several authors (e.g. [24,4,29]) have proposed and studied tableau methods for different temporal and modal logics inspired by Wolper's tableau (see [22] for a good survey). In addition, Wolper's two-pass tableau has been used in the development of decision procedures or proof techniques for logics that extend PLTL to some decidable fragment of the first-order temporal logic (e.g.[28]), or to the branching case or with other features, such as agents, knowledge, etc (e.g. [21]). The first one-pass tableau method for PLTL was developed in [34] and it avoids the second pass by adding extra information to the nodes in the tableau. Some of this information must be synthesized bottom-up and it is needed because the fulfillment of an eventuality in a single branch depends on the other branches. Hence, it carries out an on-the-fly checking of the fulfillment of every eventuality in every branch. This on-the-fly tableau method has been successfully applied to other logics such as e.g. CTL ([3]) and PDL ([23]). Another one-pass tableau method was introduced in [17] (see also [19]) that is different from the two-pass tableau started by Wolper, and that is not based on an on-the-fly check of eventualities. Instead, in [17,19], there is a tableau rule that prevents from indefinitely delaying the satisfaction of eventualities. The TRS-resolution mechanism introduced in this paper is strongly based on the tableau method in [17,19]. In Section 9, we give more details on the relation between TRS-resolution and the TTM tableau method that is its forerunner.

In this paper, we deal with clausal resolution for PLTL. The method of resolution, invented by J.A. Robinson in 1965 ([32]), is an efficient refutation proof method that has provided the basis for several well-known theorem provers for classical logics. The earliest temporal resolution method [1] uses a non-clausal approach, hence a large number of rules are required for handling general formulas instead of clauses. There is also early work (e.g. [5,8]) related to clausal resolution for (less expressive) sublogics of PLTL. The language in [5] includes no eventualities, whereas in [8] the authors consider the strictly less expressive sublanguage of PLTL defined by using only $\circ$ and $\diamond$ as temporal connectives. The early clausal method presented in [36] considers full PLTL and uses a clausal form similar to ours, but completeness is only achieved in absence of eventualities (i.e. formulas of the form $\diamond\varphi$ or $\varphi\,\mathcal{U}\,\psi$). More recently, a fruitful trend of clausal temporal resolution methods, starting with the seminal paper of M. Fisher [12], achieves completeness for full PLTL by means of a specialized *temporal resolution* rule that needs to generate an invariant formula from a set of clauses that behaves as a loop. The methods and techniques developed in such an approach have been successfully adapted to Computation Tree Logic (CTL) (see [6]), but invariant handling seems to be a handicap for further extension to more general branching temporal logics such as Full Computation Tree Logic (CTL$^\star$). In Section 9 we compare our approach with the methods in [8,1,36,12].

In this paper, we introduce a new clausal resolution method that is sound and complete for full PLTL. Our method is based on the dual methods of tableaux and sequents for PLTL presented in [19]. On this basis we are able to perform clausal resolution in the presence of eventualities avoiding the requirement of invariant generation. We define a notion of *clausal normal form* and prove that every PLTL-formula can be translated into an equisatisfiable set of clauses. Our resolution mechanism explicitly simulates the transition from one world to the next one. Inside each world, we apply two kinds of rules: (1) the resolution and subsumption rules and (2) the fixpoint rules that split a clause with an eventuality atom into a finite number of new clauses. We prove that the method is sound and complete. In fact, it finishes for any set of clauses deciding its (un)satisfiability, hence it gives rise to a new decision procedure for PLTL.

*Outline of the paper*. In Section 2 we provide the basic background on PLTL. In Section 3 we introduce the syntactic notion of clause (Subsection 3.1), we show that any PLTL-formula can be transformed into a set of clauses (Subsection 3.2) and the complexity of this transformation (Subsection 3.3). In Section 4 we introduce the system TRS of inference rules in two subsections: the first one presents the basic rules and the second one presents the rule for solving eventualities in a way that prevents their indefinite delay. Then, in Section 5 we present the notion of TRS-derivation, provide some sample derivations and study the relationship between TRS-resolution and classical (propositional) resolution. The soundness of TRS is proved in Section 6. In Section 7 we propose an algorithm for systematically obtaining, for any set of clauses $\Gamma$, a finite derivation that proves that $\Gamma$ is either satisfiable or unsatisfiable. We also show some examples of application of the algorithm in Subsection 7.2. An important issue for this algorithm is to prove its termination for every input. This proof is presented in Subsection 7.3. In Subsection 7.4 we provide a bound of the worst-case complexity of the algorithm. In Section 8, we prove the completeness of TRS-resolution on the basis of the algorithm that outputs a derivation for every set of clauses. In Section 9 we discuss significant related work. Finally, we summarize our contribution and outline some topics for future research.

## 2 The Logic PLTL

A PLTL-formula is built using propositional variables (denoted by lowercase letters $p, q, \dots$) from a set Prop, the classical connectives $\neg$ and $\wedge$, and the temporal connectives $\circ$ and $\mathcal{U}$. A lowercase Greek letter ($\varphi, \psi, \chi, \gamma, \dots$) denotes a formula and an uppercase one ($\Phi, \Delta, \Gamma, \Psi, \Omega, \dots$) denotes a finite set of PLTL-formulas. As usual other connectives can be defined in terms of the previous ones: $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \mathcal{R} \psi \equiv \neg(\neg\varphi \mathcal{U} \neg\psi)$, $\diamond \varphi \equiv \neg\varphi \mathcal{U} \varphi$, $\square \varphi \equiv \neg\diamond\neg\varphi$. Note that $\square\varphi \equiv \neg\varphi \mathcal{R} \varphi$. In this paper, these derived connectives are technically useful for expressing the clausal form of formulas. In the sequel, a *formula* means a PLTL-formula and the following kind of formulas are significant.

**Definition 1** *We call* eventuality *to any formula of the form $\varphi \mathcal{U} \psi$ or $\diamond \varphi$. Eventualities of the form $\varphi \mathcal{U} \psi$ are also called* until-formulas*.*

We use two kinds of superscripts on unary connectives. First, a superscript $i$ varying on $\mathbb{N}$ represents the sequence consisting of $i$ identical connectives, in particular the empty sequence for $i = 0$. For instance, $\circ^i$ represents the sequence $\circ \dots \circ$ of length $i$. Second, the special case of superscript $b$ varying in $\{0, 1\}$ which allows to represent a formula with or without a prefixed connective. For instance, $\square^b\varphi$ is $\square\varphi$ whenever $b$ is $1$ and $\varphi$ whenever $b$ is $0$. Along the rest of the paper superscripts starting by $b$ (from bit) range in $\{0, 1\}$.

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 $s_0, s_1, s_2, \dots$ 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 formulas is given by the truth of a formula $\varphi$ in the state $s_j$ of a PLTL-structure $\mathcal{M}$, which is denoted by $\langle \mathcal{M}, s_j \rangle \models \varphi$. This semantics is inductively defined as follows:

- $\langle \mathcal{M}, s_j \rangle \models p$ 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 $i$ such that $j \leq i < k$ it holds $\langle \mathcal{M}, s_i \rangle \models \varphi$.

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

- $\langle \mathcal{M}, s_j \rangle \models \varphi \vee \psi$ 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 $i$ such that $j \leq i < k$
- $\langle \mathcal{M}, s_j \rangle \models \Diamond \varphi$ 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 $\Phi$, denoted $\mathcal{M} \models \Phi$, iff $\langle \mathcal{M}, s_0 \rangle \models \Phi$. A satisfiable set of formulas has at least one model, otherwise it is unsatisfiable. Two sets of formulas $\Phi$ and $\Psi$ are equisatisfiable whenever $\Phi$ is satisfiable iff $\Psi$ is satisfiable. The *logical consequence* relation between a set of formulas $\Phi$ and a formula $\chi$, denoted as $\Phi \models \chi$, is defined in the following way:

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

A logic is said to be compact when it verifies that, given any set of formulas $\Phi$, if every finite subset of $\Phi$ is satisfiable then $\Phi$ is satisfiable. It is well known that PLTL is a non-compact logic. For example, the infinite set of formulas $\{\circ^i p \mid i \in I\!N\} \cup \{\Diamond \neg p\}$ is not satisfiable but every finite subset of it is satisfiable. As a consequence, the completeness of our clausal resolution method is weak in the sense that it is restricted to finite sets of clauses. Therefore, along this paper, every set of formulas, in particular clauses, is assumed to be finite.

## 3 The Clausal Language

In this section we first define the conjunctive normal form of a formula. This is the basis for our notion of clause. In the second subsection we explain how to convert any formula into a set of clauses. Thirdly, we give the worst case complexity of the translation.

### 3.1 Conjunctive Normal Form for Formulas

Our notion of literal extends the classical notion of propositional literal. This extension introduces both temporal literals and (possibly empty) prefixed chains of the connective $\circ$ in front of temporal and propositional literals. That is, using the usual BNF-notation:

$$\begin{aligned} &P ::= p \mid \neg p \\ &T ::= P_1 \mathcal{U} P_2 \mid P_1 \mathcal{R} P_2 \mid \Diamond P \mid \Box P \\ &L ::= \circ^i P \mid \circ^i T \end{aligned}$$

where $p \in \mathsf{Prop}$ and $i \in I\!N$. $P$ stands for a propositional literal, $T$ for a (basic) temporal literal and $L$ for a literal. In the sequel, we use the term literal in the latter sense and only if needed we will specify whether a literal is propositional or temporal.[1] Sub- and superscripts are used when necessary.

We extend the classical notion of *the complement* $\widetilde{L}$ *of a literal* $L$ as follows:

$$\widetilde{p} = \neg p, \quad \widetilde{\neg p} = p, \quad \widetilde{\circ L} = \circ \widetilde{L}, \quad \widetilde{P_1 \mathcal{U} P_2} = \widetilde{P_1}\, \mathcal{R}\, \widetilde{P_2} \quad \text{and} \quad \widetilde{P_1 \mathcal{R} P_2} = \widetilde{P_1}\, \mathcal{U}\, \widetilde{P_2}$$

It is easy to see that $\widetilde{\Box P} = \Diamond \widetilde{P}$ and $\widetilde{\Diamond P} = \Box \widetilde{P}$. Although $\Diamond P$ and $\Box P$ can be respectively defined by $\widetilde{P}\, \mathcal{U}\, P$ and $\widetilde{P}\, \mathcal{R}\, P$, we have intentionally introduced $\Diamond P$ and $\Box P$ as temporal literals because of technical convenience.

A *now-clause* $N$ is a finite disjunction of literals (above denoted by $L$):

$N ::= \bot \mid L \vee N$

where $\bot$ represents the empty disjunction (or the empty now-clause). We identify finite disjunctions of literals with sets of literals. Hence, we assume that there are neither repetitions nor any established order in the literals of a clause. This assumption is especially advantageous for presenting the resolution rule, because it avoids factoring and ordering problems. However, for readability, we always write the disjunction symbol between the literals of a clause.

A clause is either a now-clause or a now-clause preceded by the connective $\Box$

$C ::= N \mid \Box N$

A clause of the form $\Box N$ is called an *always-clause*. Note that the formula $\Box^b \bot$ represents the two possible syntactic forms of the empty clause, as now- or always-clause.

For a clause $C = \Box^b(L_1 \vee \ldots \vee L_n)$ we denote by $\mathsf{Lit}(C)$ the set $\{L_1, \ldots, L_n\}$ and for a set of clauses $\Gamma$ we denote by $\mathsf{Lit}(\Gamma)$ the set $\bigcup_{C \in \Gamma} \mathsf{Lit}(C)$.

**Definition 2** *The set of all clauses in* $\Gamma$ *that contain the literal* $L$ *is denoted by* $\Gamma \restriction \{L\}$*, i.e.* $\Gamma \restriction \{L\} = \{C \in \Gamma \mid L \in \mathsf{Lit}(C)\}$.

Since $\circ$ distributes over disjunction, for a given now-clause $N = L_1 \vee \ldots \vee L_n$, we denote by $\circ N$ the now-clause $\circ L_1 \vee \ldots \vee \circ L_n$. We say that a clause $C$ is $\circ$-free if $\mathsf{Lit}(C)$ does not contain any literal of the form $\circ L$.

**Definition 3** *Given a set of clauses* $\Gamma$*, we define* $\mathsf{alw}(\Gamma) = \{\Box N \mid \Box N \in \Gamma\}$ *and* $\mathsf{now}(\Gamma) = \Gamma \setminus \mathsf{alw}(\Gamma)$.

Note that a formula of the form $\Box P$, can be understood as a now-clause consisting of one temporal literal or as an always-clause consisting of one propositional literal. If a set of clauses $\Gamma$ contains this kind of formulas, by convention those formulas are considered to be in $\mathsf{alw}(\Gamma)$.

**Definition 4** *For any set of clauses* $\Gamma$

*(a)* $\mathsf{drop}_\Box(\Gamma) = \mathsf{now}(\Gamma) \cup \{N \mid \Box N \in \mathsf{alw}(\Gamma)\}$.
*(b)* $\mathsf{BTL}(\Gamma) = \{T \mid T \vee N \in \mathsf{drop}_\Box(\Gamma)\}$.
*(c)* $\mathsf{unnext}(\Gamma) = \mathsf{alw}(\Gamma) \cup \{N \mid \Box^b(\circ N) \in \Gamma\}$.

The set $\mathsf{drop}_\Box(\Gamma)$ is formed by all the now-clauses in $\Gamma$ together with the inner now-clause of all the always-clauses in $\Gamma$.

$\mathsf{BTL}(\Gamma)$ is the set of all the (basic) temporal literals that occur in $\Gamma$. Hence, $\mathsf{BTL}(\Gamma)$ is a subset of $\mathsf{Lit}(\Gamma)$. It is worth to note that any literal in $\mathsf{Lit}(\Gamma)$ that does not belong to $\mathsf{BTL}(\Gamma)$

[1] Note that $\circ$ is the only temporal connective that does not occur in the so-called (basic) temporal literals.

is either a propositional literal $P$ or a literal of the form $\circ L$, according to the grammar at the beginning of this section. Note also that $\mathsf{unnext}$ implicitly uses the equivalence between $\Box N$ and $\{N, \Box \circ N\}$.

The set $\mathsf{unnext}(\Gamma)$ consists of all the clauses that should be satisfied at the next state of a state that satisfies $\Gamma$.

A formula is in *conjunctive normal form* whenever it is a conjunction of clauses. For simplicity, we identify a set of clauses with the conjunction of the clauses in it. Concretely, we identify any formula in conjunctive normal form

$$N_1 \wedge N_2 \wedge \ldots \wedge N_r \wedge \Box N_{r+1} \wedge \ldots \wedge \Box N_k$$

with the set of clauses

$$\{N_1, N_2, \ldots, N_r, \Box N_{r+1}, \ldots, \Box N_k\}$$

where each $N_i$ is a now-clause, $k \geq 1$ and $0 \leq r \leq k$.

### 3.2 Transforming Formulas into CNF

In this subsection we present a transformation $\mathsf{CNF}$ which maps any formula $\varphi$ to its *conjunctive normal form* $\mathsf{CNF}(\varphi)$. First, we show that any formula $\varphi$ can be transformed into another formula $\mathsf{NNF}(\varphi)$, called the *negation normal form* of $\varphi$, such that every connective $\neg$ is in front of a proposition. Second, we introduce an intermediate notion of normal form, called *distributed normal form*, denoted $\mathsf{DtNF}(\varphi)$ for input formula $\varphi$. The transformations $\mathsf{NNF}$ and $\mathsf{DtNF}$ preserve logical equivalence. Finally we present the transformation of any formula to its conjunctive normal form. The formulas $\varphi$ and $\mathsf{CNF}(\varphi)$ are equisatisfiable although, in general, they are not logically equivalent.

**Proposition 5** *For any formula $\varphi$ there exists a logically equivalent formula $\mathsf{NNF}(\varphi)$ such that $\chi \in \mathsf{Prop}$ for every subformula of $\mathsf{NNF}(\varphi)$ of the form $\neg\chi$.*

*Proof* $\mathsf{NNF}(\varphi)$ is obtained by repeatedly applying to any subformula of $\varphi$ the following reduction rules until no one can be applied

$$\begin{array}{ll}
\neg\neg\psi \stackrel{\mathsf{nnf}}{\longmapsto} \psi & \neg(\psi_1 \vee \psi_2) \stackrel{\mathsf{nnf}}{\longmapsto} \neg\psi_1 \wedge \neg\psi_2 \\
\neg\circ\psi \stackrel{\mathsf{nnf}}{\longmapsto} \circ\neg\psi & \neg(\psi_1 \wedge \psi_2) \stackrel{\mathsf{nnf}}{\longmapsto} \neg\psi_1 \vee \neg\psi_2 \\
\neg\Diamond\psi \stackrel{\mathsf{nnf}}{\longmapsto} \Box\neg\psi & \neg(\psi_1\, \mathcal{U}\, \psi_2) \stackrel{\mathsf{nnf}}{\longmapsto} \neg\psi_1\, \mathcal{R}\, \neg\psi_2 \\
\neg\Box\psi \stackrel{\mathsf{nnf}}{\longmapsto} \Diamond\neg\psi & \neg(\psi_1\, \mathcal{R}\, \psi_2) \stackrel{\mathsf{nnf}}{\longmapsto} \neg\psi_1\, \mathcal{U}\, \neg\psi_2
\end{array}$$

It is routine to see that the relation $\stackrel{\mathsf{nnf}}{\longmapsto}$ (defined above) preserves logical equivalence and the process of repeatedly applying the transformation $\stackrel{\mathsf{nnf}}{\longmapsto}$ stops after a finite number of steps. Therefore, $\varphi$ and $\mathsf{NNF}(\varphi)$ are logically equivalent. ∎

Now, in the distributed normal form, every connective $\neg$ is in front of a propositional variable, every connective $\vee$ is distributed over $\wedge$, temporal connectives that are distributive over $\vee$ and $\wedge$ are distributed, for formulas of the form $\varphi\,\mathcal{U}\,(\delta\,\mathcal{U}\,\psi)$ and of the form $\varphi\,\mathcal{R}\,(\delta\,\mathcal{R}\,\psi)$ the subformulas $\varphi$ and $\delta$ are different and non-empty sequences of the form $\Diamond \ldots \Diamond$ and of the form $\Box \ldots \Box$ are of length 1.

**Definition 6** *A formula is in* distributed normal form *if it has the form* $(\gamma_1^1 \vee \ldots \vee \gamma_1^{k_1}) \wedge \ldots \wedge (\gamma_n^1 \vee \ldots \vee \gamma_n^{k_n})$ *where each* $\gamma_g^j$ *denotes a formula of one of the following forms*

- $\circ^i P$
- $\circ^i(\alpha \mathcal{R} \beta)$ *for some* $\alpha$ *and* $\beta \neq \alpha \mathcal{R} \psi$ *for any* $\psi$
- $\circ^i(\beta \mathcal{U} \alpha)$ *for some* $\beta$ *and* $\alpha \neq \beta \mathcal{U} \psi$ *for any* $\psi$
- $\circ^i \Box \beta$ *for some* $\beta \neq \Box \psi$ *for any* $\psi$
- $\circ^i \Diamond \alpha$ *for some* $\alpha \neq \Diamond \psi$ *for any* $\psi$

*where* $\alpha$ *and* $\beta$ *denote two special cases of distributed normal form. Concretely,* $\beta$ *stands for a formula of the form* $(\gamma_1^1 \vee \ldots \vee \gamma_1^{k_1})$ *with* $k_1 \geq 1$ *and* $\alpha$ *stands for either a formula* $\gamma_1^1$ *or a formula* $(\gamma_1^1 \vee \ldots \vee \gamma_1^{k_1}) \wedge \ldots \wedge (\gamma_n^1 \vee \ldots \vee \gamma_n^{k_n})$ *with* $n \geq 2$ *and* $k_h \geq 1$ *for every* $h \in \{1, \ldots, n\}$.

Note that if a formula is in distributed normal form then it is also in negation normal form.

**Proposition 7** *For any formula* $\varphi$ *there exists a logically equivalent formula* $\mathsf{DtNF}(\varphi)$ *such that* $\mathsf{DtNF}(\varphi)$ *is in distributed normal form.*

*Proof* First, we transform $\varphi$ into $\mathsf{NNF}(\varphi)$ and then we repeatedly apply to $\mathsf{NNF}(\varphi)$ the following reduction rules

$$(\varphi_1 \wedge \varphi_2) \vee \psi \xmapsto{\mathsf{dtnf}} (\varphi_1 \vee \psi) \wedge (\varphi_2 \vee \psi) \qquad \psi \vee (\varphi_1 \wedge \varphi_2) \xmapsto{\mathsf{dtnf}} (\psi \vee \varphi_1) \wedge (\psi \vee \varphi_2)$$

$$\circ(\varphi_1 \vee \varphi_2) \xmapsto{\mathsf{dtnf}} \circ\varphi_1 \vee \circ\varphi_2 \qquad \circ(\varphi_1 \wedge \varphi_2) \xmapsto{\mathsf{dtnf}} \circ\varphi_1 \wedge \circ\varphi_2$$

$$\psi \mathcal{U} (\varphi_1 \vee \varphi_2) \xmapsto{\mathsf{dtnf}} (\psi \mathcal{U} \varphi_1) \vee (\psi \mathcal{U} \varphi_2) \qquad \psi \mathcal{R} (\varphi_1 \wedge \varphi_2) \xmapsto{\mathsf{dtnf}} (\psi \mathcal{R} \varphi_1) \wedge (\psi \mathcal{R} \varphi_2)$$

$$(\varphi_1 \wedge \varphi_2) \mathcal{U} \psi \xmapsto{\mathsf{dtnf}} (\varphi_1 \mathcal{U} \psi) \wedge (\varphi_2 \mathcal{U} \psi) \qquad (\varphi_1 \vee \varphi_2) \mathcal{R} \psi \xmapsto{\mathsf{dtnf}} (\varphi_1 \mathcal{R} \psi) \vee (\varphi_2 \mathcal{R} \psi)$$

$$\Diamond (\varphi_1 \vee \varphi_2) \xmapsto{\mathsf{dtnf}} \Diamond \varphi_1 \vee \Diamond \varphi_2 \qquad \Box (\varphi_1 \wedge \varphi_2) \xmapsto{\mathsf{dtnf}} \Box \varphi_1 \wedge \Box \varphi_2$$

$$\psi_1 \mathcal{U} (\psi_1 \mathcal{U} \psi_2) \xmapsto{\mathsf{dtnf}} \psi_1 \mathcal{U} \psi_2 \qquad \psi_1 \mathcal{R} (\psi_1 \mathcal{R} \psi_2) \xmapsto{\mathsf{dtnf}} \psi_1 \mathcal{R} \psi_2$$

$$\Diamond \Diamond \psi \xmapsto{\mathsf{dtnf}} \Diamond \psi \qquad \Box \Box \psi \xmapsto{\mathsf{dtnf}} \Box \psi$$

It is routine to see that this reduction always terminates giving a formula in distributed normal form. Additionally, it is proved that every $\xmapsto{\mathsf{dtnf}}$-rule preserves logical equivalence. For that, the only non-trivial $\xmapsto{\mathsf{dtnf}}$-rules are the ones for transforming $\psi \mathcal{U} (\varphi_1 \vee \varphi_2)$, $(\varphi_1 \wedge \varphi_2) \mathcal{U} \psi$, $\psi \mathcal{R} (\varphi_1 \wedge \varphi_2)$, and $(\varphi_1 \vee \varphi_2) \mathcal{R} \psi$. Here, we give the proof details for the first one. The remaining three are similar.

Suppose that $\langle \mathcal{M}, s_j \rangle \models \psi \mathcal{U} (\varphi_1 \vee \varphi_2)$. Then, there exists $k \geq j$ such that $\langle \mathcal{M}, s_k \rangle \models \varphi_1 \vee \varphi_2$ and $\langle \mathcal{M}, s_i \rangle \models \psi$ for every $i$ such that $j \leq i < k$. Hence, for such $k$, either $\langle \mathcal{M}, s_k \rangle \models \varphi_1$ or $\langle \mathcal{M}, s_k \rangle \models \varphi_2$. In the former case, $\langle \mathcal{M}, s_j \rangle \models \psi \mathcal{U} \varphi_1$, whereas in the latter $\langle \mathcal{M}, s_j \rangle \models \psi \mathcal{U} \varphi_2$. Therefore $\langle \mathcal{M}, s_j \rangle \models (\psi \mathcal{U} \varphi_1) \vee (\psi \mathcal{U} \varphi_2)$.

Conversely, if $\langle \mathcal{M}, s_j \rangle \models (\psi \mathcal{U} \varphi_1) \vee (\psi \mathcal{U} \varphi_2)$, then either $\langle \mathcal{M}, s_j \rangle \models (\psi \mathcal{U} \varphi_1)$ or $\langle \mathcal{M}, s_j \rangle \models (\psi \mathcal{U} \varphi_2)$. Hence, there exists $k \geq j$ such that $\langle \mathcal{M}, s_i \rangle \models \psi$ for all $i$ such that $j \leq i < k$ and $\langle \mathcal{M}, s_k \rangle \models \varphi_1$ or $\langle \mathcal{M}, s_k \rangle \models \varphi_2$. Then, $\langle \mathcal{M}, s_k \rangle \models \varphi_1 \vee \varphi_2$ and $\langle \mathcal{M}, s_i \rangle \models \psi$ for every $i$ such that $j \leq i < k$. Therefore, $\langle \mathcal{M}, s_j \rangle \models \psi \mathcal{U} (\varphi_1 \vee \varphi_2)$. ∎

As the following theorem shows, we will use the distributed normal form as a preliminary step for transforming a formula into its conjunctive normal form.

**Theorem 8** *For any formula $\varphi$ there exists an equisatisfiable formula $\mathsf{CNF}(\varphi)$ such that $\mathsf{CNF}(\varphi)$ is in conjunctive normal form.*

*Proof* First, we transform $\varphi$ into $\mathsf{DtNF}(\varphi)$. Second, we repeatedly apply the following rules until no one can be applied. In the rules bellow $\psi$ is the whole formula (in distributed normal form) and the expressions of the form $\psi[\alpha \Rightarrow \beta]$ denote the formula obtained by simultaneously replacing all the occurrences of the subformula $\alpha$ in $\psi$ by the formula $\beta$, where $\alpha$ is any non-literal subformula of any conjunct of $\psi$ that is not a clause yet.

$$\psi \xmapsto{\mathsf{cnf}} \psi[\circ^i(\varphi_1 \mathcal{U} \varphi_2) \Rightarrow \circ^i(p_1 \mathcal{U} p_2)] \;\wedge\; \mathsf{CNF}(\Box(\neg p_1 \vee \varphi_1)) \;\wedge\; \mathsf{CNF}(\Box(\neg p_2 \vee \varphi_2))$$

$$\psi \xmapsto{\mathsf{cnf}} \psi[\circ^i(\varphi_1 \mathcal{R} \varphi_2) \Rightarrow \circ^i(p_1 \mathcal{R} p_2)] \;\wedge\; \mathsf{CNF}(\Box(\neg p_1 \vee \varphi_1)) \;\wedge\; \mathsf{CNF}(\Box(\neg p_2 \vee \varphi_2))$$

$$\psi \xmapsto{\mathsf{cnf}} \psi[\circ^i \Box \gamma \Rightarrow \circ^i \Box p\,] \;\wedge\; \mathsf{CNF}(\Box(\neg p \vee \gamma))$$

$$\psi \xmapsto{\mathsf{cnf}} \psi[\circ^i \Diamond \gamma \Rightarrow \circ^i \Diamond p\,] \;\wedge\; \mathsf{CNF}(\Box(\neg p \vee \gamma))$$

$$\psi \xmapsto{\mathsf{cnf}} \psi[\Box(\gamma \vee \Box \chi) \Rightarrow \Box(\gamma \vee \Box p)] \;\wedge\; \mathsf{CNF}(\Box(\neg p \vee \chi))$$

$$\psi \xmapsto{\mathsf{cnf}} \psi[\Box(\Box \chi \vee \gamma) \Rightarrow \Box(\Box p \vee \gamma)] \;\wedge\; \mathsf{CNF}(\Box(\neg p \vee \chi))$$

where $p$, $p_1$ and $p_2$ are fresh new propositional variables and the formula $\chi$ is not a propositional literal. Note that the new conjunctions of the form $\mathsf{CNF}(\Box(\neg\psi_1 \vee \psi_2))$ serve to define the fresh new symbols $\psi_1$. We will prove that the transformation from $\varphi$ to $\mathsf{CNF}(\varphi)$ stops after a finite number of steps and both formulas are equisatisfiable.

On one hand, each application of a $\xmapsto{\mathsf{cnf}}$-rule reduces the depth of (at least) one non-literal subformula of a formula in $\mathsf{DtNF}$-form. Additionally, the number of fresh new variables is bounded by the number of subformulas. These two facts ensure termination.

On the other hand we prove, by structural induction, that the formulas in both sides of each $\xmapsto{\mathsf{cnf}}$-rule are equisatisfiable. Here we only show the details for the first rule above (the remaining rules are similar or particular cases). Suppose that $\langle \mathcal{M}, s_j \rangle \models \psi$ where $\psi$ is in distributed normal form and $\circ^i(\varphi_1 \mathcal{U} \varphi_2)$ is a non-literal subformula of any conjunct of $\psi$ that is not a clause yet. Then, since $p_1$ and $p_2$ are fresh, $p_1, p_2 \notin V_{\mathcal{M}}(s_k)$ for all $k$. Therefore, we define $\mathcal{M}'$ to be the extension of $\mathcal{M}$ such that $p_h \in V_{\mathcal{M}'}(s'_k)$ iff $\langle \mathcal{M}, s_k \rangle \models \varphi_h$ for all $k$ and $h \in \{1, 2\}$. As a consequence, for all $k$, $\langle \mathcal{M}, s_k \rangle \models \circ^i(\varphi_1 \mathcal{U} \varphi_2)$ iff $\langle \mathcal{M}', s'_k \rangle \models \circ^i(p_1 \mathcal{U} p_2)$ and $\langle \mathcal{M}', s'_k \rangle \models \Box(\neg p_1 \vee \varphi_1) \wedge \Box(\neg p_2 \vee \varphi_2)$. Hence,

$$\langle \mathcal{M}', s'_k \rangle \models \psi[\circ^i(\varphi_1 \mathcal{U} \varphi_2) \Rightarrow \circ^i(p_1 \mathcal{U} p_2)] \wedge \Box(\neg p_1 \vee \varphi_1) \wedge \Box(\neg p_2 \vee \varphi_2).$$

By the induction hypothesis, the transformation of $\Box(\neg p_1 \vee \varphi_1)$ and $\Box(\neg p_2 \vee \varphi_2)$ to conjunctive normal form preserves equisatisfiability.

Conversely, consider any model $\mathcal{M}$ of the right-hand part of the first $\xmapsto{\mathsf{cnf}}$-rule. If $\langle \mathcal{M}, s_0 \rangle \not\models \circ^i(p_1 \mathcal{U} p_2)$, then $\langle \mathcal{M}, s_0 \rangle$ must satisfy some other disjunct in every conjunct of $\psi$ where $\circ^i(p_1 \mathcal{U} p_2)$ occurs in. Therefore $\mathcal{M}$ is also a model of $\psi$. If $\langle \mathcal{M}, s_0 \rangle \models \circ^i(p_1 \mathcal{U} p_2)$, then there exists a $j \geq i$ such that $\langle \mathcal{M}, s_j \rangle \models p_2$ and $\langle \mathcal{M}, s_k \rangle \models p_1$ for all $k$ such that $i \leq k < j$. Additionally, for all $k$, $\langle \mathcal{M}, s_k \rangle \models \Box(\neg p_h \vee \varphi_h)$ for $h \in \{1, 2\}$. Therefore, $\langle \mathcal{M}, s_j \rangle \models \varphi_2$ and $\langle \mathcal{M}, s_k \rangle \models \varphi_1$ for all $k$ such that $i \leq k < j$. Hence, $\langle \mathcal{M}, s_0 \rangle \models \circ^i(\varphi_1 \mathcal{U} \varphi_2)$, which means that $\mathcal{M}$ must be a model of $\psi$.

∎

**Example 9** Let us consider the following formula $\varphi = \neg(p \wedge r \wedge \Box(\neg(p \wedge r) \vee \circ(p \wedge r)))$ Note that $\varphi$ is equivalent to $\neg\Box(p \wedge r)$ by means of induction on time. First, we transform $\varphi$ into

$$\mathsf{NNF}(\varphi) = \neg p \vee \neg r \vee \Diamond(p \wedge r \wedge \circ(\neg p \vee \neg r))$$

Then, its distributed normal form is

$$\mathsf{DtNF}(\varphi) = \neg p \vee \neg r \vee \Diamond(p \wedge r \wedge (\circ\neg p \vee \circ\neg r))$$

Finally, the conjunctive (or clausal) normal form of $\varphi$ is

$$\begin{aligned}\mathsf{CNF}(\varphi) &= (\neg p \vee \neg r \vee \Diamond a) \wedge \mathsf{CNF}(\Box(\neg a \vee (p \wedge r \wedge (\circ\neg p \vee \circ\neg r)))) = \\ &= (\neg p \vee \neg r \vee \Diamond a) \wedge \Box(\neg a \vee p) \wedge \Box(\neg a \vee r) \wedge \Box(\neg a \vee \circ\neg p \vee \circ\neg r)\end{aligned}$$

where a new propositional variable $a \in \mathsf{Prop}$ has been introduced and new clauses that define the variable $a$ have been added. The formula $\mathsf{CNF}(\varphi)$ can also be understood as the set of clauses $\{(\neg p \vee \neg r \vee \Diamond a), \Box(\neg a \vee p), \Box(\neg a \vee r), \Box(\neg a \vee \circ\neg p \vee \circ\neg r)\}$.

### 3.3 Complexity of the Translation

In this subsection we show that the worst case of the translation to $\mathsf{CNF}$ is bounded by an exponential on the size of the input formula.

**Definition 10** *Given a formula $\varphi$, we define the size of $\varphi$, namely $\mathsf{size}(\varphi)$, as the number of connectives $\mathsf{cnt}(\varphi)$ plus the number of propositional variables, $\mathsf{pv}(\varphi)$ in $\varphi$.*

**Proposition 11** *For any formula $\varphi$, $\mathsf{size}(\mathsf{CNF}(\varphi)) \in 2^{\mathcal{O}(\mathsf{size}(\varphi))}$.*

*Proof* The complexity of the first transformation from $\varphi$ to $\mathsf{NNF}(\varphi)$ is linear because the worst case is when the connective $\neg$ appears only once and it occurs as the outermost connective, i.e. $\varphi$ is of the form $\neg\psi$ for some formula $\psi$. In such a case $\neg$ will end up appearing in front of every propositional variable. Hence, $\mathsf{size}(\mathsf{NNF}(\varphi)) = \mathsf{cnt}(\varphi) - 1 + 2 \times \mathsf{pv}(\varphi)$ which is smaller or equal than $2 \times \mathsf{size}(\varphi)$.
In the second transformation to $\mathsf{DtNF}(\varphi)$, each use of the distribution laws can almost double the size of the initial formula. So, we only can ensure that $\mathsf{size}(\mathsf{DtNF}(\varphi)) \leq 2^{\mathsf{size}(\mathsf{NNF}(\varphi))}$ or equivalently that $\mathsf{size}(\mathsf{DtNF}(\varphi)) \in \mathcal{O}(2^{\mathsf{size}(\varphi)})$.
Finally, the last transformation to $\mathsf{CNF}(\varphi)$ has again linear complexity. This is basically because –in the rules of Theorem 8– each new variable replaces a subformula of a formula $\psi$ that is already in $\mathsf{DtNF}$ form.
Summarizing, $\mathsf{size}(\mathsf{CNF}(\varphi)) \in \mathcal{O}(2^{\mathcal{O}(\mathsf{size}(\varphi))}) = 2^{\mathcal{O}(\mathsf{size}(\varphi))}$. ∎

We would like to remark that the exponential blow-up is only due –as in classical cnf– to the distribution laws and it can be prevented using fresh variables as it is made in the so-called *definitional cnf* (see [11]). Therefore, as in classical cnf, for practical purposes, we could use new variables to achieve a transformation to clausal form of linear complexity.

## 4 The Temporal Resolution Rules

In this section, we present the rules of our temporal resolution system. In addition to a resolution-like rule $(Res)$, this system includes a subsumption rule $(Sbm)$ and also the three so-called fixpoint rules –$(\mathcal{R}\,Fix)$, $(\mathcal{U}\,Fix)$ and $(\mathcal{U}\,Set)$– for decomposing temporal literals. The rule $(Sbm)$ is a natural extension of (traditional) clausal subsumption. The rules $(\mathcal{R}\,Fix)$ and $(\mathcal{U}\,Fix)$ are based on the usual inductive definition of the connectives $\mathcal{R}$ and $\mathcal{U}$, respectively, whereas $(\mathcal{U}\,Set)$ is based on a more complex inductive definition of $\mathcal{U}$ that is the basis of our approach. Therefore, this section is split into two subsections. The first subsection is devoted to the first four rules which we call *Basic Rules*. The details about the rule $(\mathcal{U}\,Set)$ are explained in the second subsection. The corresponding derived rules for $\Box$ and $\Diamond$ are showed in both subsections. In the sequel, the rules explained in this section are called TRS-rules and the system is called TRS.

$$(Res)\quad \frac{\Box^{\,b}(L \vee N) \qquad\qquad \Box^{\,b'}(\widetilde{L} \vee N')}{\Box^{\,b\times b'}(N \vee N')}$$

**Fig. 1** The Resolution Rule

### 4.1 Basic Rules

Considering that $\Gamma$ is the current set of clauses, the resolution rule $(Res)$ in Fig. 1 is applied to two clauses (the premises) in $\Gamma$ and obtains a new clause (the resolvent). The rule $(Res)$ is a very natural generalization of classical resolution for always-clauses, and it is written in the usual format of premises and resolvent separated by a horizontal line. $(Res)$ applies to two clauses (the premises) that contain two complementary literals. Both premises can be headed or not by an always connective (depending on superscripts $b$ and $b'$ whose range is $\{0, 1\}$). By means of the product $b \times b'$ in the superscript of the resolvent, only when both premises are always-clauses, the resolvent is also an always-clause. In particular, when $N$ and $N'$ are both $\bot$, the resolvent is $\Box^{\,b\times b'}\bot$, i.e. either $\Box\bot$ or $\bot$. The resolvent is added to $\Gamma$ while the premises remain in $\Gamma$. That is, each application of the rule $(Res)$ adds a clause to the current set of clauses. On the contrary, the remaining TRS-rules replace a set of clauses $\Sigma \subseteq \Gamma$ with another set of clauses, namely $\Psi$. We write them as transformation rules $\Sigma \mapsto \Psi$. The sets $\Sigma$ and $\Psi$ are respectively called the antecedent and the consequent and they are in general equisatisfiable but in some cases logically equivalent. So that, each application of these transformation rules removes the clauses in $\Sigma$ from the current set of clauses and adds the clauses in $\Psi$. The first transformation rule is the subsumption rule $(Sbm)$ in Fig. 2,

$$(Sbm)\quad \{\Box^{\,b}N, \Box^{\,b}N'\} \longmapsto \{\Box^{\,b}N'\} \quad \text{if } N' \subseteq N$$

**Fig. 2** The Subsumption Rule

which generalizes classical subsumption to always-clauses.[2] This rule is applied to any set that contains both $\Box^b N$ and $\Box^b N'$ to eliminate the former while $\Box^b N'$ remains.

$$(\mathcal{R}\,Fix)\ \ \{\Box^b((P_1\,\mathcal{R}\,P_2)\vee N)\}\longmapsto\{\Box^b(P_2\vee N),\ \ \Box^b(P_1\vee\circ(P_1\,\mathcal{R}\,P_2)\vee N)\}$$

$$(\mathcal{U}\,Fix)\ \ \{\Box^b((P_1\,\mathcal{U}\,P_2)\vee N)\}\longmapsto\{\Box^b(P_2\vee P_1\vee N),\ \ \Box^b(P_2\vee\circ(P_1\,\mathcal{U}\,P_2)\vee N)\}$$

**Fig. 3** The Fixpoint Rules $(\mathcal{R}\,Fix)$ and $(\mathcal{U}\,Fix)$

The fixpoint rules $(\mathcal{R}\,Fix)$ and $(\mathcal{U}\,Fix)$ in Fig. 3 serve to replace a clause of the form $\Box^b(T\vee N)$ with a logically equivalent set of clauses. The rule $(\mathcal{R}\,Fix)$ splits the temporal literal $P_1\,\mathcal{R}\,P_2$ by using the well-known inductive definition of the connective $\mathcal{R}$: $P_1\,\mathcal{R}\,P_2 \equiv P_2\wedge(P_1\vee\circ(P_1\,\mathcal{R}\,P_2))$. Likewise, the rule $(\mathcal{U}\,Fix)$ uses the inductive definition of the connective $\mathcal{U}$: $P_1\,\mathcal{U}\,P_2 \equiv P_2\vee(P_1\wedge\circ(P_1\,\mathcal{U}\,P_2))$. In both cases, a simple distribution gives the equivalent set of two clauses that is shown in the consequent of each rule. In order to illustrate this point let us consider the case of the connective $\mathcal{U}$. By the inductive definition of $\mathcal{U}$ and distributivity of $\vee$ over $\wedge$,

$$P_1\,\mathcal{U}\,P_2\equiv P_2\vee(P_1\wedge\circ(P_1\,\mathcal{U}\,P_2))\equiv(P_2\vee P_1)\wedge(P_2\vee\circ(P_1\,\mathcal{U}\,P_2)).$$

Hence, $\Box^b((P_1\,\mathcal{U}\,P_2)\vee N)$ is logically equivalent to the conjunction of the two clauses $\Box^b(P_2\vee P_1\vee N)$ and $\Box^b(P_2\vee\circ(P_1\,\mathcal{U}\,P_2)\vee N)$. So that, the antecedent of the rule $(\mathcal{U}\,Fix)$ is logically equivalent to the conjunction of the two clauses in the consequent. Since the

$$(\Box Fix)\ \{\Box^b(\Box\,P\vee N)\}\longmapsto\{\Box^b(P\vee N),\ \ \Box^b(\circ\Box\,P\vee N)\}$$

$$(\Diamond Fix)\ \{\Box^b(\Diamond\,P\vee N)\}\longmapsto\{\Box^b(P\vee\circ\Diamond\,P\vee N)\}$$

**Fig. 4** The Fixpoint Rules $(\Box\,Fix)$ and $(\Diamond\,Fix)$

connectives $\Box$ and $\Diamond$ can be seen as particular cases of $\mathcal{R}$ and $\mathcal{U}$ respectively, the rules in Fig. 4 constitute the corresponding specializations of the rules in Fig. 3.

### 4.2 The Rule $(\mathcal{U}\,Set)$

The construction of the consequent of the rule $(\mathcal{U}\,Set)$ in Fig. 5 takes into account, not only a (non-empty) set whose clauses include a temporal atom $P_1\,\mathcal{U}\,P_2$, but also the remaining clauses. Consequently, the antecedent of the rule $(\mathcal{U}\,Set)$ is

$$\Gamma\ \equiv\ \Phi\cup\{\Box^{b_i}((P_1\,\mathcal{U}\,P_2)\vee N_i)\mid 1\leq i\leq n\} \tag{1}$$

where $n\geq 1$ and $\Phi$ stands for the set consisting of all the remaining clauses in the set to which $(\mathcal{U}\,Set)$ is applied. It is worth to note that the literal $P_1\,\mathcal{U}\,P_2$ can also occur in $\Phi$.[3]

[2] Note that the same superscript $b$ occurs in both clauses.

[3] The opposite restriction is not required for soundness. However, for achieving completeness the rule $(\mathcal{U}\,Set)$ is applied over a partition of the current set of clauses into a set formed by all the clauses that include $P_1\,\mathcal{U}\,P_2$ and the remaining clauses.

$$
\begin{array}{ll}
(\mathcal{U}\,Set) & \Phi \;\cup\; \{\Box^{\,b_i}((P_1\,\mathcal{U}\,P_2) \vee N_i) \mid 1 \le i \le n\} \\
& \longmapsto \Phi \cup \{P_2 \vee P_1 \vee N_i,\; P_2 \vee \circ(a\,\mathcal{U}\,P_2) \vee N_i \;\mid\; 1 \le i \le n\} \\
& \quad \cup\, \mathsf{CNF}(\mathsf{def}(a, P_1, \Delta)) \\
& \quad \cup\, \{\Box\,(\circ(P_1\,\mathcal{U}\,P_2) \vee \circ N_i) \;\mid\; b_i = 1 \text{ and } 1 \le i \le n\}
\end{array}
$$

where $n \geq 1$
$\Delta = \mathsf{now}(\Phi)$
$a \in \mathsf{Prop}$ is fresh
$\mathsf{def}(a, P_1, \Delta) = \Box\,(\neg a \vee (P_1 \wedge \neg\Delta))$ if $\Delta \neq \emptyset$
$\mathsf{def}(a, P_1, \Delta) = \Box\,\neg a$ if $\Delta = \emptyset$

**Fig. 5** The Rule $(\mathcal{U}\,Set)$

**Example 12** Let us apply the rule $(\mathcal{U}\,Set)$ to the eventuality $r\,\mathcal{U}\,s$ in the set of clauses

$$\{p, \circ q, \Box\,u, \Box\,((r\,\mathcal{U}\,s) \vee (\circ t))\}.$$

Then $\Phi = \{p, \circ q, \Box\,u\}$ and $\Delta = \mathsf{now}(\Phi) = \{p, \circ q\}$, where $\mathsf{now}$ is the operator on sets of clauses introduced in Definition 3. Therefore, the consequent of this $(\mathcal{U}\,Set)$ application is

$$
\begin{array}{l}
\{p, \circ q, \Box\,u\} \cup \{s \vee r \vee \circ t, s \vee \circ(a\,\mathcal{U}\,s) \vee \circ t\} \\
\qquad\qquad\quad \cup \{\Box\,(\neg a \vee r), \Box\,(\neg a \vee \neg p \vee \circ\neg q)\} \\
\qquad\qquad\quad \cup \{\Box\,((\circ(r\,\mathcal{U}\,s)) \vee (\circ\circ t))\}
\end{array}
$$

where $a$ is the fresh variable and $\mathsf{def}(a, r, \Delta) = \{\Box\,(\neg a \vee r), \Box\,(\neg a \vee \neg p \vee \circ\neg q)\}$. Below we justify the construction of $\Delta = \mathsf{now}(\Phi)$ for excluding always-clauses from the definition of the fresh variable $a$. We call $\Delta$ the context. Let us give a clue on context handling through this example. If we used the whole set $\Phi$ instead of $\Delta$ in the definition of $a$, then the second clause in $\mathsf{def}(a, r, \Phi)$ would be $\Box\,(\neg a \vee \neg p \vee \circ\neg q \vee \Diamond\,\neg u)$. However, since $\Box\,u$ is in $\Phi$, the clause $\Box\,u$ also belongs to the consequent. Therefore, the disjunct $\Diamond\,\neg u$ of the above clause, would never be satisfied.

Next, we explain the intuition behind the rule $(\mathcal{U}\,Set)$ and introduce the definition of *context*. First, it is easy to see that the above set $\Gamma$ (see (1)) and the following set $\Gamma_1$ are equisatisfiable.

$$
\begin{array}{rl}
\Gamma_1 \;\equiv & \Phi \cup \{(P_1\,\mathcal{U}\,P_2) \vee N_i \mid 1 \le i \le n\} \\
& \quad \cup \{\Box^{\,b_i}(\circ(P_1\,\mathcal{U}\,P_2) \vee \circ N_i) \mid b_i = 1 \text{ and } 1 \le i \le n\}
\end{array}
$$

Second, as explained for the rule $(\mathcal{U}\,Fix)$, the set $\Gamma_1$ is equisatisfiable to the set

$$
\begin{array}{rl}
\Gamma_2 \;\equiv & \Phi \cup \{P_2 \vee P_1 \vee N_i,\;\; P_2 \vee \circ(P_1\,\mathcal{U}\,P_2) \vee N_i \;\mid\; 1 \le i \le n\} \\
& \quad \cup \{\Box^{\,b_i}(\circ(P_1\,\mathcal{U}\,P_2) \vee \circ N_i) \mid b_i = 1 \text{ and } 1 \le i \le n\}
\end{array}
$$

Now, the crucial idea is that $\Gamma_2$ is also equisatisfiable to the following set[4]

$$
\begin{array}{rl}
\Gamma_3 \;\equiv & \Phi \cup \{P_2 \vee P_1 \vee N_i,\;\; P_2 \vee \circ((P_1 \wedge \neg\Phi)\,\mathcal{U}\,P_2) \vee N_i \;\mid\; 1 \le i \le n\} \\
& \quad \cup \{\Box^{\,b_i}(\circ(P_1\,\mathcal{U}\,P_2) \vee \circ N_i) \mid b_i = 1 \text{ and } 1 \le i \le n\}
\end{array}
$$

To see that $\Gamma_2$ and $\Gamma_3$ are equisatisfiable, suppose that the set $\Gamma_2$ has a model $\mathcal{M}$ such that $\langle \mathcal{M}, s_0\rangle \models \Phi \cup \{P_1, \neg P_2, \circ(P_1\,\mathcal{U}\,P_2)\}$ and $\langle \mathcal{M}, s_1\rangle \not\models P_2$. Then, $P_2$ should be satisfied

[4] where $\neg\Phi$ stands for the disjunction of the negation of all the formulas in $\Phi$. Hence, $\Gamma_3$ is not necessarily formed by clauses.

in a later state $s_j$ with $j > 1$ and $P_1$ is true in all the states $s_h$ such that $1 \leq h < j$. Moreover, if $\Phi$ is satisfied in a state $s_k$ with $k \in \{0, \ldots, j-1\}$ and $\Phi$ is not satisfied in the states $s_{k+1}, \ldots, s_{j-1}$, then we can construct a model $\mathcal{M}'$ of $\Gamma_2$ by simply deleting the states $s_0, \ldots, s_{k-1}$ in $\mathcal{M}$. Note that at least $s_0$ satisfies $\Phi$ and also that, in particular, $k$ could be $j-1$, which means that the sequence $s_{k+1}, \ldots, s_{j-1}$ is empty and the model $\mathcal{M}'$ starts in $s_{j-1}$. This $\mathcal{M}'$ is a model of $\circ((P_1 \wedge \neg\Phi)\,\mathcal{U}\,P_2)$. In the converse direction, any model of $\circ((P_1 \wedge \neg\Phi)\,\mathcal{U}\,P_2)$ is itself a model of $\circ(P_1\,\mathcal{U}\,P_2)$. So $\Gamma_2$ and $\Gamma_3$ are equisatisfiable.
Finally, the always-clauses in $\Phi$ can be excluded from the negation of $\Phi$ since, in general, the two sets $\{\Box\,\psi, \circ((\gamma \wedge (\varphi \vee \neg\Box\,\psi))\,\mathcal{U}\,\delta)\}$ and $\{\Box\,\psi, \circ((\gamma \wedge \varphi)\,\mathcal{U}\,\delta)\}$ are logically equivalent. This fact motivates the following notion of context.

**Definition 13** *In an application of the rule* $(\mathcal{U}\,Set)$ *(see Fig. 5) to an antecedent that is partitioned in the two sets* $\Phi$ *and* $\{\Box^{\,b_i}((P_1\,\mathcal{U}\,P_2) \vee N_i) \mid 1 \leq i \leq n\}$ *we say that* $\Delta = \mathsf{now}(\Phi)$ *is the* context. [5]

Then, $\Gamma_3$ is logically equivalent to

$$\begin{aligned}\Gamma_4 \;\equiv\; \Phi \cup \{&P_2 \vee P_1 \vee N_i,\;\; P_2 \vee \circ((P_1 \wedge \neg\Delta)\,\mathcal{U}\,P_2) \vee N_i \;\mid\; 1 \leq i \leq n\} \\ \cup\; \{&\Box^{\,b_i}(\circ(P_1\,\mathcal{U}\,P_2) \vee \circ N_i) \mid b_i = 1 \text{ and } 1 \leq i \leq n\}\end{aligned}$$

Since $\circ((P_1 \wedge \neg\Delta)\,\mathcal{U}\,P_2)$ is not a literal, the rule $(\mathcal{U}\,Set)$ introduces a fresh propositional variable $a$ that replaces the formula $P_1 \wedge \neg\Delta$, hence the definition of $a$ should be given by the cnf-form of the formula $\Box\,(a \leftrightarrow (P_1 \wedge \neg\Delta))$. However, since the left-to-right implication is enough for equisatisfiability, we do not add the clauses for the reverse implication, using only the transformation to cnf-form of the formula $\Box\,(\neg a \vee (P_1 \wedge \neg\Delta))$. The correctness of the rule $(\mathcal{U}\,Set)$ is shown in detail in the proof of Proposition 28.

The rule $(\mathcal{U}\,Set)$ leads to a complete resolution method –that does not require invariant generation– mainly due to the above explained management of the so-called contexts (in the rule $(\mathcal{U}\,Set)$) that prevents from postponing indefinitely the satisfaction of $P_1\,\mathcal{U}\,P_2$. Example 17 in Section 5 illustrates how contexts are handled to cause inconsistency whenever the fulfillment of an eventuality could be infinitely delayed. There is a finite number of possible different contexts and the repetition of a previous context, while postponing an eventuality, also causes inconsistency. Therefore, there is a clear strategy to achieve termination and completeness.

$$\begin{array}{ll}(\Diamond\,Set) & \Phi \;\cup\; \{\Box^{\,b_i}(\Diamond\,P \vee N_i) \mid 1 \leq i \leq n\} \\ & \longmapsto \Phi \cup \{P \vee \circ(a\,\mathcal{U}\,P) \vee N_i \;\mid\; 1 \leq i \leq n\} \\ & \qquad \cup\; \mathsf{CNF}(\mathsf{def}(a, \Delta)) \\ & \qquad \cup\; \{\Box\,(\circ\Diamond\,P \vee \circ N_i) \;\mid\; b_i = 1 \text{ and } 1 \leq i \leq n\} \\ & \text{where } n \geq 1 \\ & \qquad \Delta = \mathsf{now}(\Phi) \\ & \qquad a \in \mathsf{Prop} \text{ is fresh} \\ & \qquad \mathsf{def}(a, \Delta) = \Box\,(\neg a \vee \neg\Delta) \text{ if } \Delta \neq \emptyset \\ & \qquad \mathsf{def}(a, \Delta) = \Box\,\neg a \text{ if } \Delta = \emptyset\end{array}$$

**Fig. 6** The Rule $(\Diamond\,Set)$

The rule $(\Diamond\,Set)$ in Fig. 6 is the specialization of $(\mathcal{U}\,Set)$ that corresponds to the equivalence of $\Diamond\,P \equiv \widetilde{P}\,\mathcal{U}\,P$. Consequently, along the rest of the paper, the rule $(\Diamond\,Set)$ is treated

[5] The operator now was introduced in Definition 3.

as a derived rule, in the sense that most technical details are given only for the general rule $(\mathcal{U}\,Set)$.

## 5 Temporal Resolution Derivations

A classical resolution derivation for a set of propositional clauses $\Gamma$ is a sequence of sets of clauses

$$\Gamma_0 \mapsto \Gamma_1 \mapsto \ldots \mapsto \Gamma_k$$

where $\Gamma = \Gamma_0$ and each $\Gamma_{i+1}$ is obtained from $\Gamma_i$ by means of a resolution-step that consists in applying the (classical) resolution rule. The sequence ends when either $\Gamma_k$ contains $\perp$ or every application of the resolution rule on formulas in $\Gamma_k$ yields a formula that is already in $\Gamma_k$. For classical propositional logic, resolution is sound, refutationally complete and, even, complete. Soundness and refutational completeness mean that the method obtains a set $\Gamma_k$ that contains $\perp$ for some $k \in I\!N$ if and only if $\Gamma$ is unsatisfiable. Moreover, in classical propositional resolution the sequence obtained is always finite (if the pairs of clauses for applying the resolution rule are selected fairly) and consequently classical propositional resolution is also complete and serves as a decision procedure.

In this section we first extend the classical notion of derivation –to the temporal case of PLTL– introducing TRS-derivations. We also provide some sample TRS-derivations. The notion of TRS-derivation is the basis of the sound, refutationally complete, and complete resolution mechanism that is presented in this paper. In the second subsection we prove technical results on the relationship between TRS-resolution and classical (propositional) resolution.

### 5.1 TRS-Derivations and Examples

Our notion of derivation explicitly simulates the transition from one state to the next one, in the sense that whenever in the current set of clauses no more resolution resolvents can be added, then we use the operator unnext (see Definition 4) to get the clauses that must be satisfied in the state that follows (is next to) the current one. Inside each state, the TRS-rules are applied, hence the so-called local derivations are (roughly speaking) an extension of classical derivations.

**Definition 14** *A* TRS-derivation *for a set of clauses $\Gamma$ is a sequence*

$$\mathcal{D} = \Gamma_0^0 \mapsto \Gamma_0^1 \mapsto \ldots \mapsto \Gamma_0^{h_0} \Mapsto \Gamma_1^0 \mapsto \Gamma_1^1 \mapsto \ldots \mapsto \Gamma_1^{h_1} \Mapsto \ldots \Mapsto \Gamma_i^0 \mapsto \Gamma_i^1 \mapsto \ldots$$

*where*

*(a)* $\Gamma_0^0 = \Gamma$
*(b)* $\mapsto$ *represents the application of a* TRS*-rule*
*(c)* $\Mapsto$ *represents the application of the* unnext *operator*

*If any set $\Gamma_i^j$ in $\mathcal{D}$ contains $\Box^b\perp$, then $\mathcal{D}$ is called a* refutation *for $\Gamma$. We say that a* TRS*-derivation is a* local derivation *if it does not contain any application of the* unnext *operator. A local derivation is called a* local refutation *if it is a refutation.*

Note that we use two different symbols ($\mapsto$ and $\Mapsto$) to highlight the difference between the application of a TRS-rule and the application of the unnext operator. The former applications produce sets $\Gamma_i^{j+1}$ from $\Gamma_i^j$ and are called TRS-*steps*. The latter applications yield $\Gamma_{i+1}^0$ from $\Gamma_i^{h_i}$ and are called *unnext-steps*.

In the sequel we only use the prefix TRS- whenever confusion might result, otherwise we simply say derivation.

Now we give four examples of refutations. For readability, the derivations are represented as vertical sequences of rule applications with the name of the applied rule at the right-hand side of each step. In addition, the formulas to which each rule affects have been underlined. The first example shows that in some cases, even if temporal literals are involved, the refutation is achieved using only the resolution rule $(Res)$ and the unnext operator. The second example illustrates that sometimes the rule $(\mathcal{U}\,Set)$ is not necessary and the rule $(\mathcal{U}\,Fix)$ is enough. The third example shows how contexts are handled to cause inconsistency whenever the fulfillment of an eventuality could be infinitely delayed. Finally, in the fourth example, the rule $(\mathcal{U}\,Set)$ is applied to a proper subset of the set of clauses that contain the literal $p\,\mathcal{U}\,q$. In general, it can be applied to any non-empty subset.

**Example 15**

$$
\begin{array}{cl}
\Gamma_0^0 = \{\Box\,(r \vee \Diamond\,p), \Box\circ\neg r, \circ\Box\,\neg p, \Box\,(\circ r \vee \neg q \vee \Diamond\,p), p \vee q, \neg q\} & \\ \hline
\Gamma_1^0 = \{\underline{\Box\,(r \vee \Diamond\,p)}, \Box\circ\neg r, \underline{\neg r}, \Box\,\neg p, \Box\,(\circ r \vee \neg q \vee \Diamond\,p)\} & (\mathsf{unnext}) \\ \hline
\Gamma_1^1 = \{\Box\,(r \vee \Diamond\,p), \Box\circ\neg r, \neg r, \underline{\Box\,\neg p}, \Box\,(\circ r \vee \neg q \vee \Diamond\,p), \underline{\Diamond\,p}\} & (Res) \\ \hline
\Gamma_1^2 = \{\Box\,(r \vee \Diamond\,p), \Diamond\,p, \Box\circ\neg r, \neg r, \Box\,\neg p, \Box\,(\circ r \vee \neg q \vee \Diamond\,p), \bot\} & (Res)
\end{array}
$$

It is worth to remark that in the TRS-step that yields $\Gamma_1^2$ from $\Gamma_1^1$ the formula $\Box\,\neg p$ is treated as a now-clause formed by a temporal literal.

**Example 16**

$$
\begin{array}{cl}
\Gamma_0^0 = \{\Box\,\neg p, \underline{\Box\,(r\,\mathcal{U}\ p)}, (\neg r)\,\mathcal{U}\ p\} & \\ \hline
\Gamma_0^1 = \{\Box\,\neg p, \underline{(\neg r)\,\mathcal{U}\ p}, \Box\,(p \vee r), \Box\,(p \vee \circ(r\,\mathcal{U}\ p))\} & (\mathcal{U}\,Fix) \\ \hline
\Gamma_0^2 = \{\underline{\Box\,\neg p}, \underline{\Box\,(p \vee r)}, \Box\,(p \vee \circ(r\,\mathcal{U}\ p)), p \vee \neg r, p \vee \circ((\neg r)\,\mathcal{U}\ p)\} & (\mathcal{U}\,Fix) \\ \hline
\Gamma_0^3 = \{\underline{\Box\,\neg p}, \Box\,(p \vee r), \Box\,(p \vee \circ(r\,\mathcal{U}\ p)), \underline{p \vee \neg r}, p \vee \circ((\neg r)\,\mathcal{U}\ p), \Box\,r\} & (Res) \\ \hline
\Gamma_0^4 = \{\Box\,\neg p, \Box\,(p \vee r), \Box\,(p \vee \circ(r\,\mathcal{U}\ p)), p \vee \neg r, p \vee \circ((\neg r)\,\mathcal{U}\ p), \underline{\Box\,r}, \underline{\neg r}\} & (Res) \\ \hline
\Gamma_0^5 = \{\Box\,\neg p, \Box\,(p \vee r), \Box\,(p \vee \circ(r\,\mathcal{U}\ p)), p \vee \neg r, p \vee \circ((\neg r)\,\mathcal{U}\ p), \Box\,r, \neg r, \bot\} & (Res)
\end{array}
$$

In this example the formulas $\Box\,\neg p$ and $\Box\,r$ are treated as always-clauses formed by one propositional literal.

**Example 17** Let $\Gamma_0^0 = \{\Box\,(\neg p \vee \circ p), p, x\,\mathcal{U}\,\neg p\}$. Then, by applying $(\mathcal{U}\,Set)$ to $x\,\mathcal{U}\,\neg p$ in $\Gamma_0^0$ where $\Phi = \{\Box\,(\neg p \vee \circ p), p\}$ and $\Delta = \{p\}$,
$\Gamma_0^1 = \{\Box\,(\neg p \vee \circ p), p, \neg p \vee x, \neg p \vee \circ(a\,\mathcal{U}\,\neg p), \Box\,(\neg a \vee \neg p), \Box\,(\neg a \vee x)\}$ where $a$ is the fresh variable whose meaning is defined to be $x \wedge \neg p$ by the last two clauses. Note that $\neg p$ is $\neg\Delta$. Then, by four applications of the rule $(Res)$ that respectively resolve the singleton clause $p$ with the four occurrences of $\neg p$,
$\Gamma_0^5 = \{\Box\,(\neg p \vee \circ p), \circ p, x, p, \neg p \vee x, \neg p \vee \circ(a\,\mathcal{U}\,\neg p), \circ(a\,\mathcal{U}\,\neg p), \neg a, \Box\,(\neg a \vee \neg p), \Box\,(\neg a \vee x)\}$.
Now, the operator unnext produces $\Gamma_1^0 = \{\Box\,(\neg p \vee \circ p), p, a\,\mathcal{U}\,\neg p, \Box\,(\neg a \vee \neg p), \Box\,(\neg a \vee x)\}$.
Hence, the application of $(\mathcal{U}\,Set)$ to $a\,\mathcal{U}\,\neg p$ in $\Gamma_1^0$ where $\Phi = \{\Box\,(\neg p \vee \circ p), p, \Box\,(\neg a \vee$

$\neg p), \Box(\neg a \vee x)\}$ and $\Delta = \{p\}$ yields
$\Gamma_1^1 = \{\Box(\neg p \vee \circ p), p, \neg p \vee a, \neg p \vee \circ(b\,\mathcal{U}\,\neg p), \Box(\neg b \vee \neg p), \Box(\neg b \vee a), \Box(\neg a \vee \neg p), \Box(\neg a \vee x)\}$
where the fresh variable $b$ is defined as $a \wedge \neg p$ by the clauses $\Box(\neg b \vee \neg p), \Box(\neg b \vee a)$. Then, the application of $(Res)$ to $p$ and $\neg p \vee a$ yields $a$. Finally, the resolution of $p$ and $\Box(\neg a \vee \neg p)$ yields $\neg a$. Hence, the empty clause is immediately obtained from $a$ and $\neg a$.
Roughly speaking, $a$ holds whenever the satisfaction of $\neg p$ (or equivalently the fullfilment of $x\,\mathcal{U}\,\neg p$) is postponed. However, $a$ means $x \wedge \neg p$, where $\neg p$ is the negated context. So that, the part of the definition of $a$ given by the clause $\Box(\neg a \vee \neg p)$ allows the inference of $\neg a$, which leads to the inconsistency.

**Example 18**

$$\begin{array}{cl}
\Gamma_0^0 = \{\Box((p\,\mathcal{U}\,q) \vee r), \underline{\Box((p\,\mathcal{U}\,q) \vee \Diamond s)}, \Box\neg q, \underline{\Box\neg s}\} & \\ \hline
\Gamma_0^1 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \Box\neg q, \Box\neg s, \underline{(p\,\mathcal{U}\,q)}\} & (Res) \\ \hline
\Gamma_0^3 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \underline{\Box\neg q}, \Box\neg s, q \vee p, \underline{q \vee \circ(a\,\mathcal{U}\,q)}, \Box\neg a\} & (\mathcal{U}\,Set) \\ \hline
\Gamma_0^4 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \Box\neg q, \Box\neg s, q \vee p, q \vee \circ(a\,\mathcal{U}\,q), \Box\neg a, \circ(a\,\mathcal{U}\,q)\} & (Res) \\ \hline
\Gamma_1^0 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \Box\neg q, \Box\neg s, \Box\neg a, \underline{a\,\mathcal{U}\,q}\} & (\mathsf{unnext}) \\ \hline
\Gamma_1^1 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \Box\neg q, \Box\neg s, \underline{\Box\neg a}, \underline{q \vee a}, q \vee \circ(b\,\mathcal{U}\,q), \Box\neg b\} & (\mathcal{U}\,Set) \\ \hline
\Gamma_1^2 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \underline{\Box\neg q}, \Box\neg s, \Box\neg a, q \vee a, q \vee \circ(b\,\mathcal{U}\,q), \Box\neg b, \underline{q}\} & (Res) \\ \hline
\Gamma_1^3 = \{\Box((p\,\mathcal{U}\,q) \vee r), \Box((p\,\mathcal{U}\,q) \vee \Diamond s), \Box\neg q, \Box\neg s, \Box\neg a, q \vee a, q \vee \circ(b\,\mathcal{U}\,q), \Box\neg b, q, \bot\} & (Res)
\end{array}$$

Note that the formula $\Box\neg s$ is treated as a literal in $\Gamma_0^0$ and as an always-clause in $\Gamma_0^1$. Besides, it is worth to note that in $\Gamma_0^1$ there are three occurrences of $p\,\mathcal{U}\,q$, but the rule $(\mathcal{U}\,Set)$ is applied by considering the set $\Phi$ to be formed by the first four clauses.

## 5.2 Relating TRS-Resolution to Classical Resolution

In this subsection we define the notion of linear local derivation and, based on it, we establish a relation between TRS-resolution and classical resolution that enables us to use well-known results from classical propositional logic.

**Definition 19** *A set of clauses $\Gamma$ is* closed *with respect to* TRS*-rules (shortly,* TRS*-closed) iff it satisfies the following three conditions:*

*(a)* $\mathsf{BTL}(\Gamma) = \emptyset$ *(i.e. any literal in $\Gamma$ is either propositional ($p$ or $\neg p$) or starts by $\circ$)*[6]
*(b) The subsumption rule $(Sbm)$ cannot be applied to $\Gamma$*
*(c) Every clause obtained from $\Gamma$ by application of the resolution rule $(Res)$ is already in $\Gamma$ or it is subsumed by some clause in $\Gamma$.*

**Definition 20** *Let $\Gamma$ be a set of clauses, we denote by $\Gamma^*$ any set such that there exists a local derivation $\Gamma \mapsto \ldots \mapsto \Gamma^*$ and either $\Box^b\bot \in \Gamma^*$ or $\Gamma^*$ is* TRS*-closed. Additionally, the non-deterministic operation that yields $\Gamma^*$ from $\Gamma$ is denoted by* close.

**Definition 21** *A set of clauses $\Gamma$ is* locally inconsistent *iff there exists a local refutation for $\Gamma$. Otherwise it is* locally consistent.

**Proposition 22** *For any* TRS*-closed set of clauses $\Gamma$, if $\Box^b\bot \notin \Gamma$ then $\Gamma$ is locally consistent.*

[6] see Subsection 3.1.

*Proof* If $\Gamma$ is TRS-closed, every clause that can be obtained by means of the rule $(Res)$ is already in $\Gamma$ or is subsumed by some other clause in $\Gamma$. If $\Box^b\bot$ is not in $\Gamma$ then there is no way to obtain it by means of a local derivation. ∎

The following notion is an adaptation of the concept of *linear resolution based on a clause* (see e.g. Section 2.6 in [33]).

**Definition 23** *A local derivation $\mathcal{D}$ for $\Gamma$ is* linear *with respect to a clause $C \in \Gamma$ iff it satisfies the following three conditions*

*(a) Every* TRS*-step in $\mathcal{D}$ is an application of the rule $(Res)$*
*(b) $C$ is one of the premises for $(Res)$ in the first* TRS*-step*
*(c) For every* TRS*-step in $\mathcal{D}$, except for the first one, one of the premises is the resolvent obtained in the previous* TRS*-step.*

Next, we formulate a useful relationship between TRS-resolution and classical propositional resolution.

**Definition 24** *Let $\Gamma$ be a set of clauses,* $\mathsf{prop}(\Gamma)$ *is the set that results from* $\mathsf{drop}_\Box(\Gamma)$ *by replacing all the occurrences of each non-propositional literal $L \in \mathsf{Lit}(\mathsf{drop}_\Box(\Gamma))$ with a fresh propositional literal in a coherent way, in the sense that complementary literals are replaced with complementary propositional literals.*

**Proposition 25** *Let $\Gamma$ be a set of clauses such that* $\mathsf{BTL}(\Gamma) = \emptyset$.

*(i)* $\mathsf{drop}_\Box(\Gamma)$ *is locally inconsistent iff* $\mathsf{prop}(\Gamma)$ *is inconsistent (in classical logic).*
*(ii) $\Gamma$ is locally inconsistent iff* $\mathsf{drop}_\Box(\Gamma)$ *is locally inconsistent.*

*Proof* (i) For the left to right implication, since $\mathsf{BTL}(\Gamma) = \emptyset$, if $\mathsf{drop}_\Box(\Gamma)$ is locally inconsistent then there exists a local refutation for $\mathsf{drop}_\Box(\Gamma)$ where every TRS-step is an application of the rule $(Res)$ or the rule $(Sbm)$. Hence, we can trivially build a classical refutation for $\mathsf{prop}(\Gamma)$ with the same number of steps and using classical resolution and subsumption instead of $(Res)$ and $(Sbm)$, respectively.
Conversely, if $\mathsf{prop}(\Gamma)$ is inconsistent then by completeness of classical propositional resolution there exists a refutation for $\mathsf{prop}(\Gamma)$ where only the classical resolution rule is used. Then, it is easy to obtain a local refutation for $\mathsf{drop}_\Box(\Gamma)$ applying the resolution rule $(Res)$ to the corresponding clauses.

(ii) Since $\mathsf{BTL}(\Gamma) = \emptyset$, if $\Gamma$ is locally inconsistent then there exists a local refutation $\mathcal{D}$ for $\Gamma$ where every TRS-step is an application of the rule $(Res)$ or the rule $(Sbm)$. From $\mathcal{D}$ we can build a local refutation for $\mathsf{drop}_\Box(\Gamma)$ in a trivial manner, by using a clause $N$ whenever the original derivation $\mathcal{D}$ uses the corresponding $\Box N$.
If $\mathsf{drop}_\Box(\Gamma)$ is locally inconsistent then, by (i) and the completeness of classical propositional resolution, there exists a refutation $\mathcal{D}$ for $\mathsf{prop}(\Gamma)$ where every TRS-step is an application of the classical resolution rule. From $\mathcal{D}$, it is straightforward to obtain a local refutation $\mathcal{D}'$ for $\mathsf{drop}_\Box(\Gamma)$ where every TRS-step is an application of the rule $(Res)$. This local refutation is trivially convertible into a local refutation for $\Gamma$, by using the clause $\Box N \in \Gamma$ instead of $N \in \mathsf{drop}_\Box(\Gamma)$ whenever $N \notin \Gamma$. ∎

Next, we provide a basic result that is used in Section 8 for proving completeness. This result is an adaptation of the completeness of classical linear resolution based on a clause (see Section 2.6 in [33]) that states

> Given a consistent set of propositional clauses $\Phi$, if for a propositional clause $\beta \notin \Phi$ the set $\Phi \cup \{\beta\}$ is inconsistent then there exists a refutation for $\Phi \cup \{\beta\}$ that is linear with respect to the clause $\beta$.

**Proposition 26** *Let $\Gamma$ be a locally consistent set of clauses such that* $\mathsf{BTL}(\Gamma) = \emptyset$ *and let $C$ be a clause that is not in $\Gamma$ such that* $\mathsf{BTL}(\{C\}) = \emptyset$*. If $\Gamma \cup \{C\}$ is locally inconsistent then there exists a local refutation for $\Gamma \cup \{C\}$ that is linear with respect to the clause $C$.*

*Proof* If $\Gamma \cup \{C\}$ is locally inconsistent, by Proposition 25 the set $\mathsf{prop}(\Gamma \cup \{C\})$ is inconsistent and, by completeness of classical linear resolution based on a clause (see above), there exists a refutation $\mathcal{D}'$ for $\mathsf{prop}(\Gamma \cup \{C\})$ that is linear with respect to the clause $C' \in \mathsf{prop}(\Gamma \cup \{C\})$ that corresponds to the clause $C$. From $\mathcal{D}'$, it is trivial to build a local refutation $\mathcal{D}$ for $\Gamma \cup \{C\}$ that is linear with respect to $C$. ■

## 6 Soundness

A resolution system is *sound* if, whenever a refutation exists for a set of clauses $\Gamma$, then $\Gamma$ is unsatisfiable. The soundness of a system can be guaranteed rule by rule, where a rule is sound whenever it preserves the satisfiability. Often some rules preserve stronger properties than satisfiability. In this section, we analize each rule from the point of view of soundness and stronger properties and prove that the resolution system TRS is sound.

**Proposition 27** *The Basic Rules of Subsection 4.1 are sound. Moreover, every application of these rules yields a new set of clauses that is logically equivalent to the initial set.*

*Proof* When $(Res)$ is applied to two clauses (the premises) $\Box^b(L \vee N)$ and $\Box^{b'}(\widetilde{L} \vee N')$ in $\Gamma$, the resolvent $\Box^{b \times b'}(N \vee N')$ is a logical consequence of $\{\Box^b(L \vee N), \Box^{b'}(\widetilde{L} \vee N')\}$ and, consequently, the new set of clauses $\Gamma' = \Gamma \cup \{\Box^{b \times b'}(N \vee N')\}$ is logically equivalent to the set of clauses $\Gamma$.

For soundness of $(Sbm)$, suppose that $\Box^b N$ and $\Box^b N'$ are in $\Gamma$ and that $N' \subsetneq N$. It is trivial that any model of $\Gamma$ is also a model of $\Gamma \setminus \{\Box^b N\}$ and vice-versa.

Given a set of clauses $\Gamma$, the rule $(\mathcal{U}\,Fix)$ replaces a clause $\Box^b((P_1\,\mathcal{U}\,P_2) \vee N) \in \Gamma$ with two clauses $\Box^b(P_2 \vee P_1 \vee N)$ and $\Box^b(P_2 \vee \circ(P_1\,\mathcal{U}\,P_2) \vee N)$ obtaining a new set $\Gamma'$ = $(\Gamma \setminus \{\Box^b((P_1\,\mathcal{U}\,P_2) \vee N)\}) \cup \{\Box^b(P_2 \vee P_1 \vee N), \Box^b(P_2 \vee \circ(P_1\,\mathcal{U}\,P_2) \vee N)\}$. The two sets, $\Gamma$ and $\Gamma'$, are logically equivalent since the clause that contains the literal of the form $P_1\,\mathcal{U}\,P_2$ is replaced with the clauses obtained by taking into account the inductive definition of the connective $\mathcal{U}$. Similarly, the rule $(\mathcal{R}\,Fix)$ replaces a clause $\Box^b((P_1\,\mathcal{R}\,P_2) \vee N) \in \Gamma$ with two clauses $\Box^b(P_2 \vee N)$ and $\Box^b(P_1 \vee \circ(P_1\,\mathcal{R}\,P_2) \vee N)$ obtaining a new set $\Gamma'$ = $(\Gamma \setminus \{\Box^b((P_1\,\mathcal{R}\,P_2) \vee N)\}) \cup \{\Box^b(P_2 \vee N), \Box^b(P_1 \vee \circ(P_1\,\mathcal{R}\,P_2) \vee N)\}$. The sets $\Gamma$ and $\Gamma'$ are logically equivalent because the clause that contains the literal of the form $P_1\,\mathcal{R}\,P_2$ is substituted by the clauses obtained by using the inductive definition of the connective $\mathcal{R}$. In particular, every application of the rules $(\Box\,Fix)$ and $(\Diamond\,Fix)$ yields a new set of clauses that is logically equivalent to the initial set. Therefore, they are also sound. ■

**Proposition 28** *The rule $(\mathcal{U}\,Set)$ is sound. Moreover, the initial and the target sets of every application of $(\mathcal{U}\,Set)$ are equisatisfiable.*

*Proof* When the rule $(\mathcal{U}\,Set)$ is applied to a set of clauses $\Gamma$, a non-empty subset $\{\Box^{b_i}(P_1\,\mathcal{U}\,P_2 \vee N_i) \mid 1 \leq i \leq n\}$ is replaced with a set of clauses

$$\begin{aligned}\Psi = \{&P_2 \vee P_1 \vee N_i,\ P_2 \vee \circ(a\,\mathcal{U}\,P_2) \vee N_i \ \mid\ 1 \leq i \leq n\}\\ &\cup \mathsf{CNF}(\mathsf{def}(a, P_1, \Delta))\\ &\cup \{\Box\,(\circ(P_1\,\mathcal{U}\,P_2) \vee \circ N_i) \ \mid\ b_i = 1 \text{ and } 1 \leq i \leq n\}\end{aligned}$$

where $\Delta = \mathsf{now}(\Gamma \setminus \{\Box^{b_i}(P_1 \mathcal{U} P_2 \vee N_i) \mid 1 \leq i \leq n\})$, $a \in \mathsf{Prop}$ is fresh, $\mathsf{def}(a, P_1, \Delta) = \Box(\neg a \vee (P_1 \wedge \neg \Delta))$ if $\Delta \neq \emptyset$ and $\mathsf{def}(a, P_1, \Delta) = \Box \neg a$ if $\Delta = \emptyset$. So the new set $\Gamma'$ is

$$(\Gamma \setminus \{\Box^{b_i}(P_1 \mathcal{U} P_2 \vee N_i) \mid 1 \leq i \leq n\}) \cup \Psi.$$

It is easy to see that if $\Gamma'$ is satisfiable then $\Gamma$ is satisfiable. Note that $a$ does not appear in $\Gamma$ and formulas of the form $\Box \varphi$ and $(\varphi_1 \wedge \varphi_2) \mathcal{U} \psi$ are equivalent to the sets of formulas $\{\varphi, \Box \circ \varphi\}$ and $\{\varphi_1 \mathcal{U} \psi, \varphi_2 \mathcal{U} \psi\}$, respectively.

We will show the converse implication. Let $\langle \mathcal{M}, s_0 \rangle \models \Gamma$, since $a$ does not appear in the $N_i$'s, we build a model of $\Gamma'$ in the following two cases. First, consider that $\langle \mathcal{M}, s_0 \rangle \models N_i$ for all $i \in \{1, \ldots, n\}$. Then we can define a model $\mathcal{M}'$ for $\Gamma'$ as follows

- $a \notin V_{\mathcal{M}'}(s'_j)$ for every $j \in \mathbb{N}$
- $p \in V_{\mathcal{M}'}(s'_j)$ iff $p \in V_{\mathcal{M}}(s_j)$ for all $j \in \mathbb{N}$ and all $p \in \mathsf{Prop}$ such that $p \neq a$

Second, if $\langle \mathcal{M}, s_0 \rangle \not\models N_i$ for some $i \in \{1, \ldots, n\}$, then it should be that $\langle \mathcal{M}, s_0 \rangle \models P_1 \mathcal{U} P_2$. Let $x$ be the least $z \geq 0$ such that $\langle \mathcal{M}, s_z \rangle \models P_2$. If $x = 0$ then, since $a$ does not appear in $P_2$, a model $\mathcal{M}'$ of $\Gamma'$ can be built just as above. If $x > 0$, let $y$ be the greatest $z$ such that $0 \leq z < x$ and

$$\langle \mathcal{M}, s_z \rangle \models \mathsf{now}(\Gamma \setminus \{\Box^{b_i}(P_1 \mathcal{U} P_2 \vee N_i) \mid 1 \leq i \leq n\}) \cup \{P_1 \mathcal{U} P_2\}.$$

Note that at least $z = 0$ must satisfy the above set of clauses. As a consequence of the choice of $x$ and $y$, it holds that

$$\langle \mathcal{M}, s_y \rangle \models \{P_1, \neg P_2, \circ((P_1 \wedge \neg \mathsf{now}(\Gamma \setminus \{\Box^{b_i}(P_1 \mathcal{U} P_2 \vee N_i) \mid 1 \leq i \leq n\})) \mathcal{U} P_2)\}.$$

Besides, $\langle \mathcal{M}, s_y \rangle \models \mathsf{now}(\Gamma \setminus \{\Box^{b_i}(P_1 \mathcal{U} P_2 \vee N_i) \mid 1 \leq i \leq n\})$. So that, we can define a model $\mathcal{M}'$ for $\Gamma'$ as follows

- $p \in V_{\mathcal{M}'}(s'_j)$ iff $p \in V_{\mathcal{M}}(s_{j+y})$ for all $j \in \mathbb{N}$ and all $p \in \mathsf{Prop}$ such that $p \neq a$
- $a \notin V_{\mathcal{M}'}(s'_0)$
- $a \in V_{\mathcal{M}'}(s'_j)$ for every $j \in \{1, \ldots, x - y - 1\}$
- $a \notin V_{\mathcal{M}'}(s'_j)$ for every $j \geq x - y$. ∎

As a particular case of Proposition 28, the derived rule $(\Diamond\, Set)$ is also sound.

**Proposition 29** *The operator* unnext *(see Definition 4) preserves satisfiability.*

*Proof* If $\mathcal{M}$ is a model of $\Gamma$ then $\mathsf{unnext}(\Gamma)$ is true in the state $s_1$ of $\mathcal{M}$, which obviously gives a model for $\mathsf{unnext}(\Gamma)$. ∎

Note that the equisatisfiability, in general, of initial and target sets of unnext cannot be ensured. For example, $\{p, \neg p, \circ q\}$ is unsatisfiable, but $\mathsf{unnext}(\{p, \neg p, \circ q\}) = \{q\}$ is satisfiable.

As a direct consequence of the above Propositions 27, 28 and 29, we have the following soundness theorem:

**Theorem 30** *If the resolution system* TRS *produces a refutation from $\Gamma$, then $\Gamma$ is unsatisfiable.* ∎

**Input:** A finite set of clauses $\Gamma$

**Output:** A resolution proof for $\Gamma$ called $\mathcal{D}(\Gamma)$

$\Gamma_0^0 := \Gamma;\ i := 0;\ j := 0;$
$\mathsf{sel\_ev\_set}_0 := \mathsf{fair\_select}(\Gamma_0^0);$
**loop**
  **if** $\mathsf{sel\_ev\_set}_i \neq \emptyset$
    **then** $(\Gamma_i^1, \mathsf{sel\_ev\_set}_i^*) := \mathsf{apply\_}\mathcal{U}\mathsf{\_Set}(\Gamma_i^0, \mathsf{sel\_ev\_set}_i);\ j := 1;$
    **else** $\mathsf{sel\_ev\_set}_i^* := \emptyset$
  **end if**;
  $\Gamma_i^* := \mathsf{close}(\Gamma_i^j);$
  **if** $\Box^b\bot \in \Gamma_i^*$ **or** $\mathsf{is\_cycling}(\mathcal{D}(\Gamma))$ **then exit**; **end if**;
  $\Gamma_{i+1}^0 := \mathsf{unnext}(\Gamma_i^*);$
  **if** $\mathsf{sel\_ev\_set}_i^* \cap \mathsf{event}(\Gamma_{i+1}^0) = \emptyset$ **then** $\mathsf{sel\_ev\_set}_{i+1} := \mathsf{fair\_select}(\Gamma_{i+1}^0);$
    **else** $\mathsf{sel\_ev\_set}_{i+1} := \mathsf{sel\_ev\_set}_i^*$
  **end if**;
  $i := i + 1;\ j := 0;$
**end loop**;

**Fig. 7** The Algorithm $\mathcal{SR}$

## 7 The Algorithm $\mathcal{SR}$ for Systematic TRS-Resolution

The nondeterministic application of the set of TRS-rules yields sound derivations but it does not guarantee completeness, even with the proviso of fairness. In this section we first introduce an algorithm called $\mathcal{SR}$ that uses the system TRS in a more (not fully) deterministic way which ensures completeness. Then, in the Subsection 7.2 we provide some detailed examples of application of $\mathcal{SR}$. In the last two subsections we respectively provide the termination and worst case complexity results for $\mathcal{SR}$.

### 7.1 The Algorithm $\mathcal{SR}$

The algorithm $\mathcal{SR}$, for any input set of clauses $\Gamma$, obtains a finite *resolution proof* –called $\mathcal{D}(\Gamma)$– of the form

$$\Gamma_0^0 \mapsto \ldots \mapsto \Gamma_0^{h_0} \Mapsto \Gamma_1^0 \mapsto \ldots \mapsto \Gamma_1^{h_1} \Mapsto \ldots \Mapsto \Gamma_k^0 \mapsto \ldots \mapsto \Gamma_k^{h_k}$$

As we will respectively show in Subsection 7.3 and Section 8, $\mathcal{D}(\Gamma)$ is always finite and $\mathcal{D}(\Gamma)$ is a refutation whenever the input set $\Gamma$ is unsatisfiable. When convenient, we represent $\mathcal{D}(\Gamma)$ by sequences of pairs

$$(\Gamma_0, \Gamma_0^*) \Mapsto (\Gamma_1, \Gamma_1^*) \Mapsto \ldots \Mapsto (\Gamma_k, \Gamma_k^*)$$

where $\Gamma_i$ and $\Gamma_i^*$ coincide with $\Gamma_i^0$ and $\Gamma_i^{h_i}$ respectively, for every $i \in \{0, \ldots, k\}$.

The construction of $\mathcal{D}(\Gamma)$, for any input $\Gamma$, is expressed by means of a while-program in Fig. 7, called the algorithm $\mathcal{SR}$, which we explain next. In order to ensure that $\mathcal{D}(\Gamma)$ is finite, the rule $(\mathcal{U}\,Set)$ is applied exactly to one eventuality[7] (if there is any) between each two consecutive unnext-steps (see Definition 14). For that purpose, the algorithm $\mathcal{SR}$

[7] see Definition 1.

keeps two variables $\mathsf{sel\_ev\_set}_i$ and $\mathsf{sel\_ev\_set}_i^*$ for every $i \geq 0$. Both variables $\mathsf{sel\_ev\_set}_i$ and $\mathsf{sel\_ev\_set}_i^*$ take as value a set that is empty or a singleton, depending on whether $\Gamma_i^0$ contains at least one eventuality or not, respectively. The variable $\mathsf{sel\_ev\_set}_i$ stands for the selected eventuality in $\Gamma_i^0$, whereas $\mathsf{sel\_ev\_set}_i^*$ corresponds to the eventuality selected in every set of the sequence from $\Gamma_i^1$ until $\Gamma_i^{h_i}$.
The algorithm $\mathcal{SR}$ (see Fig. 7) initializes both the set of clauses for starting the derivation $\Gamma_0^0$ to be the input set $\Gamma$ and the variable $\mathsf{sel\_ev\_set}_0$ to be either, a fairly selected eventuality in $\Gamma_0^0$ if there is any, or empty, otherwise. The expression $\mathsf{fair\_select}(\Gamma_i^j)$ encapsulates the *fair* selection of an eventuality in $\Gamma_i^j$, where fairness means that an eventuality cannot be indefinitely unselected.
After initialization, the algorithm $\mathcal{SR}$ iterates the following process.

- The lines 4 to 8 serve to extend the derivation from $\Gamma_i^0$ to $\Gamma_i^*$.
  First, by lines 4-7, the rule $(\mathcal{U}\,Set)$ is applied exactly to the selected eventuality provided that $\mathsf{sel\_ev\_set}_i \neq \emptyset$. More precisely, if $\mathsf{sel\_ev\_set}_i = \{T\}$, then the rule $(\mathcal{U}\,Set)$ is applied to a partition of $\Gamma_i^0$ of the form $\Phi \cup (\Gamma_i^0 \upharpoonright \mathsf{sel\_ev\_set}_i)$,[8] producing the set $\Gamma_i^1$ in $\mathcal{D}(\Gamma)$. Additionally, as part of this application of the rule $(\mathcal{U}\,Set)$, the variable $\mathsf{sel\_ev\_set}_i^*$ gets the value $\{a\,\mathcal{U}\,P\}$ where $a\,\mathcal{U}\,P$ is the new eventuality introduced by the rule $(\mathcal{U}\,Set)$ with a fresh variable $a$. Otherwise, if $\mathsf{sel\_ev\_set}_i$ is empty, the rule $(\mathcal{U}\,Set)$ is not applied and $\mathsf{sel\_ev\_set}_i^*$ gets the value $\emptyset$.
  Second, by line 8, the remaining TRS-rules are repeatedly applied to $\Gamma_i^j$ (where $j = 0$ or $j = 1$) to construct $\Gamma_i^*$. The operation $\mathsf{close}$ is introduced in Definition 20. Hence, $\Gamma_i^*$ is either TRS-closed (see Definition 19) or contains the empty clause. Moreover, the variable $\mathsf{sel\_ev\_set}_i^*$ is not changed by the operation $\mathsf{close}$. Hence, at line 11 the value of $\mathsf{sel\_ev\_set}_i^*$ is the same as at line 7.
- In line 9, the loop is exited if either the empty clause has been added to $\Gamma_i^*$ or a cycle in $\mathcal{D}(\Gamma)$ is detected according to the following definition.

  **Definition 31** *Let* $\mathcal{D} = (\Gamma_0, \Gamma_0^*) \mapsto (\Gamma_1, \Gamma_1^*) \mapsto \ldots \mapsto (\Gamma_j, \Gamma_j^*) \mapsto \ldots \mapsto (\Gamma_k, \Gamma_k^*)$ *be a derivation (where* $0 \leq j \leq k$*), we say that* $\mathcal{D}$ *is* cycling *with respect to* $j$ *and* $k$ *iff* $\mathcal{D}$ *satisfies the following conditions*
  1. $\Box^b\bot \notin \Gamma_i^*$ *for every* $i \in \{0, \ldots, k\}$
  2. $\mathsf{now}(\mathsf{unnext}(\Gamma_k^*)) = \mathsf{now}(\Gamma_j)$
  3. *For every eventuality* $T$ *such that* $T \in \mathsf{Lit}(\mathsf{now}(\Gamma_g))$ *for all* $g \in \{j, \ldots, k\}$*, there exists* $h \in \{j, \ldots, k\}$ *such that* $\mathsf{sel\_ev\_set}_h = \{T\}$. ∎

  The function $\mathsf{is\_cycling}$ (line 9) is supposed to implement a test of the conditions (2) and (3) in Definition 31 on the current derivation $\mathcal{D}(\Gamma) = (\Gamma_0, \Gamma_0^*) \mapsto \ldots \mapsto (\Gamma_i, \Gamma_i^*)$.
- Otherwise, if the loop is not exited, the $\mathsf{unnext}$ operator (Definition 4) is applied to the TRS-closed set $\Gamma_i^*$ to yield $\Gamma_{i+1}^0$ (line 10), which will be the $\Gamma_i^0$ of the next step, after increasing $i$ (line 14).
- Finally, the lines 11 to 13 serve to initialize the variable $\mathsf{sel\_ev\_set}_{i+1}$. Note that, after the application of the subsumption rule and/or of the $\mathsf{unnext}$ operator, every clause that includes the selected eventuality $\mathsf{sel\_ev\_set}_i^*$ could have disappeared from the current $\Gamma_{i+1}^0$. In other words, although $\circ(a\,\mathcal{U}\,P)$ occurs in some $\Gamma_i^j$, it could happen that the selected eventuality $a\,\mathcal{U}\,P$ does not occur in $\Gamma_{i+1}^0$. The function $\mathsf{event}$ (line 11) returns the set of all eventualities occurring in an input set of clauses, that is

  **Definition 32** *Let* $\Psi$ *be a set of clauses,* $\mathsf{event}(\Psi) = \{P_1\,\mathcal{U}\,P_2 \mid \Box^b((P_1\,\mathcal{U}\,P_2) \vee N) \in \Psi\}$.

[8] See Definition 2.

Therefore, if $\mathsf{sel\_ev\_set}_i^* \cap \mathsf{event}(\Gamma_{i+1}^0)$ is non-empty, then the selected eventuality remains selected. Otherwise, the function $\mathsf{fair\_select}$ is used to fairly select an eventuality from $\mathsf{event}(\Gamma_{i+1}^0)$.

We would like to remark the following three issues about the construction of $\mathcal{D}(\Gamma)$ by the algorithm $\mathcal{SR}$

1. Although $(Sbm)$ can be correctly applied whenever it is possible, in order to guarantee termination it suffices to apply $(Sbm)$ just before testing for a cycling derivation.
2. For achieving completeness the $\mathsf{unnext}$ operator must always be applied to TRS-closed sets.
3. In the intermediate sets $\Gamma_i^j$ of the process for obtaining $\Gamma_i^*$ from $\Gamma_i$, literals can appear that are neither in $\Gamma_i^*$ nor in $\Gamma_i$. This fact can be easily observed applying the algorithm $\mathcal{SR}$ to (e.g.) the set $\Gamma = \{p\,\mathcal{U}\,q, q\}$.

## 7.2 Examples

In this subsection we apply the algorithm $\mathcal{SR}$ to some illustrative examples. For readability, the selected eventualities appear between quotation symbols.

**Example 33** The following derivation is a refutation of $\{p, \Box(\neg p \vee \circ p), \Diamond\neg p\}$ that has been obtained following the algorithm $\mathcal{SR}$.

$$\begin{array}{cl}
\Gamma_0 = \Gamma_0^0 = \{p, \Box(\neg p \vee \circ p), \underline{\text{“}\Diamond\neg p\text{”}}\} & \\ \hline
\Gamma_0^1 = \{\underline{p}, \underline{\Box(\neg p \vee \circ p)}, \neg p \vee \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”}), \Box(\neg a \vee \neg p)\} & (\Diamond\, Set) \\ \hline
\Gamma_0^2 = \{\underline{p}, \Box(\neg p \vee \circ p), \underline{\neg p \vee \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})}, \Box(\neg a \vee \neg p), \circ p\} & (Res) \\ \hline
\Gamma_0^3 = \{\underline{p}, \Box(\neg p \vee \circ p), \neg p \vee \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”}), \underline{\Box(\neg a \vee \neg p)}, \circ p, \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})\} & (Res) \\ \hline
\Gamma_0^4 = \{p, \Box(\neg p \vee \circ p), \underline{\neg p \vee \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})}, \Box(\neg a \vee \neg p), \circ p, \underline{\circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})}, \neg a\} & (Res) \\ \hline
\Gamma_0^* = \Gamma_0^5 = \{p, \Box(\neg p \vee \circ p), \Box(\neg a \vee \neg p), \circ p, \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”}), \neg a\} & (Sbm) \\ \hline
\Gamma_1 = \Gamma_1^0 = \{\Box(\neg p \vee \circ p), \Box(\neg a \vee \neg p), p, \underline{\text{“}a\,\mathcal{U}\,\neg p\text{”}}\} & (\mathsf{unnext}) \\ \hline
\Gamma_1^1 = \{\Box(\neg p \vee \circ p), \Box(\neg a \vee \neg p), \underline{p}, \underline{\neg p \vee a}, \neg p \vee \circ(\text{“}b\,\mathcal{U}\,\neg p\text{”}), \Box(\neg b \vee a), \Box(\neg b \vee \neg p)\} & (\mathcal{U}\, Set) \\ \hline
\Gamma_1^2 = \{\Box(\neg p \vee \circ p), \underline{\Box(\neg a \vee \neg p)}, \underline{p}, \neg p \vee a, \varphi, \Box(\neg b \vee a), \Box(\neg b \vee \neg p), a\} & (Res) \\ \hline
\Gamma_1^3 = \{\Box(\neg p \vee \circ p), \Box(\neg a \vee \neg p), p, \neg p \vee a, \varphi, \Box(\neg b \vee a), \Box(\neg b \vee \neg p), \underline{a}, \underline{\neg a}\} & (Res) \\ \hline
\Gamma_1^* = \Gamma_1^4 = \{\Box(\neg p \vee \circ p), \Box(\neg a \vee \neg p), p, \neg p \vee a, \varphi, \Box(\neg b \vee a), \Box(\neg b \vee \neg p), a, \neg a, \bot\} & (Res)
\end{array}$$

where $\varphi = \neg p \vee \circ(\text{“}b\,\mathcal{U}\,\neg p\text{”})$

First of all, in $\Gamma_0$ the selected eventuality is $\Diamond\neg p$ and the context is $\{p\}$, since always-clauses are excluded from the negation of the context. Then, the rule $(\Diamond\, Set)$ is applied. This introduces a new propositional variable $a$ and transforms the selected eventuality into the last two clauses in $\Gamma_0^1$. From now, the selected eventuality is the until-formula in the third clause in $\Gamma_0^1$. After that, the resolution rule $(Res)$ is applied to the first two clauses in $\Gamma_0^1$. This produces the last clause $\circ p$ in $\Gamma_0^2$. Now again, $(Res)$ is applied to the first and third clauses in $\Gamma_0^2$, giving the last clause $\circ(a\,\mathcal{U}\,\neg p)$ in $\Gamma_0^3$. Again, by resolution of the first and fourth clauses in $\Gamma_0^3$, we obtain the clause $\neg a$ in $\Gamma_0^4$. By subsumption, the third clause is dropped, since it is subsumed by the sixth one, giving $\Gamma_0^5$. Now, since no other rule can be applied, the $\mathsf{unnext}$ operator transforms $\Gamma_0^5$ into $\Gamma_1$. The latter represents the clauses that must be satisfied in the state $s_1$, provided that the state $s_0$ satisfies $\Gamma_0$. Since the selected eventuality must be immediately handled (after $\mathsf{unnext}$), the rule $(\mathcal{U}\, Set)$ is applied to it.

Note that, the context is again $\{p\}$. Then, $\Gamma_1^1$ contains four new clauses that substitute the clause $a\,\mathcal{U}\,\neg p$. A new propositional variable $b$ occurs in the new clauses. Finally, by three consecutive applications of the rule $(Res)$ to the three underlined pairs of clauses, the empty clause is obtained. Note that the repeated context in $\Gamma_0$ and $\Gamma_1$ has led to find a contradiction in three resolution steps.

In the previous example, if we had used the rules $(\Diamond\, Fix)$ and $(\mathcal{U}\, Fix)$ instead of the rules $(\Diamond\, Set)$ and $(\mathcal{U}\, Set)$, we would have not obtained the empty clause. The following example illustrates this fact.

**Example 34** Below, we start with the same $\Gamma_0^0$ as in the previous Example 33. We firstly apply $(\Diamond\, Fix)$ (instead of $(\Diamond\, Set)$) and get a set $\Gamma_0^1$ with an atom $p$ that is resolved with two clauses that contain $\neg p$. Then, by subsumption and unnext we get $\Gamma_1^0 = \Gamma_0^0$. Repeating this process we could obtain an endless resolution derivation. Indeed, we will never obtain the empty clause unless we use the rules $(\Diamond\, Set)$ and $(\mathcal{U}\, Set)$ in an appropriate manner.

$$
\begin{array}{cl}
\Gamma_0^0 = \{p, \Box(\neg p \lor \circ p), \underline{\Diamond \neg p}\} & \\ \hline
\Gamma_0^1 = \{\underline{p}, \underline{\Box(\neg p \lor \circ p)}, \neg p \lor \circ\Diamond \neg p\} & (\Diamond\, Fix) \\ \hline
\Gamma_0^2 = \{\underline{p}, \Box(\neg p \lor \circ p), \underline{\neg p \lor \circ\Diamond \neg p}, \circ p\} & (Res) \\ \hline
\Gamma_0^3 = \{p, \Box(\neg p \lor \circ p), \underline{\neg p \lor \circ\Diamond \neg p}, \circ p, \underline{\circ\Diamond \neg p}\} & (Res) \\ \hline
\Gamma_0^4 = \{p, \Box(\neg p \lor \circ p), \circ p, \circ\Diamond \neg p\} & (Sbm) \\ \hline
\Gamma_1^0 = \{p, \Box(\neg p \lor \circ p), \Diamond \neg p\} & (\mathsf{unnext}) \\ \hline
\cdots &
\end{array}
$$

Obviously, this derivation does not follow the algorithm $\mathcal{SR}$.

The next example shows how the systematic TRS-resolution deals with clauses of the form $\Box P$.

**Example 35**

$$
\begin{array}{cl}
\Gamma_0 = \Gamma_0^0 = \{\Box p, \underline{\text{“}\Diamond \neg p\text{”}}\} & \\ \hline
\Gamma_0^1 = \{\underline{\Box p}, \underline{\neg p \lor \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})}, \Box \neg a\} & (\Diamond\, Set) \\ \hline
\Gamma_0^2 = \{\Box p, \underline{\neg p \lor \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})}, \underline{\circ(\text{“}a\,\mathcal{U}\,\neg p\text{”})}, \Box \neg a\} & (Res) \\ \hline
\Gamma_0^* = \Gamma_0^3 = \{\Box p, \circ(\text{“}a\,\mathcal{U}\,\neg p\text{”}), \Box \neg a\} & (Sbm) \\ \hline
\Gamma_1 = \Gamma_1^0 = \{\Box p, \underline{\text{“}a\,\mathcal{U}\,\neg p\text{”}}, \Box \neg a\} & (\mathsf{unnext}) \\ \hline
\Gamma_1^1 = \{\underline{\Box p}, \Box \neg a, \underline{\neg p \lor a}, \neg p \lor \circ(\text{“}b\,\mathcal{U}\,\neg p\text{”}), \Box \neg b\} & (\mathcal{U}\, Set) \\ \hline
\Gamma_1^2 = \{\Box p, \underline{\Box \neg a}, \neg p \lor a, \neg p \lor \circ(\text{“}b\,\mathcal{U}\,\neg p\text{”}), \Box \neg b, \underline{a}\} & (Res) \\ \hline
\Gamma_1^* = \Gamma_1^3 = \{\Box p, \Box \neg a, \neg p \lor a, \neg p \lor \circ(\text{“}b\,\mathcal{U}\,\neg p\text{”}), \Box \neg b, a, \bot\} & (Res)
\end{array}
$$

Since the procedure close in $\mathcal{SR}$ uses the function BTL (see Definition 4) for selecting temporal literals and since BTL is based on the function $\mathsf{drop}_\Box$, clauses of the form $\Box P$ are considered always-clauses formed by one propositional literal and not now-clauses formed by one (basic) temporal literal. So following $\mathcal{SR}$ we obtain the above refutation. But it is worthy to remark that if we do not follow $\mathcal{SR}$ it is possible to build the following refutation

$$
\frac{\Gamma_0^0 = \{\underline{\Box p}, \underline{\Diamond \neg p}\}}{\Gamma_0^1 = \{\Box p, \Diamond \neg p, \bot\}}\ (Res)
$$

The following two examples show that the subsumption rule $(Sbm)$ is required to guarantee the termination of the algorithm $\mathcal{SR}$. In the case of Example 36 the concerned set of clauses is satisfiable, whereas in Example 37 is not.

**Example 36** Consider the following derivation for the set of clauses $\{(p\mathcal{U}\,q)\vee\Box\, r, \Box\,\neg p, \Box\,\neg q\}$, which is only developed until the first application of (unnext).

$$
\begin{array}{ll}
\Gamma_0 = \Gamma_0^0 = \{\underline{\text{“}(p\mathcal{U}\,q)\text{”} \vee \Box\, r}, \Box\,\neg p, \Box\,\neg q\} & (\mathcal{U}\,Set) \\ \hline
\Gamma_0^1 = \{\underline{q \vee p \vee \Box\, r}, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \Box\, r, \Box\,\neg p, \Box\,\neg q, \Box\,\neg a_1\} & (\Box Fix) \\ \hline
\Gamma_0^2 = \{q \vee p \vee r, q \vee p \vee \circ\Box\, r, \underline{q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \Box\, r}, \Box\,\neg p, \Box\,\neg q, \Box\,\neg a_1\} & (\Box Fix) \\ \hline
\Gamma_0^3 = \{\underline{q \vee p \vee r}, q \vee p \vee \circ\Box\, r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, \Box\,\neg p, \underline{\Box\,\neg q}, & \\
\quad \Box\,\neg a_1\} & (Res) \\ \hline
\Gamma_0^4 = \{q \vee p \vee r, \underline{q \vee p \vee \circ\Box\, r}, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, \Box\,\neg p, \underline{\Box\,\neg q}, & \\
\quad \Box\,\neg a_1, p \vee r\} & (Res) \\ \hline
\Gamma_0^5 = \{q \vee p \vee r, q \vee p \vee \circ\Box\, r, \underline{q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r}, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, \Box\,\neg p, \underline{\Box\,\neg q}, & \\
\quad \Box\,\neg a_1, p \vee r, p \vee \circ\Box\, r\} & (Res) \\ \hline
\Gamma_0^6 = \{q \vee p \vee r, q \vee p \vee \circ\Box\, r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, \underline{q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r}, \Box\,\neg p, \underline{\Box\,\neg q}, & \\
\quad \Box\,\neg a_1, p \vee r, p \vee \circ\Box\, r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r\} & (Res) \\ \hline
\Gamma_0^7 = \{\underline{q \vee p \vee r}, q \vee p \vee \circ\Box\, r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, \underline{\Box\,\neg p}, \Box\,\neg q, & \\
\quad \Box\,\neg a_1, p \vee r, p \vee \circ\Box\, r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r\} & (Res) \\ \hline
\Gamma_0^8 = \{q \vee p \vee r, \underline{q \vee p \vee \circ\Box\, r}, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, \underline{\Box\,\neg p}, \Box\,\neg q, & \\
\quad \Box\,\neg a_1, p \vee r, p \vee \circ\Box\, r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, q \vee r\} & (Res) \\ \hline
\Gamma_0^9 = \{q \vee p \vee r, q \vee p \vee \circ\Box\, r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, \underline{\Box\,\neg p}, \Box\,\neg q, & \\
\quad \Box\,\neg a_1, \underline{p \vee r}, p \vee \circ\Box\, r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, q \vee r, q \vee \circ\Box\, r\} & (Res) \\ \hline
\Gamma_0^{10} = \{q \vee p \vee r, q \vee p \vee \circ\Box\, r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, & \\
\quad \underline{\Box\,\neg p}, \Box\,\neg q, \Box\,\neg a_1, p \vee r, \underline{p \vee \circ\Box\, r}, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, & \\
\quad \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, q \vee r, q \vee \circ\Box\, r, r\} & (Res) \\ \hline
\Gamma_0^{11} = \{q \vee p \vee r, q \vee p \vee \circ\Box\, r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, q \vee \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, & \\
\quad \Box\,\neg p, \Box\,\neg q, \Box\,\neg a_1, p \vee r, p \vee \circ\Box\, r, \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee r, & \\
\quad \text{“}\circ(a_1\,\mathcal{U}\,q)\text{”} \vee \circ\Box\, r, q \vee r, q \vee \circ\Box\, r, r, \circ\Box\, r\} & (Sbm) \\ \hline
\Gamma_0^* = \Gamma_0^{12} = \{\Box\,\neg p, \Box\,\neg q, \Box\,\neg a_1, r, \circ\Box\, r\} & (\mathsf{unnext}) \\ \hline
\Gamma_1 = \Gamma_1^0 = \{\Box\,\neg p, \Box\,\neg q, \Box\,\neg a_1, \Box\, r\} &
\end{array}
$$

It is worthy to note that if $(Sbm)$ were not applied in the step just before (unnext), then the above set $\Gamma_1$ would be

$$\{\text{“}(a_1\,\mathcal{U}\,q)\text{”} \vee \Box\, r, \Box\,\neg p, \Box\,\neg q, \Box\,\neg a_1, \Box\, r\}$$

Indeed, every set $\Gamma_i$ ($i \geq 1$) obtained after $i$ unnext-steps would be of the form $\{(a_i\,\mathcal{U}\,q) \vee \Box\, r, \Box\,\neg p, \Box\,\neg q, \Box\, r\} \cup \{\Box\,\neg a_h \mid 1 \leq h \leq i\}$. Consequently, it would be impossible to obtain two sets $\Gamma_j$ and $\Gamma_k$ such that $0 \leq j \leq k$ and $\mathsf{now}(\Gamma_j) = \mathsf{now}(\mathsf{unnext}(\Gamma_k^*))$. Hence, the resolution process would not stop.

**Example 37** For the set of clauses $\{(p\mathcal{U}\,q) \vee (r\,\mathcal{U}\,s), \Box\,\neg p, \Box\,\neg q, \Box\,\neg s\}$, if the first selected eventuality is $p\mathcal{U}\,q$ then the same problem as in the previous Example 36 happens, but with $(a_i\,\mathcal{U}\,q) \vee (r\,\mathcal{U}\,s)$ instead of $(a_i\,\mathcal{U}\,q) \vee \Box\, r$, where $a_i$ is a fresh variable.

*Remark 1* Note that when $\Gamma$ is a satisfiable set of (non-temporal) classical propositional clauses, the derivation $\mathcal{D}(\Gamma)$ obtained by the algorithm $\mathcal{SR}$ is of the form $\Gamma_0^0 \mapsto \ldots \mapsto \Gamma_0^{h_0} \Mapsto \Gamma_1^0$, and it can also be represented as $(\Gamma_0, \Gamma_0^*) \Mapsto (\Gamma_1, \Gamma_1^*)$, where $\Gamma_0 = \Gamma_0^0 = \Gamma$, $\Gamma_0^{h_0} = \Gamma_0^*$, $\Gamma_1 = \Gamma_1^* = \mathsf{unnext}(\Gamma_0^*) = \emptyset$. The set $\Gamma_1^0$ –which is at the same time $\Gamma_1$ and $\Gamma_1^*$– is TRS-closed and additionaly produces a cycle because $\mathcal{D}(\Gamma)$ verifies the three items of Definition 31 and, in particular the second one since $\mathsf{now}(\mathsf{unnext}(\Gamma_1^*)) = \mathsf{now}(\Gamma_1)$. So the cycle is from $\Gamma_1^0$ to $\Gamma_1^0$. Sets of temporal clauses, e.g. the singleton $\{\circ P\}$, can also give rise to this kind of cycling derivation ended in an empty set. However, the singleton $\{\Box\, P\}$

produces a cycle with non-empty set of clauses. In general, every systematic derivation that is not a refutation becomes cyclic.

Along the rest of the paper, we will denote by $\mathcal{D}(\Gamma)$ any derivation of the form $(\Gamma_0, \Gamma_0^*) \mapsto (\Gamma_1, \Gamma_1^*) \mapsto \ldots \mapsto (\Gamma_j, \Gamma_j^*) \mapsto \ldots \mapsto (\Gamma_k, \Gamma_k^*)$ obtained by $\mathcal{SR}$ with initial set $\Gamma_0 = \Gamma$. In particular, $\mathcal{D}(\Gamma)$ may be a refutation or a cycling derivation with respect to $j$ and $k$.

### 7.3 Termination

In this section we show that the algorithm $\mathcal{SR}$ always obtains either a refutation or a cycling derivation after a finite number of iterations. Remember that we assume that $\mathcal{SR}$ uses a fair strategy for selecting eventualities.

The termination proof of $\mathcal{SR}$ requires to show that the algorithm cannot generate an infinite number of new propositional variables. A priori, there are two ways for generating new propositional variables in $\mathcal{SR}$. The first is the translation to CNF applied in the output to the rule $(\mathcal{U}\,Set)$. However, no new variable is introduced by $\mathcal{SR}$ in this way. The reason is that the translation to CNF is applied to a formula that only needs DtNF-rules to be in CNF and DtNF-rules do not use extra variables (see Proposition 7).

The second source of new propositional variables is the fresh variable that explicitly occurs in the consequent of the rule $(\mathcal{U}\,Set)$. However, as we will show, the sequence of new eventualities produced by successive applications of the rule $(\mathcal{U}\,Set)$ is always finite. There is a twofold reason for the latter. On one hand, the clauses defining a new variable (see function def in Fig. 5) are always-clauses, which are excluded from the negated context. On the other hand, in the algorithm $\mathcal{SR}$, the rule $(\mathcal{U}\,Set)$ is always applied to sets where the propositional variables introduced (as fresh) by previous applications of $(\mathcal{U}\,Set)$ are also out of the context.

In order to prove the termination result, we first define the closure (Definition 39) of a set of clauses $\Gamma$ that contains all the clauses that can be generated from the literals that could appear in the clauses obtained from $\Gamma$ by means of all the TRS-rules with the exception of the rule $(\mathcal{U}\,Set)$ (and the derived rule $(\diamond\,Set)$).

**Definition 38** *Let $\Gamma$ be a set of clauses. The set* $\mathsf{univlit}(\Gamma)$ *is the smallest set of literals defined as follows*[9]

- $\mathsf{Lit}(\Gamma) \subseteq \mathsf{univlit}(\Gamma)$
- *If* $L \in \mathsf{univlit}(\Gamma)$*, then* $\widetilde{L} \in \mathsf{univlit}(\Gamma)$
- *If* $P_1\,\mathcal{U}\,P_2 \in \mathsf{univlit}(\Gamma)$*, then* $\{\circ(P_1\,\mathcal{U}\,P_2), P_1, P_2\} \subseteq \mathsf{univlit}(\Gamma)$
- *If* $P_1\,\mathcal{R}\,P_2 \in \mathsf{univlit}(\Gamma)$*, then* $\{\circ(P_1\,\mathcal{R}\,P_2), P_1, P_2\} \subseteq \mathsf{univlit}(\Gamma)$
- *If* $\diamond P \in \mathsf{univlit}(\Gamma)$*, then* $\{\circ\diamond P, P\} \subseteq \mathsf{univlit}(\Gamma)$
- *If* $\Box P \in \mathsf{univlit}(\Gamma)$*, then* $\{\circ\Box P, P\} \subseteq \mathsf{univlit}(\Gamma)$
- *If* $\circ L \in \mathsf{univlit}(\Gamma)$*, then* $L \in \mathsf{univlit}(\Gamma)$.

The set $\mathsf{univlit}(\Gamma)$ is finite for any set of clauses $\Gamma$ since we only consider finite sets of clauses and finite clauses. Now, we define the closure of a set of clauses.

**Definition 39** *Let $\Gamma$ be a set of clauses. The set* $\mathsf{closure}(\Gamma)$ *is the set formed by all the clauses $C$ such that* $\mathsf{Lit}(C) \subseteq \mathsf{univlit}(\Gamma)$.

[9] Remember that $\mathsf{Lit}(\Box^b(L_1 \vee \ldots \vee L_n)) = \{L_1, \ldots, L_n\}$ and $\mathsf{Lit}(\Gamma) = \bigcup_{C \in \Gamma} \mathsf{Lit}(C)$.

The rule $(\mathcal{U}\,Set)$ introduces new eventualities involving fresh variables. In order to justify that derivations that (potentially) use $(\mathcal{U}\,Set)$ are finite, we have to show that the cycling conditions in Definition 31, in particular its third requirement, will be satisfied after a finite number of iteration steps.

**Definition 40** *Let $\mathcal{D}(\Gamma) = (\Gamma_0, \Gamma_0^*) \mapsto \ldots \mapsto (\Gamma_k, \Gamma_k^*)$ be the derivation constructed by the algorithm $\mathcal{SR}$ (Fig. 7). We say that an eventuality $T'$ is the* direct descendant *of an eventuality $T$ in $\mathcal{D}(\Gamma)$ iff for some $i \in \{0, \ldots, k\}$:* $\mathsf{sel\_ev\_set}_i = \{T\}$ *and* $\mathsf{sel\_ev\_set}_i^* = \{T'\}$.
*Let $S = T_0, T_1, \ldots, T_n$ be a sequence of eventualities. We say that $S$ is the* sequence of descendants *of $T_0$ in $\mathcal{D}(\Gamma)$ iff $T_{i+1}$ is a direct descendant of $T_i$ in $\mathcal{D}(\Gamma)$ for all $i \in \{0, \ldots, n-1\}$.*

For example, $\diamond \neg p, a\,\mathcal{U} \neg p, b\,\mathcal{U} \neg p$ is the sequence of descendants of $\diamond \neg p$ in the derivation in Example 35.

**Lemma 41** *For all $\mathcal{D}(\Gamma)$ and every selected eventuality $T$ in $\mathcal{D}(\Gamma)$, the sequence of descendants of $T$ in $\mathcal{D}(\Gamma)$ is finite.*

*Proof* Let $T$ be $P_0\,\mathcal{U}\,P$. Suppose that $T$ occurs in the set $\Gamma_0^0$ in $\mathcal{D}(\Gamma)$, $\mathsf{sel\_ev\_set}_0 = \{P_0\,\mathcal{U}\,P\}$ and the sequence of descendants of $T$ in $\mathcal{D}(\Gamma)$ is infinite. When the rule $(\mathcal{U}\,Set)$ is applied to a partition of $\Gamma_0^0$ of the form $\Phi_0 \;\cup\; \Gamma_0^0 \upharpoonright \{P_0\,\mathcal{U}\,P\}$, the set $\Gamma_0^0 \upharpoonright \{P_0\,\mathcal{U}\,P\}$ is replaced with the union of the following five disjoint sets of clauses

$$
\begin{aligned}
&\Psi_0^1 = \{P \vee P_0 \vee N_0 \mid \Box^b((P_0\,\mathcal{U}\,P) \vee N_0) \in \Gamma_0\}\\
&\Psi_0^2 = \{P \vee \circ(a_1\,\mathcal{U}\,P) \vee N_0 \mid \Box^b((P_0\,\mathcal{U}\,P) \vee N_0) \in \Gamma_0\}\\
&\Psi_0^3 = \{\Box\,(\circ(P_0\,\mathcal{U}\,P) \vee \circ N_0) \mid \Box((P_0\,\mathcal{U}\,P) \vee N_0) \in \Gamma_0\}\\
&\Psi_0^4 = \{\Box\,(\neg a_1 \vee P_0)\}\\
&\Psi_0^5 = \mathsf{CNF}(\Box\,(\neg a_1 \vee \neg\mathsf{now}(\Phi_0)))
\end{aligned}
$$

where $\Psi_0^4 \cup \Psi_0^5$ corresponds to $\mathsf{CNF}(\mathsf{def}(a_1, P_0, \mathsf{now}(\Phi_0)))$ (see Fig. 5).

Hence, the set $\Gamma_0^1$ is the union of $\Phi_0$ and the above five sets, and the new selected eventuality is $a_1\,\mathcal{U}\,P$, i.e., $\mathsf{sel\_ev\_set}_0^* = \{a_1\,\mathcal{U}\,P\}$. The fresh variable $a_1$ only occurs in $\Psi_0^2$ and $\Psi_0^4 \cup \Psi_0^5$. The latter is a set of always-clauses, and the occurrences of $a_1$ in $\Psi_0^4 \cup \Psi_0^5$ are not preceded by $\circ$. Consequently, after the operations $\mathsf{close}$ and $\mathsf{unnext}$ (lines 8 and 10 in Fig. 7), all the occurrences of $a_1$ in the set $\Gamma_1^0$ are either in an always-clause or in a now-clause that comes from $\Psi_0^2$. Hence, the only now-clauses where $a_1$ occurs in $\Gamma_1^0$ are of the form $N \vee a_1\,\mathcal{U}\,P$, where $a_1\,\mathcal{U}\,P$ is the new selected eventuality. Hence, the next application of the rule $(\mathcal{U}\,Set)$ does not introduce any occurrence of $a_1$ in the negated context, because always-clauses and clauses containing $a_1\,\mathcal{U}\,P$ are both excluded from the context. Moreover, $\mathsf{CNF}(\Box\,(\neg a_1 \vee \neg\mathsf{now}(\Phi_0)))$ does not contain any other fresh variable (apart from $a_1$). The reason is that $\mathsf{DtNF}(\Box\,(\neg a_1 \vee \neg\mathsf{now}(\Phi_0)))$ is already in conjunctive normal form, so the only transformation that uses new fresh variables –which is detailed in the proof of Theorem 8– is left out.

The above reasoning about the construction of $\Gamma_1^0$ from $\Gamma_0^0$ can be generalized to the construction of $\Gamma_{i+1}^0$ from $\Gamma_i^0$ with selected eventuality $a_i\,\mathcal{U}\,P$ to obtain a direct descendant $a_{i+1}\,\mathcal{U}\,P$ as follows. When the rule $(\mathcal{U}\,Set)$ is applied to a partition of $\Gamma_i^0$ of the form $\Phi_i \;\cup\; \Gamma_i^0 \upharpoonright \{a_i\,\mathcal{U}\,P\}$, then the consequent $\Gamma_i^1$ is the union of $\Phi_i$ and the following five disjoint sets

$$
\begin{aligned}
\Psi_i^1 &= \{P \vee a_i \vee N_i \mid \Box^b((a_i\,\mathcal{U}\,P) \vee N_i) \in \Gamma_i\}\\
\Psi_i^2 &= \{P \vee \circ(a_{i+1}\,\mathcal{U}\,P) \vee N_i \mid \Box^b((a_i\,\mathcal{U}\,P) \vee N_i) \in \Gamma_i\}\\
\Psi_i^3 &= \{\Box\,(\circ(a_i\,\mathcal{U}\,P) \vee \circ N_i) \mid \Box\,((a_i\,\mathcal{U}\,P) \vee N_i) \in \Gamma_i\}\\
\Psi_i^4 &= \{\Box\,(\neg a_1 \vee P_0), \Box\,(\neg a_2 \vee a_1), \dots, \Box\,(\neg a_i \vee a_{i-1}), \Box\,(\neg a_{i+1} \vee a_i)\}\\
\Psi_i^5 &= \mathsf{CNF}(\Box\,(\neg a_{i+1} \vee \neg \mathsf{now}(\Phi_i)))
\end{aligned}
$$

where $(\Psi_i^4 \setminus \Psi_{i-1}^4) \cup \Psi_i^5$ corresponds to $\mathsf{CNF}(\mathsf{def}(a_{i+1}, a_i, \mathsf{now}(\Phi_i)))$ whenever $i \geq 1$ (see Fig. 5). Now, the fresh variables $a_1, \dots, a_i, a_{i+1}$ occur in the above five sets $\Psi_i^j$. The occurrences of fresh variables in $\Psi_i^2 \cup \Psi_i^4 \cup \Psi_i^5$ are not filtered to the negated context in $\Gamma_{i+1}^0$ by the reasons explained above for $\Gamma_1^0$. Regarding the occurrences of $a_i$ in the set $\Psi_i^1$, since they are not preceded by $\circ$, no one of them can be filtered to $\Gamma_{i+1}^0$. Additionally, $\Psi_i^3$ is empty for all $i \geq 1$. To realize this fact, it suffices to check the following three facts. First, whenever the rule $(\mathcal{U}\,Set)$ is applied to the set $\Gamma_{i-1}^0$, by considering the partition $\Phi_{i-1} \cup (\Gamma_{i-1}^0 \upharpoonright \mathsf{sel\_ev\_set}_{i-1})$, the new literal $\circ(a_i\,\mathcal{U}\,P)$ appears only in now-clauses. Second, the remaining basic rules (resolution, subsumption and fixpoint rules), that are applied to obtain the TRS-closed set $\Gamma_{i-1}^*$ from $\Gamma_{i-1}^1$, cannot introduce (in $\Gamma_{i-1}^*$) an always-clause $C$ such that $\circ(a_i\,\mathcal{U}\,P) \in \mathsf{Lit}(C)$. Third, since $\Gamma_i^0$ is obtained from $\Gamma_{i-1}^*$ by $\mathsf{unnext}$, then $\Gamma_i^0$ cannot include an always-clause $C$ such that $\circ(a_i\,\mathcal{U}\,P) \in \mathsf{Lit}(C)$.
Consequently, every fresh variable $a_\ell$ is not in $\mathsf{Lit}(\mathsf{now}(\Gamma_h^0))$ for all $h \geq \ell$ and all $\ell \geq 1$. Therefore, fresh variables do not occur in any context of any application of the rule $(\mathcal{U}\,Set)$. So that, the successive contexts are exclusively formed by formulas from the closure of $\Gamma_0^0$. Since the set $\mathsf{closure}(\Gamma_0^0)$ is finite, if the sequence of descendants of $P_0\,\mathcal{U}\,P$ were infinite, there would necessarily be two sets $\Gamma_g^0$ and $\Gamma_h^0$ such that $g < h$ and $\mathsf{now}(\Gamma_g^0 \setminus \Gamma_g^0 \upharpoonright \mathsf{sel\_ev\_set}_g) = \mathsf{now}(\Gamma_h^0 \setminus \Gamma_h^0 \upharpoonright \{a_h\,\mathcal{U}\,P\})$[10]. Without loss of generality, we consider $g = 0$ and $h = i$. By repeatedly applying the rule $(Res)$ to $\mathsf{now}(\Gamma_0^0 \setminus \Gamma_0^0 \upharpoonright \{P_0\,\mathcal{U}\,P\})$ and $\mathsf{CNF}(\Box\,(\neg a_1 \vee \neg \mathsf{now}(\Gamma_0 \setminus \Gamma_0 \upharpoonright \{P_0\,\mathcal{U}\,P\})))$, the algorithm $\mathcal{SR}$ obtains $\neg a_1$ which resolves with $\Box\,(\neg a_2 \vee a_1)$ producing $\neg a_2$. Then $\neg a_2$ resolves with $\Box\,(\neg a_3 \vee a_2)$. At the end of this process $\neg a_{i-1}$ resolves with $\Box\,(\neg a_i \vee a_{i-1})$ producing $\neg a_i$. This literal resolves with every clause in $\{P \vee a_i \vee N_i \mid (a_i\,\mathcal{U}\,P) \vee N_i \in \Gamma_i\}$ producing the clauses in $\{P \vee N_i \mid (a_i\,\mathcal{U}\,P) \vee N_i \in \Gamma_i\}$ which subsume the clauses in $\{P \vee \circ(a_{i+1}\,\mathcal{U}\,P) \vee N_i \mid (a_i\,\mathcal{U}\,P) \vee N_i \in \Gamma_i\}$. Therefore, the selected temporal literal $a_{i+1}\,\mathcal{U}\,P$ disappears after the following unnext-step. Hence, $a_{i+1}\,\mathcal{U}\,P$ cannot be the selected eventuality at the next step, i.e., $\mathsf{sel\_ev\_set}_{i+1} \neq \{a_{i+1}\,\mathcal{U}\,P\}$. This is a contradiction because the sequence of descendants of $P_0\,\mathcal{U}\,P$ has been supposed to be infinite. ∎

In the above proof we have considered that $(\mathcal{U}\,Set)$ is always applied with a non-empty context. The proof for possibly empty contexts is just a special case. Note also that the application of the subsumption rule, together with the subsequent use of the $\mathsf{unnext}$ operator, is essential in the above proof.

**Theorem 42** *The algorithm $\mathcal{SR}$, for each input $\Gamma$, terminates giving a resolution proof.*

*Proof* Suppose that $\mathcal{SR}$ does not produce $\Box^b\bot$. On the one hand, by Lemma 41, $\mathcal{SR}$ cannot generate an infinite sequence of descendants of any selected eventuality. Besides, when the sequence of descendants of one eventuality finishes because the last one, namely $T$, ceases to be the selected eventuality in $\Gamma_i$ for some $i \geq 1$ (i.e. $\mathsf{sel\_ev\_set}_{i-1}^* = \{T\}$ and

[10] $\mathsf{sel\_ev\_set}_g = \{P_0\,\mathcal{U}\,P\}$ if $g = 0$, and $\mathsf{sel\_ev\_set}_g = \{a_g\,\mathcal{U}\,P\}$ if $g > 0$.

$\mathsf{sel\_ev\_set}_i \neq \{T\}$), then the set $\mathsf{now}(\Gamma_i)$ is included in $\mathsf{closure}(\Gamma)$ because the fresh variables introduced by $(\mathcal{U}\,Set)$ only occur in $\mathsf{alw}(\Gamma_i)$. If the process continues and the algorithm $\mathcal{SR}$ selects another eventuality, finiteness of sequences of descendants (Lemma 41) guarantees the existence of $\Gamma_g$, with $g > i$, such that $\mathsf{now}(\Gamma_g)$ is included in $\mathsf{closure}(\Gamma)$. As the closure is finite, there must exist $j$ and $k$ such that $j \leq k$ and the set of now-clauses of $\Gamma_j$ is exactly the set of now-clauses of $\mathsf{unnext}(\Gamma_k^*)$.
On the other hand, fairness ensures that the third condition in Definition 31 must be satisfied at some moment. ∎

Note that the third condition in Definition 31 is persistent in the sense that once it is satisfied in a derivation, it cannot be broken.

### 7.4 Complexity

In order to analyze the worst case complexity of the algorithm $\mathcal{SR}$, we first consider the set $\mathsf{closure}(\Gamma)$ (see Definition 39) of all the possible clauses formed using the literals in $\mathsf{univlit}(\Gamma)$ (see Definition 38).

**Proposition 43** *The number of clauses in* $\mathsf{closure}(\Gamma)$ *is* $2^n$*, where* $n$ *is the number of literals in* $\mathsf{univlit}(\Gamma)$. ∎

Then, the set of all possible sets of clauses that could appear as context when applying $(\mathcal{U}\,Set)$ has double-exponential size in $n$.

**Proposition 44** *Let* $\mathsf{contexts}(\Gamma) = \{\Delta \mid \Delta \subseteq \mathsf{closure}(\Gamma)\}$*, then the number of sets in* $\mathsf{contexts}(\Gamma)$ *is* $2^{2^n}$. ∎

Therefore, the worst case complexity of the algorithm $\mathcal{SR}$ can be bounded to $\mathcal{O}(2^{\mathcal{O}(2^n)})$.

**Proposition 45** *The number of clauses generated by the resolution method is bounded by* $\mathcal{O}(2^{\mathcal{O}(2^n)})$ *and the number of new variables is also bounded by* $\mathcal{O}(2^{\mathcal{O}(2^n)})$ *where* $n$ *is the number of literals in* $\mathsf{univlit}(\Gamma)$.

*Proof* In the worst case, each clause in $\mathsf{closure}(\Gamma)$ contains a selected eventuality that generates a sequence of descendants with an eventuality for each possible context in $\mathsf{contexts}(\Gamma)$ plus a repeated context. That is, each of the $2^n$ initial clauses may generate $1 + 2^{2^n}$ clauses with new eventualities. So, $f(n) = 2^n \times (1 + 2^{2^n}) = 2^n + 2^{n+2^n}$ is the maximum number of different clauses (with new eventualities) that can appear in a derivation. Since, each new eventuality is associated to a new variable, $2^n + 2^{n+2^n}$ also bounds the number of fresh variables. In the worst case, the definition of each new variable generates $2^n$ new clauses. So that, $g(n) = 2^{2.n} + 2^{2.n+2^n}$ bounds the number of clauses defining new variables. To sum up, the worst case is bounded to

$$2^n + f(n) + g(n) = 2^n + 2^n + 2^{n+2^n} + 2^{2.n} + 2^{2.n+2^n}$$

where the leftmost $2^n$ stands for the size of the closure which bounds the initial set of clauses. That is, in the worst case, the number of clauses is in $\mathcal{O}(2^{\mathcal{O}(2^n)})$ and the number of new variables is in $\mathcal{O}(2^{\mathcal{O}(2^n)})$ . ∎

## 8 Completeness

A resolution method is *refutationally complete* if, whenever a set of clauses $\Gamma$ is unsatisfiable, a refutation for $\Gamma$ can be constructed. In our case we prove the refutational completeness of TRS-resolution showing that there exists a model of $\Gamma$ whenever the resolution proof $\mathcal{D}(\Gamma)$ obtained by the algorithm $\mathcal{SR}$ is a cycling derivation. This result together with the proof of termination (Theorem 42) shows that our algorithm for systematic resolution (Fig. 7) is complete and, hence, a decision procedure for PLTL.

For the rest of this section we fix the derivation

$$\mathcal{D}(\Gamma) \equiv (\Gamma_0, \Gamma_0^*) \mapsto (\Gamma_1, \Gamma_1^*) \mapsto \ldots \mapsto (\Gamma_j, \Gamma_j^*) \mapsto \ldots \mapsto (\Gamma_k, \Gamma_k^*)$$

to be cycling with respect to $j$ and $k$. In order to prove the existence of a model of $\Gamma$ from the existence of $\mathcal{D}(\Gamma)$ we will show that the sets $\Gamma_i^*$ in $\mathcal{D}(\Gamma)$ can be extended (with literals of their own clauses) preserving its local consistency. These extensions are literal-closed in the sense that they contain at least one literal from each clause in $\Gamma_i^*$. Remember that the sets $\Gamma_i^*$ in $\mathcal{D}(\Gamma)$ are TRS-closed (see Definition 19) which, in particular, means that $\mathsf{BTL}(\Gamma_i^*) = \emptyset$. Actually, inside the collection of all the locally consistent literal-closed (lclc, in short) extensions of each $\Gamma_i^*$, we define the subclass of the so-called *standard extensions*. In particular, standard lclc-extensions of the sets $\Gamma_i^*$ in $\mathcal{D}(\Gamma)$ allow us to ensure the model existence. We define a *successor relation* on lclc-extensions of the sets $\Gamma_i^*$ that gives rise to infinite paths of standard lclc-extensions. These infinite paths can be used to characterize or define PLTL-structures. Finally we show that at least one of those paths satisfies the suitable conditions for defining a model of $\Gamma$. Hence, this section is divided into a first subsection devoted to the notion of lclc-extensions of sets of clauses and their main properties, including the existence of a non-empty subclass of standard lclc-extensions for any locally consistent and TRS-closed set of clauses. In the second subsection, we define the notion of successor and prove the existence of infinite paths. Lastly, in the third subsection, we prove the existence of a model of $\Gamma$.

### 8.1 Extending Locally Consistent TRS-Closed Sets of Clauses

In this subsection we show that every TRS-closed set of clauses has at least one locally consistent extension that is literal-closed and standard. We gradually define the notions and prove the results.

**Definition 46** *A set of clauses $\Gamma$ is* literal-closed *iff $\Gamma \cap \mathsf{Lit}(C) \neq \emptyset$ for every $C \in \Gamma$.*[11] *Besides, $\mathsf{lclc}(\Gamma)$ denotes the collection of all locally consistent sets of clauses $\widehat{\Gamma}$ such that $\Gamma \subseteq \widehat{\Gamma} \subseteq \Gamma \cup \mathsf{Lit}(\Gamma)$ and $\widehat{\Gamma}$ is literal-closed. We say that each $\widehat{\Gamma} \in \mathsf{lclc}(\Gamma)$ is an* lclc-extension *of $\Gamma$.*

Note that if $\Box^b\bot$ is in $\Gamma$ then $\mathsf{lclc}(\Gamma) = \emptyset$ by local inconsistency. Besides, since only literals included in some clause in $\Gamma$ are used to build the elements in $\mathsf{lclc}(\Gamma)$, if no clause in $\Gamma$ includes any (basic) temporal literal (i.e. $\mathsf{BTL}(\Gamma) = \emptyset$, see Subsection 3.1) then every $\widehat{\Gamma} \in \mathsf{lclc}(\Gamma)$ also satisfies that $\mathsf{BTL}(\widehat{\Gamma}) = \emptyset$. In particular, if $\Gamma = \emptyset$ then $\mathsf{lclc}(\Gamma) = \{\emptyset\}$.

Next, we show that for every locally consistent set of clauses $\Gamma$ that does not contain (basic) temporal literals there exists at least one lclc-extension of $\Gamma$.

[11] Note that literals in $\mathsf{Lit}(C)$ are viewed as singleton clauses.

**Proposition 47** *If $\mathit{\Gamma}$ is a locally consistent set of clauses such that* $\mathsf{BTL}(\mathit{\Gamma}) = \emptyset$ *then* $\mathsf{lclc}(\mathit{\Gamma}) \neq \emptyset$.

*Proof* We will show that there exists a sequence $S = \Omega_0, \Omega_1, \Omega_2, \ldots, \Omega_g$ such that $g \geq 0$, $\Omega_0 = \mathit{\Gamma}$ and $\Omega_{h+1} = \Omega_h \cup \{L\}$ (for every $h \in \{0, \ldots, g-1\}$) for some $L \in \mathsf{Lit}(C)$ and some $C \in \Omega_h$ such that $\mathsf{Lit}(C) \cap \Omega_h = \emptyset$ and $\Omega_h \cup \{L\}$ is locally consistent. In addition, $\Omega_g \in \mathsf{lclc}(\mathit{\Gamma})$ whereas $\Omega_h \notin \mathsf{lclc}(\mathit{\Gamma})$ for all $h \in \{0, \ldots, g-1\}$. Since the number of clauses is finite, this inductive construction is also finite and shows that $\mathsf{lclc}(\mathit{\Gamma}) \neq \emptyset$.

We have to show that, for every $h$ such that $\Omega_h \notin \mathsf{lclc}(\mathit{\Gamma})$, there exists a locally consistent $\Omega_{h+1}$ that extends $\Omega_h$ with a new literal from some clause in $\mathit{\Gamma}$. Since $\Omega_h \notin \mathsf{lclc}(\mathit{\Gamma})$ there exists (at least one) clause $C = \Box^b(L_1 \vee \ldots \vee L_n) \in \Omega_h$ such that $L_i \notin \Omega_h$ for all $i \in \{1, \ldots, n\}$. Suppose that $\Omega_h \cup \{L_i\}$ is not locally consistent for all $i \in \{1, \ldots, n\}$. Then, by Proposition 26, there exists a local refutation $\mathcal{D}_i$ for $\Omega_h \cup \{L_i\}$ that is linear with respect to $L_i$, for every $i \in \{1, \ldots, n\}$. From these $n$ local refutations we are able to construct a local refutation $\mathcal{D}$ for $\Omega_h$ that is linear with respect to $C$, contradicting the assumption that $\Omega_h$ is locally consistent. Hence, $\Omega_h \cup \{L_i\}$ must be locally consistent for some $i \in \{1, \ldots, n\}$. ■

**Definition 48** *Let $\mathit{\Gamma}$ be a set of clauses such that* $\mathsf{lclc}(\mathit{\Gamma}) \neq \emptyset$ *and let* $\mathit{\Lambda} \subseteq \mathsf{Lit}(\mathit{\Gamma})$. *We say that $\mathit{\Lambda}$* represents *$\mathit{\Gamma}$ if $\widehat{\mathit{\Gamma}} \cap \mathit{\Lambda} \neq \emptyset$ for all $\widehat{\mathit{\Gamma}} \in \mathsf{lclc}(\mathit{\Gamma})$. If, in addition, for every $\mathit{\Lambda}' \subsetneq \mathit{\Lambda}$ there exists $\widehat{\mathit{\Gamma}} \in \mathsf{lclc}(\mathit{\Gamma})$ such that $\widehat{\mathit{\Gamma}} \cap \mathit{\Lambda}' = \emptyset$, then we say that $\mathit{\Lambda}$* minimally represents *$\mathit{\Gamma}$.*

The following result shows that the minimal representatives of a TRS-closed set of clauses $\mathit{\Gamma}$ are included (as clauses) in $\mathit{\Gamma}$.

**Proposition 49** *For every $\mathit{\Lambda}$ that minimally represents a non-empty locally consistent* TRS-*closed set of clauses $\mathit{\Gamma}$ there is a clause $C \in \mathit{\Gamma}$ such that* $\mathsf{Lit}(C) = \mathit{\Lambda}$.

*Proof* First we will show that $\mathit{\Gamma}$ must contain at least one clause $C$ such that $\mathsf{Lit}(C) \subseteq \mathit{\Lambda}$. We partition $\mathit{\Gamma}$ into the following two sets:

$$\mathit{\Pi}_1 = \{C \in \mathit{\Gamma} \mid \mathsf{Lit}(C) \cap \mathit{\Lambda} = \emptyset\}$$
$$\mathit{\Pi}_2 = \{C \in \mathit{\Gamma} \mid \mathsf{Lit}(C) \cap \mathit{\Lambda} \neq \emptyset\}$$

We split the clauses in $\mathit{\Pi}_2$ into the sub-clauses formed by literals that do not appear in $\mathit{\Lambda}$ and the sub-clauses formed by literals that appear in $\mathit{\Lambda}$. These sets of clauses respectively are the following sets $\mathit{\Sigma}_1$ and $\mathit{\Sigma}_2$.

$$\mathit{\Sigma}_1 = \{N \mid \Box^b(N \vee N') \in \mathit{\Pi}_2, \mathsf{Lit}(N) \cap \mathit{\Lambda} = \emptyset \text{ and } \mathsf{Lit}(N') \subseteq \mathit{\Lambda}\}$$
$$\mathit{\Sigma}_2 = \{N' \mid \Box^b(N \vee N') \in \mathit{\Pi}_2, \mathsf{Lit}(N) \cap \mathit{\Lambda} = \emptyset \text{ and } \mathsf{Lit}(N') \subseteq \mathit{\Lambda}\}$$

Since $\mathit{\Gamma}$ is locally consistent, $\mathit{\Pi}_1$, $\mathit{\Pi}_2$ and also their proper subsets are locally consistent. In addition, $\mathit{\Gamma}$ is TRS-closed, hence $\mathsf{BTL}(\mathit{\Gamma}) = \emptyset$ and every set of clauses considered along the rest of this proof does not contain any clause that includes any (basic) temporal literal.

Now we show, by contradiction, that $\bot \in \mathit{\Pi}_1 \cup \mathit{\Sigma}_1$ and, since $\mathit{\Pi}_1$ is locally consistent, it follows that $\bot \in \mathit{\Sigma}_1$ and, consequently, there exists a clause $C \in \mathit{\Gamma}$ such that $\mathsf{Lit}(C) \subseteq \mathsf{Lit}(\mathit{\Sigma}_2)$, i.e., $\mathsf{Lit}(C) \subseteq \mathit{\Lambda}$.

Let us suppose that $\bot \notin \mathit{\Pi}_1 \cup \mathit{\Sigma}_1$. First, suppose that $\mathit{\Pi}_1 \cup \mathit{\Sigma}_1$ is locally consistent. By Proposition 47, the set $\mathsf{lclc}(\mathit{\Pi}_1 \cup \mathit{\Sigma}_1)$ is non-empty and for every $\mathit{\Psi} \in \mathsf{lclc}(\mathit{\Pi}_1 \cup \mathit{\Sigma}_1)$ the set $\Omega = \mathit{\Gamma} \cup \{L \mid L \in \mathit{\Psi}\}$ is in $\mathsf{lclc}(\mathit{\Gamma})$ and satisfies $\Omega \cap \mathit{\Lambda} = \emptyset$. This contradicts that $\mathit{\Lambda}$ minimally represents $\mathit{\Gamma}$.

Second, suppose that $\mathit{\Pi}_1 \cup \mathit{\Sigma}_1$ is locally inconsistent, there exists some minimal locally inconsistent subset $\mathit{\Phi}$ of $\mathit{\Pi}_1 \cup \mathit{\Sigma}_1$ (i.e. $\mathit{\Phi}$ does not contain locally inconsistent proper subsets of $\Pi_1 \cup \Sigma_1$). Since every subset of $\Pi_1$ is locally consistent, then $\Phi \cap \Sigma_1 \neq \emptyset$. Let $N$ be any clause in $\Phi \cap \Sigma_1$. By Proposition 26, there exists a local refutation $\mathcal{D}$ for $\Phi$ that is linear with respect to $N$. By using the original clauses in $\Pi_2$ instead of their sub-clauses in $\Phi \cap \Sigma_1$, we can build from $\mathcal{D}$ a derivation $\mathcal{D}'$ whose last set contains a clause $C$ such that $\mathsf{Lit}(C) \subseteq \mathsf{Lit}(\Sigma_2)$. Hence, $\bot \in \Sigma_1$ and this contradicts that $\bot \notin \Pi_1 \cup \Sigma_1$.

So, since considering $\bot \notin \Pi_1 \cup \Sigma_1$ leads to a contradiction when we consider that $\Pi_1 \cup \Sigma_1$ is locally consistent and when we consider that $\Pi_1 \cup \Sigma_1$ is locally inconsistent, it follows that $\bot \in \Pi_1 \cup \Sigma_1$. Therefore $\bot \in \Sigma_1$ because $\Pi_1$ is locally consistent and, consequently, there are a clause $C \in \Gamma$ such that $\mathsf{Lit}(C) \subseteq \Lambda$.

Finally, $\mathsf{Lit}(C)$ cannot be a proper subset of $\Lambda$ because $\mathsf{Lit}(C)$ also represents $\Gamma$ and that would contradict the minimality of the representation of $\Gamma$ by $\Lambda$ (see Definition 48). Henceforth, $\mathsf{Lit}(C) = \Lambda$. ∎

Next we introduce the notion of *standard* lclc-extensions of a set of clauses.

**Definition 50** *Let $\Gamma$ be a locally consistent* TRS*-closed set of clauses. We say that $\widehat{\Gamma} \in \mathsf{lclc}(\Gamma)$ is* standard *iff it satisfies the following conditions:*

*(a) If $\circ L \in \widehat{\Gamma}$, then there exists a clause $\Box^b(\circ L \vee \circ N) \in \Gamma$*
*(b) For every propositional literal $P \in \mathsf{Lit}(\Gamma)$, if $\widehat{\Gamma} \cup \{P\}$ is locally consistent, then $P \in \widehat{\Gamma}$.*
*(c) If $\circ L \in \widehat{\Gamma}$, then $\widehat{\Gamma} \setminus \{\circ L\}$ is not literal-closed.*

The following lemma ensures the existence of at least one standard lclc-extension of any locally consistent TRS-closed set of clauses.

**Lemma 51** *Let $\Gamma$ be a locally consistent* TRS*-closed set of clauses. There exists at least one standard set in $\mathsf{lclc}(\Gamma)$.*

*Proof* We first prove that there exists $\Omega \in \mathsf{lclc}(\Gamma)$ that satisfies item (a) in Definition 50. Second, we show that there exists $\Sigma \supseteq \Omega$ such that $\Sigma \in \mathsf{lclc}(\Gamma)$ and satisfies (a) and (b) in Definition 50. Third, we show that there exists $\Delta \subseteq \Sigma$ such that $\Delta \in \mathsf{lclc}(\Gamma)$ and satisfies (a), (b) and (c) in Definition 50.

1. By Proposition 47, $\mathsf{lclc}(\Gamma)$ is non-empty. Now, let us suppose that for every set in $\mathsf{lclc}(\Gamma)$ there exists a literal of the form $\circ L$ such that $\circ L \notin \mathsf{Lit}(\Box^b \circ N)$ for every clause $\Box^b \circ N \in \Gamma$. Then, for every $\widehat{\Gamma} \in \mathsf{lclc}(\Gamma)$, there exists some $L \in \widehat{\Gamma}$ that belongs to the following set
$$\Psi = \{\circ L \in \mathsf{Lit}(\Gamma) \mid \circ L \notin \mathsf{Lit}(\Box^b \circ N) \text{ for every clause } \Box^b \circ N \in \Gamma\}$$
Hence $\Psi$ represents $\Gamma$ and there should exist some $\Lambda \subseteq \Psi$ that minimally represents $\Gamma$. Therefore, by Proposition 49, there exists a clause $C \in \Gamma$ such that $\mathsf{Lit}(C) = \Lambda$. This is a contradiction because the literals in $\Psi$, and in particular the literals in $\Lambda$, do not belong to any clause of the form $\Box^b \circ N$ in $\Gamma$. Therefore, there exists some set $\Omega$ in $\mathsf{lclc}(\Gamma)$ that satisfies Definition 50(a).
2. Since $\Omega$ is locally consistent and $\mathsf{BTL}(\Omega) = \emptyset$, the sequence $\Omega_0, \Omega_1, \Omega_2, \ldots, \Omega_g$ in the proof of Proposition 47 is easily adapted for ensuring that each $\Omega_i$ satisfies Definition 50(a) and that $\Omega_g$ satisfies Definition 50(b). So that $\Sigma = \Omega_g$.
3. We show that $\Sigma$ should contain a subset $\Delta$ that satisfies the lemma. Since $\Sigma$ belongs to $\mathsf{lclc}(\Gamma)$, verifies Definition 50(a) and (b) and is a finite set, we can ensure the existence of a finite sequence $\Sigma_0, \Sigma_1, \Sigma_2, \ldots, \Sigma_r$ such that $r \geq 0$, $\Sigma_0 = \Sigma$, $\Sigma_r \setminus \{\circ L\} \notin \mathsf{lclc}(\Gamma)$ for all $\circ L \in \Sigma_r$, and $\Sigma_{h+1} = \Sigma_h \setminus \{\circ L_h\}$ for some $\circ L_h \in \Sigma_h$ and $\Sigma_{h+1} \in \mathsf{lclc}(\Gamma)$ for every $h \in \{0, \ldots, r-1\}$. Therefore, $\Sigma_h$ satisfies Definition 50(a) and (b) for all $h \in \{0, \ldots, r\}$ and $\Sigma_r$ additionally satisfies (c). Hence, $\Sigma_r$ is the set $\Delta$ we were looking for. ∎

For locally consistent TRS-closed sets, the subclass of their standard lclc-extensions represents the whole class of their lclc-extensions with respect to sets of next-literals in the sense shown by the following proposition.

**Proposition 52** *Let $\Gamma$ be any locally consistent* TRS*-closed set of clauses and $\Lambda \subseteq \mathsf{Lit}(\Gamma)$ be a set such that every literal in $\Lambda$ is of the form $\circ L$. If $\widehat{\Gamma} \cap \Lambda \neq \emptyset$ for every standard set $\widehat{\Gamma} \in \mathsf{lclc}(\Gamma)$, then $\Lambda$ represents $\Gamma$.*

*Proof* Consider any $\Lambda$ that satisfies the hypothesis but does not represent $\Gamma$. Hence, there exists some non-standard set $\Psi \in \mathsf{lclc}(\Gamma)$ such that $\Psi \cap \Lambda = \emptyset$. Now, let

$$\Pi = \{N \mid \Box^b(N \vee N') \in \Gamma, \mathsf{Lit}(N) \cap \Lambda = \emptyset \text{ and } \mathsf{Lit}(N') \subseteq \Lambda\}$$
$$\Phi = \{N \in \Pi \mid \text{ no clause in } \Pi \text{ subsumes } N\}$$

Then, $\Phi$ is TRS-closed and locally consistent. The former holds because $\Gamma$ is TRS-closed. For the latter suppose that $\Phi$ is not locally consistent. By Proposition 22, $\bot \in \Phi$. Hence, by definition of $\Phi$, there exists a clause $C \in \Gamma$ such that $\mathsf{Lit}(C) \subseteq \Lambda$. But this contradicts the assumption $\Psi \cap \Lambda = \emptyset$ because $\Psi$ is an lclc-extension of $\Gamma$ and, consequently, $\mathsf{Lit}(C) \cap \Psi$ cannot be empty.

Since $\Phi$ is TRS-closed and locally consistent, by Lemma 51, there is some $\Omega \in \mathsf{lclc}(\Phi)$ that is standard. Hence, consider $\Sigma = \Gamma \cup \{L \mid L \in \Omega\}$ for some standard $\Omega \in \mathsf{lclc}(\Phi)$. First, $\Sigma$ is an lclc-extension of $\Gamma$ because $\mathsf{Lit}(\Omega) \subseteq \mathsf{Lit}(\Gamma)$ and because for every clause $C \in \Gamma$ there exists a clause $N \in \Phi$ such that $\mathsf{Lit}(N) \subseteq \mathsf{Lit}(C)$. Second, $\Sigma$ is standard because $\Omega$ is a standard lclc-extension of $\Phi$ and $\Lambda$ contains only literals of the form $\circ L$, so that $\Sigma$ satisfies Definition 50. Consequently, $\Sigma$ is a standard lclc-extension of $\Gamma$ such that $\Sigma \cap \Lambda = \emptyset$. This contradicts that $\widehat{\Gamma} \cap \Lambda \neq \emptyset$ for all standard $\widehat{\Gamma} \in \mathsf{lclc}(\Gamma)$. Therefore, $\Lambda$ represents $\Gamma$. ∎

## 8.2 Building Infinite Paths of Standard Lclc-Extensions

In order to build sequences of standard lclc-extensions of the TRS-closed sets $\Gamma_i^*$ –in the cycling derivation $\mathcal{D}(\Gamma)$– that represent models of $\Gamma$, such sequences must be coherent with respect to the meaning of temporal connectives. We mean that, e.g. if $\circ p$ belongs to a set $\Omega$ in the sequence, then $p$ must belong to the set that is the successor of $\Omega$ in the sequence. Similarly, for eventualities where also the selections performed along $\mathcal{D}(\Gamma)$ are relevant. As a consequence a successor relation is defined for the lclc-extensions of the TRS-closed sets that appear in the derivation $\mathcal{D}(\Gamma)$:

$$(\Gamma_0, \Gamma_0^*) \mapsto (\Gamma_1, \Gamma_1^*) \mapsto \ldots \mapsto (\Gamma_j, \Gamma_j^*) \mapsto \ldots \mapsto (\Gamma_k, \Gamma_k^*)$$

which is cycling with respect to $j$ and $k$. This successor relation on

$$\{\mathsf{lclc}(\Gamma_i^*) \times \mathsf{lclc}(\Gamma_{i+1}^*) \mid 0 \leq i < k\} \cup (\mathsf{lclc}(\Gamma_k^*) \times \mathsf{lclc}(\Gamma_j^*))$$

is presented in Definition 53. Along the rest of this paper, $\widehat{\Gamma_i^*}$ denotes a member of $\mathsf{lclc}(\Gamma_i^*)$.

**Definition 53** *Let $i = h + 1$ if $h \in \{0, \ldots, k-1\}$ and let $i = j$ if $h = k$, we say that $\widehat{\Gamma_i^*}$ is a* successor *of $\widehat{\Gamma_h^*}$ or that $\widehat{\Gamma_h^*}$ is a* predecessor *of $\widehat{\Gamma_i^*}$ if for every $\circ L \in \widehat{\Gamma_h^*}$ there is some $S \in \mathsf{nxclo}_i(\circ L)$ such that $S \subseteq \widehat{\Gamma_i^*}$, where $\mathsf{nxclo}_i$ is defined as follows*

– $\mathsf{nxclo}_i(\circ P) = \{\{P\}\}$ *where $P$ is a propositional literal.*

- $\mathsf{nxclo}_i(\circ\circ L) = \{\{\circ L\}\}$
- $\mathsf{nxclo}_i(\circ(P_1\,\mathcal{U}\,P_2)) = \begin{cases} \{\{P_2\}, \{P_1, \circ(P_1\,\mathcal{U}\,P_2)\}\} & \textit{if } P_1\,\mathcal{U}\,P_2 \notin \mathsf{sel\_ev\_set}_i \\ \{\{P_2\}, \{P_1, \circ(a\,\mathcal{U}\,P_2)\}\} & \textit{otherwise} \\ \textit{where } a\,\mathcal{U}\,P_2 \in \mathsf{sel\_ev\_set}_i^* & \end{cases}$
- $\mathsf{nxclo}_i(\circ\Diamond\,P) = \begin{cases} \{\{P\}, \{\circ\Diamond\,P\}\} & \textit{if } \circ\Diamond\,P \notin \mathsf{sel\_ev\_set}_i \\ \{\{P\}, \{\circ(a\,\mathcal{U}\,P)\}\} & \textit{otherwise} \\ \textit{where } a\,\mathcal{U}\,P \in \mathsf{sel\_ev\_set}_i^* & \end{cases}$
- $\mathsf{nxclo}_i(\circ(P_1\,\mathcal{R}\,P_2)) = \{\{P_2, P_1\}, \{P_2, \circ(P_1\,\mathcal{R}\,P_2)\}\}$
- $\mathsf{nxclo}_i(\circ\Box\,P) = \{\{P, \Box\,P\}, \{P, \circ\Box\,P\}\}$.

*The set of successors of a given set* $\widehat{\Gamma_h^*}$ *is denoted by* $\mathsf{succ}(\widehat{\Gamma_h^*})$.

The definition of $\mathsf{nxclo}_i(\circ\Box\,P)$ arises from the fact that the literal $\circ\Box\,P$ can be either a singleton now-clause or a literal properly contained in a clause $C$. In the first case, $\Gamma_i$ contains the always-clause $\Box\,P$ which will not be affected by the rule $(\Box\,Fix)$. Consequently, in such a case $\Gamma_i^*$ contains necessarily $\Box\,P$. However, in the second case, the literal $\circ\Box\,P$ is introduced by application of the rule $(\Box\,Fix)$ to the clause $C$.

The existence of infinite paths of standard lclc-extensions is based on the existence of a predecessor for each standard lclc-extension of a TRS-closed set in the derivation which is a standard lclc-extension of the previous TRS-closed set in the derivation.

**Proposition 54** *For every* $i \in \{1, \ldots, k\}$ *and every standard* $\widehat{\Gamma_i^*} \in \mathsf{lclc}(\Gamma_i^*)$*, there exists a standard* $\widehat{\Gamma_{i-1}^*} \in \mathsf{lclc}(\Gamma_{i-1}^*)$ *such that* $\widehat{\Gamma_i^*} \in \mathsf{succ}(\widehat{\Gamma_{i-1}^*})$*.*

*Proof* Let $W_\ell = \{\widehat{\Gamma_\ell^*} \in \mathsf{lclc}(\Gamma_\ell^*) \mid \widehat{\Gamma_\ell^*}$ is standard $\}$ for each $\ell \in \{0, \ldots, k\}$. If there exists some $\widehat{\Gamma_{i-1}^*} \in W_{i-1}$ such that $\widehat{\Gamma_{i-1}^*}$ does not contain any clause of the form $\circ L$, then $\widehat{\Gamma_i^*} \in \mathsf{succ}(\widehat{\Gamma_{i-1}^*})$ for all $\widehat{\Gamma_i^*}$. Otherwise, every set $\widehat{\Gamma_{i-1}^*} \in W_{i-1}$ contains at least one clause of the form $\circ L$. We proceed by contradiction. Let us suppose that $\widehat{\Gamma_i^*}$ is a member of $W_i$ such that $\widehat{\Gamma_i^*} \notin \mathsf{succ}(\widehat{\Gamma_{i-1}^*})$ for all $\widehat{\Gamma_{i-1}^*} \in W_{i-1}$. Hence, there exists at least one $\circ L$ in every $\widehat{\Gamma_{i-1}^*} \in W_{i-1}$ such that $S \not\subseteq \widehat{\Gamma_i^*}$ for all $S \in \mathsf{nxclo}_i(\circ L)$. Therefore, the set

$$\Lambda = \{\circ L \mid \circ L \in \textstyle\bigcup_{\widehat{\Gamma_{i-1}^*} \in W_{i-1}} \widehat{\Gamma_{i-1}^*} \text{ such that } S \not\subseteq \widehat{\Gamma_i^*} \text{ for all } S \in \mathsf{nxclo}_i(\circ L)\}$$

satisfies that $\Lambda \cap \widehat{\Gamma_{i-1}^*} \neq \emptyset$ for all $\widehat{\Gamma_{i-1}^*} \in W_{i-1}$. Therefore, by Proposition 52, $\Lambda$ represents $\Gamma_{i-1}^*$ and, consequently there exists some set $\Omega \subseteq \Lambda$ that minimally represents $\Gamma_{i-1}^*$. By Proposition 49, there exists a clause $C = \Box^b(\circ L_1 \vee \ldots \vee \circ L_r)$ in $\Gamma_{i-1}^*$ such that $\mathsf{Lit}(C) = \Omega$ and $r \geq 1$. Since $\mathsf{unnext}(\{C\}) \subseteq \Gamma_i$, then the clause $C' = L_1 \vee \ldots \vee L_r$ is in $\Gamma_i$. Now, let

$$\{S_1, \ldots, S_n\} = \bigcup_{g=1}^{r} \mathsf{nxclo}_i(\circ L_g)$$

(note that $n \geq 1$) and let $\{C_1, \ldots, C_m\}$ be the set of all clauses of the form $L_1 \vee \ldots \vee L_n$ such that $L_h \in S_h$ for all $h \in \{1, \ldots, n\}$. By subsumption, $\Gamma_i^*$ contains a non-empty set of (non-empty) clauses $\{D_1, \ldots, D_m\}$ such that $\mathsf{Lit}(D_t) \subseteq \mathsf{Lit}(C_t)$ for all $t \in \{1, \ldots, m\}$.

By construction $S \not\subseteq \widehat{\Gamma_i^*}$ for all $S \in \mathsf{nxclo}_i(\circ L_g)$ and all $g \in \{1, \ldots, r\}$. Hence, for each pair $(g, S)$ such that $g \in \{1, \ldots, r\}$ and $S \in \mathsf{nxclo}_i(\circ L_g)$, we can choose at least one literal $L$ such that $L \in S$ and $L \notin \widehat{\Gamma_i^*}$. As a consequence, there exists a clause $D_t \in \Gamma_i^*$ with $t \in \{1, \ldots, m\}$ such that $\mathsf{Lit}(D_t) \subseteq \mathsf{Lit}(C_t)$ where $D_t \cap \widehat{\Gamma_i^*} = \emptyset$. This contradicts the fact that $\widehat{\Gamma_i^*}$ contains at least one literal from each clause in $\Gamma_i^*$. ∎

**Proposition 55** *For every $i \in \{1, \ldots, k\}$ and every standard $\widehat{\Gamma_i^*}$, there exists a sequence $\widehat{\Gamma_0^*}, \widehat{\Gamma_1^*}, \ldots, \widehat{\Gamma_i^*}$ of standard sets such that $\widehat{\Gamma_h^*} \in \mathsf{succ}(\widehat{\Gamma_{h-1}^*})$ for every $h \in \{1, \ldots, i\}$.*

*Proof* By Lemma 51 and Proposition 54. ∎

**Proposition 56** *For every standard $\widehat{\Gamma_j^*}$ there exists at least one standard $\widehat{\Gamma_k^*}$ such that $\widehat{\Gamma_j^*} = \mathsf{succ}(\widehat{\Gamma_k^*})$.*

*Proof* The proof is very similar to the one of Proposition 54, but using that $\mathsf{now}(\Gamma_j) = \mathsf{now}(\mathsf{unnext}(\Gamma_k^*))$ instead of $\Gamma_i = \mathsf{unnext}(\Gamma_{i-1}^*)$ and also using the fact that the set $\{N \mid \Box \circ N \in \Gamma_k^*\}$ is contained into the set $\mathsf{now}(\mathsf{unnext}(\Gamma_k^*))$ (by definition of the $\mathsf{unnext}$ operator). ∎

Now, we are going to construct a pre-model of $\Gamma$ by means of sequences of standard lclc-extensions of the sets in $\mathcal{D}(\Gamma)$ which will be ordered by the successor relation. For that, we need some notation on such sequences. For $g$ and $h$, where $0 \leq g \leq h \leq k$, we denote by $\mathcal{D}(\Gamma)_{[g..h]}$, the set of all *intervals* of standard lclc-extensions $\widehat{\Gamma_g^*}, \widehat{\Gamma_{g+1}^*}, ..., \widehat{\Gamma_h^*}$ such that $\widehat{\Gamma_i^*} \in \mathsf{succ}(\widehat{\Gamma_{i-1}^*})$ for every $i \in \{g+1, \ldots, h\}$. The functions $\mathsf{first}$ and $\mathsf{last}$ respectively return the first and the last set of a given interval. We use superscripts notation to denote subsequences of an interval $s \in \mathcal{D}(\Gamma)_{[g..h]}$ as follows. For $n$ and $m$ such that $g \leq n \leq m \leq h$, the subsequence $s^{n..m}$ denotes the subsequence formed by the sets $\widehat{\Gamma_n^*}, \widehat{\Gamma_{n+1}^*}, \ldots, \widehat{\Gamma_m^*}$ of $s$. In particular, if $n = m$ we write $s^n$ instead of $s^{n..n}$ and intentionally confuse the sequence of one set with the set itself. For $s \in \mathcal{D}(\Gamma)_{[g..h]}$, we denote by $\mathsf{range}(s)$ the set of natural numbers $\{n \mid g \leq n \leq h\}$. Since $\mathcal{D}(\Gamma)$ is cycling with respect to $j$ and $k$, the two sets of intervals $\mathcal{D}(\Gamma)_{[0..j-1]}$ and $\mathcal{D}(\Gamma)_{[j..k]}$ are respectively called *initial* and *inner*. Note that, since $j$ could be 0, the set $\mathcal{D}(\Gamma)_{[0..j-1]}$ could be empty, but $\mathcal{D}(\Gamma)_{[j..k]}$ is non-empty for any $\mathcal{D}(\Gamma)$.

**Proposition 57** *For each standard $\widehat{\Gamma_j^*}$ there exists $s \in \mathcal{D}(\Gamma)_{[j..k]}$ such that $\widehat{\Gamma_j^*} \in \mathsf{succ}(\mathsf{last}(s))$.*

*Proof* By Propositions 55 and 56. ∎

Note that in the above proposition $\widehat{\Gamma_j^*}$ and $\mathsf{first}(s)$ can be different.

Now, we define when a sequence of elements from $\mathcal{D}(\Gamma)_{[j..k]}$ forms a cycle, which is called a $\mathcal{D}(\Gamma)$-cycle. Then we prove that there exists at least one $\mathcal{D}(\Gamma)$-cycle.

**Definition 58** *A $\mathcal{D}(\Gamma)$-cycle is a finite non-empty sequence $s_0, s_1, \ldots, s_n$ such that*

*(i)* $s_i \in \mathcal{D}(\Gamma)_{[j..k]}$ *for all* $i \in \{0, \ldots, n\}$
*(ii)* $\mathsf{first}(s_{i+1}) \in \mathsf{succ}(\mathsf{last}(s_i))$ *for all* $i \in \{0, \ldots, n-1\}$ *and*
*(iii)* $\mathsf{first}(s_0) \in \mathsf{succ}(\mathsf{last}(s_n))$.

**Proposition 59** *There exists at least one $\mathcal{D}(\Gamma)$-cycle.*

*Proof* By Lemma 51, there exists at least one standard set in $\mathsf{lclc}(\Gamma_j^*)$. Let us consider any standard $\widehat{\Gamma_j^*}$ in $\mathsf{lclc}(\Gamma_j^*)$. By Proposition 57, there exists an interval $r_0 \in \mathcal{D}(\Gamma)_{[j..k]}$ such that $\widehat{\Gamma_j^*} \in \mathsf{succ}(\mathsf{last}(r_0))$. Additionally, by repeatedly applying Proposition 57, we can build an infinite sequence of intervals $r_0, r_1, \ldots$ in $\mathcal{D}(\Gamma)_{[j..k]}$ such that $\mathsf{first}(r_{i-1}) \in \mathsf{succ}(\mathsf{last}(r_i))$ for every $i \geq 1$. Since $\mathcal{D}(\Gamma)_{[j..k]}$ is finite, $r_g = r_h$ must hold for some $g$ and $h$ such that $0 \leq g < h$. Then, the reverse of the sequence $r_g, \ldots, r_{h-1}$, i.e. the sequence $r_{h-1}, \ldots, r_g$ is a $\mathcal{D}(\Gamma)$-cycle. ∎

Note that the minimal cycles consist of exactly one interval $s \in \mathcal{D}(\Gamma)_{[j..k]}$ such that $\mathsf{first}(s) \in \mathsf{succ}(\mathsf{last}(s))$.

## 8.3 Model Existence

In this subsection we prove that there exists at least one model of $\Gamma$ on the basis of the cycling derivation $\mathcal{D}(\Gamma)$. First, we define a graph structure $\mathcal{G}_{\mathcal{D}(\Gamma)}$ whose nodes are intervals in $\mathcal{D}(\Gamma)_{[0..j-1]}$ and $\mathcal{D}(\Gamma)_{[j..k]}$. There is a (directed) edge $(s, s')$ in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ whenever $\mathsf{first}(s') \in \mathsf{succ}(\mathsf{last}(s))$. Note that every node in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ is related to a node from $\mathcal{D}(\Gamma)_{[j..k]}$. Second, we define a notion of self-fulfilling path in this graph. Then, we prove that $\mathcal{G}_{\mathcal{D}(\Gamma)}$ contains at least one strongly connected component (a $\mathcal{D}(\Gamma)$-cycle) that is self-fulfilling. Finally, we define a model of $\Gamma$ on the basis of this strongly connected component in $\mathcal{G}_{\mathcal{D}(\Gamma)}$.

**Definition 60** *We associate to $\mathcal{D}(\Gamma)$ the graph $\mathcal{G}_{\mathcal{D}(\Gamma)}$ that is formed by the following set of nodes $S_{\mathcal{D}(\Gamma)}$ and the following edge-relation $R_{\mathcal{D}(\Gamma)}$ on $S_{\mathcal{D}(\Gamma)}$:*

- $S_{\mathcal{D}(\Gamma)} = \mathcal{D}(\Gamma)_{[0..j-1]} \cup \mathcal{D}(\Gamma)_{[j..k]}$
- $sR_{\mathcal{D}(\Gamma)}s'$ *iff* $s' \in \mathcal{D}(\Gamma)_{[j..k]}$ *and* $\mathsf{first}(s') \in \mathsf{succ}(\mathsf{last}(s))$.

Paths and strongly connected components in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ are defined as usual in graph theory. The notion of $\mathcal{D}(\Gamma)$-cycle (see Definition 58) has an obvious extension to $\mathcal{G}_{\mathcal{D}(\Gamma)}$. Therefore, by Proposition 59, the graph $\mathcal{G}_{\mathcal{D}(\Gamma)}$ has at least one cycle. The minimal graphs $\mathcal{G}_{\mathcal{D}(\Gamma)}$ consist of exactly one node $n$ with one edge from $n$ to $n$.

We would like to remark that, from a locally consistent literal-closed set, interleaved unnext-steps and TRS-steps could yield a TRS-refutation. As a consequence, there could exist some interval $s$ in $S_{\mathcal{D}(\Gamma)}$ such that no $s' \in S_{\mathcal{D}(\Gamma)}$ satisfies $sR_{\mathcal{D}(\Gamma)}s'$ and, hence, there could exist lclc-extensions that do not belong to any interval in $S_{\mathcal{D}(\Gamma)}$.

The paths in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ are formed by standard lclc-extensions of TRS-closed sets which do not include any (basic) temporal literal. Consequently, any occurrence of an eventuality in the states of $\mathcal{G}_{\mathcal{D}(\Gamma)}$ must be preceded by a $\circ$ connective. This fact leads us to define the following notion of eventuality fulfillment in the paths of $\mathcal{G}_{\mathcal{D}(\Gamma)}$.

**Definition 61** *Let $\pi = s_0, s_1, \ldots$ be a path in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ such that $\circ(P_1 \mathcal{U} P_2) \in s_g^i$ for some $g \geq 0$ and $i \in \mathsf{range}(s_g)$. We say that $\pi$* fulfills *$\circ(P_1 \mathcal{U} P_2)$ iff either*

- *there exists $h \in \mathsf{range}(s_g)$ such that $h > i$, $P_2 \in s_g^h$ and $P_1 \in s_g^\ell$ for all $\ell \in \{i+1, \ldots, h-1\}$, or*
- *there exist $r > g$ and $h \in \mathsf{range}(s_r)$ such that $P_2 \in s_r^h$ and $P_1 \in s_z^\ell$ for all $(z, \ell)$ such that $g < z < r$ and $\ell \in \mathsf{range}(s_z)$ and $P_1 \in s_r^\ell$ for all $\ell \in \{j, \ldots, h-1\}$ and $P_1 \in s_g^\ell$ for all $\ell \in \{i+1, \ldots, m\}$ where $m$ is the maximum in $\mathsf{range}(s_g)$.*

*A path $\pi$ is self-fulfilling iff $\pi$ fulfills every $\circ(P_1 \mathcal{U} P_2)$ that occurs in any of its sets. Besides, a $\mathcal{D}(\Gamma)$-cycle $\sigma$ in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ is* self-fulfilling *if the path $\sigma^\omega$ is self-fulfilling.*

Since $\circ\Diamond P$ and $\circ(\widetilde{P}\,\mathcal{U}\,P)$ are equivalent, the fulfillment notion for $\circ\Diamond P$ is a particular case of Definition 61.

The next three propositions are auxiliary results about the fulfillment of eventualities, which are useful for proving the Lemma 65. Note that, by means of rule $(\Diamond\,Set)$, the literal $T$ in every selected $\circ T$ is always an until-formula. Consequently, in the next two propositions, only this kind of eventualities are considered.

**Proposition 62** *Let $s$ be an interval in $\mathcal{D}(\Gamma)_{[g..k]}$ for some $g \in \{0,\dots,k-1\}$. If $\circ(P_g\,\mathcal{U}\,P) \in s^g$ and $P_g\,\mathcal{U}\,P \in \mathsf{sel\_ev\_set}_{g+1}$, then $P \in s^i$ for some $i \in \{g+1,\dots,k\}$.*

*Proof* Let us suppose that $P \notin s^i$ for every $i \in \{g+1,\dots,k\}$. Then, since $s$ is an interval, $s^i \in \mathsf{succ}(s^{i-1})$ for every $i \in \{g+1,\dots,k\}$. Hence, by Definition 53, there exists a sequence of literals of the form $P_{g+1}\,\mathcal{U}\,P,\dots,P_k\,\mathcal{U}\,P$ such that $\mathsf{sel\_ev\_set}^*_h = \{P_h\,\mathcal{U}\,P\}$ for every $h \in \{g+1,\dots,k\}$ and $P_h\,\mathcal{U}\,P$ is the direct descendant of $P_{h-1}\,\mathcal{U}\,P$ in $\mathcal{D}(\Gamma)$ for every $h \in \{g+1,\dots,k\}$. Since $s^k$ is standard, by item (a) in Definition 50, there exists a clause of the form $\circ N \in \Gamma^*_k$ such that $\circ(P_k\,\mathcal{U}\,P) \in \mathsf{Lit}(\circ N)$. Consequently, since $\mathcal{D}(\Gamma)$ is a cycling derivation with respect to $j$ and $k$, there exists $N \in \Gamma_j$ such that $P_k\,\mathcal{U}\,P \in \mathsf{Lit}(N)$. This contradicts the fact that $P_k$ is (according to the rule $(\mathcal{U}\,Set)$) a fresh variable that cannot appear in the set $\Gamma_j$. ∎

**Proposition 63** *Let $s$ be an interval in $\mathcal{D}(\Gamma)_{[g..h]}$ for some $g$ and $h$ such that $0 \le g < h \le k-1$. If $\circ(P_g\,\mathcal{U}\,P) \in s^g$, $P_g\,\mathcal{U}\,P \in \mathsf{sel\_ev\_set}_{g+1}$ and $P \notin s^i$ for all $i \in \{g+1,\dots,h\}$, then $P_g \in s^i$ for all $i \in \{g+1,\dots,h\}$.*

*Proof* If $h = g+1$ then $P_g \in s^h$ because $s^h$ is a successor of $s^g$ (see Definition 53). Now, in the case of $h \ge g+2$, let us suppose that there exists some $r \in \{g+2,\dots,h\}$ such that $P_g \notin s^r$. Since $s$ is an interval, $s^\ell \in \mathsf{succ}(s^{\ell-1})$ for every $\ell \in \{g+1,\dots,h\}$. Hence, by Definition 53, there exists a sequence of literals of the form $P_{g+1}\,\mathcal{U}\,P,\dots,P_h\,\mathcal{U}\,P$ such that $P_\ell\,\mathcal{U}\,P$ is the direct descendant of $P_{\ell-1}\,\mathcal{U}\,P$ in $\mathcal{D}(\Gamma)$, $\mathsf{sel\_ev\_set}^*_\ell = \{P_\ell\,\mathcal{U}\,P\}$ and $\{P_{\ell-1}, \circ(P_\ell\,\mathcal{U}\,P)\} \subseteq s^\ell$ for every $\ell \in \{g+1,\dots,h\}$. Then, $P_{r-1} \in s^r$. Additionally, by construction of $\mathcal{D}(\Gamma)$, there exists either a clause of the form $C_i = \Box(\neg P_i \vee P_{i-1})$ or $C_i = \Box\neg P_i$ in $s^r$ for every $i \in \{g+1,\dots,r\}$.[12] Since we are supposing that $P_g \notin s^r$, then $\{\neg P_{g+1},\dots,\neg P_r\} \subseteq s^r$ must hold because $s^r$ is literal-closed. Then, $\neg P_{r-1}$ is also in $s^r$. Therefore $\{P_{r-1}, \neg P_{r-1}\} \subseteq s^r$, which contradicts the fact that $s^r$ is locally consistent. ∎

**Proposition 64** *Let $\pi = s_0, s_1, \dots, s_n$ be a $\mathcal{D}(\Gamma)$-cycle. If there exists a literal $\circ(P_0\,\mathcal{U}\,P) \in \mathsf{univlit}(\Gamma)$ such that $\circ(P_0\,\mathcal{U}\,P) \in s^i_\ell$ for some $\ell \in \{0,\dots,n\}$ and some $i \in \{j,\dots,k\}$, and the path $\pi^\omega$ does not fulfill $\circ(P_0\,\mathcal{U}\,P)$, then $P_0\,\mathcal{U}\,P \notin \mathsf{sel\_ev\_set}_g$ and $\{P_0, \circ(P_0\,\mathcal{U}\,P)\} \subseteq s^g_h$ for every $h \in \{0,\dots,n\}$ and every $g \in \{j,\dots,k\}$.*

*Proof* Since $\pi$ is a $\mathcal{D}(\Gamma)$-cycle and $\pi^\omega$ does not fulfill $\circ(P_0\,\mathcal{U}\,P)$, we can ensure, by Definitions 58, 53 and 61 that $P_0 \in s^g_h$ and $P \notin s^g_h$ for every $h \in \{0,\dots,n\}$ and every $g \in \{j,\dots,k\}$. Therefore, by using Proposition 62 and Proposition 63, we can ensure that $P_0\,\mathcal{U}\,P \notin \mathsf{sel\_ev\_set}_g$ for every $g \in \{j,\dots,k\}$, since otherwise $\pi^\omega$ would fulfill $\circ(P_0\,\mathcal{U}\,P)$. Consequently, by Definition 53 and Definition 58, we can ensure that $\{P_0, \circ(P_0\,\mathcal{U}\,P)\} \subseteq s^g_h$ for every $h \in \{0,\dots,n\}$ and every $g \in \{j,\dots,k\}$. ∎

Next, we prove that every $\mathcal{D}(\Gamma)$-cycle in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ is self-fulfilling. As a consequence, we know that there exists at least one self-fulfilling infinite path in the graph $\mathcal{G}_{\mathcal{D}(\Gamma)}$.

[12] The form of the clause respectively depends on whether the context is empty or not when the rule $(\mathcal{U}\,Set)$ is applied to $\Gamma_i$.

**Lemma 65** *For any cycling derivation $\mathcal{D}(\Gamma)$, the graph $\mathcal{G}_{\mathcal{D}(\Gamma)}$ contains at least one self-fulfilling $\mathcal{D}(\Gamma)$-cycle.*

*Proof* By Proposition 59 there is at least one $\mathcal{D}(\Gamma)$-cycle in $\mathcal{G}_{\mathcal{D}(\Gamma)}$. We show, by contradiction, that every $\mathcal{D}(\Gamma)$-cycle in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ is self-fulfilling. For that, let us suppose that there is a $\mathcal{D}(\Gamma)$-cycle $\pi = s_0, s_1, \ldots, s_n$ in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ that is non-self-fulfilling, i.e., the path $\pi^\omega$ does not fulfill a literal $\circ(P_0 \,\mathcal{U}\, P) \in s^i_\ell$ for some $\ell \in \{0, \ldots, n\}$ and some $i \in \{j, \ldots, k\}$. Then, by Proposition 64, $P_0 \,\mathcal{U}\, P \notin \mathsf{sel\_ev\_set}_g$ for every $g \in \{j, \ldots, k\}$ and $\{P_0, \circ(P_0 \,\mathcal{U}\, P)\} \subseteq s^i_\ell$ for every $\ell \in \{0, \ldots, n\}$ and every $i \in \{j, \ldots, k\}$. Since $s^g_h$ is standard for every $\ell \in \{0, \ldots, n\}$ and every $i \in \{j, \ldots, k\}$, we conclude that, for every $i \in \{j, \ldots, k\}$, the set $\Gamma^*_i$ contains a clause $C = \Box^b \circ N$ such that $\circ(P_0 \,\mathcal{U}\, P) \in \mathsf{Lit}(C)$ and, consequently, $P_0 \,\mathcal{U}\, P \in \mathsf{Lit}(\mathsf{now}(\Gamma_i))$ for every $i \in \{j, \ldots, k\}$. Therefore, by Definition 31(3), $\mathcal{D}(\Gamma)$ is not a cycling derivation, which is a contradiction. ∎

The particular case of Lemma 65 for eventualities of the form $\diamond P$ follows easily.

Next, we introduce pre-models as a kind of paths along $\mathcal{G}_{\mathcal{D}(\Gamma)}$.

**Definition 66** $\mathsf{PMod}(\mathcal{G}_{\mathcal{D}(\Gamma)})$ *is the collection of all finite paths $\pi = s_0, s_1, s_2, \ldots, s_n$ in $\mathcal{G}_{\mathcal{D}(\Gamma)}$ such that*

*(a)* $s_0 \in \mathcal{D}(\Gamma)_{[0..j-1]}$ *and* $\sigma = s_1, s_2, \ldots, s_n \in \mathsf{cycles}(\mathcal{G}_{\mathcal{D}(\Gamma)})$*, if* $\mathcal{D}(\Gamma)_{[0..j-1]} \neq \emptyset$
*(b)* $\pi = s_0, s_1, \ldots, s_n \in \mathsf{cycles}(\mathcal{G}_{\mathcal{D}(\Gamma)})$*, if* $\mathcal{D}(\Gamma)_{[0..j-1]} = \emptyset$

*where* $\mathsf{cycles}(\mathcal{G}_{\mathcal{D}(\Gamma)})$ *is the collection of all the self-fulfilling cycles in* $\mathcal{G}_{\mathcal{D}(\Gamma)}$.

As a direct consequence of Propositions 55 and 59 and Lemma 65, there exists at least one pre-model in the graph $\mathcal{G}_{\mathcal{D}(\Gamma)}$.

**Proposition 67** $\mathsf{PMod}(\mathcal{G}_{\mathcal{D}(\Gamma)})$ *is non-empty.* ∎

Finally, the above pre-model allows us to construct a model of $\Gamma$. This proves the completeness of our TRS-resolution system.

**Theorem 68** *For any set of clauses $\Gamma$, if $\Gamma$ is unsatisfiable then there exists a* TRS*-refutation for $\Gamma$.*

*Proof* Suppose that there is no TRS-refutation for $\Gamma$, then the algorithm $\mathcal{SR}$ in Fig. 7 produces a cycling derivation $\mathcal{D}(\Gamma)$. By Proposition 67, there exists a pre-model $\pi = s_0, s_1, s_2, \ldots, s_n$ in $\mathsf{PMod}(\mathcal{G}_{\mathcal{D}(\Gamma)})$. If $\mathcal{D}(\Gamma)_{[0..j-1]} = \emptyset$ we define $\sigma$ as the infinite path $\pi^\omega$. Otherwise $\sigma = s_0 \cdot \rho^\omega$ where $\rho = s_1, s_2, \ldots, s_n$. Now, we define the PLTL-structure $\mathcal{M}_\sigma = (\sigma, V_{\mathcal{M}_\sigma})$ where the states are the standard lclc-extensions that form the intervals in $\sigma$ which can be seen as

$$\Omega^0_0, \ldots, \Omega^r_0, \Omega^j_1, \ldots, \Omega^k_1, \Omega^j_2, \ldots, \Omega^k_2, \ldots, \Omega^j_n, \ldots, \Omega^k_n, \Omega^j_\ell, \ldots, \Omega^k_\ell, \ldots$$

where $r = j-1$ and $\ell = 1$ if $\mathcal{D}(\Gamma)_{[0..j-1]} \neq \emptyset$, whereas $r = k$ and $\ell = 0$ if $\mathcal{D}(\Gamma)_{[0..j-1]} = \emptyset$. Additionally, $\Omega^g_h$ is in $\mathsf{lclc}(\Gamma^*_g)$ and $V_{\mathcal{M}_\sigma}(\Omega^g_h) = \{p \in \mathsf{Prop} \mid p \in \Omega^g_h\}$ for every $g \in \{0, \ldots, k\}$ and every $h \in \{0, \ldots, n\}$. It is routine to see that $\langle \mathcal{M}_\sigma, \Omega^i_h \rangle \models C$ holds for all $C \in \Gamma^*_i$. Since any lclc-extension contains at least one literal of $C$, this is made by structural induction on the form of the literal and using Definition 53 and the fact that $\sigma$ is self-fulfilling (by Lemma 65). In particular, $\mathcal{M}_\sigma$ is a model of $\Gamma^*_0$ and, by Propositions 27 and 28, the set $\Gamma_0$ is satisfiable. Hence, since $\Gamma = \Gamma_0$, the set of clauses $\Gamma$ is satisfiable. ∎

## 9 Related Work

In this section we describe the contributions in the literature that are more closely related to our approach to clausal temporal resolution. First, we explain the relation with the tableau method ([17,19]) that inspired TRS-resolution. And then, we discuss and compare the four clausal resolution methods ([8,1,36,12]) that are more similar to TRS-resolution.

### 9.1 The TTM Tableau Method [17,19]

The TRS-resolution method is strongly inspired in the TTM tableau method introduced in [17,19]. Indeed, the TRS-rule $(\mathcal{U}\, Set)$ is a clausal variant of the TTM-rule $(\mathcal{U})_2$. In [18,19], the idea behind the rule $(\mathcal{U})_2$ is used for achieving cut-freeness (in particular, invariant-freeness) in the framework of sequent calculi for PLTL. In [19], a cut-free sequent calculus that is dual to the one-pass tableau method TTM is presented.

The crucial point –in both rules $(\mathcal{U})_2$ and $(\mathcal{U}\, Set)$– is the fact that whenever a set of formulas $\Delta \cup \{\varphi \mathcal{U} \psi\}$ is satisfiable, there must exist a model $\mathcal{M}$ (with states $s_0, s_1, \ldots$) that is minimal in the following sense:

$$\mathcal{M} \textit{ satisfies either } \Delta \cup \{\psi\} \textit{ or } \Delta \cup \{\varphi, \circ((\varphi \wedge \neg\Delta) \mathcal{U} \psi)\}$$

In other words, in a minimal model $\mathcal{M}$ such that $\langle \mathcal{M}, s_0 \rangle \not\models \psi$, the so-called *context* $\Delta$ cannot be satisfied from the state $s_1$ until the state where $\psi$ is true. Regarding tableaux, the rule $(\mathcal{U})_2$ –which is crucial in our approach for getting a one-pass method– allows to split a branch containing a node labelled by $\Delta \cup \{\varphi \mathcal{U} \psi\}$ into two branches respectively labelled by $\Delta \cup \{\psi\}$ and $\Delta \cup \{\varphi, \circ((\varphi \wedge \neg\Delta) \mathcal{U} \psi)\}$. Hence, the negation of the successive contexts $\Delta$ will be required by the postponed eventuality. Provided that the number of possible contexts $\Delta$ is finite, the fulfillment of $\psi$ cannot be indefinitely postponed, without getting a contradiction. Of course, the procedure must fairly select an eventuality to ensure termination. Tableau rules handle general formulas, whereas resolution needs a preliminary transformation to the clausal language before the rules can be applied. The rule $(\mathcal{U}\, Set)$ introduced in this paper is an adaptation –to the clausal language setting– of the tableau rule $(\mathcal{U})_2$. That is, $(\mathcal{U}\, Set)$ is applied to a set of clauses and the eventuality is inside a clause whereas in $(\mathcal{U})_2$ the eventuality is itself a formula.
Regarding worst-case complexity, the upper bound given in [19] coincides with the one for TRS-resolution (see Proposition 45). The computational cost of introducing the negation of the context in postponed eventualities not only depends on the size of the context but also on its form. There are syntactically detectable classes of formulas that can be disregarded when negating the context. In particular the most remarkable class is formed by formulas of the form $\Box \varphi$. Since often most of the clauses are formulas of the form $\Box \varphi$ where $\varphi$ is in some normal form, the rule $(\mathcal{U}\, Set)$ is specifically well suited for clausal resolution.

### 9.2 The Resolution Method of Cavali & Fariñas del Cerro [8]

The complete resolution method presented in [8] deals with a language that is strictly less expressive than full PLTL since only the temporal connectives $\circ$, $\Box$ and $\Diamond$ are allowed. The normal form is based only on distribution laws, and renaming is not used to remove any nesting of operators. Consequently, their translation into the normal form does not introduce new variables, at the price of achieving little reduction of nesting of classical and temporal connectives. A formula in Conjunctive Normal Form is a conjunction of clauses $C_1 \wedge \ldots \wedge C_r$ where every clause $C_j$ has the following recursive structure

$$L_1 \vee \ldots \vee L_n \vee \Box\,\delta_1 \vee \ldots \vee \Box\,\delta_m \vee \Diamond\,\kappa_1 \vee \ldots \vee \Diamond\,\kappa_h$$

Here each $L_j$ is of the form $\circ^i p$ or $\circ^i \neg p$ with $p$ being a propositional atom, each $\delta_j$ is a clause and each $\kappa_j$ is a conjunction where every conjunct is a clause. The resolution method is based on considering different cases in order to check whether formulas that must be satisfied at the same state are contradictory or not. For instance, for deciding whether $\Sigma = \{\Box\,\varphi, \Diamond\,\psi\}$ is unsatisfiable, the unsatisfiability of $\Sigma' = \{\Box\,\varphi, \psi\}$ is analyzed. This case actually represents a jump to an indeterminate state, i.e. the number of states between the state $s$ where $\Sigma$ is satisfied and the state $s'$ where $\Sigma'$ is satisfied is unknown. Similarly, in order to decide whether $\{\Diamond\,\varphi, \Diamond\,\psi\}$ is unsatisfiable, the unsatisfiability of $\{\Diamond\,\varphi, \psi\}$ and $\{\varphi, \Diamond\,\psi\}$ is analyzed. Also formulas of the form $\varphi \vee \circ\varphi \vee \ldots \vee \circ^i\varphi$ and of the form $\neg\varphi \wedge \circ\neg\varphi \wedge \ldots \wedge \circ^{i-1}\neg\varphi \wedge \circ^i\varphi$ are considered for dealing with $\Diamond\,\varphi$ and formulas of the form $\varphi \wedge \circ\varphi \wedge \ldots \wedge \circ^i\varphi$ for dealing with $\Box\,\varphi$. However, there is not a clear algorithm to construct derivations and, therefore, complexity cannot be analyzed. In our approach, the nesting of connectives in the normal form is much more restricted. Our resolution method is based on reasoning "forwards in time" state by state (without uncontrolled jumps). And, finally, our method is complete for full PLTL and we provide a terminating algorithm to construct derivations. In [7] an extension of the resolution method presented in [8] is shown and the full expressiveness of PLTL is achieved by means of the connectives $\circ$ and $\mathcal{P}$ ("precedes") such that $\varphi\,\mathcal{P}\,\psi$ is equivalent to the until-formula $(\neg\psi)\,\mathcal{U}\,(\varphi \wedge \neg\psi)$, but the completeness result for the extended method is not provided.

### 9.3 The Nonclausal Resolution Method of Abadi & Manna [1]

A nonclausal resolution method for full PLTL is presented in [1] (see also [2]). Eventualities are expressed by means of the connectives $\Diamond$ and $\mathcal{P}$ ("precedes"). Since they deal with general formulas (instead of clauses), the provided rules enable the manipulation and simplification of subformulas at any level but with some restrictions for preserving soundness. The resolution rule is of the form

$$\varphi[\chi], \psi[\chi] \longmapsto \varphi[true] \vee \psi[false]$$

where the occurrences of the subformula $\chi$ in $\varphi$ and $\psi$ that are replaced with $true$ and $false$, respectively, are all in the scope of the same number of $\circ$'s and are not in the scope of any other modal operator in either $\varphi$ or $\psi$. They also use modality rules, such as e.g. $\Box\,\varphi, \Diamond\,\psi \longmapsto \Diamond\,((\Box\,\varphi) \wedge \psi)$ and $\Diamond\,\varphi, \Diamond\,\psi \longmapsto \Diamond\,((\Diamond\,\varphi) \wedge \psi) \vee \Diamond\,(\varphi \wedge \Diamond\,\psi)$, that makes this nonclausal method very different from our proposal. However, they also introduce *induction rules* for dealing with eventualities. These induction rules are very close to our rule $(\mathcal{U}\,Set)$. Here, for simplicity and clarity, we only describe the induction rule for $\Diamond$, which in terms of the present paper says

$$\Delta, \Delta', \Diamond\,\varphi \longmapsto \Delta, \Delta', \Diamond\,(\neg\varphi \wedge \circ(\varphi \wedge \neg\Delta)) \text{ if } \vdash \neg(\Delta \wedge \varphi)$$

where $\Delta$ and $\Delta'$ are set of formulas. This rule states that if $\Delta$ and $\varphi$ cannot hold at the same time but $\varphi$ eventually holds, then there must be a sate $s_j$ where $\varphi$ does not hold and at the next state $s_{j+1}$ the formulas $\varphi$ and $\neg\Delta$ hold. Hence, the above $\Delta$ (called a *fringe* in [1]) resembles our context, but the technical handling of fringes in [1] is quite different from our treatment of contexts. The first important difference is that induction rules use an aside condition (see $\vdash \neg(\Delta \wedge \varphi)$ above) for choosing the fringe $\Delta$. In our approach, contexts are syntactically determined without any auxiliary derivation. Second, in $(\mathcal{U}\, Set)$ accumulation of the contexts is made in the non-eventuality part of the until-formula, i.e. the left-hand subformula of the until-formula. Indeed, the consequent of the TRS-rule $(\diamond\, Set)$ introduces an until-formula with the negated context in the left-hand subformula. In contrast, negated fringes are accumulated in the eventuality part. Third, the method in [1] does not impose any deterministic or systematic strategy to apply the induction rules although the completeness proof outlines a strategy based on the finiteness of the set of possible fringes. We provide, by means of the algorithm $\mathcal{SR}$, a systematic method. Additionally, in our method when a context is repeated, the derivation of a refutation is straightforward, whereas in [1] obtaining a refutation after a repetition is not so direct. The reason is that our forward reasoning approach keeps a better structure for detecting the contradiction between a context and its negation. This fact can be seen by looking at the following example $\{p, \Box(\neg p \vee \circ p), \diamond \neg p\}$. In our method a refutation is easily achieved when the context $\{p\}$ is repeated (see Example 33). However, by using the induction rule in [1] with $\Delta = \{p\}$ and $\Delta' = \{\Box(\neg p \vee \circ p)\}$, they get

$$\{p, \Box(\neg p \vee \circ p), \diamond(\neg\neg p \wedge \circ(\neg p \wedge \neg p))\}.$$

Applying some other rules, which we cannot detail here, this set is transformed into

$$\{p, \circ p, \circ\Box(\neg p \vee \circ p), \diamond(p \wedge \circ\neg p)\}.$$

The resolution rule is not enough for achieving a contradiction from the latter set. Fourth, [1] does not address the problem of satisfiable input sets, whereas we ensure the existence of a model for any satisfiable input through the notion of cycling derivation. Finally, complexity is not discussed in [1, 2] and is difficult to assess due to the lack of a clear strategy for applying the rules.

## 9.4 Venkatesh's Temporal Resolution [36]

The resolution method presented in [36] is very similar to ours in everything but the way of dealing with eventualities. The normal form and even the way in which the new variables are used during the translation process are the same as ours. The resolution rule and the way of unwinding temporal literals –in the case of our rules $(\mathcal{U}\, Fix)$ and $(\mathcal{R}\, Fix)$– follow the same idea. Also the approach of reasoning forwards, i.e., jumping to the next state carrying the clauses that must be necessarily satisfied in the next state, appears in both methods. However, in sharp contrast to our TRS-resolution, the method in [36] needs invariant property generation for dealing with eventualities that can unwind indefinitely (or whose fulfillment can be delayed indefinitely). More precisely, cyclic sequences of sets of clauses that contain the so-called *persistent eventualities* –eventualities that can be unwound indefinitely and cannot be satisfied– must be detected and the persistent eventualities must be removed. Detecting those cycles can be seen as finding an invariant property $\chi$ that ensures that a given eventuality $\varphi\, \mathcal{U}\, \psi$ cannot be fulfilled because $\Box\neg\psi$ follows from $\chi$. Finding the invariant property requires an additional process whose development is not tackled in [36], therefore the complexity of the method cannot be directly assessed. Instead of invariant properties, we use the concept of context –in the applications of the rule $(\mathcal{U}\, Set)$– for preventing indefinite unwinding of eventualities.

9.5 Fisher's Temporal Resolution [12]

The resolution method presented in [12] is also for full PLTL. The structure of a formula in the Separated Normal Form (SNF) is $\Box C_1 \wedge \ldots \wedge \Box C_r$ and since it is equivalent to $\Box (C_1 \wedge \ldots \wedge\ C_r)$, the calculations are made using only the so-called PLTL-clauses $C_1, \ldots, C_r$, without $\Box$. Each $C_j$ is of one of the following three forms

$$\mathbf{start} \to \delta \qquad\qquad \kappa \to \circ\,\delta \qquad\qquad \kappa \to \Diamond\,\lambda$$

where $\to$ denotes the connective for logical implication, $\mathbf{start}$ is a nullary connective that is only true in the initial state, $\delta$ is a disjunction of propositional literals, $\kappa$ is a conjunction of propositional literals and $\lambda$ is a propositional literal. The use of $\mathbf{start}$ makes possible to differentiate the clauses that refer only to the first state and the clauses that refer to all the states. Additionally, in SNF only the temporal connectives $\circ$ and $\Diamond$ are kept, since any clause involving one of the remaining connectives ($\mathcal{U}$, $\Box$, etc.) is expressed by a set of new clauses whose only temporal connectives are $\circ$ and $\Diamond$. The three kinds of clauses are called, respectively, *initial* PLTL-clauses, *step* PLTL-clauses and *sometime* PLTL-clauses. Resolution between the former two kinds of clauses is a straightforward generalization of classical resolution but the so-called *temporal resolution rule* for sometime PLTL-clauses is more complicated:

$$\frac{\kappa_0 \to \circ\,\delta_0, \ldots, \kappa_n \to \circ\,\delta_n, \kappa_{n+1} \to \Diamond\,\lambda}{\text{SNF}(\kappa_{n+1} \to (\neg\kappa_0 \wedge \ldots \wedge \neg\kappa_n)\mathcal{W}\lambda)}$$

where the *unless* or *weak until* connective $\mathcal{W}$ is defined as $\varphi\mathcal{W}\psi \equiv (\varphi\,\mathcal{U}\,\psi) \vee \Box\varphi$. Additionally the following *loop side conditions* must be valid

$$\delta_j \to \neg\lambda \text{ and } \delta_j \to (\kappa_0 \vee \ldots \vee \kappa_n) \text{ for every } j \in \{0, \ldots, n\}$$

The idea is that if the set $\Omega = \{\kappa_0 \to \circ\,\delta_0, \ldots, \kappa_n \to \circ\,\delta_n\}$ satisfies the loop side conditions, then it follows that $(\kappa_0 \vee \ldots \vee \kappa_n) \to \circ\Box\neg\lambda$. In such a case $\Omega$ is called a loop in $\Diamond\,\lambda$ and $\kappa_0 \vee \ldots \vee \kappa_n$ is called a loop formula (also called invariant) in $\neg\lambda$. So the method is based on searching for the existence of these invariant properties. This task requires specialized graph search algorithms (see [14, 10]) and is the most intricate part of this approach. The worst-case complexity is discussed in [14], where the translation to SNF is proved to be linear in the length of the input, whereas resolution is doubly exponential in the number of proposition symbols. An improved and simplified version of the resolution method in [12] can be found in [9]. The main differences with respect to TRS-resolution method are three. First, although the technique of renaming complex subformulas by a new proposition symbol is used in both approaches, in our normal form the temporal connectives $\mathcal{U}$ and $\mathcal{R}$ are kept. Second, we follow the approach of reasoning forwards and jumping to the next state when necessary, whereas the method presented in [12] involves reasoning backwards. Actually, contradictions are achieved at the initial state. Third, the most remarkable difference is the way of dealing with eventualities, since we dispense with invariant generation by means of the rule $(\mathcal{U}\,Set)$.

## 10 Conclusion

We have presented a new method for temporal resolution that is sound and complete for PLTL and does not require invariant generation. We have provided the conversion of any formula to clausal form, a resolution system called TRS that extends classical resolution, and an easily implementable algorithm that decides the satisfiability of any set of clauses. Moreover, together with its yes/no answer, the algorithm provides an (un/)satisfiability proof. That is, either a systematic refutation or a canonical model of the set of clauses that has been given as input.

We believe that the presented work opens many interesting topics for future research. The extension of our resolution method to more expressive logics is a wide area of work. In particular, we hope that the presented method gives an opportunity to develop the first resolution method for Full Computation Tree Logic $\mathsf{CTL}^\star$. Although the first complete tableau system for $\mathsf{CTL}^\star$ has been recently published in [30], a resolution procedure for $\mathsf{CTL}^\star$ is not known yet. Additionally, the extension of TRS-resolution to first-order linear temporal logic (shortly, FLTL), besides its own relevance, could produce a new class of decidable fragments of FLTL along with their associated decision procedures based on TRS-resolution. For instance, one may consider the clausal FLTL-language that is obtained from our clausal language by allowing, as atoms, predicate symbols applied to first-order terms, instead of propositional variables. A syntactical restriction of this clausal FLTL-language would be decidable provided that the set of all possible different contexts –in any application of the rule $(\mathcal{U}\, Set)$– were ensured to be finite in the restricted language. Moreover, particular syntactical restrictions could allow to specialize the general TRS-procedure in order to gain efficiency. The TRS-resolution method could also be applied to other extensions of PLTL like spatial, dynamic, etc. Regarding the opposite case of restricting the language (instead of extending it), we would like to remark that temporal logic programming languages could be obtained as concrete subsets of our clausal language and their operational semantics could be defined in terms of TRS-resolution. Indeed, we already have some results in this direction. The development of practical automated reasoning tools based on TRS-resolution constitutes a broad area of present and future work. At the moment, a preliminary prototype is available online in `http://www.sc.ehu.es/jiwlucap/TRS.html`. This prototype is a direct implementation of the transformation to CNF and the algorithm $\mathcal{SR}$. There is only a small amount of nondeterminism in $\mathcal{SR}$. Moreover, the form of nondeterminism in $\mathcal{SR}$ is sometimes called *angelic* nondeterminism, in the sense that backtracking is not required to ensure termination. The crucial actions upon which the implementation of $\mathcal{SR}$ depends are the fair selection of eventualities, the application of each rule, and the test for termination. We plan to gradually improve this prototype and to compare it with other available automated reasoners for PLTL. In particular with the temporal resolution prover TRP++ [25] that implements the method introduced in [12], which is very close to TRS-resolution. We are also interested in comparison with the implementations of the tableau-based methods presented in [27, 34] that are available in the Logics Workbench Version 1.1 (`http://www.lwb.unibe.ch`) and with our own *TTM Theorem Prover* (`http://www.sc.ehu.es/jiwlucap/TTM.html`), which implements the method introduced in [17,19]. We are also considering the possibility of combining TRS-resolution with the one-pass tableau method (inside our TTM Theorem Prover) to produce a kind of *hyper tableaux* that would be also interesting for practical implementation purposes.
Finally, the accurate study of complexity of TRS-resolution seems to be also interesting.

**Acknowledgements** We would like to thank the referees for their careful reading and many helpful comments leading to significant improvements of the paper.

## References


1. M. Abadi and Z. Manna. Nonclausal temporal deduction. In R. Parikh, editor, *Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1985.
2. M. Abadi and Z. Manna. Nonclausal deduction in first-order temporal logic. *J. ACM*, 37(2):279–317, 1990.
3. P. Abate, R. Goré, and F. Widmann. One-pass tableaux for computation tree logic. In N. Dershowitz and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 2007.
4. B. Banieqbal and H. Barringer. Temporal logic with fixed points. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification, Altrincham, UK, April 8-10, 1987, Proceedings*, volume 398 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1987.
5. M. Baudinet. Temporal logic programming is complete and expressive. In *Proceedings, Sixteenth Annual ACM Symposium on Principles of Programming Languages (POPL), Austin, Texas*, pages 267–280, 1989.
6. A. Bolotov and M. Fisher. A clausal resolution method for CTL branching-time temporal logic. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):77–93, 1999.
7. A. R. Cavalli. A method of automatic proof for the specification and verification of protocols. *Computer Communication Review*, 14(2):100–106, 1984.
8. A. R. Cavalli and L. F. del Cerro. A decision method for linear temporal logic. In R. E. Shostak, editor, *7th International Conference on Automated Deduction, Napa, California, USA, May 14-16, 1984, Proceedings*, volume 170 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 1984.
9. A. Degtyarev, M. Fisher, and B. Konev. A simplified clausal resolution procedure for propositional linear-time temporal logic. In *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2002, Copenhagen, Denmark, July 30 - August 1, 2002, Proceedings*, volume 2381 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2002.
10. C. Dixon. Search strategies for resolution in temporal logics. In M. A. McRobbie and J. K. Slaney, editors, *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, New Brunswick, NJ, USA, July 30 - August 3, 1996, Proceedings*, volume 1104 of *Lecture Notes in Computer Science*, pages 673–687. Springer, 1996.
11. E. Eder. *Relative complexities of first order calculi*. Artificial intelligence. Vieweg, 1992.
12. M. Fisher. A resolution method for temporal logic. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI), Sydney, Australia.*, pages 99–104, 1991.
13. M. Fisher. *An Introduction to Practical Formal Methods Using Temporal Logic*. John Wiley & Sons, Ltd, June, 2011.
14. M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, 2001.
15. D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal logic (vol. 1): mathematical foundations and computational aspects*. Oxford University Press, Inc., New York, USA, 1994.
16. D. M. Gabbay, M. A. Reynolds, and M. Finger. *Temporal logic (vol. 2): mathematical foundations and computational aspects*. Oxford University Press, Inc., New York, USA, 2000.
17. J. Gaintzarain, M. Hermo, P. Lucio, and M. Navarro. Systematic semantic tableaux for PLTL. In E. Pimentel, editor, *Proceedings of the Seventh Spanish Conference on Programming and Computer Languages (PROLE 2007), Selected Papers*, volume 206 of *Electronic Notes in Theoretical Computer Science*, pages 59–73, 2008.
18. J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. A cut-free and invariant-free sequent calculus for PLTL. In J. Duparc and T. A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2007.
19. J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. Dual systems of tableaux and sequents for PLTL. *Journal of Logic and Algebraic Programming*, 78(8):701–722, 2009.
20. V. Goranko, A. Kyrilov, and D. Shkatov. Tableau tool for testing satisfiability in LTL: Implementation and experimental analysis. *Electronic Notes in Theoretical Computer Science*, 262:113–125, 2010.
21. V. Goranko and D. Shkatov. Tableau-based decision procedure for full coalitional multiagent temporal-epistemic logic of linear time. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 969–976, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.
22. R. Goré. *Tableau methods for modal and temporal logics*, pages 297–396. Kluwer Academic Publishers, 1999.

23. R. Goré and F. Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *Proceedings of the 22nd International Conference on Automated Deduction*, CADE-22, pages 437–452, Berlin, Heidelberg, 2009. Springer-Verlag.
24. G. D. Gough. *Decision Procedures for Temporal Logic*. Master's thesis. Department of Computer Science, University of Manchester, England, 1984.
25. U. Hustadt and B. Konev. Trp++2.0: A temporal resolution prover. In F. Baader, editor, *Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August 2, 2003, Proceedings*, volume 2741 of *Lecture Notes in Computer Science*, pages 274–278. Springer, 2003.
26. U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4):479–522, 1999.
27. G. Janssen. Logics for digital circuit verification - theory, algorithms, and applications, phd thesis, eindhoven university of technology, the netherland, 1999.
28. R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. Temporalising tableaux. *Studia Logica*, 76(1):91–134, 2004.
29. O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.
30. M. Reynolds. A tableau for $\mathsf{CTL}^\star$. In *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands, November 2-6, 2009. Proceedings*, volume 5850 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2009.
31. M. Reynolds and C. Dixon. Theorem-proving for discrete temporal logic. In *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 279–314. Elsevier, 2005.
32. J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, January 1965.
33. U. Schöning. *Logic for Computer Scientists*. Birkhäuser Boston-Basel-Berlin, 1989.
34. S. Schwendimann. A new one-pass tableau calculus for $\mathsf{PLTL}$. In *Analytic Tableaux and Related Methods*, volume 1397 of *Lecture Notes in Computer Science*, pages 277–292, 1998.
35. A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.
36. G. Venkatesh. A decision method for temporal logic based on resolution. In S. N. Maheshwari, editor, *Foundations of Software Technology and Theoretical Computer Science, Fifth Conference, New Delhi, India, December 16-18, 1985, Proceedings*, volume 206 of *Lecture Notes in Computer Science*, pages 272–289. Springer, 1985.
37. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.