Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Standpoint Linear Temporal Logic

Nicola Gigante[1] , **Lucía Gómez Álvarez**[2] , **Tim S. Lyon**[2]
[1]Free University of Bozen-Bolzano, Italy
[2]TU Dresden, Germany
nicola.gigante@unibz.it, {lucia.gomez_alvarez,timothy_stephen.lyon}@tu-dresden.de

Abstract

Many complex scenarios require the coordination of agents possessing unique points of view and distinct semantic commitments. In response, *standpoint logic* (SL) was introduced in the context of knowledge integration, allowing one to reason with diverse and potentially conflicting viewpoints by means of indexed modalities. Another multi-modal logic of import is *linear temporal logic* (LTL)—a formalism used to express temporal properties of systems and processes, having prominence in formal methods and fields related to artificial intelligence. In this paper, we present *standpoint linear temporal logic* (SLTL), a new logic that combines the temporal features of LTL with the multi-perspective modelling capacity of SL. We define the logic SLTL, its syntax, its semantics, establish its decidability and complexity, and provide a terminating tableau calculus to automate SLTL reasoning. Conveniently, this offers a clear path to extend existing LTL reasoners with practical reasoning support for temporal reasoning in multi-perspective settings.

1 Introduction

Reasoning about systems involving multiple agents is a core problem in artificial intelligence, with countless applications studied over the last few decades. A variety of formalisms have been introduced to model and reason about multi-agent scenarios; e.g. STIT (See To It That) logic (Belnap and Perloff 1988; Belnap, Perloff, and Xu 2001), BDI (Belief-Desire-Intention) logic (Rao and Georgeff 1998; Bratman 1999), deontic agency logic (van Berkel and Lyon 2021; Murakami 2004), and epistemic logic (Ditmarsch, Hoek, and Kooi 2007; Plaza 2007). Despite the usefulness of such logics in modelling the internal states of an agent, including the agent's attitudes, beliefs, and knowledge, such logics tend to be difficult to handle computationally (Balbiani, Herzig, and Troquard 2008; Lutz 2006; Rao and Georgeff 1998).

In contrast to these approaches, *standpoint logic* (SL) has been recently introduced (Gómez Álvarez and Rudolph 2021) as a relatively low-cost multi-agent logic with applications in the context of knowledge integration. Within the framework of standpoint logic, propositions may be 'wrapped' within modalities of the form $\Box_{\mathsf{s}}$ and $\Diamond_{\mathsf{s}}$ with s a standpoint, allowing for declarations of the form $\Box_{\mathsf{s}}\,\varphi$ ('according to s, it is *unequivocal* that φ') and $\Diamond_{\mathsf{s}}\,\varphi$ ('according to s, it is *conceivable* that φ'). Such modalities capture the semantic commitments occurring at a particular standpoint and do not require the *nesting* of semantic commitments within semantic commitments, which allows for SL to recover favourable computational properties; indeed, the satisfiability problem for SL is NP-complete (Gómez Álvarez and Rudolph 2021).

A natural application of standpoint-based frameworks arises in the context of distributed and multi-agent systems, since they support the establishment of different, possibly conflicting specifications and their coordination. And, as temporal considerations often arise when modelling the behaviour of a system, it appears worthwhile to enhance standpoint logic with temporal operators, thus allowing for mutable states-of-affairs and changing standpoints to be explicitly described. To endow standpoint logic with the capacity to express dynamic concepts, we use the preferred formalism for modelling and expressing temporal notions, i.e., *linear temporal logic* or LTL (Pnueli 1977).

LTL is a propositional modal logic interpreted over discrete, infinite sequences of states. In the nearly five decades since its inception, LTL has gained popularity as a specification language for systems, and has found many applications in AI. For instance, LTL has been applied in *automated planning*, temporally extended goals (Bacchus and Kabanza 1998), temporal planning (Fox and Long 2003; Mayer et al. 2007), timeline-based planning (Della Monica et al. 2017), planning over (partially observable) Markov decision processes (Brafman and De Giacomo 2019; Brafman, De Giacomo, and Patrizi 2018), reinforcement learning (De Giacomo et al. 2020; Hammond et al. 2021), temporal description logics (Artale et al. 2014), and temporal epistemic logics (van Benthem et al. 2009). Computationally, temporal logics are usually quite hard; e.g. LTL satisfiability is PSPACE-complete. However, many efficient techniques and tools exists to deal with LTL specifications (*e.g.*, Geatti, Gigante, and Montanari (2019); Li et al. (2014); Cavada et al. (2014)), allowing for the logic to be routinely used in practice and industry.

In this paper, we *fuse* the multi-perspective capabilities of SL with the temporal features of LTL, resulting in *standpoint linear temporal logic* (SLTL). SLTL inherits the features of both SL and LTL, letting us model both the evolution of a system as well as changing standpoints over

time. The result is a flexible formalism that maintains the favourable computational properties of its components. In particular, we provide:

1. A detailed syntax and semantics for SLTL;
2. A *tree-shaped* tableau calculus to facilitate and automate SLTL-reasoning;
3. An analysis of the computational complexity of SLTL, which is found to be PSPACE-complete (i.e., SLTL is no harder than LTL).

Our tableau calculus uses *quasi-model* based methods (Wolter and Zakharyaschev 1998). It is inspired by the nested sequent calculus for propositional standpoint logic (Lyon and Gómez Álvarez 2022) and built atop the tree-shaped tableau calculus for LTL provided by Reynolds (Reynolds 2016; Geatti et al. 2021), which has many interesting features. In particular, the tree shape of Reynolds tableau calculus allows it to be easily and efficiently traversed *symbolically*, *i.e.*, by means of Boolean formulas solved by off-the-shelf SAT solvers, as done by the BLACK satisfiability checker (Geatti, Gigante, and Montanari 2021). By maintaining the tree shape of the tableau and its overall structure, which is extended to support standpoints, we pave the way for the adoption of symbolic techniques for efficient reasoning in SLTL.

The paper is structured as follows: In Section 2, we introduce the syntax and semantics of SLTL, as well as exemplify how the logic may be applied. In Section 3, we define our tableau calculus for SLTL, and describe the tableau-based algorithm that decides SLTL formulae. Subsequently, in Section 4 we prove the calculus sound and complete, and in Section 5, we prove that the satisfiability problem for SLTL is PSPACE-complete. Section 6 concludes and discusses future work.

2 Standpoint Linear Temporal Logic

We formally introduce *standpoint linear temporal logic* (SLTL), which fuses together propositional standpoint logic (SL) (Gómez Álvarez and Rudolph 2021) and linear temporal logic (LTL) (Pnueli 1977). We begin by explaining the various logical operators and modalities included in the language and demonstrate their applicability by means of an example. Subsequently, we provide a semantics for SLTL, defining the models used (*temporal standpoint structures*) and clarifying how formulae are interpreted.

2.1 Language

The logic SLTL is built atop classical propositional logic, and therefore, employs propositional variables along with the connectives for negation $\neg$, disjunction $\vee$, and conjunction $\wedge$. In addition, our logic incorporates the temporal modalities from LTL; in particular, (1) the unary modalities X, F, and G, and (2) the binary modalities $\mathcal{U}$ and $\mathcal{R}$. These modalities are read as follows: $\mathsf{X}\varphi$ states 'at the next moment φ holds', $\mathsf{F}\varphi$ states 'eventually φ holds', $\mathsf{G}\varphi$ states 'always φ holds', $\varphi\mathcal{U}\psi$ states 'φ holds until ψ holds', and $\varphi\mathcal{R}\psi$ is interpreted as the dual of $\varphi\mathcal{U}\psi$. The formal semantics of these formulae can be found in Definition 4 below.

We also employ the standpoint modalities $\Box_{\mathsf{s}}$ and $\Diamond_{\mathsf{s}}$, where s is taken from a finite set $\mathcal{S}$ of standpoints, $\Box_{\mathsf{s}}\varphi$ is read as 'according to s, it is unequivocal that φ' and $\Diamond_{\mathsf{s}}\varphi$ is read as 'according to s, it is conceivable that φ'. Furthermore, we include formulae of the form $\mathsf{s} \preceq \mathsf{s}'$ indicating that the standpoint s is *sharper* than s', i.e. s complies with s'.

Definition 1 (Formulae)**.** Let $\mathcal{V} = \langle \mathcal{P}, \mathcal{S} \rangle$ be a *vocabulary*, where $\mathcal{P}$ is a non-empty set of propositional variables and $\mathcal{S}$ is a set of standpoint symbols containing the distinguished symbol $*$, called the *universal standpoint*. We define the language $\mathcal{L}_{\mathcal{V}}$ to be the collection of all standpoint expressions of the form $\mathsf{s} \preceq \mathsf{s}'$ where $\mathsf{s}, \mathsf{s}' \in \mathcal{S}$, and of all formulae φ generated via the following grammar in BNF:

$$\varphi ::= p \mid \neg p \mid (\varphi \circ \varphi) \mid \nabla\varphi$$

where $\circ \in \{\vee, \wedge, \mathcal{U}, \mathcal{R}\}$, $\nabla \in \{\mathsf{X}, \mathsf{F}, \mathsf{G}\} \cup \{\Diamond_{\mathsf{s}}, \Box_{\mathsf{s}} \mid \mathsf{s} \in \mathcal{S}\}$ and $p \in \mathcal{P}$. We use p, q, r ... (potentially annotated) to denote propositional variables and φ, ψ, χ, ... (potentially annotated) to denote formulae from $\mathcal{L}_{\mathcal{V}}$.[1] We define the formulae $\neg\varphi$ with φ a complex formula and $\varphi \rightarrow \psi$ as usual.

In order to calculate complexity bounds for our tableaux, it will be helpful to define the size of formulae.

Definition 2 (Subformula, Size)**.** We define the set of *subformulae* of φ, denoted $\mathrm{sufo}(\varphi)$, recursively as follows:

- $\mathrm{sufo}(p) := \{p\}$ and $\mathrm{sufo}(\neg p) := \{\neg p\}$;
- $\mathrm{sufo}(\nabla\psi) := \{\nabla\psi\} \cup \mathrm{sufo}(\psi)$;
- $\mathrm{sufo}(\psi \circ \chi) := \{\psi \circ \chi\} \cup \mathrm{sufo}(\psi) \cup \mathrm{sufo}(\chi)$.

with $\circ \in \{\vee, \wedge, \mathcal{U}, \mathcal{R}\}$, $\nabla \in \{\mathsf{X}, \mathsf{F}, \mathsf{G}\} \cup \{\Diamond_{\mathsf{s}}, \Box_{\mathsf{s}} \mid \mathsf{e} \in \mathcal{E}_{\mathcal{S}}\}$, and $p \in \mathcal{P}$. We say that ψ is a *subformula* of φ *iff* $\psi \in \mathrm{sufo}(\varphi)$. We define the *size* of a formula φ in $\mathcal{L}_{\mathcal{V}}$, denoted $|\varphi|$, accordingly: $|\varphi| := |\mathrm{sufo}(\varphi)|$. Last, we define the *size* of a set of formulae Φ, denoted $|\Phi|$, as: $|\Phi| := \Sigma_{\varphi \in \Phi} |\mathrm{sufo}(\varphi)|$.

Example. Medical devices require testing and certification prior to marketing and use by medical professionals. Albeit, regulations differ from country to country, giving rise to potentially conflicting standards and safety qualifications. For instance, Germany (DE) and Italy (IT) may agree that a medical device X has been deemed *safe according to testing* (X_TestSafe), so long as it has been found that it never *malfunctions* (Malf). This judgement can be expressed by the SLTL formula shown below, where $*$ is the universal standpoint which encodes that the formula is unequivocal from all perspectives (i.e. from the perspective of both DE and IT).

$$\Box_{*}(\texttt{X\_TestSafe} \rightarrow \mathsf{G}\neg\texttt{Malf}) \tag{1}$$

Yet, despite the agreement on what makes X safe according to testing, each country may differ in how it considers X to be *safe overall* (X_Safe). It could be that Italy deems a device safe so long as it has been deemed safe according to testing or has been found *safe by comparison* (X_SafeComp). We could formalise this perspective in our language as:

$$\Box_{\mathsf{IT}}(\texttt{X\_Safe} \rightarrow \texttt{X\_SafeComp} \vee \texttt{X\_TestSafe}) \tag{2}$$

[1] We have opted to employ formulae for standpoint LTL in *negation normal form* as it will simplify the presentation of our tableaux later on.

The notion of 'safe by comparison' relies on X's relation to other devices within its domain of application. If a *comparable* device Y exists in terms of architecture, materials used, applicability, etc. (X_Comp_Y), and this device has been deemed safe by testing (Y_TestSafe), then it can be argued that X is safe by comparison. We can express this as the formula shown below, where Devices consists of the set of devices within X's domain of application (of which there are only finitely many).

$$\Box_{\mathsf{IT}}(\texttt{X\_SafeComp} \rightarrow \bigvee_{\texttt{Y} \in \text{Devices}} \texttt{X\_Comp\_Y} \wedge \texttt{Y\_TestSafe}) \quad (3)$$

In contrast to Italy's perspective, Germany may qualify X as safe, so long as it has been tested safe, that is:

$$\Box_{\mathsf{DE}}(\texttt{X\_Safe} \rightarrow \texttt{X\_TestSafe}) \quad (4)$$

We can see that Germany's standpoint on what counts as safe is more stringent than, or subsumed by, Italy's standpoint, a fact which may be encoded as $\mathsf{DE} \preceq \mathsf{IT}$. As each nation takes a different stance on what it deems safe, it is clear that propositions may be inconsistent with one perspective as opposed to the other. For example, the proposition 'it is conceivable that X is safe overall though not safe according to testing', i.e. the formula $\Diamond_*(\texttt{X\_Safe} \wedge \neg\texttt{X\_TestSafe})$ is consistent with formulae (1)-(4) above. This is due to Italy's standpoint, which allows for X to be safe by comparison. Still, the formula $\Box_*(\texttt{X\_Safe} \wedge \neg\texttt{X\_TestSafe})$ is inconsistent with formulae (1)-(4) since Germany regards X as not safe overall if it has not been determined safe by testing.

As demonstrated above, the use of standpoint modalities permits the modelling of distinct, conflicting perspectives. Without the use of modalities to 'wrap' reasoning, we would achieve undesirable inconsistencies that are unrepresentative of the actual (consistent) scenario we aim to model. Thus, the use of standpoint modalities with LTL improves our capacity to represent knowledge more faithfully, and as shown in Sect. 5, this does not come at a cost in terms of complexity. Indeed, an attractive feature of standpoint logic is its low complexity in relation to other knowledge integration approaches, being NP-complete (Gómez Álvarez and Rudolph 2021; Lyon and Gómez Álvarez 2022).

2.2 Semantics

The models used for SLTL are variants of relational models, employing *state sequences* (as in LTL) as opposed to worlds and *standpoint interpretations* rather than accessibility relations; cf. (Gómez Álvarez and Rudolph 2021).

Definition 3 (Temporal Standpoint Structure)**.** Let $\mathcal{V} = \langle \mathcal{P}, \mathcal{S} \rangle$ be a vocabulary. We define a *state sequence* (relative to $\mathcal{V}$) to be an infinite sequence of states $\overline{\sigma} = \langle \sigma_0, \sigma_1, \ldots \rangle$, where each state $\sigma_i \subseteq \mathcal{P}$. A *temporal standpoint structure* is an ordered-pair $\mathfrak{M} = \langle \Pi, \lambda \rangle$ such that Π is a non-empty set of state sequences $\langle \sigma_0, \sigma_1, \ldots \rangle$ and $\lambda : \mathcal{S} \rightarrow 2^{\Pi}$ is a *standpoint interpretation* mapping each standpoint symbol to a non-empty subset of Π with $\lambda(*) = \Pi$.

Definition 4 (Satisfaction)**.** Let $\varphi \in \mathcal{L}_{\mathcal{V}}$ and $\mathfrak{M} = \langle \Pi, \lambda \rangle$ be a temporal standpoint structure such that $\overline{\sigma} \in \Pi$. We recursively define the *satisfaction* of φ on $\overline{\sigma}$ at the time point $n \geq 0$, written $\overline{\sigma}, n \models \varphi$, as follows:

Rule	$\varphi \subseteq \Delta$ for $(\epsilon)\Delta \in \Gamma$	$\Gamma_1(\varphi)$	$\Gamma_2(\varphi)$
DIS	$\alpha \vee \beta$	$\{\alpha\}$	$\{\beta\}$
UNT	$\alpha \mathcal{U} \beta$	$\{\beta\}$	$\{\alpha, \mathsf{X}(\alpha \mathcal{U} \beta)\}$
REL	$\alpha \mathcal{R} \beta$	$\{\alpha, \beta\}$	$\{\beta, \mathsf{X}(\alpha \mathcal{R} \beta)\}$
EVE	$\mathsf{F}\alpha$	$\{\alpha\}$	$\{\mathsf{XF}\alpha\}$
CON	$\alpha \wedge \beta$	$\{\alpha, \beta\}$	
ALW	$\mathsf{G}\alpha$	$\{\alpha, \mathsf{XG}\alpha\}$	

Table 1: Expansion rules 1: When a singleton φ of the types shown in the table is found in one indexed set $(\epsilon)\Delta$ of the set $\Gamma(u)$ of a node u, one or two children nodes u' and u'' are created, each with a copy $\Gamma(u')$ and $\Gamma(u'')$ of $\Gamma(u)$ in which φ has been replaced by $\Gamma_1(\varphi)$ and $\Gamma_2(\varphi)$ (respectively) in the set indexed $(\epsilon)\Delta$.

- $\mathfrak{M}, \overline{\sigma}, n \models \mathsf{s}' \preceq \mathsf{s}$ *iff* $\lambda(\mathsf{s}') \subseteq \lambda(\mathsf{s})$;
- $\mathfrak{M}, \overline{\sigma}, n \models p$ *iff* $p \in \sigma_n$;
- $\mathfrak{M}, \overline{\sigma}, n \models \neg p$ *iff* $p \notin \sigma_n$;
- $\mathfrak{M}, \overline{\sigma}, n \models \varphi \vee \psi$ *iff* $\mathfrak{M}, \overline{\sigma}, n \models \varphi$ or $\mathfrak{M}, \overline{\sigma}, n \models \psi$;
- $\mathfrak{M}, \overline{\sigma}, n \models \varphi \wedge \psi$ *iff* $\mathfrak{M}, \overline{\sigma}, n \models \varphi$ and $\mathfrak{M}, \overline{\sigma}, n \models \psi$;
- $\mathfrak{M}, \overline{\sigma}, n \models \mathsf{X}\varphi$ *iff* $\mathfrak{M}, \overline{\sigma}, n+1 \models \varphi$;
- $\mathfrak{M}, \overline{\sigma}, n \models \mathsf{F}\varphi$ *iff* there is a $j > n$ such that $\mathfrak{M}, \overline{\sigma}, j \models \psi$;
- $\mathfrak{M}, \overline{\sigma}, n \models \varphi \mathcal{R} \psi$ *iff* either $\mathfrak{M}, \overline{\sigma}, i \models \psi$ for all $i \geq n$, or there exists a $k \geq n$ such that $\mathfrak{M}, \overline{\sigma}, k \models \varphi$ and $\mathfrak{M}, \overline{\sigma}, j \models \psi$ for all $n \leq j \leq k$;
- $\mathfrak{M}, \overline{\sigma}, n \models \varphi \mathcal{U} \psi$ *iff* there exists a $j \geq n$ such that $\mathfrak{M}, \overline{\sigma}, j \models \psi$ and for every $n \leq i < j$, $\mathfrak{M}, \overline{\sigma}, i \models \varphi$;
- $\mathfrak{M}, \overline{\sigma}, n \models \Diamond_{\mathsf{s}} \psi$ *iff* for some $\overline{\sigma}' \in \lambda(\mathsf{s})$, $\mathfrak{M}, \overline{\sigma}', n \models \psi$;
- $\mathfrak{M}, \overline{\sigma}, n \models \Box_{\mathsf{s}} \psi$ *iff* for each $\overline{\sigma}' \in \lambda(\mathsf{s})$, $\mathfrak{M}, \overline{\sigma}', n \models \psi$;
- $\mathfrak{M}, \overline{\sigma} \models \varphi$ *iff* $\mathfrak{M}, \overline{\sigma}, 0 \models \varphi$;
- $\mathfrak{M} \models \varphi$ *iff* for all $\overline{\sigma} \in \Pi$ then $\mathfrak{M}, \overline{\sigma} \models \varphi$.

We say that that a set of formulas Φ is *valid*, written $\models \Phi$, *iff* $\mathfrak{M} \models \varphi$ for each $\varphi \in \Phi$ and each temporal standpoint structure $\mathfrak{M}$. For a vocabulary $\mathcal{V}$, we define the *standpoint logic* $\text{SLTL}^{\mathcal{V}} := \{\varphi \in \mathcal{L}_{\mathcal{V}} \mid \models \varphi\}$.

3 Automating Reasoning via Tableaux

This section presents the tableau calculus for SLTL, which we will later show to be terminating, sound, and complete. The tableau introduces support for standpoint reasoning into the structure of a tableau calculus for LTL. This is done by encapsulating a full set of standpoint interpretations (at a given time) within each node and exploiting *quasi-model* based techniques (Wolter and Zakharyaschev 1998). In particular, we extend the tree-shaped tableau by Reynolds (Reynolds 2016), which proved to be quite amenable for extensions of LTL to different logics (Geatti et al. 2021) and for efficient implementations (Geatti, Gigante, and Montanari 2019). Classic tableaux for LTL (Lichtenstein and Pnueli 2000) usually build a graph structure and then, in a second pass, look for models inside it. In contrast,

Reynolds' tableau builds a tree structure where each branch can be explored independently from the others in a single pass (which also aids parallelization (McCabe-Dansted and Reynolds 2017)). Furthermore, its modular rule-based structure and it's model-theoretic completeness proofs (Geatti et al. 2021) are easy to extend to different logics.

Here, we exploit these features to obtain a tree-shaped tableau calculus for SLTL. We remark that the overall structure of Reynolds' tableau remains unchanged, but the nodes' labels are enriched, and additional rules are devised, to deal with standpoint modalities. Let us begin by presenting a sequence of useful definitions.

Definition 5 (Closure)**.** Let $\Phi \subseteq \mathcal{L}_\mathcal{V}$. The closure $\mathcal{C}(\Phi)$ is the set defined as follows

1. For every $\varphi \in \Phi$ and $\psi \in \mathrm{sufo}(\varphi)$, $\psi \in \mathcal{C}(\Phi)$;
2. For every $\neg p \in \mathcal{P}$, $p \in \mathcal{C}(\Phi)$ if and only if $\neg p \in \mathcal{C}(\Phi)$;
3. If $\varphi_1 \circ \varphi_2 \in \mathcal{C}(\Phi)$, for $\circ \in \{\mathcal{U}, \mathcal{R}\}$, then $\mathsf{X}(\varphi_1 \circ \varphi_2) \in \mathcal{C}(\Phi)$;
4. If $\triangledown \varphi \in \mathcal{C}(\Phi)$, for $\triangledown \in \{\mathsf{G}, \mathsf{F}\}$, then $\mathsf{X}\varphi \in \mathcal{C}(\Phi)$.

Intuitiely, the closure of a formula is the set of all the formulas that is sufficient to take care of when reasoning about the formula. A tableau for Φ is a tree made of sets of constraint sets, defined below. Each constraint set is associated with the set of standpoints to which it belongs and with the set of formulas (from the closure) that hold for that point.

Definition 6 (Standpoint Encoding)**.** The encoding of a standpoint s in Φ is defined as $\mathrm{enc}(\mathsf{s}) = \{\mathsf{s}\} \cup \{\mathsf{s}' \mid \mathsf{s} \preceq \mathsf{s}' \in \Phi\}$. We use ϵ, ϵ', ... to refer to standpoint encodings. We let $E = \{\mathrm{enc}(\mathsf{s}) \mid \mathsf{s} \in \Phi\}$ and also $\epsilon \preceq \epsilon'$ iff $\epsilon' \subseteq \epsilon$.

Definition 7 (Constraint Set)**.** Let $c, c', c'', \ldots$ be *constraint sets* of the form $\langle \epsilon, \ell, \Delta \rangle$ where:

- ϵ is a standpoint encoding;
- ℓ is a label (that may be empty, in which case $\ell = \mathbf{f}$, where **f** stands for 'false'.);
- $\Delta = \{\varphi_1, \ldots, \varphi_n\}$ is a subset of $\mathcal{C}(\Phi)$.

We use the shortcut functions $\epsilon(c_i) = \epsilon_i$, $\ell(c_i) = \ell_i$ and $\Delta(c_i) = \Delta_i$, for $c_i = \langle \epsilon_i, \ell_i, \Delta_i \rangle$. Finally, we use $\langle c \rangle$ to denote to a constraint set such that $\ell(c) \neq \mathbf{f}$ (referred to as a *labelled or diamond set*) and $[c]$ to denote a constraint set such that $\ell(c) = \mathbf{f}$ (referred to as an *unlabelled or box set*).

Definition 8 (Node Constraint Set for Φ)**.** Let $u, u', u'', \ldots$ be the nodes of a tableau checking the satisfiability of a set of formulas Φ. The constraint set of a node u, denoted $\Gamma(u)$, is a set $\Gamma(u) := \{c_1, \ldots, c_n\}$ of constraint sets such that

(a) For each $\mathsf{s} \in \mathcal{S}$, there is exactly one constraint set $[c] \in \Gamma(u)$ such that $\epsilon(c) = \mathrm{enc}(\mathsf{s})$ (i.e., there is one unlabelled set per standpoint);

(b) For each $\langle c \rangle \in \Gamma(u)$, there is no $c' \in \Gamma(u)$ with $\ell(c) = \ell(c')$ (i.e., labels are unique).

We say that $\Gamma(u) = \Gamma(u')$ if for each $c \in \Gamma(u)$ there is some $c' \in \Gamma(u')$ such that $\epsilon(c) = \epsilon(c')$ and $\Delta(c) = \Delta(c')$ and *vice versa*.

Intuitively, $\Gamma(u)$ represents the structure of standpoints at a certain time point. Later, we will show how to construct a model $\mathfrak{M}$ from a tableau branch. In doing that, each labelled set c will correspond to a point in a sequence belonging to the standpoint determined by $\epsilon(c)$. Unlabelled sets encode 'standpoint types' and are necessary to ensure that full precisification sequences can be reconstructed during model construction.

Let us define some useful functions. We will use $\iota(\Gamma, \epsilon, \ell) = \Delta(c)$, if there is some $c \in \Gamma$ such that $\epsilon(c) = \epsilon$ and $\ell(c) = \ell$; otherwise, $\iota(\Gamma, \epsilon, \ell) = \emptyset$. The *union* and *set difference* operations are defined as follows (respectively) for sets of constraint sets:

- $\Gamma \cup \Gamma' = \{ \langle \epsilon, \ell, \Delta \cup \Delta' \rangle \mid \langle \epsilon, \ell, \Delta \rangle \in \Gamma, \iota(\Gamma', \epsilon, \ell) = \Delta' \}$;
- $\Gamma \setminus \Gamma' = \{ \langle \epsilon, \ell, \Delta \setminus \Delta' \rangle \mid \langle \epsilon, \ell, \Delta \rangle \in \Gamma, \iota(\Gamma', \epsilon, \ell) = \Delta' \}$.

Branches within a tableau will either become *ticked* (✓) or *crossed* (✗) throughout the processing and expansion of the tableau. Once all branches are ticked or crossed, the tableau is deemed *complete*, and a model can be extracted from each ticked branch. A counter-model can be extracted if all branches are crossed.

An X-*eventuality* is defined to be a formula of the form $\mathsf{X}(\varphi \mathcal{U} \psi)$, and if such a formula occurs in the constraint set of a node, this implies that a pending request still needs to be fulfilled at a future moment. A *poised branch* is defined to be a branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$ such that u_n contains only constraint sets with formulae of the form p and $\neg p$ with $p \in \mathcal{P}$ and $\mathsf{X}\alpha$ with $\alpha \in \mathcal{L}_\mathcal{V}$, and such that no expansion rules are applicable. Let us now describe how our tableaux are initialised and how rules are applied to them.

Initialisation: Tableaux are generated by taking r as the root node with $\Gamma(r) = \{\langle \{*\}, \ell_0, \Phi \rangle\} \cup \{\langle \mathrm{enc}(\mathsf{s}), \mathbf{f}, \emptyset \rangle \mid \mathsf{s} \in \mathcal{S}\}$ as input.

Expansion: We expand a tableau by repeatedly applying the rules in Fig. 1 as well as the rules defined below.

The *expansion rules*, shown in Fig. 1, work as follows: each rule looks for constraint set $\langle \epsilon, \ell, \Delta \rangle \in \Gamma(u)$ and a formula $\varphi \in \Delta$, where u is the current node. By applying the rules, one or two children u' and u'' are created with $\Gamma(u') = \Gamma(u) \setminus \{\langle \epsilon, \ell, \{\varphi\} \rangle\} \cup \Gamma_1(\varphi)$ and $\Gamma(u'') = \Gamma(u) \setminus \{\langle \epsilon, \ell, \{\varphi\} \rangle\} \cup \Gamma_2(\varphi)$, respectively. In addition, we have the following rules:

BOX^1 Given a node constraint set $\Gamma(u)$ and a constraint set $c \in \Gamma(u)$ with $\Box_\mathsf{s} \alpha \in \Delta(c)$ and $\bot \notin \Delta(c)$, a child node u' is created with $\Gamma(u') = \Gamma(u) \setminus \{\langle \epsilon, \ell, \{\Box_\mathsf{s} \alpha\} \rangle\} \cup \{\langle \mathrm{enc}(\mathsf{s}), \mathbf{f}, \{\alpha\} \rangle\}$, and if $\ell(c) = \mathbf{f}$, then a second child node is created with $\Gamma(u'') = \Gamma(u) \setminus \{\langle \epsilon, \ell, \{\Box_\mathsf{s} \alpha\} \rangle\} \cup \{\langle \epsilon, \mathbf{f}, \{\bot\} \rangle\}$.

BOX^2 Given a node constraint set $\Gamma(u)$ and two constraint sets $[c], c' \in \Gamma(u)$ with $\epsilon(c') \supseteq \epsilon(c)$, $\alpha \in \Delta(c)$, and $\alpha \notin \Delta(c')$, then a child node u' is created with $\Gamma(u') = \Gamma(u) \cup \{\langle \epsilon(c'), \ell(c'), \{\alpha\} \rangle\}$.

DIA^1 Given a node constraint set $\Gamma(u)$ and an indexed set $\langle \epsilon, \ell, \Delta \rangle \in \Gamma(u)$ with $\Diamond_\mathsf{s} \alpha \in \Delta$, then a child node u' is created such that $\Gamma(u') = \Gamma(u) \setminus \{\langle \epsilon, \ell, \{\Diamond_\mathsf{s} \alpha\} \rangle\} \cup \{\langle \mathrm{enc}(\mathsf{s}), \ell_i, \{\alpha\} \rangle\}$, with $\ell_i = \ell'$ if there is $\langle \mathrm{enc}(\mathsf{s}), \ell', \Delta' \rangle \in \Gamma(u)$ with $\alpha \in \Delta'$ and otherwise ℓ_i is a fresh label.

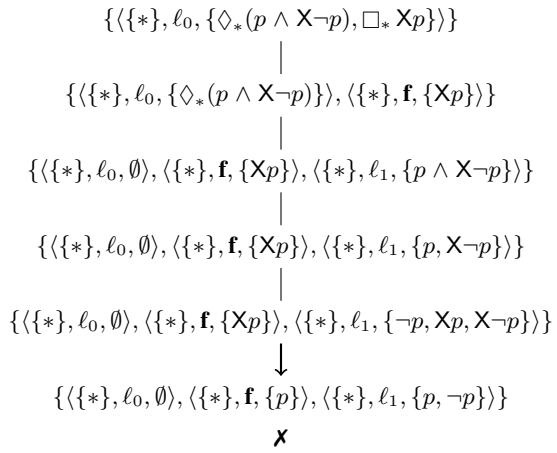

Figure 1: We provide an example demonstrating a run of our tableau algorithm with the input $\{\Diamond_*(p \wedge \mathsf{X}\neg p), \Box_* \mathsf{X}p\}$. The rules applied (in order) are BOX^1, DIA^1, CON, BOX^2, STEP, and CONTRADICTION, thus showing that the tableau creates a crossed branch, i.e. the input set is unsatisfiable.

DIA^2 Given a node constraint set $\Gamma(u)$ and two indexed sets $\langle \epsilon, \ell, \Delta\rangle, \langle \epsilon, \ell', \Delta'\rangle \in \Gamma(u)$ with $\Delta' \subseteq \Delta$ and $\ell' \neq \mathbf{f}$, then a child node u' is created with $\Gamma(u') = \Gamma(u) \setminus \{\langle \epsilon, \ell', \Delta'\rangle\}$.

STEP Given a poised branch $\overline{u} = \langle u_0, \dots, u_n\rangle$, a child u' is added to $\overline{u}$ with $\Gamma(u') = \{\langle \epsilon, \mathbf{f}, \{\alpha \mid \mathsf{X}\alpha \in \Delta\}\rangle \mid \langle \epsilon, \mathbf{f}, \Delta\rangle \in \Gamma(u)\} \cup \{\langle \epsilon, \ell, \{\alpha \mid \mathsf{X}\alpha \in \Delta\}\rangle \mid \langle \epsilon, \ell, \Delta\rangle \in \Gamma(u), \Delta \neq \emptyset\}$

CONTRADICTION If $c \in \Gamma(u_n)$ and $\bot \in \Delta(c)$, then u_n is *crossed*.[2]

EMPTY Given a branch $\overline{u} = \langle u_0, \dots, u_n\rangle$, if for all $c \in \Gamma(u_n)$ we have $\Delta(c) = \emptyset$, then u_n is *ticked*.

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

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

4 Soundness and Completeness

We now prove soundness, completeness, and termination of the tableau calculus described above. The proofs are inspired by (Geatti et al. 2021) and adapted introducing the *quasi-model* infrastructure (Wolter and Zakharyaschev 1998) when needed, seldom referring to existing proofs while being as self-contained as possible.

We start with the definition of *pre-models*, which are structures that summarize a set of models of a formula. We will see that each branch in a tableau is associated with a pre-model, and *vice versa*. We start with the notion of *atom*, which is a consistent set of formulas from the closure.

[2] $\bot$ is a shortcut for $\{p, \neg p\}$ for some $p \in \mathcal{P}$.

Definition 9 (Atom)**.** An *atom* for a set of formulas $\Phi \subseteq \mathcal{L}_\mathcal{V}$ is a set of formulas Δ such that

1. $\Delta \subseteq \mathcal{C}(\Phi)$;
2. $\bot \notin \Delta$;
3. If $\psi \in \Delta$, then either $\Gamma_1(\psi) \subseteq \Delta$, or $\Gamma_2(\psi) \neq \emptyset$ and $\Gamma_2(\psi) \subseteq \Delta$, where $\Gamma_1(\psi)$ and $\Gamma_2(\psi)$ are defined as in Fig. 1;
4. For each $\psi, \psi' \in \mathcal{C}(\Phi)$, if $\psi \in \Delta$ and $\psi \Vdash \psi'$, then $\psi' \in \Delta$, i.e., Δ is closed by logical deduction (as far as the closure is concerned);

Atoms are assigned to standpoint encodings and collected into *timestamps*.

Definition 10 (Timestamp)**.** We define a *timestamp* $\mathcal{T} = \{a_1, \dots, a_n\}$ to be a set of *indexed atoms*, that is, each a_i is of the form $\langle \epsilon_i, \ell_i, \Delta_i\rangle$, with ϵ_i a standpoint encoding, ℓ_i a label, and Δ_i an atom such that

T1 For each $a \in \mathcal{T}$, if $\Diamond_\mathsf{s} \psi \in \Delta(a)$, then there exists a $a' \in \mathcal{T}$ such that $\epsilon(a') \preceq \mathrm{enc}(\mathsf{s})$ and $\psi \in \Delta(a')$;

T2 For each $a \in \mathcal{T}$, if $\Box_\mathsf{e} \psi \in \Delta(a)$, then $\psi \in \Delta(a')$ for each $a' \in \mathcal{T}$ with $\epsilon(a') \preceq \mathrm{enc}(\mathsf{s})$;

T3 For each $\epsilon \in E$ with $E = \{\mathrm{enc}(\mathsf{s}) \mid \mathsf{s} \in \mathcal{S}\}$, there is $a_\epsilon \in \mathcal{T}$ such that $\epsilon(a_\epsilon) = \epsilon$ and $\Delta(a_\epsilon) = \bigcap\{\Delta(a') \mid a' \in \mathcal{T}, \epsilon(a) \preceq \epsilon\}$

As with the constraint sets, we use the shortcut functions $\epsilon(a_i) = \epsilon_i$, $\ell(a_i) = \ell_i$ and $\Delta(a_i) = \Delta_i$. Moreover, we say that $a_i = a_j$ iff $\epsilon(a_i) = \epsilon(a_j)$ and $\Delta(a_i) = \Delta(a_j)$.

Timestamps are collected in sequences, and *runs* map each point in time with an atom of the respective timestamp.

Definition 11 (Run)**.** Let $\overline{\mathcal{T}} = \langle \mathcal{T}_0, \mathcal{T}_1, \dots\rangle$ be an infinite sequence of timestamps. A run r on $\overline{\mathcal{T}}$ is a map associating each $i \in \mathbb{N}$ with an indexed atom $a \in \mathcal{T}_i$ in such a way that:

R1 For all $a_i = r(i)$ and $a_j = r(j)$, $\epsilon(a_i) = \epsilon(a_j)$.

R2 If $\mathsf{X}\psi \in \Delta(a_i)$ for $a_i = r(i)$, then $\psi \in \Delta(a_{i+1})$ for $a_{i+1} = r(i+1)$;

R3 If $\psi_1 \mathcal{U} \psi_2 \in r(i)$, then there is a $j \geq i$ with $\psi_2 \in \Delta(a_j)$ for $a_j = r(j)$ and $\psi_1 \in \Delta(a_k)$ for $a_k = r(k)$ for all $i \leq k < j$;

Now, a pre-model is made of a sequence of timestamps and a set of runs that encode the sequences of a usual model.

Definition 12 (Pre-Model)**.** Let $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R}\rangle$ be a tuple consisting of an infinite sequence of timestamps $\overline{\mathcal{T}}$ and a set of runs $\mathbf{R}$ such that for every timestamp $\mathcal{T}_i$ and every atom $a \in \mathcal{T}_i$ there is a run such that $r(i) = a$. $\mathcal{D}$ is a *pre-model* of $\Phi \subseteq \mathcal{L}_\mathcal{V}$ if the following constraints are satisfied:

P1 There exists an $a \in \overline{\mathcal{T}}(0)$ such that $\epsilon(a) = \mathrm{enc}(*)$ and $\Phi \subseteq \Delta(a)$;

P2 For each $i \in \mathbb{N}$ and $a \in \mathcal{T}_i$, $\Delta(a)$ is *minimal*, i.e. for any $\psi \in \Delta(a)$, the set $\Delta(a) \setminus \{\psi\}$ is (a) not an atom or (b) $\mathcal{T}$ ceases to be a timestamp or (c) there ceases to be a run.

The core feature of pre-models is their correspondence with actual models of formulae, stated in the following.

Lemma 13. *Let $\varphi \in \mathcal{L}_\mathcal{V}$, then φ has a pre-model iff φ is satisfiable.*

Proof sketch. The proof is articulated but straightforward, following the definitions of pre-models and the semantics of SLTL. See **??** for the full proof. □

With the concept of pre-models we can now prove soundness and completeness.

4.1 Soundness

Here we prove that the tableau system is sound, that is, if a complete tableau for a set of formulae Φ has a successful branch, then Φ is satisfiable. The proof shows how a pre-model for Φ can be extracted from a successful branch of a complete tableau. Intuitively, the expansion of non-poised nodes builds the set of constraint sets that will form the current timestamp, and the application of the STEP rule to a poised node marks the advancement to the next one.

Definition 14 (Step node)**.** Consider a branch $\overline{u} = \langle u_0, \ldots, u_n \rangle$ of a complete tableau T. A poised node u_i is said to be a step node if either $u_i = u_n$ or u_{i+1} is the child of u_i added by applying the STEP rule.

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

Definition 15 (Atom of a tableau node)**.** Let $\Phi \subseteq \mathcal{L}_\mathcal{V}$, u_i be a step node of a tableau for Φ and $c \in \Gamma(u_i)$. The atom extracted from c, written $\mathrm{a}(c)$, is the indexed atom c^a such that $\epsilon(c^\mathrm{a}) = \epsilon(c)$ and $\Delta(c^\mathrm{a}) = \Delta$ where Δ is the closure by logical entailment (within $\mathcal{C}(\Phi)$) of $\Delta(c)$, $\Delta = \mathrm{cl}(\Delta(c))$.

Definition 16 (Timestamp of a tableau node)**.** Let u_i be a step node of a tableau for Φ. The timestamp extracted from $\Gamma(u_i)$ is defined as $\mathrm{a}(\Gamma(u_i)) = \{\mathrm{a}(c) \mid c \in \Gamma(u_i)\}$

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

Lemma 17. *Let $\Phi \subseteq \mathcal{L}_\mathcal{V}$ and let T be a complete tableau for Φ. If T has a successful branch, then there exists a pre-model for Φ.*

Proof. Let $\overline{u} = \langle u_0, \ldots, u_n \rangle$ be a successful branch of T and let $\overline{v} = \langle v_0, \ldots, v_m \rangle$ be the subsequence of step nodes of $\overline{u}$. Intuitively, a pre-model for Φ can be obtained from $\overline{v}$ by building the atoms from the labels of the step nodes, and extending them to infinite sequences. If the branch has been accepted by the LOOP rule, we can identify a position $0 \leq k \leq m$ in $\overline{v}$ such that $\Gamma(v^k) = \Gamma(v^m)$ and all the X-eventualities requested in v^k are fulfilled in $v_{[k \ldots m]}$. If instead $\overline{v}$ has been accepted by the EMPTY rule and in particular there are no X-eventualities requested, hence setting $k = m$ we obtain the same effect.

Therefore, we can extract from v the periodic sequence of timestamps $\overline{\mathcal{T}} = \overline{\mathcal{T}}_0 \overline{\mathcal{T}}_t^\omega$, where $\overline{\mathcal{T}}_0 = \langle \mathrm{a}(\Gamma(v_0)), \ldots, \mathrm{a}(\Gamma(v_k)) \rangle$, and either $\overline{\mathcal{T}}_t^\omega = \langle \mathrm{a}(\Gamma(v_k + 1)), \ldots, \mathrm{a}(\Gamma(v_m)) \rangle$ or $\overline{\mathcal{T}}_t^\omega = \langle \mathrm{a}(\Gamma(v_m)) \rangle$ depending on which rule accepted the branch, respectively the LOOP or the EMPTY rule. In other words, we build a periodic pre-model that infinitely repeats the fulfilling loop identified by the LOOP rule, or the last empty node otherwise. Then let $K : \mathbb{N} \to \mathbb{N}$ be the map from positions in the pre-model to their original positions in the branch, which is defined as $K(i) = i$ for $0 \leq i < k$, and for $i \geq k$ is defined either as $K(i) = k + ((i - k) \bmod T)$, with $T = m - k$ (LOOP rule), or as $K(i) = k$ (EMPTY rule).

Then, we set $\mathbf{R}$ to be the set of all possible runs on $\overline{\mathcal{T}}$ and we show that $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ is a pre-model of Φ. Notice that by the initialisation of the tableau we have $\Phi \in \Delta(a_0)$ for some $a_0 \in \overline{\mathcal{T}}(0)$ and atoms are minimal by definition.

First, we show that the three conditions of timestamp in Definition 10 are satisfied for $\overline{\mathcal{T}}(i)$ with $i \geq 1$. It is easily checked that T1 is satisfied by the non-applicability of DIA^1, T2 by the non-applicability of both BOX^1 and BOX^2 and T3 by the non-applicability of DIA^2.

Let us now show that each $r \in \mathbf{R}$ fulfils the conditions in Definition 11 of a run for $i \geq 1$. For condition R1, we notice that for every $\epsilon \in E$ there is some $a \in \overline{\mathcal{T}}(i)$ by the construction of the tableau, since during initialisation one box-indexed atom is created per standpoint expression and such atoms are always propagated by the STEP rule. Hence for each $i \geq 1$ there is some $a_j \in \overline{\mathcal{T}}(j)$ such that $r(i) = a$ and $r(j) = a_j$ and $\epsilon(a) = \epsilon(a_j)$.

Assume now that $\mathsf{X}\psi \in \Delta(r(i))$. Being an elementary formula, we can observe that we must have $\mathsf{X}\psi \in \Delta(c)$ for some $c \in \Gamma(u_i)$. Two cases have to be considered. If $v_{K(i+1)} = v_{K(i)+1}$, i.e., the next atom comes from the actual successor of the current one in the tableau branch, then, by the STEP rule, there is $a_{i+1} \in \overline{\mathcal{T}}(i+1)$ such that $\epsilon(a_{i+1}) = \epsilon(a)$ and $\psi \in \Delta(a_{i+1})$. Otherwise, $\overline{\mathcal{T}}(i) = \overline{\mathcal{T}}(m) = \mathrm{a}(\Gamma(v_m))$, and v_m was ticked by the LOOP rule (since at least $\Delta(a)$ is not empty), and thus $\overline{\mathcal{T}}(i+1) = \mathrm{a}(\Gamma(v_k + 1))$ for some $k < m$ such that $\Gamma(v_k) = \Gamma(v_m)$. Hence $\mathsf{X}\psi \in \Delta(c')$ with $c' \in \Gamma(v_k)$, and by the step rule applied to v_k we obtain that $\psi \in \Delta(c'')$ with $c'' \in \Gamma(v_k + 1)$, hence there is $a''_{i+1} \in \overline{\mathcal{T}}(i+1)$ such that $\epsilon(a''_{i+1}) = \epsilon(a)$ and $\psi \in \Delta(a''_{i+1})$ as well in this case, hence condition R2 is also satisfied. Finally, the case of $\psi_1 \mathcal{U} \psi_2 \in \Delta(r(i))$ (condition R3) is then straightforward in view of the expansion rules definition. With this, we have shown that a run can be created for each $a \in \mathcal{T}_i$ and $i \geq 0$ satisfying all the conditions, and hence $\mathcal{D}$ is a pre-model. □

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

Theorem 18 (Soundness)**.** *Let $\Phi \subseteq \mathcal{L}_\mathcal{V}$ and let T be a complete tableau for Φ. If T has a successful branch, then Φ is satisfiable.*

Proof. Extract a pre-model for Φ from the successful branch of T as shown in Lemma 17, and then obtain from it an actual model for the formula as shown by Lemma 13. □

4.2 Completeness

We now prove the completeness of the tableau system, i.e., if a set of formulae Φ is satisfiable, then any complete tableau T for it has an accepting branch. The proof uses a pre-model for Φ, which we know exists if the formula is satisfiable,

as a guide to suitably descend through the tableau to look for an accepted branch. We first describe how to perform such a descent. Then, we will show how to make sure that this descent must obtain an accepted branch. The descent is performed as follows.

Lemma 19 (Extraction of the branch)**.** *Let $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ be a pre-model for a set of formulae Φ. Then, any complete tableau T for Φ has a label mapping $\mathcal{L}$ and a branch u, with a sequence of step nodes $\overline{v} = \langle v_0, \dots, v_m \rangle$, such that for $0 \leq i \leq m$ and $a \in \overline{\mathcal{T}}(i)$, we have one $c \in \Gamma(v_i)$ such that $a = \mathrm{a}(c)$.*

Proof. Let $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ be a pre-model. To find $\overline{u}$, we traverse the tree using $\overline{\mathcal{T}}$ as a guide, starting from the root u_0, building a sequence of branch prefixes $\overline{u_i} = \langle u_0, \dots, u_i \rangle$, suitably choosing $\overline{u_{i+1}}$ at each step among the children of $\overline{u_i}$. We maintain a non-decreasing function $J : \mathbb{N} \to \mathbb{N}$ that maps positions in $\overline{u_i}$ to positions in $\overline{\mathcal{T}}$ such that for each $a \in \overline{\mathcal{T}}(J(k))$, there is some $c \in \Gamma(u_k)$ such that $\epsilon(c^{\mathrm{a}}) = \epsilon(a)$ and $\Delta(c^{\mathrm{a}}) \subseteq \Delta(a)$ with $\mathrm{a}(c) = c^{\mathrm{a}}$, for each $0 \leq k \leq i$. Moreover, we maintain a labelling function $\mathcal{L}$ witnessing that relation, $\mathcal{L}(c) = a$ We start from $u_0 = \langle u_0 \rangle$ and $J(0) = 0$. Let $c_0 = \langle \{*\}, \ell_0, \Phi \rangle$ Notice that the tableau is initialised with

$$\Gamma(r) = \{c_0\} \cup \{c_\epsilon = \langle \epsilon, \mathbf{f}, \emptyset \rangle \mid \epsilon \in E\}$$

Moreover, we have $\Phi \subseteq \Delta(a_0)$ for some $a_0 \in \overline{\mathcal{T}}(0)$, and there is some $a_\epsilon \in \overline{\mathcal{T}}(0)$ for each $\epsilon \in E$, by the definition of a pre-model. Thus, we set $\mathcal{L}(c_0) = a_0$, and for each $\epsilon \in E$ we set $\mathcal{L}(c_\epsilon) = a_\epsilon$. Thus, the base case clearly holds.

Then, at each step $i > 0$, we choose u_{i+1} among the children of u_i as follows, depending on the expansion rule. We show that for each $a \in \mathcal{T}(J(i))$, there is some $c \in \Gamma(u_k)$ such that $\epsilon(c^{\mathrm{a}}) = \epsilon(a)$ and $\Delta(c^{\mathrm{a}}) \subseteq \Delta(a)$ with $\mathrm{a}(c) = c^{\mathrm{a}}$. We show the main cases and direct the reader to (Geatti et al. 2021) for the rest.

BOX^1 If u_i is not a step node and was expanded by the BOX^1 rule then it has a single child which is chosen as u_{i+1}. We leave $\mathcal{L}$ unchanged and define $J(i+1) = J(i)$, since we do not advance to the next position in the pre-model; We show that for the node u_{i+1} the condition holds. By the definition of BOX^1 we have $\Gamma(u_{i+1}) = \Gamma(u_i) \setminus \{\langle \epsilon, \ell, \{\Box_{\mathsf{e}} \psi\}\rangle\} \cup \{\langle \mathrm{enc}(\mathsf{s}), \mathbf{f}, \{\psi\}\rangle\}$. By the definition of a pre-model, if $\Box_{\mathsf{e}} \psi \in \Delta(a)$ for some $a \in \overline{\mathcal{T}}(J(i))$, then $\psi \in \Delta(a')$ for each $a' \in \overline{\mathcal{T}}(J(i))$ with $\epsilon(a') \preceq \mathrm{enc}(\mathsf{s})$, and hence the condition will still be satisfied.

BOX^2 If u_i is not a step node and was expanded by the BOX^2 rule then it has a single child which is chosen as u_{i+1}. We leave $\mathcal{L}$ unchanged and define $J(i+1) = J(i)$, since we do not advance to the next position in the pre-model; We show that for the node u_{i+1} the condition holds. By the definition of BOX^2 we have $\Gamma(u_{i+1}) = \Gamma(u_i) \cup \{\langle \epsilon(c'), \ell(c'), \{\psi\}\rangle\}$ for two constraint sets $[c], c' \in \Gamma(u_i)$ with $\epsilon(c') \preceq \epsilon(c)$, $\psi \in \Delta(c)$, and $\psi \notin \Delta(c')$. And again by the definition of a pre-model, this means that $\psi \in \Delta(a_\epsilon)$ and thus for all $a' \in \overline{\mathcal{T}}(J(i))$ with $\epsilon(a') \preceq \epsilon$ then also $\psi \in \Delta(a')$, and hence the condition will still be satisfied.

DIA^1 If u_i is not a step node and was expanded by the DIA^1 rule then it has a single child which is chosen as u_{i+1}. We define $J(i+1) = J(i)$ since we do not advance to the next position in the pre-model; We show that for the node u_{i+1} the condition holds. By the definition of DIA^1 we have $\Gamma(u') = \Gamma(u) \setminus \{\langle \epsilon, \ell, \{\Diamond_{\mathsf{s}} \psi\}\rangle\} \cup \{\langle \mathrm{enc}(\mathsf{s}), \ell_i, \{\psi\}\rangle\}$. If ℓ_i is not a fresh label, then $\mathcal{L}$ is unchanged and $\Gamma(u')$ only removes $\Diamond_{\mathsf{s}} \psi$ from $\Gamma(u)$, and hence the condition holds trivially. Assume ℓ_i is a fresh label. By construction, we have $\langle \epsilon, \ell, \Delta \rangle \in \Gamma(u)$ with $\Diamond_{\mathsf{s}} \psi \in \Delta$, and a newly introduced $c' \in \Gamma(u')$ with $c' = \langle \mathrm{enc}(\mathsf{s}), \ell_i, \{\psi\}\rangle$. By induction, there is some $a \in \overline{\sigma}(J(i))$ such that $\Diamond_{\mathsf{s}} \psi \in \Delta(a)$ and by the definition of a pre-model, there is some $a' \in \overline{\sigma}(J(i))$ such that $\epsilon(s) \preceq \mathrm{enc}(\mathsf{s})$ and $\psi \in \Delta(a')$. Thus, we update $\mathcal{L}$ such that $\mathcal{L}(c') = a'$ and with this the condition is satisfied.

DIA^2 If u_i is not a step node and was expanded by the DIA^2 rule then it has a single child which is chosen as u_{i+1}. We define $J(i+1) = J(i)$ since we do not advance to the next position in the pre-model; We show that for the node u_{i+1} the condition holds. By the definition of DIA^2 we have $\Gamma(u') = \Gamma(u) \setminus \{\langle \epsilon, \ell', \Delta' \rangle\}$ with $\Diamond_{\mathsf{s}} \psi \in \Delta$ for $\langle \epsilon, \ell, \Delta \rangle, \langle \epsilon, \ell', \Delta' \rangle \in \Gamma(u)$ with $\Delta' \subseteq \Delta$ and $\ell' \neq \mathbf{f}$. Thus, we update $\mathcal{L}$ to the shrunk set and the condition is still satisfied.

DIS, UNT, REL, CON, EVE ALW These are straightforward. Refer to Lemma 3 in (Geatti et al. 2021).

STEP If u_i is a step node but not a leaf, then it has a single child which is chosen as u_{i+1}, leaving $\mathcal{L}$ unchanged and defining $J(i+1) = J(i) + 1$ since we need to advance to the next position in the pre-model as well; We show that for the node u_{i+1} the condition holds: For each atom c^{a} extracted from the tableau node, there is some $a \in \overline{\mathcal{T}}(J(i+1))$ such that $\epsilon(c^{\mathrm{a}}) = \epsilon(a)$ and $\Delta(c^{\mathrm{a}}) \subseteq \Delta(a)$. By the definition of STEP we have $\Gamma(u') = \{\langle \epsilon, \mathbf{f}, \{\psi \mid \mathsf{X}\psi \in \Delta\}\rangle \mid \langle \epsilon, \mathbf{f}, \Delta \rangle \in \Gamma(u)\} \cup \{\langle \epsilon, \ell, \{\psi \mid \mathsf{X}\psi \in \Delta\}\rangle \mid \langle \epsilon, \ell, \Delta \rangle \in \Gamma(u), \Delta \neq \emptyset\}$. By the definition of a pre-model, if $\mathsf{X}\psi \in \Delta(a)$ for some $a \in \overline{\mathcal{T}}(J(i))$ then $\psi \in \Delta(a')$ for some $a' \in \overline{\mathcal{T}}(J(i))$. Thus the condition follows.

Now, let $\overline{u} = \langle u_0, \dots, u_n \rangle$ be the branch found as described above, and let $\overline{v} = \langle v_0, \dots, v_m \rangle$ be the sequence of its step nodes. Since the value of $J(i)$ is incremented only when an application of the STEP rule is traversed, for each $a \in \overline{\mathcal{T}}(i)$, there is some $c \in \Gamma(v_i)$ such that $\epsilon(c^{\mathrm{a}}) = \epsilon(a)$ and $\Delta(c^{\mathrm{a}}) \subseteq \Delta(a)$ with $\mathrm{a}(c) = c^{\mathrm{a}}$. Moreover, for the minimality of the atoms in the pre-model, required by Definition 12, we know that for all $a \in \mathcal{T}_i$ then any formula $\psi \in \Delta(a)$ is either $\psi \in \Phi$ and $a = a_0$ or it has been obtained by the expansion of X, $\Box_{\mathsf{s}}$ or $\Diamond_{\mathsf{s}}$, or by the closure by logical entailment. Similarly, by the construction of the tableau, any $\varphi \in c^{\mathrm{a}}$ has been obtained either by an application of a STEP fulfilling the X-eventualities or by an application of DIA^1, BOX^1 or BOX^2, fulfilling the standpoint eventualities, or by an expansion rule fulfilling the closure by logical

entailment. Finally, DIA^2 fulfils the minimality criteria by deleting subsumed diamond-atoms. Hence we can conclude that for each $a \in \overline{\mathcal{T}}(i)$, there is exactly one $c \in \Gamma(v_i)$ such that $\epsilon(c^{\mathrm{a}}) = \epsilon(a)$ and $\Delta(c^{\mathrm{a}}) \subseteq \Delta(a)$ with $\mathrm{a}(c) = c^{\mathrm{a}}$. □

The particular branch found as described above might, in general, be crossed. However, it is immediate to note that it cannot possibly have been crossed by an application of the CONTRADICTION rule, since this would imply that the pre-model itself is contradictory (which cannot happen because of Lemma 13). Hence, if a crossed leaf is found, it has been crossed by the PRUNE rule. We can, however, defin a particular class of models (and their pre-models) such that when we descend through the tableau following any model of this class, we cannot possibly find a node crossed by the PRUNE rule, neither. This class is called *greedy pre-models*.

To define the notion of greedy pre-models, consider a pre-model $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$. Let $X \subset \mathcal{C}(\varphi)$ be the set of all the X-eventualities in $\mathcal{C}(\varphi)$. For each run $r \in \mathbf{R}$ and each $i \geq 0$, the *distance vector* $d_i^r \in \mathbb{N}^X$ is a function mapping each X-eventuality ψ to the distance $d_i^r(\psi)$ of the first position where it is fulfilled, if $\psi \in r(i)$, or zero otherwise. For example, if $\psi_1 \mathcal{U} \psi_2 \in r(4)$, and $\psi_2 \in r(7)$, then $d_4^r(\psi_1 \mathcal{U} \psi_2) = 3$.

Two distance vectors are compared component-wise, *i.e.*, we define a partial order such that $v \prec v'$ iff $v(\psi) < v'(\psi)$ for all $\psi \in X$. Then, we compare two runs r and r' lexicographically, that is, $r \prec r'$ iff there is an $i \geq 0$ such that $d_j^r = d_j^{r'}$ for all $j < i$ and $d_i^r \prec d_i^{r'}$.

Intuitively, if $r \prec r'$, there is a point i where r' uselessly delays the fulfillment of an X-eventuality, while r fullfils it before, all the rest being equal before i.

Definition 20 (Greedy pre-models)**.** A run r for a formula φ is *greedy* if there is no other r' such that $r' \prec r$. A pre-model $\mathcal{D}$ for φ is greedy if all its runs are greedy.

One can show that greedy pre-models actually exist.

Lemma 21 (Limit of a sequence of runs)**.** *Let $r_1 \succ r_2 \succ \ldots$ be an infinite descending sequence of runs. Then, there exists a run r^ω such that $r^\omega \preceq r_i$ for all $i \geq 0$.*

Proof sketch. The proof is combinatorical, exploiting the properties of the lexicographic ordeding between distance vectors and the fact that the closure is finite. It is identical to the proof of Lemma 4 in (Geatti et al. 2021). □

Lemma 22 (Existence of greedy pre-models)**.** *Let $\mathcal{D}$ be a pre-model for a formula φ. Then, there is a greedy pre-model $\mathcal{D}' \preceq \mathcal{D}$.*

Proof. A direct consequence of Lemma 21. See also Lemma 5 in (Geatti et al. 2021). □

Now, we can connect greedy pre-models to the tableau, the PRUNE rule, and the descent through the tree done in Lemma 19. Consider a pre-model $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$. We define the *segment* $\mathcal{D}_{[j,k]} = \langle \overline{\mathcal{T}}_{[j,k]}, \mathbf{R}_{[j,k]} \rangle$ as the result of isolating the time points between j and k. That is, $\overline{\mathcal{T}}_{[j,k]} = \langle \overline{\mathcal{T}}(j), \ldots, \overline{\mathcal{T}}(k) \rangle$ and for every $r \in \mathbf{R}$, we have $r' \in \mathbf{R}_{[j,k]}$ where $r'(i) = r(i - j)$.

We say that a X-eventuality $\psi_1 \mathcal{U} \psi_2$ is *requested* in $\mathcal{D}_{[j,k]}$ if there is a $r \in \mathbf{R}_{[j,k]}$ such that $\psi_1 \mathcal{U} \psi_2 \in r_{[j,k]}(0)$, and that is is *fulfilled* in $\mathbf{R}_{[j,k]}$ if it is requested and $\psi_2 \in r_{[k,j]}(w)$ for some $0 \leq w < (k - j)$.

Similarly, we define $\mathcal{D}_{]j,k[} = \langle \overline{\mathcal{T}}_{]j,k[}, \mathbf{R}_{]j,k[} \rangle$ as the pre-model where the segment $\mathcal{D}_{[j,k]}$ has been cut away. That is, $\overline{\mathcal{T}}_{]j,k[} = \langle \overline{\mathcal{T}}(0), \ldots, \overline{\mathcal{T}}(j), \overline{\mathcal{T}}(k) \ldots \rangle$ and for all and only $r \in \mathbf{R}$, we have $r' \in \mathbf{R}_{[j,k]}$ where $r'(i) = r(i)$ for $0 \leq i \leq j$ and $r'(w) = r(w - (k - j))$ for $w > j$.

Definition 23 (Redundant segments)**.** Let $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ be a pre-model for φ and let $i < j < k$ be three positions such that $\overline{\mathcal{T}}(i) = \overline{\mathcal{T}}(j) = \overline{\mathcal{T}}(k)$. Then, the segment $\mathcal{D}_{[j+1 \ldots k]}$ of $\mathcal{D}$ is redundant if not all the X-eventualities requested in $\mathcal{D}_{[j+1 \ldots k]}$ are fulfilled in $\mathcal{D}_{[j+1 \ldots k]}$, and all those fulfilled in $\mathcal{D}_{[j+1 \ldots k]}$ are fulfilled in $\mathcal{D}_{[i+1 \ldots j]}$ as well.

We notice how Definition 23 is similar to the definition of the PRUNE rule, but focuses on the pre-model instead of the branch of the tableau. The core feature of redundant segments, as the name suggests, is that they can be removed from any pre-model, obtaining again a pre-model.

Lemma 24 (Removal of redundant segments: correctness)**.** *Let $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ be a pre-model for a formula φ with a redundant segment $\mathcal{D}_{[j+1 \ldots k]}$. Then, $\mathcal{D}_{]j,k+1[}$ is a pre-model.*

Proof. We start by noting that since $\overline{\mathcal{T}}(i) = \overline{\mathcal{T}}(j) = \overline{\mathcal{T}}(k)$, then $r(i) = r(j) = r(k)$ for all $r \in \mathbf{R}$. By Definition 11, for every $\mathsf{X}\psi \in \Delta(r(j))$, we have $\psi \in \Delta(r(j+1))$, hence also $\psi \in \Delta(r(k+1))$. Moreover, for any $\psi_1 \mathcal{U} \psi_2 \in r(k)$, there is a $w \leq k$ such that $\psi_2 \in \Delta(r(w))$ and $\psi_2 \in \Delta(r(\ell))$ for all $k \leq \ell < w$. Hence, note that in $\mathcal{D}_{]j,k+1[}$, the conditions of Definition 11 on $r(j)$ are satisfied. Hence $\mathcal{D}_{]j,k+1[}$ is a pre-model. □

Now, we can observe that when a redundant segment is removed from a pre-model, the runs in the resulting pre-models *decrease* in the $\prec$ ordering.

Lemma 25 (Removal of redundant segments: ordering)**.** *Let $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ be a pre-model for a formula φ with a redundant segment $\mathcal{D}_{[j+1 \ldots k]}$. Let $r \in \mathbf{R}$ and let r' be the corresponding run in $\mathcal{D}_{[j+1 \ldots k]}$. Then, $r' \prec r$.*

Proof sketch. This is a direct consequence of the definition of the $\prec$ ordering. When a redundant segment is removed, all the X-eventualities fulfilled in the removed segment do not change any distance vector, while those fulfilled later decrese their distance from the requesting points. For details, refer to Lemma 6 in (Geatti et al. 2021). □

Theorem 26 (Completeness)**.** *Let $\varphi \in \mathcal{L}_{\mathcal{V}}$ and T a complete tableau for Φ. If Φ is satisfiable, then T has a successful branch.*

Proof. Let $\mathfrak{M}$ be a model for Φ. As already noted, it is straightforward to build a pre-model for Φ from $\mathfrak{M}$. Then, given a pre-model for Φ, Lemma 21 ensures that a greedy pre-model for Φ exists. We can thus consider $\mathcal{D} = \langle \overline{\mathcal{T}}, \mathbf{R} \rangle$ to be a greedy pre-model for Φ. Now, given a complete tableau T for Φ, thanks to Lemma 19 we can obtain a branch from

T, with a sequence of step nodes $\overline{v} = \langle v_0, \ldots, v_m \rangle$ such that for each $a \in \overline{\mathcal{T}}(k)$, there is some $c \in \Gamma(v_k)$ such that $\epsilon(c^{\mathrm{a}}) = \epsilon(a)$ and $\Delta(c^{\mathrm{a}}) \subseteq \Delta(a)$ with $\mathrm{a}(c) = c^{\mathrm{a}}$ for all $0 \leq k \leq m$. As already noted, we know that if v_m is crossed, then it has to have been crossed by the PRUNE rule. If this was the case, however, it would mean there are other two-step nodes v_i and v_j with $i < j < m$ and $\Gamma(v_i) = \Gamma(v_j) = \Gamma(v_m)$, and such that all the X-eventualities requested in the three nodes and fulfilled between v_{j+1} and v_m are fulfilled between v_{i+1} and v_j as well. Since for each $a \in \overline{\mathcal{T}}(k)$ we have one $c \in \Gamma(v_k)$ such that $a = \mathrm{a}(c)$ for all $0 \leq k \leq m$, this fact reflects onto the pre-model, hence $\overline{\mathcal{T}}(i) = \overline{\mathcal{T}}(j) = \overline{\mathcal{T}}(k)$, and all the X-eventualities requested in these atoms and fulfilled in $\mathcal{D}_{[j+1\ldots m]}$ are fulfilled in $\mathcal{D}_{[i+1\ldots j]}$ as well. That is, $\mathcal{D}_{[j+1\ldots m]}$ is a redundant segment, which contradicts the assumption that $\mathcal{D}$ is greedy by Lemma 24 and 25. □

5 Computational Complexity

We now employ our tableau calculus to show that the satisfiability for SLTL is PSPACE-complete. As SLTL is a direct extension of LTL, it immediately follows that SLTL is PSPACE-hard. Hence, we need only show that the satisfiability of SLTL formulae can be decided within PSPACE.

If our tableau-based decision procedure were to explicitly construct tableaux, then the resulting procedure would fail to be PSPACE as the tableau branches could be exponentially long (but not longer, as we will see) and the entirety of each branch would need to be kept in memory in order to check the LOOP and PRUNE rules. However, such limitations can be overcome by incorporating *non-determinism* into our tableau-based decision algorithm. Therefore, we let n be the size of Φ, and we confirm that any branch of a tableau is at most exponentially long in n.

Lemma 27. *Given a set of formulae $\Phi \subseteq \mathcal{L}_{\mathcal{V}}$, the length of a branch of the tableau for Φ is at most* exponential *in $|\Phi|$.*

Proof. First, note that the construction of a branch stops whenever the LOOP or PRUNE rule is triggered. The following two facts imply that the number of possible labels of a node is exponential: (1) The number of formulas that can appear in a constraint set is polynomial, (2) The number of different constraint sets is polynomial as there are no constraint sets $\langle c \rangle$ and $\langle c' \rangle$ in the same label such that $\langle c \rangle \subseteq \langle c' \rangle$.

Hence, in any branch, after at most an exponential number of nodes, two nodes u_i and u_j with $\Gamma(u_i) = \Gamma(u_j)$ must appear. If this pair satisfies the conditions of the LOOP rule, we are done (and the branch is exponential length). Otherwise, the branch construction continues. If the LOOP rule is never triggered, the PRUNE will. Now, suppose by contradiction that the PRUNE rule is triggered only $2^{\Omega(p(n))}$ nodes later with $p(n)$ a polynomial. Then, an exponential number of positions $j_0 < j_1 \ldots < j_k$, with $k \in \mathcal{O}(2^n)$, have to exist such that $\Gamma(u_{j_i}) = \Gamma(u_{j_0})$ for all $0 \leq i \leq k$. But then, observe that the set of X-eventualities fulfilled between each j_i and j_{i+1} can only grow. This means that after a polynomial number of repetitions of $\Gamma(u_{j_0})$, the PRUNE rule must be triggered. Note that since $j_{i+1} - j_i \in \mathcal{O}(2^{c(n)})$, this contradicts the assumption. □

By means of the above Lemma, we can exploit and employ non-determinism in our tableau-based decision procedure to determine the complexity of SLTL.

Theorem 28. *Satisfiability of* SLTL *is* PSPACE*-complete.*

Proof. We show the existence of a deterministic procedure to decide the satisfiability of a set of SLTL formulae Φ, by providing a *non-deterministic* one, and then appealing to the classic theorem by Savitch (Papadimitriou 1994) to state the existence of a deterministic one.

Our procedure traverses a tableau non-deterministically by *guessing* at each step which child of the current node to visit among the many created following the tableau rules. If a node accepted by the LOOP rule is found, the procedure returns *yes*. If a number of steps that exceeds the exponential upper bound established in Lemma 27 is reached, the procedure returns *no* (i.e., the computation branch is rejected). To be able to do this in polynomial space, the procedure also guesses at each step a Boolean flag *loop* that says whether the current node will be the one checked by the LOOP rule to find a loop, i.e., if the current position will be the one with the same label as the leaf. If *loop* is guessed to be true, the current label is saved and kept in memory for later. Also, from now on, the set of eventualities fulfilled is kept in memory. When any node with the same label as the saved one is found, it means the LOOP rule triggers on that branch, and the procedure can return *yes*. Otherwise, the procedure proceeds. Note that the procedure illustrated here keeps the following polynomial amount of data in memory: (1) The current label, (2) The saved loop label, if any, and (3) The set of X-eventualities fulfilled since the saved loop point, if any. Hence, the procedure works in polynomial space, showing SLTL satisfiability is PSPACE. □

6 Concluding Remarks

We have introduced *standpoint linear temporal logic* (SLTL)—a logic fusing the multi-perspective reasoning offered by standpoint logic with the dynamic reasoning offered by LTL. As discussed, SLTL permits one to model diverse and potentially conflicting semantic commitments held by a set of agents, while expressing how such commitments relate to temporal notions or change throughout time. To automate SLTL-reasoning, we define a sound, complete, and terminating tableau calculus, which supports (counter-)model extraction witnessing the (non-)satisfaction of SLTL formulae. We employed our tableau calculus in an analysis of SLTL's complexity, finding that the incorporation of standpoint modalities—despite increasing the modelling capacity of LTL—preserves the PSPACE-completeness of LTL.

Our contribution has practical implications. Not only it formally confirms that standpoint and temporal reasoning can be combined at no cost in complexity, but the employment of a tree-shaped tableau *à la* Reynolds paves the way for its efficient implementation. Indeed, we plan to transfer the *symbolic tableau* technique applied in the BLACK satisfiability checker (Geatti, Gigante, and Montanari 2019) to our tableau, thus providing an efficient SAT-based procedure to reason in standpoint linear temporal logic.

Acknowledgments

Tim S. Lyon has received funding from the European Research Council (Grant Agreement no. 771779, DeciGUT). Nicola Gigante acknowledges the support of the PURPLE project, 1st Open Call for Innovators of the AIPlan4EU H2020 project, a project funded by EU Horizon 2020 research and innovation programme under GA n. 101016442 (since 2021).

References

Artale, A.; Kontchakov, R.; Ryzhikov, V.; and Zakharyaschev, M. 2014. A cookbook for temporal conceptual data modelling with description logics. *ACM Transactions on Computational Logic* 15(3):25:1–25:50.

Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics in Artificial Intelligence* 22(1-2):5–27.

Balbiani, P.; Herzig, A.; and Troquard, N. 2008. Alternative axiomatics and complexity of deliberative STIT theories. *J. Philosophical Logic* 37(4):387–406.

Belnap, N., and Perloff, M. 1988. Seeing to it that: a canonical form for agentives. *Theoria* 54(3):175–199.

Belnap, N.; Perloff, M.; and Xu, M. 2001. *Facing the future: agents and choices in our indeterminist world*. Oxford University Press, Oxford.

Brafman, R. I., and De Giacomo, G. 2019. Planning for ltlf /ldlf goals in non-markovian fully observable nondeterministic domains. In Kraus, S., ed., *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1602–1608.

Brafman, R. I.; De Giacomo, G.; and Patrizi, F. 2018. Ltlf/ldlf non-markovian rewards. In McIlraith, S. A., and Weinberger, K. Q., eds., *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 1771–1778. AAAI Press.

Bratman, M. 1999. *Intention, Plans, and Practical Reason*. The David Hume Series. Cambridge University Press.

Cavada, R.; Cimatti, A.; Dorigatti, M.; Griggio, A.; Mariotti, A.; Micheli, A.; Mover, S.; Roveri, M.; and Tonetta, S. 2014. The nuXmv symbolic model checker. In Biere, A., and Bloem, R., eds., *Proceedings of the 26th International Conference on Computer Aided Verification*, 334–342. Springer.

De Giacomo, G.; Favorito, M.; Iocchi, L.; and Patrizi, F. 2020. Imitation learning over heterogeneous agents with restraining bolts. In Beck, J. C.; Buffet, O.; Hoffmann, J.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the 13th International Conference on Automated Planning and Scheduling*, 517–521. AAAI Press.

Della Monica, D.; Gigante, N.; Montanari, A.; Sala, P.; and Sciavicco, G. 2017. Bounded timed propositional temporal logic with past captures timeline-based planning with bounded constraints. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1008–1014.

Ditmarsch, H.; Hoek, W.; and Kooi, B. 2007. *Dynamic Epistemic Logic*, volume 337. Springer.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Geatti, L.; Gigante, N.; Montanari, A.; and Reynolds, M. 2021. One-pass and tree-shaped tableau systems for TPTL and TPTL_b+Past. *Information and Computation* 278:104599.

Geatti, L.; Gigante, N.; and Montanari, A. 2019. A SAT-Based encoding of the one-pass and tree-shaped tableau system for LTL. In Cerrito, S., and Popescu, A., eds., *Proceedings of the 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 11714 of *Lecture Notes in Computer Science*, 3–20. Springer.

Geatti, L.; Gigante, N.; and Montanari, A. 2021. BLACK: A fast, flexible and reliable LTL satisfiability checker. In Monica, D. D.; Pozzato, G. L.; and Scala, E., eds., *Proceedings of the 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, volume 2987 of *CEUR Workshop Proceedings*, 7–12. CEUR-WS.org.

Gómez Álvarez, L., and Rudolph, S. 2021. Standpoint logic: Multi-perspective knowledge representation. In Neuhaus, F., and Brodaric, B., eds., *Proceedings of the 12th International Conference on Formal Ontology in Information Systems*, volume 344 of *FAIA*, 3–17. IOS Press.

Hammond, L.; Abate, A.; Gutierrez, J.; and Wooldridge, M. J. 2021. Multi-agent reinforcement learning with temporal logic specifications. In Dignum, F.; Lomuscio, A.; Endriss, U.; and Nowé, A., eds., *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*, 583–592. ACM.

Li, J.; Yao, Y.; Pu, G.; Zhang, L.; and He, J. 2014. Aalta: an LTL satisfiability checker over infinite/finite traces. In Cheung, S.; Orso, A.; and Storey, M. D., eds., *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 731–734. ACM.

Lichtenstein, O., and Pnueli, A. 2000. Propositional Temporal Logics: Decidability and Completeness. *Logic Journal of the IGPL* 8(1):55–85.

Lutz, C. 2006. Complexity and succinctness of public announcement logic. In Nakashima, H.; Wellman, M. P.; Weiss, G.; and Stone, P., eds., *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 137–143. Association for Computing Machinery.

Lyon, T. S., and Gómez Álvarez, L. 2022. Automating Reasoning with Standpoint Logic via Nested Sequents. In Kern-Isberner, G.; Lakemeyer, G.; and Meyer, T., eds., *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, 257–266.

Mayer, M. C.; Limongelli, C.; Orlandini, A.; and Poggioni, V. 2007. Linear temporal logic as an executable semantics for planning languages. *Journal of Logic, Language and Information* 16(1):63–89.

McCabe-Dansted, J. C., and Reynolds, M. 2017. A parallel linear temporal logic tableau. In Bouyer, P.; Orlandini,

A.; and Pietro, P. S., eds., *Proceedings Eighth International Symposium on Games, Automata, Logics and Formal Verification*, volume 256, 166–179.

Murakami, Y. 2004. Utilitarian deontic logic. In Schmidt, R. A.; Pratt-Hartmann, I.; Reynolds, M.; and Wansing, H., eds., *Advances in Modal Logic 5, Papers from the 5th Conference on Advances in Modal logic*, 211–230. King's College Publications.

Papadimitriou, C. H. 1994. *Computational complexity*. Addison-Wesley.

Plaza, J. 2007. Logics of public communications. *Synthese* 158(2):165–179.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, 46–57. IEEE Computer Society.

Rao, A. S., and Georgeff, M. P. 1998. Decision Procedures for BDI Logics. *Journal of Logic and Computation* 8(3):293–343.

Reynolds, M. 2016. A New Rule for LTL Tableaux. In Cantone, D., and Delzanno, G., eds., *Proceedings of the 7th International Symposium on Games, Automata, Logics and Formal Verification*, volume 226 of *EPTCS*, 287–301.

van Benthem, J.; Gerbrandy, J.; Hoshi, T.; and Pacuit, E. 2009. Merging frameworks for interaction. *J. Philos. Log.* 38(5):491–526.

van Berkel, K., and Lyon, T. 2021. The varieties of ought-implies-can and deontic stit logic. In Liu, F.; Marra, A.; Portner, P.; and Putte, F. V. D., eds., *Proceedings of the 15th International Conference on Deontic Logic and Normative Systems*, 57–76. College Publications.

Wolter, F., and Zakharyaschev, M. 1998. Satisfiability problem in description logics with modal operators. In Cohn, A. G.; Schubert, L. K.; and Shapiro, S. C., eds., *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, 512–523. Morgan Kaufmann.