Temporal Answer Set Programming

FELICIDAD AGUADO and PEDRO CABALAR
University of Corunna, Spain

MARTÍN DIÉGUEZ
LERIA, Université d'Angers, France

GILBERTO PÉREZ
University of Corunna, Spain

TORSTEN SCHAUB and ANNA SCHUHMANN
University of Potsdam, Germany

CONCEPCIÓN VIDAL
University of Corunna, Spain

submitted [n/a]; revised [n/a]; accepted [n/a]

Abstract

In this paper we present an overview on Temporal Logic Programming under the perspective of its application for Knowledge Representation and declarative problem solving. The syntax of temporal logic programs is the result of combining usual rules with temporal modal operators, as those from *Linear-time Temporal Logic* (LTL). In the paper, we focus on the main recent results of the non-monotonic formalism called *Temporal Equilibrium Logic* (TEL) that is defined for the full syntax of LTL, but performs a model selection criterion based on *Equilibrium Logic*, a well known logical characterization of Answer Set Programming (ASP). As a result, we obtain a proper extension of the stable models semantics for the general case of arbitrary temporal formulas. We recall the basic definitions for TEL and its monotonic basis, the temporal logic of Here-and-There (THT), and study the differences between infinite versus finite trace length. We also provide other useful results, such as the translation into other formalisms like Quantified Equilibrium Logic or second-order LTL, and some techniques for computing temporal stable models based on automata construction. In a second part of the paper, we focus on practical aspects, defining a syntactic fragment called *temporal logic programs* closer to ASP, and explaining how this has been exploited in the construction of the solver `telingo`, a temporal extension of the well-known ASP solver `clingo` that uses its incremental solving capabilities.

1 Introduction

Temporal Logic Programming (TLP) is an extension of logic programs that incorporates modal temporal operators, usually from Linear-time Temporal Logic (LTL; Kamp 1968; Pnueli 1977). The origins of TLP date back to the mid 1980s, when several LTL extensions were proposed (Moszkowski 1986; Fujita et al. 1986; Gabbay 1987b; Abadi and Manna 1989; Baudinet 1992; Orgun and Wadge 1992). However, the initial interest has gradually cooled down over time and LTL extensions have neither become a common feature nor had a substantial impact in Prolog. With the appearance of *Answer Set Programming* (ASP; Lifschitz 2002) and its use as a formalism for practical knowledge representation, the interest in the specification of dynamic systems was renewed. ASP has been commonly used for temporal reasoning and the representation of action theories, following the methodology proposed in (Gelfond and Lifschitz 1993).

In this paper, we focus on the combination of LTL and ASP and, in particular, in the TLP formalism called *Temporal Equilibrium Logic* (TEL; Cabalar and Vega 2007). As suggested by its name, it combines LTL with the ASP logical characterization based on *Equilibrium Logic* (Pearce 1997). Although TEL was introduced more than a decade ago and a first survey was already presented in (Aguado et al. 2013), several important advances have taken place since then, providing significant breakthroughs. On the theoretical side, apart from new results on complexity and expressiveness (Bozzelli and Pearce 2015; Balbiani and Diéguez 2016; Aguado et al. 2017), the most important feature has arguably been the redefinition of TEL to cope not only with infinite traces, as in its original version, but also with finite traces (Cabalar et al. 2018), following similar steps to the same variation introduced for LTL in (De Giacomo and Vardi 2013). On the practical side, finite traces are much better adapted to the way in which ASP is used to solve planning problems and have paved the way for the introduction of temporal operators into the ASP solver `clingo`, giving birth to the first full-fledged temporal ASP solver called `telingo` (Cabalar et al. 2019). In the paper at hand, we give a revised definition of TEL that incorporates the new advances, present the most general version of the logic (which neither imposes nor forbids finiteness of traces) and then specify the two variants: TEL_ω for infinite traces, as first defined in (Cabalar and Vega 2007), and TEL_f for finite ones, as recently introduced in (Cabalar et al. 2018). We revisit previous results for TEL_ω under the more general umbrella of TEL and then compare their effects to the TEL_f case, providing an homogeneous presentation of the main achievements in the topic.

The rest of the paper is organized as follows. In the next section, we introduce TEL semantics, whose basic definition is made in terms of a models selection criterion on top of the monotonic basis, the logic of *Temporal Here-and-There* (THT). We also explain the relation of TEL to standard ASP by proving that the former can be seen as a fragment of Quantified Equilibrium Logic with monadic predicates and a linear order relation. In Section 3, we study the monotonic basis, THT, in full detail, providing some properties and relations to other formalisms. Section 4 is focused on different techniques to compute temporal equilibrium models via automata construction. This is done in two steps: first defining the temporal equilibrium models of a temporal theory in terms of a second-order formula expressed in the logic of *Quantified LTL* (QLTL), and then building the automata from this QLTL formula. In the next section, we study a normal form for temporal theories under TEL semantics: *temporal logic programs*. This normal form is used as the basis for a pair of translations from temporal logic programs (for finite traces) into standard ASP programs. We also define a syntactic fragment, called *present-centered* rules, that facilitates the application of incremental reasoning and, eventually, has led to the construction of the tool `telingo`, an extension of `clingo` with temporal operators. Finally, Section 7 discusses some related work and Section 8 concludes the paper.

2 Temporal Equilibrium Logic

The definition of *(Linear-time) Temporal Equilibrium Logic* (TEL) is done in two steps. First, we define a monotonic logic called *(Linear-time) Temporal Here-and-There* (THT), a temporal extension of the intermediate logic of Here-and-There (Heyting 1930). In a second step, we select some models from THT that are said to be *in equilibrium*, obtaining in this way a non-monotonic entailment relation.

2.1 Monotonic basis: Temporal Here-and-There

The syntax of THT (and TEL) is the same as for LTL with past operators. In particular, in this paper, we use the following notation. Given a set $\mathcal{A}$ of propositional variables (called *alphabet*), *temporal formulas* φ are defined by the grammar:

$$\varphi ::= a \mid \bot \mid \varphi_1 \otimes \varphi_2 \mid \bullet\varphi \mid \varphi_1 \,\mathbf{S}\, \varphi_2 \mid \varphi_1 \,\mathbf{T}\, \varphi_2 \mid \circ\varphi \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \varphi_1 \,\mathbf{R}\, \varphi_2$$

where $a \in \mathcal{A}$ is an atom and $\otimes$ is any binary Boolean connective $\otimes \in \{\to, \wedge, \vee\}$. The last six cases correspond to the temporal connectives whose names are listed below:

Past	$\bullet$	for *previous*	*Future*	$\circ$	for *next*
	$\mathbf{S}$	for *since*		$\mathbf{U}$	for *until*
	$\mathbf{T}$	for *trigger*		$\mathbf{R}$	for *release*
				$\mathbb{W}$	for *while*

We also define several common derived operators like the Boolean connectives $\top \stackrel{def}{=} \neg\bot$, $\neg\varphi \stackrel{def}{=} \varphi \to \bot$, $\varphi \leftrightarrow \psi \stackrel{def}{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$, and the following temporal operators:

$\blacksquare\varphi$	$\stackrel{def}{=}$	$\bot \,\mathbf{T}\, \varphi$	*always before*	$\square\varphi$	$\stackrel{def}{=}$	$\bot \,\mathbf{R}\, \varphi$	*always afterward*
$\blacklozenge\varphi$	$\stackrel{def}{=}$	$\top \,\mathbf{S}\, \varphi$	*eventually before*	$\lozenge\varphi$	$\stackrel{def}{=}$	$\top \,\mathbf{U}\, \varphi$	*eventually afterward*
$\mathbf{I}$	$\stackrel{def}{=}$	$\neg\bullet\top$	*initial*	$\mathbf{F}$	$\stackrel{def}{=}$	$\neg\circ\top$	*final*
$\widehat{\bullet}\varphi$	$\stackrel{def}{=}$	$\bullet\varphi \vee \mathbf{I}$	*weak previous*	$\widehat{\circ}\varphi$	$\stackrel{def}{=}$	$\circ\varphi \vee \mathbf{F}$	*weak next*

A *(temporal) theory* is a (possibly infinite) set of temporal formulas. Note that we use solid operators to refer to the past, while future-time operators are denoted by outlined symbols.

Although THT and LTL share the same syntax, they have a different semantics, the former being a weaker logic than the latter. The semantics of LTL relies on the concept of a *trace*, a (possibly infinite) sequence of *states*, each of which is a set of atoms. For defining traces, we start by introducing some notation to deal with intervals of integer time points. Given $a \in \mathbb{N}$ and $b \in \mathbb{N} \cup \{\omega\}$, we let $[a..b]$ stand for the set $\{i \in \mathbb{N} \mid a \leq i \leq b\}$, $[a..b)$ for $\{i \in \mathbb{N} \mid a \leq i < b\}$ and $(a..b]$ for $\{i \in \mathbb{N} \mid a < i \leq b\}$. In LTL, a *trace* $\mathbf{T}$ of length λ over alphabet $\mathcal{A}$ is a sequence $\mathbf{T} = (T_i)_{i\in[0..\lambda)}$ of sets $T_i \subseteq \mathcal{A}$. We sometimes use the notation $|\mathbf{T}| \stackrel{def}{=} \lambda$ to stand for the length of the trace. We say that $\mathbf{T}$ is *infinite* if $|\mathbf{T}| = \omega$ and *finite* if $|\mathbf{T}| \in \mathbb{N}$. To represent a given trace, we write a sequence of sets of atoms concatenated with '$\cdot$'. For instance, the finite trace $\{a\} \cdot \emptyset \cdot \{a\} \cdot \emptyset$ has length 4 and makes a true at even time points and false at odd ones. For infinite traces, we sometimes use ω-regular expressions like, for instance, in the infinite trace $(\{a\} \cdot \emptyset)^\omega$ where all even positions make a true and all odd positions make it false.

At each state T_i in a trace, an atom a can only be true, viz. $a \in T_i$, or false, $a \notin T_i$. The logic THT weakens this truth assignment, following the same intuitions as the (non-temporal) logic of HT. In THT, atom a in a state i can be in one of these three possibilities: *false*, *assumed* (or true by default) or *proven* (or certainly true). Anything proved has to be assumed, but the opposite does not necessarily hold. Following this idea, state i is represented as a pair of sets of atoms $\langle H_i, T_i\rangle$ with $H_i \subseteq T_i \subseteq \mathcal{A}$ where H_i (standing for "here") contains the proven atoms, whereas T_i (standing for "there") contains the assumed atoms. On the other hand, false atoms are just the ones not assumed, captured by $\mathcal{A} \setminus T_i$. Accordingly, *Here-and-There trace* (for short HT-*trace*) of length λ over alphabet $\mathcal{A}$ is a sequence of these pairs $(\langle H_i, T_i\rangle)_{i\in[0..\lambda)}$ with $H_i \subseteq T_i$ for any $i \in [0..\lambda)$. For convenience, we usually represent the HT-trace as the pair $\langle \mathbf{H}, \mathbf{T}\rangle$ of traces $\mathbf{H} = (H_i)_{i\in[0..\lambda)}$ and $\mathbf{T} = (T_i)_{i\in[0..\lambda)}$. Given $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$, we also denote its length as $|\mathbf{M}| \stackrel{def}{=} |\mathbf{H}| = |\mathbf{T}| = \lambda$. Note that the two traces $\mathbf{H}$, $\mathbf{T}$ must satisfy a kind of order relation, since $H_i \subseteq T_i$ for each time point i. Formally, we define the ordering $\mathbf{H} \le \mathbf{T}$ between two traces of the same length λ as $H_i \subseteq T_i$ for each $i \in [0..\lambda)$. Furthermore, we define $\mathbf{H} < \mathbf{T}$ as both $\mathbf{H} \le \mathbf{T}$ and $\mathbf{H} \ne \mathbf{T}$. Thus, an HT-trace can also be defined as any pair $\langle \mathbf{H}, \mathbf{T}\rangle$ of traces such that $\mathbf{H} \le \mathbf{T}$. The particular type of HT-traces satisfying $\mathbf{H} = \mathbf{T}$ are called *total*.

We proceed by generalizing the extension of HT with temporal operators, called THT in (Aguado et al. 2013), to HT-traces of fixed length in order to integrate finite as well as infinite traces. Given any HT-trace $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$, we define the THT satisfaction of formulas as follows.

Definition 1 (THT-*satisfaction; Aguado et al. 2013; Cabalar et al. 2018* [1])
An HT-trace $\mathbf{M} = \langle \mathbf{H}, \mathbf{T}\rangle$ of length λ over alphabet $\mathcal{A}$ *satisfies* a temporal formula φ at time point $k \in [0..\lambda)$, written $\mathbf{M}, k \models \varphi$, if the following conditions hold:

1. $\mathbf{M}, k \models \top$ and $\mathbf{M}, k \not\models \bot$
2. $\mathbf{M}, k \models a$ if $a \in H_k$ for any atom $a \in \mathcal{A}$
3. $\mathbf{M}, k \models \varphi \wedge \psi$ iff $\mathbf{M}, k \models \varphi$ and $\mathbf{M}, k \models \psi$
4. $\mathbf{M}, k \models \varphi \vee \psi$ iff $\mathbf{M}, k \models \varphi$ or $\mathbf{M}, k \models \psi$
5. $\mathbf{M}, k \models \varphi \to \psi$ iff $\langle \mathbf{H}', \mathbf{T}\rangle, k \not\models \varphi$ or $\langle \mathbf{H}', \mathbf{T}\rangle, k \models \psi$, for all $\mathbf{H}' \in \{\mathbf{H}, \mathbf{T}\}$
6. $\mathbf{M}, k \models \bullet\varphi$ iff $k > 0$ and $\mathbf{M}, k{-}1 \models \varphi$
7. $\mathbf{M}, k \models \varphi \,\mathbf{S}\, \psi$ iff for some $j \in [0..k]$, we have $\mathbf{M}, j \models \psi$ and $\mathbf{M}, i \models \varphi$ for all $i \in (j..k]$
8. $\mathbf{M}, k \models \varphi \,\mathbf{T}\, \psi$ iff for all $j \in [0..k]$, we have $\mathbf{M}, j \models \psi$ or $\mathbf{M}, i \models \varphi$ for some $i \in (j..k]$
9. $\mathbf{M}, k \models \circ\varphi$ iff $k + 1 < \lambda$ and $\mathbf{M}, k{+}1 \models \varphi$
10. $\mathbf{M}, k \models \varphi \,\mathbf{U}\, \psi$ iff for some $j \in [k..\lambda)$, we have $\mathbf{M}, j \models \psi$ and $\mathbf{M}, i \models \varphi$ for all $i \in [k..j)$
11. $\mathbf{M}, k \models \varphi \,\mathbf{R}\, \psi$ iff for all $j \in [k..\lambda)$, we have $\mathbf{M}, j \models \psi$ or $\mathbf{M}, i \models \varphi$ for some $i \in [k..j)$
12. $\mathbf{M}, k \models \varphi \,\mathbb{W}\, \psi$ iff for all $j \in [k..\lambda)$, we have $\langle \mathbf{H}', \mathbf{T}\rangle, j \models \varphi$ or $\langle \mathbf{H}', \mathbf{T}\rangle, i \not\models \psi$ for some $i \in [k..j)$ and for all $\mathbf{H}' \in \{\mathbf{H}, \mathbf{T}\}$

⊠

An HT-trace $\mathbf{M}$ is a *model* of a temporal theory Γ if $\mathbf{M}, 0 \models \varphi$ for all $\varphi \in \Gamma$. We write $\mathrm{THT}(\Gamma, \lambda)$ to stand for the set of THT-models of length λ of a theory Γ, and define $\mathrm{THT}(\Gamma) \stackrel{def}{=} \mathrm{THT}(\Gamma, \omega) \cup \bigcup_{\lambda\in\mathbb{N}} \mathrm{THT}(\Gamma, \lambda)$. That is, $\mathrm{THT}(\Gamma)$ is the whole set of models of Γ of any length. For $\Gamma = \{\varphi\}$, we just write $\mathrm{THT}(\varphi, \lambda)$ and $\mathrm{THT}(\varphi)$. We can analogously define $\mathrm{LTL}(\Gamma, \lambda)$, that is, the set of traces of length λ that satisfy theory Γ, and $\mathrm{LTL}(\Gamma)$, that is, the LTL-models of Γ any length. We omit specifying LTL satisfaction since it coincides with THT when HT-traces are total.

[1] The while operator $\mathbb{W}$ is introduced in (Aguado et al. 2020).

Proposition 1 (*Aguado et al. 2013; Cabalar et al. 2018*)
For any trace $\mathbf{T}$ of length λ, any temporal formula φ, and any time point $k \in [0..\lambda)$, we have: $\mathbf{T}, k \models \varphi$ in LTL iff $\langle \mathbf{T}, \mathbf{T} \rangle, k \models \varphi$. ⊠

In fact, total models can be forced by adding the following set of *excluded middle* axioms:

$$\Box(a \vee \neg a) \qquad \text{for each atom } a \in \mathcal{A} \text{ in the alphabet.} \tag{EM}$$

Proposition 2 (*Aguado et al. 2013; Cabalar et al. 2018*)
Let $\langle \mathbf{H}, \mathbf{T} \rangle$ be an HT-trace and γ the theory containing all excluded middle axioms for all atoms in $\mathcal{A}$. Then, $\langle \mathbf{H}, \mathbf{T} \rangle, 0 \models \gamma$ iff $\mathbf{H} = \mathbf{T}$. ⊠

Satisfaction of derived operators can be easily deduced:

Proposition 3 (*Aguado et al. 2013; Cabalar et al. 2018*)
Let $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ be an HT-trace of length λ over $\mathcal{A}$. Given the respective definitions of derived operators, we get the following satisfaction conditions:

12. $\mathbf{M}, k \models \mathsf{I}$ iff $k = 0$
13. $\mathbf{M}, k \models \widehat{\bullet}\varphi$ iff $k = 0$ or $\mathbf{M}, k{-}1 \models \varphi$
14. $\mathbf{M}, k \models \blacklozenge\varphi$ iff $\mathbf{M}, i \models \varphi$ for some $i \in [0..k]$
15. $\mathbf{M}, k \models \blacksquare\varphi$ iff $\mathbf{M}, i \models \varphi$ for all $i \in [0..k]$
16. $\mathbf{M}, k \models \mathsf{F}$ iff $k = \lambda{-}1$
17. $\mathbf{M}, k \models \widehat{\circ}\varphi$ iff $k + 1 = \lambda$ or $\mathbf{M}, k{+}1 \models \varphi$
18. $\mathbf{M}, k \models \Diamond\varphi$ iff $\mathbf{M}, i \models \varphi$ for some $i \in [k..\lambda)$
19. $\mathbf{M}, k \models \Box\varphi$ iff $\mathbf{M}, i \models \varphi$ for all $i \in [k..\lambda)$

⊠

A formula φ is a *tautology* (or is *valid*), written $\models \varphi$, iff $\mathbf{M}, k \models \varphi$ for any HT-trace $\mathbf{M}$ and any $k \in [0..\lambda)$. We call the logic induced by the set of all tautologies *Temporal logic of Here-and-There* (THT for short).

Several types of equivalence can be defined in THT. In this sense, it is important to observe that being equivalent is something generally stronger than simply having the same set of models. Two formulas φ, ψ are (globally) *equivalent*, written $\varphi \equiv \psi$, iff $\models \varphi \leftrightarrow \psi$, that is, $\mathbf{M}, k \models \varphi \leftrightarrow \psi$ for any HT-trace $\mathbf{M}$ of length λ and any $k \in [0..\lambda)$. Whenever φ and ψ are equivalent, they are completely interchangeable in any theory Γ without altering the semantic interpretation of Γ. We say that φ, ψ are just *initially equivalent*, written $\varphi \equiv_0 \psi$, if they have the same models $\mathrm{THT}(\varphi) = \mathrm{THT}(\psi)$, that is, $\mathbf{M}, 0 \models \varphi$ iff $\mathbf{M}, 0 \models \psi$, for any HT-trace $\mathbf{M}$. Obviously, $\varphi \equiv \psi$ implies $\varphi \equiv_0 \psi$ but not vice versa. For example, note that $\bullet a \equiv_0 \bot$, since $\bullet a$ is always false at the initial situation, whereas in the general case $\bullet a \not\equiv \bot$ or, otherwise, we could always replace $\bullet a$ by $\bot$ in any context.

One important remark is that the finiteness of a trace $\langle \mathbf{H}, \mathbf{T} \rangle$ only affects the satisfaction of formulas dealing with future-time operators. In particular, if $\langle \mathbf{H}, \mathbf{T} \rangle$ has some finite length $\lambda = n$, then, in the semantics for $\mathbf{U}$, $\mathbf{R}$ and $\mathbb{W}$ in Definition 1, index j ranges over the finite interval $\{k, \ldots, n-1\}$. Besides, if $\lambda = n$, the satisfaction of $\circ$ forces $k < n$, which implies that there does exist a next state $k+1$. As a result, the formula $\circ\top$ is not always satisfied, since it is false whenever $k = n = \lambda$.

Operators I and F exclusively depend on the value of time point k, so that the valuation of atoms in $\langle \mathbf{H}, \mathbf{T} \rangle$ is irrelevant to them. As a result, they behave "classically" and satisfy the law of the excluded middle, that is, $\models \mathsf{I} \vee \neg \mathsf{I}$ and $\models \mathsf{F} \vee \neg \mathsf{F}$ are THT tautologies. Besides, operator F can only be true in finite traces. This implies that the inclusion of axiom $\Diamond \mathsf{F}$ in any theory forces its models to be finite traces, while including its negation $\neg \Diamond \mathsf{F}$ causes the opposite effect, that is, all models of the theory are infinite traces.

Several logics stronger than THT can be obtained by the addition of axioms (or the corresponding restriction on sets of traces). For instance, THT_ω is defined as $\mathrm{THT}+\{\neg \Diamond \mathsf{F}\}$, that is, THT where we exclusively consider infinite HT-traces.[2] THT_f, the finite-trace version, corresponds to $\mathrm{THT} + \{\Diamond \mathsf{F}\}$. Linear Temporal Logic for possibly infinite traces, LTL, can be obtained as $\mathrm{THT} + \{(\mathrm{EM})\}$, that is, THT with total HT-traces, LTL_ω is captured by $\mathrm{THT}_\omega + \{(\mathrm{EM})\}$, i.e. infinite and total HT-traces, and finally LTL_f can be obtained as $\mathrm{THT}_f + \{(\mathrm{EM})\}$, that is, LTL on finite traces (De Giacomo and Vardi 2013).

We study more properties and results about THT later on, but we proceed next to define its non-monotonic extension, TEL.

2.2 Non-monotonic extension: Temporal Equilibrium Logic

Given a set of THT-models, we define the ones in equilibrium as follows.

Definition 2 (*Temporal Equilibrium/Stable Model*)
Let $\mathfrak{S}$ be some set of HT-traces.

A total HT-trace $\langle \mathbf{T}, \mathbf{T} \rangle \in \mathfrak{S}$ is a *temporal equilibrium model* of $\mathfrak{S}$ iff there is no other $\mathbf{H} < \mathbf{T}$ such that $\langle \mathbf{H}, \mathbf{T} \rangle \in \mathfrak{S}$.

The trace $\mathbf{T}$ is called a *temporal stable model* (TS-model) of $\mathfrak{S}$. ⊠

We further talk about temporal equilibrium or temporal stable models of a theory Γ when $\mathfrak{S} = \mathrm{THT}(\Gamma)$, respectively. Moreover, we write $\mathrm{TEL}(\Gamma, \lambda)$ and $\mathrm{TEL}(\Gamma)$ to stand for the temporal equilibrium models of $\mathrm{THT}(\Gamma, \lambda)$ and $\mathrm{THT}(\Gamma)$ respectively. The corresponding sets of TS-models are denoted as $\mathrm{TSM}(\Gamma, \lambda)$ and $\mathrm{TSM}(\Gamma)$ respectively. One interesting observation is that, since temporal equilibrium models are total models $\langle \mathbf{T}, \mathbf{T} \rangle$, due to Proposition 1, we obtain $\mathrm{TSM}(\Gamma, \lambda) \subseteq \mathrm{LTL}(\Gamma, \lambda)$ that is, temporal stable models are a subset of LTL-models.

Since the ordering relation among traces is only defined for a fixed λ, the following can be easily observed:

Proposition 4 (*Cabalar et al. 2018*)
The set of temporal equilibrium models of Γ can be partitioned by the trace length λ, that is, $\bigcup_{\lambda=0}^{\omega} \mathrm{TEL}(\Gamma, \lambda) = \mathrm{TEL}(\Gamma)$. ⊠

Temporal Equilibrium Logic (TEL) is the (non-monotonic) logic induced by temporal equilibrium models. We can also define the variants TEL_ω and TEL_f by applying the corresponding restriction to infinite and finite traces, respectively.

As an example of non-monotonicity, consider the formula

$$\Box(\bullet \mathit{loaded} \wedge \neg \mathit{unloaded} \rightarrow \mathit{loaded}) \tag{1}$$

along with literal *loaded* which combines the inertia for *loaded* with the initial state for that

[2] This corresponds to the (stronger) version of THT considered previously by Aguado et al. (2013).

fluent. Without entering into too much detail, this formula behaves as the logic program consisting of fact `loaded(0)` and rule "`loaded(T) :- loaded(T-1), not unloaded(T)`" for any time point `T>0`. As expected, for some fixed λ, we get a unique temporal stable model of the form $(\ \{loaded\}\ )^{\lambda}$. This entails that *loaded* is always true, $\square loaded$, as there is no reason for *unloaded* to become true. Note that in the most general case of TEL, we actually get one stable model per each possible λ, including $\lambda = \omega$. Now, consider formula (1) along with $loaded \wedge \circ\circ unloaded$ which amounts to adding the fact `unloaded(2)`. As expected, for each λ, the only temporal stable model now is $\mathbf{T} = \{loaded\} \cdot \{loaded\} \cdot \{unloaded\} \cdot \emptyset^{\alpha}$ where α can be $*$ or ω. Note that by making $\circ\circ unloaded$ true, we are also forcing $|\mathbf{T}| \geq 3$, that is, there are no temporal stable models (nor even THT-models) of length smaller than three. Thus, by adding the new information $\circ\circ unloaded$ some conclusions that could be derived before, such as $\square loaded$, are not derivable any more.

As an example emphasizing the behavior of finite traces, take the formula

$$\square(\neg a \rightarrow \circ a) \tag{2}$$

which can be seen as a program rule "`a(T+1) :- not a(T)`" for any natural number `T`. As expected, temporal stable models make a false in even states and true in odd ones. However, we cannot take finite traces making a false at the final state $\lambda - 1$, since the rule would force $\circ a$ and this implies the existence of a successor state. As a result, the temporal stable models of this formula have the form $(\emptyset \cdot \{a\})^{+}$ for finite traces in TEL_f, or the infinite trace $(\emptyset \cdot \{a\})^{\omega}$ in TEL_{ω}.

Another interesting example is the temporal formula

$$\square(\neg\circ a \rightarrow a) \wedge \square(\circ a \rightarrow a).$$

The corresponding rules "`a(T) :- not a(T+1)`" and "`a(T) :- a(T+1)`" have no stable model (Fages 1994) when grounded for all natural numbers `T`. This is because there is no way to build a finite proof for any `a(T)`, as it depends on infinitely many next states to be evaluated. The same happens in TEL_{ω}, that is, we get no infinite temporal stable model. However in TEL_f, we can use the fact that $\circ a$ is always false in the last state. Then, $\square(\neg\circ a \rightarrow a)$ supports a in that state and therewith $\square(\circ a \rightarrow a)$ inductively supports a everywhere.

As an example of a temporal expression not so close to logic programming, consider, for instance, the formula $\square\Diamond a$, which is normally used in LTL_{ω} to assert that a occurs infinitely often. As discussed in (De Giacomo and Vardi 2013), if we assume finite traces, then the formula collapses to $\square(\mathbf{F} \rightarrow a)$ in LTL_f, that is, a is true at the final state (and either true or false everywhere else). The same behavior is obtained in THT_{ω} and THT_f, respectively. However, if we move to TEL, a truth minimization is additionally required. As a result, in TEL_f, we obtain a unique temporal stable model for each fixed $\lambda \in \mathbb{N}$, in which a is true at the last state, and false everywhere else. Unlike this, TEL_{ω} yields no temporal stable model at all. This is because for any $\mathbf{T}$ with an infinite number of a's we can always take some $\mathbf{H}$ from which we remove a at some state, and still have an infinite number of a's in $\mathbf{H}$. Thus, for any total THT_{ω}-model $\langle \mathbf{T}, \mathbf{T} \rangle$ of $\square\Diamond a$ there always exists some model $\langle \mathbf{H}, \mathbf{T} \rangle$ with strictly smaller $\mathbf{H} < \mathbf{T}$. Note that we can still specify infinite traces with an infinite number of occurrences of a, but at the price of *removing the truth minimization* for that atom. This can be done, for instance, by adding the excluded middle axiom (EM). In this way, infinite traces satisfying $\square\Diamond a \wedge \square(a \vee \neg a)$ are those that contain an infinite number of a's. In fact, if we add the excluded middle axiom for all atoms, TEL collapses into LTL, as stated below.

Proposition 5 (*Aguado et al. 2013; Cabalar et al. 2018*)
Let Γ be a temporal theory over $\mathcal{A}$ and γ be the set of all excluded middle axioms for all atoms in $\mathcal{A}$.

Then, $\mathrm{TSM}(\Gamma \cup \gamma) = \mathrm{LTL}(\Gamma)$. ⊠

2.3 Relation to ASP

As we saw in the above examples, there seems to be a connection between temporal formulas over propositional atoms like a and logic programs with atoms for (monadic) predicates, viz. `a(T)`, with an integer argument `T` representing a time point. This connection is strongly related to Kamp's well-known theorem (Kamp 1968) that allows for translating LTL into Monadic First-Order Logic with a linear order $<$ relation, MFO($<$) for short. In this section, we show how Kamp's translation is also applicable to TEL, so that the latter can also be reduced to (Monadic) Quantified Equilibrium Logic, which essentially covers ASP for predicates with one argument. This connection reinforces the adequacy of TEL as a suitable temporal extension of ASP.

We begin by revisiting the definition of *Quantified Equilibrium Logic* (QEL; Pearce and Valverde 2006). This logic allows first-order logic programs in ASP to be partially simplified before grounding. Moreover, its monotonic basis, *Quantified Here-and-There* (QHT), can be used to check the property of strong equivalence, analogous to HT in the propositional case. The definition of QHT is based on a first-order language denoted by $\langle C, F, P\rangle$, where C, F and P are three disjoint sets representing constants, functions and predicates, respectively. Given a set of constants D, we define:

- $\mathcal{T}(D, F)$ as the set of all ground terms that can be built with functions in F and constants in D, and
- $\mathcal{A}(D, P)$ as the set of all ground atomic sentences that can be formed with predicates in P and constants in D.

In its most general version, QHT allows for different domains in the here and there worlds and the interpretation of equality can also be varied in each world. In this paper, though, we use the most common option when applied to ASP, that is, QHT with "*static domains and decidable equality*," dealing with a common universe and a fixed interpretation of equality. In this setting, a QHT-interpretation is a tuple $\mathcal{M} = \langle (D, \sigma), H, T\rangle$ such that:

- D is a (possibly infinite) non-empty set of constant names identifying each element in the universe. For simplicity, we use the same name for the constant and the universe element.
- $\sigma : \mathcal{T}(C \cup D, F) \to D$ is a mapping from ground terms into elements of D satisfying $\sigma(d) = d$ and structural recursion $\sigma(f(t_1, \ldots, t_n)) = \sigma(f(\sigma(t_1), \ldots, \sigma(t_n)))$.
- H and T are sets of atomic sentences satisfying $H \subseteq T \subseteq \mathcal{A}(D, P)$.

Given two QHT-interpretations, $\mathcal{M} = \langle (D, \sigma), H, T\rangle$ and $\mathcal{M}' = \langle (D', \sigma'), H', T'\rangle$, we say that $\mathcal{M} \leq \mathcal{M}'$ iff $D = D'$, $\sigma = \sigma'$, $T = T'$ and $H \subseteq H'$. If, additionally, $H \subset H'$ we say that the relation is strict and denote by $\mathcal{M} < \mathcal{M}'$.

Definition 3 (*QHT-satisfaction*)
A QHT-interpretation $\mathcal{M} = \langle (D, \sigma), H, T \rangle$ satisfies a first-order formula α, written $\mathcal{M} \models \alpha$, if the following conditions hold:

- $\mathcal{M} \models \top$ and $\mathcal{M} \not\models \bot$
- $\mathcal{M} \models p(\tau_1, \cdots, \tau_n)$ iff $p(\sigma(\tau_1), \cdots, \sigma(\tau_n)) \in H$
- $\mathcal{M} \models \tau = \tau'$ iff $\sigma(\tau) = \sigma(\tau')$
- $\mathcal{M} \models \varphi \wedge \psi$ iff $\mathcal{M} \models \varphi$ and $\mathcal{M} \models \psi$
- $\mathcal{M} \models \varphi \vee \psi$ iff $\mathcal{M} \models \varphi$ or $\mathcal{M} \models \psi$
- $\mathcal{M} \models \varphi \rightarrow \psi$ iff $\langle (D, \sigma), X, T \rangle \not\models \varphi$ or $\langle (D, \sigma), X, T \rangle \models \psi$, for $X \in \{H, T\}$
- $\mathcal{M} \models \forall x\ \varphi(x)$ iff $\mathcal{M} \models \varphi(d)$, for all $d \in D$
- $\mathcal{M} \models \exists x\ \varphi(x)$ iff $\mathcal{M} \models \varphi(d)$, for some $d \in D$ ⊠

Equilibrium models for first-order theories are defined as follows.

Definition 4 (*Quantified Equilibrium Model*)
Let φ be a first-order formula. A total QHT-interpretation $\mathcal{M} = \langle (D, \sigma), T, T \rangle$ is a first-order equilibrium model of φ if $\mathcal{M} \models \varphi$ and there is no model $\mathcal{M}' < \mathcal{M}$ of φ. ⊠

We now focus on a particular fragment of QHT, called MHT($<$), by imposing the following limitations:

1. $C = [i..\lambda)$ where $\lambda \in \mathbb{N}$ or $\lambda = \omega$ and $D = \{\mathbf{u}\} \cup C$ where $\mathbf{u}$ stands for "undefined."
2. We only allow for (unary) functions "+1" and "−1" with the expected meaning:

$$\sigma(\tau + 1) \stackrel{def}{=} \begin{cases} \sigma(\tau) + 1 & \text{if } \sigma(\tau) \neq \mathbf{u} \text{ and } \sigma(\tau) + 1 < \lambda \\ \mathbf{u} & \text{otherwise} \end{cases}$$

$$\sigma(\tau - 1) \stackrel{def}{=} \begin{cases} \sigma(\tau) - 1 & \text{if } \sigma(\tau) \neq \mathbf{u} \text{ and } \sigma(\tau) - 1 \geq 0 \\ \mathbf{u} & \text{otherwise} \end{cases}$$

3. All predicates are unary, except binary predicates = and $<$, interpreted as:
   (a) $\mathcal{M} \models \tau = \tau'$ if $\sigma(\tau) = \sigma(\tau') \neq \mathbf{u}$.
   (b) $\mathcal{M} \models \tau < \tau'$ if $\sigma(\tau) < \sigma(\tau')$ and both $\sigma(\tau) \neq \mathbf{u}$ and $\sigma(\tau') \neq \mathbf{u}$.

Note that the interpretation for equality requires now that both terms are different from $\mathbf{u}$. We define the abbreviation $x \leq y$ as $x < y \vee x = y$. Given these limitations, we can simply represent an MHT($<$) interpretation as $\mathcal{M} = \langle \lambda, H, T \rangle$. Moreover, it is easy to see that we can establish a one-to-one mapping between the latter and an HT-trace $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ with $\lambda = |\mathbf{M}|$ so that $H = \{a(i) \mid a \in H_i, i \in [0..\lambda)\}$ and $T = \{a(i) \mid a \in T_i, i \in [0..\lambda)\}$. When this happens, we say that $\mathbf{M}$ and $\mathcal{M}$ are *corresponding* interpretations.

Example 1
The HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ with $\mathbf{H} = \{a\} \cdot \emptyset \cdot \{b\}$ and $\mathbf{T} = \{a, b\} \cdot \{a\} \cdot \{b\}$ corresponds to the MHT($<$) interpretation $\langle 3, H, T \rangle$ where $H = \{a(0), b(2)\}$ and $T = \{a(0), b(0), a(1), b(2)\}$.

We proceed to adapt now Kamp's translation to our setting in the following way.

Definition 5 (*Kamp's translation*)
Let φ be a temporal formula over $\mathcal{A}$. Kamp's translation of φ for some time point $k \in \mathbb{N}$, denoted by $[\varphi]_k$, is defined as follows:

$$
\begin{array}{rcl}
[\bot]_k & \stackrel{def}{=} & \bot \\
[a]_k & \stackrel{def}{=} & a(k), \text{ with } a \in \mathcal{A} \\
[\alpha \otimes \beta]_k & \stackrel{def}{=} & [\alpha]_k \otimes [\beta]_k \quad \text{for any connective } \otimes \in \{\wedge, \vee, \rightarrow\} \\
[\circ \alpha]_k & \stackrel{def}{=} & \exists x \ \big(x = k+1 \wedge [\alpha]_x\big) \\
[\alpha \ \mathbf{U} \ \beta]_k & \stackrel{def}{=} & \exists x \ \big(k \leq x \wedge [\beta]_x \wedge \forall y \ (k \leq y \wedge y < x \rightarrow [\alpha]_y)\big) \\
[\alpha \ \mathbf{R} \ \beta]_k & \stackrel{def}{=} & \forall x \ \big(k \leq x \rightarrow [\beta]_x \vee \exists y \ (k \leq y \wedge y < x \wedge [\alpha]_y)\big) \\
[\alpha \ \mathbb{W} \ \beta]_k & \stackrel{def}{=} & \forall x \ \big(k \leq x \wedge \forall y \ (k \leq y \wedge y < x \rightarrow [\beta]_y) \rightarrow [\alpha]_x\big) \\
[\bullet \alpha]_k & \stackrel{def}{=} & \exists x \ \big(x = k-1 \wedge [\alpha]_x\big) \\
[\alpha \ \mathbf{S} \ \beta]_k & \stackrel{def}{=} & \exists x \ \big(x \leq k \wedge [\beta]_x \wedge \forall y \ (x < y \wedge y \leq k \rightarrow [\alpha]_y)\big) \\
[\alpha \ \mathbf{T} \ \beta]_k & \stackrel{def}{=} & \forall x \ \big(x \leq k \rightarrow [\beta]_x \vee \exists y \ (x < y \wedge y \leq k \wedge [\alpha]_y)\big)
\end{array}
$$

⊠

We now prove that, when considering the model correspondence between MHT($<$) and THT, Kamp's translation is sound.

Theorem 1
Let φ be a temporal formula over $\mathcal{A}$, $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ a THT-interpretation over $\mathcal{A}$ and $\mathcal{M} = \langle (D, \sigma), H, T \rangle$ its corresponding MHT($<$)-interpretation.
Then, $\mathbf{M}, k \models \varphi$ in THT iff $\mathcal{M} \models [\varphi]_k$ in MHT($<$). ⊠

Corollary 1
A total THT-interpretation $\mathbf{M} = \langle \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of a temporal formula φ iff its corresponding MHT($<$)-interpretation $\mathcal{M} = \langle |\mathbf{T}|, T, T \rangle$ is an equilibrium model of $[\varphi]_0$. ⊠

The translation of derived operators can be simplified in MHT($<$) as follows:

$$
\begin{array}{rcl}
[\mathbf{I}]_k & \equiv & \neg \exists x \ (x = k-1) \quad \equiv \quad k = 0 \\
[\widehat{\bullet} \alpha]_k & \equiv & \forall x \ (x = k-1 \rightarrow [\alpha]_x) \quad \equiv \quad k = 0 \ \vee \ [\alpha]_{k-1} \\
[\blacklozenge \alpha]_k & \equiv & \exists x \ (x \leq k \wedge [\alpha]_x) \\
[\blacksquare \alpha]_k & \equiv & \forall x \ (x \leq k \rightarrow [\alpha]_x) \\
[\mathbf{F}]_k & \equiv & \neg \exists x \ (x = k+1) \\
[\widehat{\circ} \alpha]_k & \equiv & \forall x \ (x = k+1 \rightarrow [\alpha]_x) \\
[\Diamond \alpha]_k & \equiv & \exists x \ (k \leq x \wedge [\alpha]_x) \\
[\Box \alpha]_k & \equiv & \forall x \ (k \leq x \rightarrow [\alpha]_x)
\end{array}
$$

As an example, the translation of formula (2), viz. $\Box(\neg a \rightarrow \circ a)$, for $k = 0$ amounts to:

$$
\begin{array}{rl}
& \forall x \ (\ 0 \leq x \rightarrow (\neg a(x) \rightarrow \exists y \ (y = x+1 \wedge a(y))) \ ) \\
\equiv & \forall x \ (\ \neg a(x) \rightarrow \exists y \ (y = x+1 \wedge a(y)) \ )
\end{array}
$$

since in MHT(<), $x \geq 0$ for any x. For infinite traces, the existence of some $y = x + 1$ is always guaranteed, and the formula above can be further simplified into:

$$\forall x\ (\ \neg a(x) \rightarrow a(x+1)\ )$$

which is just a first-order logic representation of the ASP rule[3] "`a(X+1) :- not a(X).`" Similarly, it is not difficult to check that the translation of (1) amounts to:

$$\begin{aligned} & \forall x\ (0 \leq x \wedge \exists y\ (y = x - 1 \wedge \mathit{loaded}(y)) \wedge \neg \mathit{unloaded}(x) \rightarrow \mathit{loaded}(x)) \\ \equiv\ & \forall x\ (0 < x \wedge \mathit{loaded}(x-1) \wedge \neg \mathit{unloaded}(x) \rightarrow \mathit{loaded}(x)) \end{aligned}$$

that corresponds to the ASP rule:

```
loaded(X) :- loaded(X-1), not unloaded(X), X>0.
```

3 Foundations of Temporal Here-and-There

In this section, we explore some of the fundamental properties of THT, the monotonic basis of TEL. The importance of THT with respect to TEL is analogous to the relevance of HT for Equilibrium Logic and ASP. In particular, THT is a suitable framework to study the TEL-equivalence of two alternative representations. In what follows, we prove that THT-equivalence is a necessary and sufficient condition for *strong equivalence*, we provide several interesting equivalences in THT and we also present an alternative three-valued characterization of this logic. Besides, we explain how THT can be translated to LTL adding auxiliary atoms, something that allows reducing the strong equivalence problem to LTL-satisfiability checking. The section concludes with some results for THT for infinite traces, including properties about inter-definability of operators and an axiomatization.

3.1 Strong Equivalence

As happens in ASP, given that TEL is a non-monotonic formalism, it may be the case that two different temporal formulas α and β share the same temporal equilibrium models but behave differently when a common context γ is added. For this reason, it is usual to consider the notion of *strong equivalence* instead. Two temporal formulas α, β are *strongly equivalent* iff $\mathrm{TEL}(\alpha \wedge \gamma) = \mathrm{TEL}(\beta \wedge \gamma)$ for any arbitrary temporal formula γ. As expected, the THT-equivalence of $\alpha \equiv \beta$ is a sufficient condition for strong equivalence, since temporal equilibrium models are the result of a selection among THT-models.

Theorem 2 (*Aguado et al. 2013; Cabalar et al. 2018*)
If two temporal formulas α and β satisfy $\alpha \equiv \beta$ then they are strongly equivalent. ⊠

The interest of THT is that equivalence in that logic is also a *necessary* condition for strong equivalence, as we prove next.

[3] Although variable `X` is unsafe, in a practical implementation, we would add a domain predicate `time(X)` to specify that `X` is a time point.

Lemma 1
Let α and β be two LTL-equivalent formulas and let γ be the theory containing a formula $\beta \to \Box(a \vee \neg a)$ for each atom $a \in \mathcal{A}$.

Then, the following conditions are equivalent:

1. There exists some $\mathbf{H} < \mathbf{T}$ such that $\langle \mathbf{H}, \mathbf{T} \rangle \not\models \alpha \to \beta$;
2. $\mathbf{T}$ is a TS-model of $\{\beta\} \cup \gamma$ but not a TS-model of $\{\alpha\} \cup \gamma$. ⊠

Theorem 3
If two temporal formulas α and β are strongly equivalent then $\alpha \equiv \beta$. ⊠

Proof
We prove that if $\alpha \not\equiv \beta$ then there is some context theory γ for which $\{\alpha\} \cup \gamma$ and $\{\beta\} \cup \gamma$ have different TS-models. Assume first that α and β have different total models, i.e., different LTL-models. Then, take γ as the set of excluded middle axioms (EM) for every $a \in \mathcal{A}$. The LTL-models of $\{\alpha\} \cup \gamma$ and $\{\beta\} \cup \gamma$ also differ (since γ is a set of LTL tautologies). But by Proposition 1, LTL-models of these theories are exactly their TS-models, and so, they also differ.

Suppose now that α and β are LTL-equivalent but still, $\alpha \not\equiv \beta$. Then, there is some THT-countermodel $\langle \mathbf{H}, \mathbf{T} \rangle$ of either $(\alpha \to \beta)$ or $(\beta \to \alpha)$, and given LTL-equivalence of α and β, the countermodel is non-total, $\mathbf{H} < \mathbf{T}$. Without loss of generality, assume $\langle \mathbf{H}, \mathbf{T} \rangle \not\models \alpha \to \beta$. By Lemma 1, taking the theory γ consisting of an implication $\beta \to \Box(a \vee \neg a)$ per each atom $a \in \mathcal{A}$, we get that $\mathbf{T}$ is TS-model of $\{\beta\} \cup \gamma$ but not TS-model of $\{\alpha\} \cup \gamma$. □

As a consequence of Theorems 2 and 3, we obtain the following characterization:

Theorem 4
Two temporal formulas α and β are strongly equivalent iff $\alpha \equiv \beta$. ⊠

3.2 Some interesting properties of THT

THT-equivalences have a crucial role for deciding how to rewrite a formula without caring about the possible context in which it is included. We analyze next several useful THT-equivalences. The following are some De Morgan laws satisfied by negation and other operators:

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi \quad (3)$$
$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi \quad (4)$$
$$\neg(\varphi \,\mathbf{U}\, \psi) \equiv \neg\varphi \,\mathbf{R}\, \neg\psi \quad (5)$$
$$\neg(\varphi \,\mathbf{R}\, \psi) \equiv \neg\varphi \,\mathbf{U}\, \neg\psi \quad (6)$$
$$\neg(\circ\varphi) \equiv \widehat{\circ}\neg\varphi \quad (7)$$
$$\neg(\widehat{\circ}\varphi) \equiv \circ\neg\varphi \quad (8)$$

and the same happens for the past-version of the temporal operators above.

In the infinite case, we also have:

$$\circ(\varphi \oplus \psi) \equiv \circ\varphi \oplus \circ\psi \quad (9)$$
$$\circ \otimes \varphi \equiv \otimes\circ\varphi \quad (10)$$

for any formula φ and ψ and any binary connective $\oplus$ and any unary connective $\otimes$.

Also, THT satisfies *persistence*, a characteristic property of HT-based logics.

Proposition 6 (*Persistence; Aguado et al. 2013; Cabalar et al. 2018*)
Let $\langle \mathbf{H}, \mathbf{T} \rangle$ be an HT-trace of length λ and φ be a temporal formula.

Then, for any $k \in [0..\lambda)$, if $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ then $\langle \mathbf{T}, \mathbf{T} \rangle, k \models \varphi$ (or, if preferred, $\mathbf{T}, k \models \varphi$). ⊠

As a corollary, we have that $\langle \mathbf{H}, \mathbf{T} \rangle \models \neg\varphi$ iff $\mathbf{T} \not\models \varphi$ in LTL. All THT tautologies are LTL tautologies but not vice versa. However, they coincide for some types of equivalences, as we show next.

Proposition 7 (*Aguado et al. 2013; Cabalar et al. 2018*)
Let φ and ψ be temporal formulas without implications (and so, without negations either).

Then, $\varphi \equiv \psi$ in LTL iff $\varphi \equiv \psi$ in THT. ⊠

As an example, the usual inductive definition of the until operator from LTL

$$\varphi \,\mathbf{U}\, \psi \equiv \psi \vee (\varphi \wedge \circ(\varphi \,\mathbf{U}\, \psi)) \tag{11}$$

is also valid in THT due to Proposition 7. In fact, by De Morgan laws, LTL satisfies a kind of duality guaranteeing, for instance, that (11) iff:

$$\varphi \,\mathbf{R}\, \psi \equiv \psi \wedge (\varphi \vee \widehat{\circ}(\varphi \,\mathbf{R}\, \psi)) \tag{12}$$

and, by Proposition 7 again, this is also a valid equivalence in THT. If we define all the pairs of dual connectives as follows: $\wedge/\vee$, $\top/\bot$, $\mathbf{U}/\mathbf{R}$, $\circ/\widehat{\circ}$, $\Box/\Diamond$, $\mathbf{S}/\mathbf{T}$, $\bullet/\widehat{\bullet}$, $\blacksquare/\blacklozenge$, we can extend this to any formula φ without implications and define $\delta(\varphi)$ as the result of replacing each connective by its dual operator. Then, we get the following corollary of Proposition 7.

Corollary 2 (*Boolean Duality; Cabalar et al. 2018*)
Let φ and ψ be formulas without implication.

Then, THT satisfies: $\varphi \equiv \psi$ iff $\delta(\varphi) \equiv \delta(\psi)$. ⊠

In a similar manner, the temporal symmetry in the system can be exploited in order to switch the temporal direction of operators to conclude, for instance, that (11) iff $\varphi \,\mathbf{S}\, \psi \equiv \psi \vee (\varphi \wedge \bullet(\varphi \,\mathbf{S}\, \psi))$. However, this duality has some obvious limitations when we allow for infinite traces. For instance, the past has a beginning $\blacklozenge\mathbf{I} \equiv \top$ but the future may have no end $\Diamond\mathbf{F} \not\equiv \top$. If we restrict ourselves to finite traces, we get the following result.

To this end, let $\mathbf{U}/\mathbf{S}$, $\mathbf{R}/\mathbf{T}$, $\circ/\bullet$, $\widehat{\circ}/\widehat{\bullet}$, $\Box/\blacksquare$, and $\Diamond/\blacklozenge$ denote all pairs of swapped-time connectives and let $\sigma(\varphi)$ denote the replacement in φ of each connective by its swapped-time version.

Lemma 2 (*Cabalar et al. 2018*)
There exists a mapping ϱ on finite HT-traces of the same length $\lambda \geq 0$ such that for any $k \in [0..\lambda)$, $\mathbf{M}, k \models \varphi$ iff $\varrho(\mathbf{M}), \lambda{-}1{-}k \models \sigma(\varphi)$. ⊠

Theorem 5 (*Temporal Duality Theorem; Cabalar et al. 2018*)
A temporal formula φ is a THT$_f$-tautology iff $\sigma(\varphi)$ is a THT$_f$-tautology. ⊠

3.3 Three-valued Characterization of THT

As mentioned in Section 2, HT-traces can be seen as three-valued models where each atom can be false, assumed or proven (in a given state). This stems from the fact that the intermediate logic of HT actually corresponds to Gödel's three-valued logic G_3 (Gödel 1932). In particular, the HT-satisfaction of formulas can be replaced by a three-valued function that assigns, to each formula, a value 0, 1, or 2, standing for "false" (it does not hold in T), "assumed" (it holds in T but not in H) and "proven" (it holds in H, and so, in T, too), respectively. The interest of a multi-valued truth assignment $\boldsymbol{m}(k,\alpha)$ of a formula α at time point k is that THT-equivalence $\alpha \equiv \beta$ can be reduced to a comparison of truth values such as $\boldsymbol{m}(k,\alpha) = \boldsymbol{m}(k,\beta)$. This has an important application when introducing an auxiliary atom a to represent α so that, we can safely replace α by a provided that we add formulas to guarantee $\boldsymbol{m}(k,a) = \boldsymbol{m}(k,\alpha)$. Following these ideas, we proceed with the following formal definitions.

Given an HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ of length λ, we define its associated truth valuation as a function $\boldsymbol{m}(k,\varphi)$ that assigns a truth value in $\{0,1,2\}$ to formula φ at time point $k \in [0..\lambda)$ according to the following rules:

$$
\begin{aligned}
\boldsymbol{m}(k,\bot) &\stackrel{def}{=} 0\\
\boldsymbol{m}(k,a) &\stackrel{def}{=} \begin{cases} 0 & \text{if } a \notin T_k \\ 1 & \text{if } a \in T_k \setminus H_k \\ 2 & \text{if } a \in H_k \end{cases} \qquad \text{for any atom } a\\
\boldsymbol{m}(k,\varphi \wedge \psi) &\stackrel{def}{=} \min(\boldsymbol{m}(k,\varphi), \boldsymbol{m}(k,\psi))\\
\boldsymbol{m}(k,\varphi \vee \psi) &\stackrel{def}{=} \max(\boldsymbol{m}(k,\varphi), \boldsymbol{m}(k,\psi))\\
\boldsymbol{m}(k,\varphi \to \psi) &\stackrel{def}{=} \begin{cases} 2 & \text{if } \boldsymbol{m}(k,\varphi) \leq \boldsymbol{m}(k,\psi) \\ \boldsymbol{m}(k,\psi) & \text{otherwise} \end{cases}\\
\boldsymbol{m}(k,\bullet\varphi) &\stackrel{def}{=} \begin{cases} 0 & \text{if } k = 0 \\ \boldsymbol{m}(k-1,\varphi) & \text{if } k > 0 \end{cases}\\
\boldsymbol{m}(k,\varphi \,\mathbf{S}\, \psi) &\stackrel{def}{=} \max\{\min(\boldsymbol{m}(j,\psi), \min\{\boldsymbol{m}(i,\varphi) \mid j < i \leq k\}) \mid 0 \leq j \leq k\}\\
\boldsymbol{m}(k,\varphi \,\mathbf{T}\, \psi) &\stackrel{def}{=} \min\{\max(\boldsymbol{m}(j,\psi), \max\{\boldsymbol{m}(i,\varphi) \mid j < i \leq k\}) \mid 0 \leq j \leq k\}\\
\boldsymbol{m}(k,\circ\varphi) &\stackrel{def}{=} \begin{cases} 0 & \text{if } k+1 = \lambda \ (\neq \omega) \\ \boldsymbol{m}(k+1,\varphi) & \text{if } k+1 < \lambda \end{cases}\\
\boldsymbol{m}(k,\varphi \,\mathbf{U}\, \psi) &\stackrel{def}{=} \max\{\min(\boldsymbol{m}(j,\psi), \min\{\boldsymbol{m}(i,\varphi) \mid k \leq i < j\}) \mid k \leq j < \lambda\}\\
\boldsymbol{m}(k,\varphi \,\mathbf{R}\, \psi) &\stackrel{def}{=} \min\{\max(\boldsymbol{m}(j,\psi), \max\{\boldsymbol{m}(i,\varphi) \mid k \leq i < j\}) \mid k \leq j < \lambda\}
\end{aligned}
$$

The valuation of derived operators can be easily concluded from their definitions:

$$
\begin{array}{rcl}
\boldsymbol{m}(k,\top) & = & 2\\
\boldsymbol{m}(k,\neg\varphi) & \stackrel{def}{=} & \begin{cases} 2 & \text{if } \boldsymbol{m}(k,\varphi)=0\\ 0 & \text{otherwise}\end{cases}\\
\boldsymbol{m}(k,\mathsf{I}) & \stackrel{def}{=} & \begin{cases} 2 & \text{if } k=0\\ 0 & \text{if } k>0\end{cases}\\
\boldsymbol{m}(k,\widehat{\bullet}\varphi) & \stackrel{def}{=} & \begin{cases} 2 & \text{if } k=0\\ \boldsymbol{m}(k-1,\varphi) & \text{if } k>0\end{cases}\\
\boldsymbol{m}(k,\blacksquare\varphi) & = & \min\{\boldsymbol{m}(i,\varphi) \mid 0\leq i\leq k\}\\
\boldsymbol{m}(k,\blacklozenge\varphi) & = & \max\{\boldsymbol{m}(i,\varphi) \mid 0\leq i\leq k\}\\
\boldsymbol{m}(k,\mathsf{F}) & \stackrel{def}{=} & \begin{cases} 2 & \text{if } k=\lambda\neq\omega\\ 0 & \text{if } k<\lambda\end{cases}\\
\boldsymbol{m}(k,\widehat{\circ}\varphi) & \stackrel{def}{=} & \begin{cases} 2 & \text{if } k+1=\lambda\ (\neq\omega)\\ \boldsymbol{m}(k+1,\varphi) & \text{if } k+1<\lambda\end{cases}\\
\boldsymbol{m}(k,\Box\varphi) & = & \min\{\boldsymbol{m}(i,\varphi) \mid k\leq i<\lambda, i\neq\omega\}\\
\boldsymbol{m}(k,\Diamond\varphi) & = & \max\{\boldsymbol{m}(i,\varphi) \mid k\leq i<\lambda, i\neq\omega\}
\end{array}
$$

For the restriction to THT_f, it suffices to impose the condition $\lambda\neq\omega$.

Proposition 8
Let $\langle\mathbf{H},\mathbf{T}\rangle$ be a HT-trace of length λ, $\boldsymbol{m}$ its associated valuation and $k\in[0..\lambda)$.
Then, we have that

- $\langle\mathbf{H},\mathbf{T}\rangle, k\models\varphi$ iff $\boldsymbol{m}(k,\varphi)=2$
- $\langle\mathbf{T},\mathbf{T}\rangle, k\models\varphi$ iff $\boldsymbol{m}(k,\varphi)\neq 0$ ⊠

As an illustration, consider the HT-trace from Example 1. This trace corresponds to the three-valued assignment $\boldsymbol{m}(0,a)=2$, $\boldsymbol{m}(0,b)=1$, $\boldsymbol{m}(1,a)=1$, $\boldsymbol{m}(1,b)=0$, $\boldsymbol{m}(2,a)=0$ and $\boldsymbol{m}(2,b)=2$. The formula $\Diamond b$ is proven at $k=0$, that is, $\boldsymbol{m}(0,\Diamond b)=2$, because $\boldsymbol{m}(0,b)=2$ and 2 is the maximum possible value. The formula $\Box b$ is false at $k=0$, $\boldsymbol{m}(0,\Box b)=0$ because $\boldsymbol{m}(1,b)=0$ and 0 is the minimum value. On the other hand, the formula $\Box(a\vee b)$ is just assumed but not proven at 0. To see why, note that the value $\boldsymbol{m}(k,a\vee b)$ is the maximum of both disjuncts and that, in our trace, this is equal to 2 for time points $k=0$ and $k=2$ but is equal to 1 for $k=1$. Since $\Box(a\vee b)$ takes the minimum of all these values for $k\geq 0$, we get $\boldsymbol{m}(0,\Box(a\vee b))=1$.

3.4 From THT *to* LTL

Propositions 1 and 2 tell us that the addition of excluded middle axioms (EM) allows for an easy encoding of LTL into THT. In fact, Proposition 5 even guarantees that the addition of these axioms makes also the non-monotonic formalism of TEL collapse into LTL. An interesting question is whether the other direction is possible, namely, whether THT can be encoded into LTL. In fact, this can be done by adapting the encoding of (non-temporal) HT into classical propositional logic, as presented, for instance, in (Pearce et al. 2001).

Given a propositional signature $\mathcal{A}$, let us define a new propositional signature in LTL by $\mathcal{A}^* = \mathcal{A} \cup \{a' \mid a \in \mathcal{A}\}$. For any temporal formula φ, we define its translation φ^* as follows:

1. $\bot^* \stackrel{def}{=} \bot$
2. $a^* \stackrel{def}{=} a'$ for any $a \in \mathcal{A}$
3. $(\odot\varphi)^* \stackrel{def}{=} \odot\varphi^*$, if $\odot \in \{\circ, \bullet\}$
4. $(\varphi \odot \psi)^* \stackrel{def}{=} \varphi^* \odot \psi^*$, when $\odot \in \{\wedge, \vee, \mathbf{U}, \mathbf{R}, \mathbf{S}, \mathbf{T}\}$
5. $(\varphi \to \psi)^* \stackrel{def}{=} (\varphi \to \psi) \wedge (\varphi^* \to \psi^*)$

We associate with any HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ of length λ the trace $\mathbf{T}^*$ in LTL defined as the sequence of sets of atoms $T_i^* = T_i \cup \{a' \mid a \in H_i\}$ for any $i \in [0..\lambda)$. Informally speaking, $\mathbf{T}^*$ considers a new primed atom a' per each $a \in \mathcal{A}$ in the original signature. In the trace, the primed atom a' represents the fact that a occurs at some point in the $\mathbf{H}$ components, whereas the original symbol a is used to represent an atom in $\mathbf{T}$. As an HT-trace satisfies $H_i \subseteq T_i$ by construction, we may have traces that do not correspond to any HT-trace, since the set of primed atoms must be a subset of the non-primed ones. To force this structural condition on traces, we can just add the axiom:

$$\Box(a' \to a) \tag{13}$$

for any atom $a \in \mathcal{A}$. By including this axiom, we obtain a one-to-one correspondence between HT-traces and traces for the extended signature. In particular, if we take any arbitrary trace $\mathbf{T}^*$ for signature $\mathcal{A}^*$ satisfying (13), we can now define its corresponding HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ as $T_i \stackrel{def}{=} T_i^* \cap \mathcal{A}$ and $H_i \stackrel{def}{=} \{a \mid a' \in \mathbf{T}_i^*\}$.

Theorem 6
Let $\langle \mathbf{H}, \mathbf{T} \rangle$ be an HT-trace of length λ, $k \in [0..\lambda)$ and φ a temporal formula.
Then, $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ iff $\mathbf{T}^*, k \models \varphi^* \wedge (13)$ in LTL. ⊠

Example 2
Take the HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ from Example 1. Its corresponding trace is $\mathbf{T}^* = \{a, b, a'\} \cdot \{a\} \cdot \{b, b'\}$. Axiom (13) in this case is the formula:

$$\Box(a' \to a) \wedge \Box(b' \to b) \tag{14}$$

Now, for instance, given the formula:

$$\Box(\neg b \to a) \tag{15}$$

its translation $(15)^*$ corresponds to:

$$\begin{aligned} & \Box\big((\neg b \to a) \wedge ((\neg b)^* \to a^*)\big) \\ = \; & \Box\big((\neg b \to a) \wedge (\neg b \wedge \neg b' \to a')\big) \\ \equiv \; & \Box(\neg b \to a) \wedge \Box(\neg b \to a') \end{aligned} \tag{16}$$

It is easy to see that $\mathbf{T}^*$ satisfies $(15)^* \wedge (14)$ in LTL.

To conclude this comparison to LTL, we include a brief comment on the complexity of THT. As shown above, LTL-satisfaction of a temporal formula φ can be reduced to THT satisfaction with the simple addition of (EM) axioms. On the other hand, reducing THT-satisfaction of φ to LTL-satisfaction consists of adding axiom (13) and applying translation φ^*, which can be easily proved to be quadratic in the worse case. As a result, LTL and THT satisfaction share the same complexity, regardless of the trace length under consideration. In particular, it is well-known that LTL_ω-satisfiability is PSPACE-complete (Sistla and Clarke 1985). Also, LTL_f was also proved to be PSPACE-complete in (De Giacomo and Vardi 2013). As a result, both THT_ω- and THT_f-satisfiability have the same complexity, too.

3.5 Automata-based Checking of Strong Equivalence

The translation from THT to LTL can be used to reduce strong equivalence of temporal theories to LTL-satisfiability checking. This feature was exploited by the tool `abstem` (Cabalar and Diéguez 2014) that provides us with several functionalities for temporal theories under TEL_ω semantics.

For checking whether two temporal formulas φ and ψ are LTL-equivalent, `abstem` checks the validity of $\varphi \leftrightarrow \psi$, which is reduced to the satisfiability of $\neg(\varphi \leftrightarrow \psi)$. Such satisfiability is performed by translating $\neg(\varphi \leftrightarrow \psi)$ into a Büchi automaton $\mathfrak{A}$ by means of the LTL-model checker `SPoT` (Duret-Lutz et al. 2016). As explained in Section 4, the language accepted by $\mathfrak{A}$ corresponds to the LTL-models of $\neg(\varphi \leftrightarrow \psi)$. Therefore, if the accepting language of $\mathfrak{A}$ is empty it means that $\neg(\varphi \leftrightarrow \psi)$ is not satisfiable and φ and ψ are LTL-equivalent. Otherwise, any word accepted by $\mathfrak{A}$ can be seen as a counterexample of $\varphi \leftrightarrow \psi$.

In (Cabalar and Diéguez 2014), determining whether φ and ψ are strongly equivalent is reduced to checking the validity of both $\varphi \to \psi$ and $\psi \to \varphi$ in THT. Let us consider, without loss of generality, that we want to check the validity of $\varphi \to \psi$. This is equivalent to checking the satisfiability of

$$\neg\left(\Box\left(\bigwedge_{a\in\mathcal{A}} a' \to a\right) \to (\varphi \to \psi)^*\right) \tag{17}$$

in LTL and it can be done by using the procedure explained above. `abstem` obtains a Büchi automaton $\mathfrak{A}_{(17)}$ that accepts the LTL-models of (17). If (17) is LTL-satisfiable then `abstem` filters all the variables of the type a' from the accepting language of $\mathfrak{A}_{(17)}$ as explained in Section 4). The resulting automaton, denoted by $h(\mathfrak{A}_{(17)})$, captures the TEL-models of $\varphi \wedge \gamma$ which are not models of $\psi \wedge \gamma$, where

$$\gamma \stackrel{def}{=} \bigwedge_{a\in\mathcal{A}} \psi \to \Box(a \vee \neg a)$$

for every atom a in the signature (assuming a finite alphabet).

With minor modifications, this technique could be adapted to the TEL_f case. The only required change is the target automata: for TEL_ω we need to use Büchi automata since we are dealing with infinite computations while, in the finite case, we need to use non deterministic finite automata on words. The rest of the algorithm would remain the same.

3.6 Definability of Temporal Operators in THT

If we consider intuitionistic propositional logic (van Dalen 1986), propositional connectives are not interdefinable. However, when it comes to propositional HT, some definability results can be obtained. For instance, the HT-valid formula (Lukasiewicz 1941)

$$p \vee q \leftrightarrow ((p \to q) \to q) \wedge ((q \to p) \to p)\,.$$

allows us to determine that disjunction in HT can be defined in terms of the remaining propositional connectives. Unfortunately, this is the only definable operator (Balbiani and Diéguez 2016; Aguado et al. 2015). When considering QHT, the equivalence

$$\exists x\; p(x) \leftrightarrow (\forall x\; \forall y\;\; (p(x) \to p(y)) \to p(y))$$

is valid, so existential quantifiers can be reformulated in terms of the remaining connectives as well (Mints 2010). Since THT can be seen as a syntactic subclass of QHT, a similar result could be expected. Balbiani et al. (Balbiani et al. 2020) have proved that the connectives $\Diamond p$ can be defined in terms of a $\mathbf{U}$-free formula, as stated in the following lemma.

Lemma 3 (*Balbiani et al. 2020*)
The formulas $\Diamond p$ and

$$(\Box(p \to \Box(p \vee \neg p)) \wedge \Box(\circ\Box(p \vee \neg p) \to p \vee \neg p \vee \circ\Box\neg p)) \to (\Box(p \vee \neg p) \wedge \neg\Box\neg p) \quad (18)$$

are equivalent in THT.

Broadly speaking the equivalence of Lemma 3 captures the three different ways of satisfying $\Diamond p$ on any THT-model $\langle \mathbf{H}, \mathbf{T} \rangle$ at $i \geq 0$: 1) $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \Box\,(p \vee \neg p)$ holds, which in this case $\langle \mathbf{H}, \mathbf{T} \rangle$ behaves classically after i; at least for formulas whose only variable is p. In this case $\Diamond p$ would behave as $\neg\Box\neg p$. 2) If $\langle \mathbf{H}, \mathbf{T} \rangle, i \not\models \Box\,(p \vee \neg p)$, then $\langle \mathbf{H}, \mathbf{T} \rangle$ does not behave classically after i. Therefore, for some $j \geq i$, $\langle \mathbf{H}, \mathbf{T} \rangle, i \not\models p \vee \neg p$. In order to make the right part of the equivalence true, either $\langle \mathbf{H}, \mathbf{T} \rangle, i \not\models \Box\,(p \to \Box\,(p \vee \neg p))$ or $\langle \mathbf{H}, \mathbf{T} \rangle, i \not\models \Box(\circ\Box(p \vee \neg p) \to p \vee \neg p \vee \circ\Box\neg p))$ holds. The former formula fails when there is $k \geq i$ such that $\langle \mathbf{H}, \mathbf{T} \rangle, j \models p$ (so $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \Diamond p$) and $\langle \mathbf{H}, \mathbf{T} \rangle$ does not behave classically after k. 3) Similarly, $\Box(\circ\Box(p \vee \neg p) \to p \vee \neg p \vee \circ\Box\neg p))$ fails exactly where there is $k \geq i$ satisfying p but $\langle \mathbf{H}, \mathbf{T} \rangle$ behaves classically after k. In other words $\langle \mathbf{H}, \mathbf{T} \rangle$ falsifies $p \vee \neg p$ only for $i < k$. In this case, $\circ\Box(p \vee \neg p) \to p \vee \neg p \vee \circ\Box\neg p$ is falsified exactly at the greatest such k. From this equivalence the following corollary follows

Corollary 3 (*Balbiani et al. 2020*)
$p \,\mathbf{U}\, q$ is $\mathbf{U}$-free definable by using the equivalence $q \,\mathbf{U}\, p \leftrightarrow (p \,\mathbf{R}\, (q \vee p)) \wedge$ (18).

Balbiani et al. (Balbiani et al. 2020) also give a negative answer for the definability of $\Box p$ or $p \,\mathbf{R}\, q$ as a $\mathbf{R}$-free formula. To do so, they consider the following THT model $\langle \mathbf{H}, \mathbf{T} \rangle$ defined as 1) $H_0 = \emptyset$; 2) $T_0 = \{p\}$; 3) $H_i = T_i = \{p\}$ if $i > 0$. It is evident that $\langle \mathbf{H}, \mathbf{T} \rangle, 0 \not\models \Box p$ and $\langle \mathbf{T}, \mathbf{T} \rangle, 0 \models \Box p$. The following lemma shows that $\langle \mathbf{H}, \mathbf{T} \rangle$ and $\langle \mathbf{T}, \mathbf{T} \rangle$ satisfy the same $\mathbf{R}$-free formulas at 0.

Lemma 4 (*Balbiani et al. 2020*)
Let us consider the THT model defined above. For all $\mathbf{R}$-free formula φ it follows that

$$\langle \mathbf{H}, \mathbf{T} \rangle, 0 \models \varphi \text{ iff } \langle \mathbf{T}, \mathbf{T} \rangle, 0 \models \varphi.$$

As a consequence, if $\Box p$ were definable in terms of a $\mathbf{R}$-free formula ψ, this would mean that $\langle \mathbf{H}, \mathbf{T} \rangle, 0 \not\models \psi$ while $\langle \mathbf{T}, \mathbf{T} \rangle, 0 \models \psi$, which contradicts the aforementioned lemma. Lemma 4 is proved by showing that $\langle \mathbf{H}, \mathbf{T} \rangle$ and $\langle \mathbf{T}, \mathbf{T} \rangle$ are *bisimilar* at time 0 with respect to the temporal language excluding the $\mathbf{R}$ connective. This means both models satisfy the same $\mathbf{R}$-free formulas.

Although there is no formal result, we conjecture that the equivalence of Definition 3 and Lemma 4 also hold for finite traces. This would allow us to provide positive and negative definability results for THT_f.

Another interesting observation regarding expressiveness of operators is the presence of the newly introduced temporal operator $\varphi \mathbb{W} \psi$ read as "repeat φ while ψ." This operator is not usual in LTL because, in that logic, it can be easily defined in terms of $\mathbf{U}$ or $\mathbf{R}$:

Proposition 9
In LTL, we have the following equivalences:

$$\begin{aligned} \alpha \ \mathbb{W} \ \beta &\equiv \neg(\beta \ \mathbf{U} \ \neg\alpha) \equiv \neg\beta \ \mathbf{R} \ \alpha \\ \alpha \ \mathbf{U} \ \beta &\equiv \neg\neg(\alpha \ \mathbf{U} \ \neg\neg\beta) \\ &\equiv \neg(\neg\beta \ \mathbb{W} \ \alpha) \qquad \Box \end{aligned}$$

⊠

In THT, however, these equivalences do not hold. Unlike $\mathbf{U}$ and $\mathbf{R}$, operator $\mathbb{W}$ has an implicational nature. As a consequence, there is no clear way to represent $\mathbb{W}$ in terms of the other operators[4]. The following equivalences describe these three operators, $\mathbf{U}$, $\mathbf{R}$ and $\mathbb{W}$, by an inductive unfolding.

Proposition 10
In THT we have the following equivalences:

$$\begin{aligned} \alpha \, \mathbf{U} \, \beta &\equiv \beta \vee (\alpha \wedge \circ(\alpha \, \mathbf{U} \, \beta)) \\ \alpha \, \mathbf{R} \, \beta &\equiv \beta \wedge (\alpha \vee \circ(\alpha \, \mathbf{R} \, \beta)) \\ \alpha \, \mathbb{W} \, \beta &\equiv \alpha \wedge (\beta \rightarrow \circ(\alpha \, \mathbb{W} \, \beta)) \end{aligned}$$

⊠

The first two equivalences are well-known from LTL and also preserved in THT. The third equivalence shows that the $\mathbb{W}$ operator is formed by a repeated application of implications. This is even clearer in the following expansion:

Proposition 11
The formula $(\alpha \, \mathbb{W} \, \beta)$ is THT equivalent to the (possibly infinite) conjunction of rules:

$$\left(\bigwedge_{j=0}^{i-1} \circ^j \beta \right) \quad \rightarrow \quad \circ^i \alpha$$

for all $i \geq 0$. ⊠

[4] We conjecture that this may be, in fact, a fundamental operator in THT.

To see how this expression works, think about the formula $(w \mathbb{W} f)$ where w means "pouring water" and f means "fire." Notice that, for $i = 0$, the rule body becomes an empty conjunction $(\top)$ and so this expands to $\top \rightarrow \circ^0 w$ which is just equivalent to fact w. For instance, for $i \in [0, 3]$ we get the rules:

$$\begin{array}{rcl} & & w \\ f & \rightarrow & \circ w \\ f \wedge \circ f & \rightarrow & \circ^2 w \\ f \wedge \circ f \wedge \circ^2 f & \rightarrow & \circ^3 w \\ & \vdots & \end{array}$$

That is, we start pouring water at situation 0 and, if we have fire, we keep pouring water at 1 and check fire again, and so on. As we can see, the effect of each f test is placed at the next situation. Thus, the reading of $(w \mathbb{W} f)$ is like a procedural program "**do** pour-water **while** fire". If we want to move the test to the same situation of its effect, we would write instead $\big((f \rightarrow w) \mathbb{W} f\big)$ whose reading would be ''**while** fire **do** pour-water" and whose expansion becomes:

$$\begin{array}{rcl} f & \rightarrow & w \\ f \wedge \circ f & \rightarrow & \circ w \\ f \wedge \circ f \wedge \circ^2 f & \rightarrow & \circ^2 w \\ & \vdots & \end{array}$$

The expansion of $(w \mathbb{W} f)$ as a set of rules reveals its behaviour from a logic programming point of view. For instance, a theory only containing the formula $(w \mathbb{W} f)$ has a unique temporal stable model (per each length $\lambda > 0$) in which w is only true at the initial state, whereas f is always false, since there is no additional evidence about fire. In LTL, the formula $(w \mathbb{W} f)$ is equivalent to $(\neg f \, \mathbf{R} \, w)$, that is, $\square w \vee (w \, \mathbf{U} \, (w \wedge \neg f))$. In our formalism, however, this last formula produces temporal stable models with arbitrary prefix sequences of w, and even the case in which w holds in all the states of the trace. In other words, water may be poured arbitrarily many times, even though fire is false all over the trace.

3.7 Axiomatisation of THT

The axiomatic system of THT (Balbiani and Diéguez 2016) is obtained by combining the axioms of intuitionistic modal logic K (Simpson 1994) (IK), the Hosoi Axiom (Hosoi 1966) and the LTL axiomatisation of Goldblatt (Goldblatt 1992). Such axiomatic system is described below.

Hosoi axiom: $p \vee (p \rightarrow q) \vee \neg q$;
IK **Axioms for** $\circ$ **and** $\widehat{\circ}$ (Simpson 1994):

1) $\widehat{\circ} p \leftrightarrow \circ p$;
2) $\widehat{\circ} (p \rightarrow q) \rightarrow \big(\widehat{\circ} p \rightarrow \widehat{\circ} q\big)$;
3) $\widehat{\circ} (p \rightarrow q) \rightarrow (\circ p \rightarrow \circ q)$;
4) $\circ (p \vee q) \leftrightarrow \circ p \vee \circ q$;
5) $\big(\circ p \rightarrow \widehat{\circ} q\big) \rightarrow \widehat{\circ} (p \rightarrow q)$;
6) $\neg \circ \bot$

IK **axioms for $\Box$ and $\Diamond$**:

7) $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$;
8) $\Box(p \rightarrow q) \rightarrow (\Diamond p \rightarrow \Diamond q)$;
9) $\Diamond(p \vee q) \rightarrow \Diamond p \vee \Diamond q$;
10) $(\Diamond p \rightarrow \Box q) \rightarrow \Box(p \rightarrow q)$;
11) $\neg \Diamond \bot$

Axioms combining $\circ$, $\widehat{\circ}$, $\Box$ and $\Diamond$: 12) $\Box p \rightarrow p \wedge \widehat{\circ}\Box p$; 13) $p \vee \circ\Diamond p \rightarrow \Diamond p$;

Induction: 14) $\dfrac{p \rightarrow \widehat{\circ} p}{p \rightarrow \Box p}$; 15) $\dfrac{\circ p \rightarrow p}{\Diamond p \rightarrow p}$;

Axioms for $\mathbf{U}$ and $\mathbf{R}$:

16) $p \,\mathbf{U}\, q \rightarrow \Diamond q$
17) $p \,\mathbf{U}\, q \leftrightarrow (q \vee (p \wedge \circ\, (p \,\mathbf{U}\, q)))$;
18) $\Box q \rightarrow p \,\mathbf{R}\, q$
19) $p \,\mathbf{R}\, q \leftrightarrow \big(q \wedge \big(p \vee \widehat{\circ}\, (p \,\mathbf{R}\, q)\big)\big)$;

Modus ponens: $\dfrac{p \rightarrow q,\ p}{q}$. **Necessitation**: $\dfrac{p}{\Box p}$; $\dfrac{p}{\widehat{\circ} p}$.

Due to the intuitionistic basis of THT all the axioms need to be duplicated in order to cope with the potential non-definability of modal operators (Simpson 1994). It should be also noticed that the induction axiom is replaced by the corresponding inference rule. These inference rules can be can be derived from the axioms and vice versa. The corresponding proof of soundness and completeness is achieved by adapting Goldblatt's proof for classical LTL (Goldblatt 1992) to the here-and-there case: the soundness is proved by checking that all axioms are valid in THT and the inference rules preserve validity. The proof of completeness relies on a variations of well-known methods for proving completeness: canonical model construction for birelational models (Simpson 1994), filtration and unwinding (Balbiani and Diéguez 2016).

From the axiomatic point of view, the main difference between THT and THT_f relies on Axiom 1), which guarantees that every time instant has a successor. From the point of view of the completeness proof, this axiom makes possible the unwinding of the filtrated model in (Balbiani and Diéguez 2016). Since Axiom 1) is not valid in THT_f because it fails at the end of the trace, this unwinding may not be directly applied to obtain a sound and complete axiomatisation of THT_f.

4 From TEL to automata

The relation between traditional temporal logics and automata is well-known and has been thoroughly studied in the literature (Demri et al. 2016). For infinite traces, LTL_ω unsatisfiability can be reduced to checking the emptiness of a language with a given automaton that accepts *infinite words*. For this, we can use, for instance, the well-known translation of (Gerth et al. 1995) mapping a temporal formula φ into a *Büchi automaton* (Büchi 1962) that accepts the ω-regular language corresponding to the (infinite trace) models of φ. When we jump to finite traces, the type of automata required is simpler, as they just need to cover languages with words of finite length. Thus, in (Zhu et al. 2019; De Giacomo and Vardi 2013), translations from LTL_f into different types of automata such as *Non-deterministic finite automata* (NFA) and *Alternating Automata on Words* (AFW; Chandra et al. 1981) have been proposed.

The use of automata for deciding THT-satisfiability can naturally follow the same steps, provided that the translation from Section 3.4 allows us to reduce the problem to standard LTL-satisfiability, regardless of the trace length. This fact was used in (Cabalar and Diéguez 2014) to decide strong equivalence of two temporal formulas α and β in TEL_ω as follows. By Theorem 4, strong equivalence amounts to THT_ω-equivalence, that is, checking the validity of $\varphi := \alpha \leftrightarrow \beta$ in THT_ω. But then, by Theorem 6, this amounts to checking the satisfiability of (13) $\wedge \neg(\varphi^*)$ in LTL_ω and the latter is then done by the standard Büchi-automata construction method. Similar steps can be followed for THT_f, replacing Büchi-automata for LTL_ω by NFA or AFW for LTL_f.

The use of automata for a *non-monotonic* temporal formalism like TEL becomes a more involved process, since temporal equilibrium models are obtained by a model selection (or minimization) process. The complexity of TEL_ω, for instance, suggests that there is no efficient method to reduce TEL_ω- to LTL_ω-satisfiability.

Theorem 7 (*Bozzelli and Pearce 2015*)
The problem of deciding whether a temporal formula has some (infinite) temporal equilibrium model in TEL_ω is ExpSpace-complete. ⊠

A possible way of overcoming this difficulty is to observe that Definition 2 (of temporal equilibrium models) involves a kind of *quantification*. That is, we must find a total HT-model $\langle \mathbf{T}, \mathbf{T} \rangle$ of some formula such that *there is no smaller* $\mathbf{H} < \mathbf{T}$ such that $\langle \mathbf{H}, \mathbf{T} \rangle$ is also a model of the formula. This quantification seems to have a second-order nature, as it has to do with different configurations of the truth of propositions. In fact, in the non-temporal case, equilibrium models of a formula have been captured using Quantified Boolean Formulas (Pearce et al. 2001) and subsequently generalized by using a syntactic operator SM[·] (Ferraris et al. 2007). This gives a second-order formula that uses the translation from Section 3.4 and quantifies over primed propositions. This approach opens the natural possibility of capturing TEL through *Quantified* LTL (QLTL), an extension of LTL in which we can use second-order quantifiers over propositions, and define a temporal version of the SM[·] operator as a QLTL formula. The advantage of having a quantified temporal formula SM[φ] capturing the TS-models of φ is that there exist several methods for reducing that formula to an automaton, as we see below.

4.1 TS-*models in terms of* QLTL*: the* SM *operator*

We start by recalling the syntax and semantics of *Quantified Linear-Time Temporal Logic* (QLTL; Sistla et al. 1987; Sistla 1983; Demri et al. 2016). The syntax of QLTL extends the one of temporal formulas by allowing for the use of quantifiers on the propositional variables, that is, formulas of the type $\exists a\, \varphi$ and $\forall a\, \varphi$ are added.

Interpreting the additional formulas requires the concept of a variant trace.

Definition 6 (*X-variant trace; French and Reynolds 2002*)
Given two traces $\mathbf{T}$ and $\mathbf{T}'$ of length λ and $X \subseteq \mathcal{A}$, we say that $\mathbf{T}$' is an *X-variant trace* of $\mathbf{T}$ if $T_i \setminus X = T'_i \setminus X$ for all $i \in [0..\lambda)$.

With this, the semantics for QLTL is obtained from that of LTL by extending its satisfaction relation with the cases of quantified formulas as follows:

- $\mathbf{T}, i \models \forall a\, \varphi$ iff $\mathbf{T}', i \models \varphi$ for all $\{a\}$-variant traces $\mathbf{T}'$ of $\mathbf{T}$,
- $\mathbf{T}, i \models \exists a\, \varphi$ iff $\mathbf{T}', i \models \varphi$ for some $\{a\}$-variant traces $\mathbf{T}'$ of $\mathbf{T}$.

It is known that QLTL_ω is as expressive as Büchi automata (Sistla et al. 1987), and so, more expressive than LTL_ω. For instance, the indicative "even states" property (Wolper 1983) can be expressed by the quantified temporal formula $\exists q\ (\neg q \wedge \Box\,(q \leftrightarrow \circ\neg q)) \wedge \Box\,(q \rightarrow p)$.

Now, in order to encode TS-models of φ using a quantified temporal formulas, we use translation φ^* from Section 3.4 and define the following notation. Let $\boldsymbol{a}$ and $\boldsymbol{a'}$ stand for the tuples $\langle a_1, \ldots, a_n\rangle$ and $\langle a'_1, \ldots, a'_n\rangle$ of n atoms, respectively. We define the following expressions:

$$\boldsymbol{a'} \leq \boldsymbol{a} \stackrel{def}{=} \bigwedge_{i=1}^{n} \Box(a'_i \rightarrow a_i) \qquad\qquad \boldsymbol{a'} < \boldsymbol{a} \stackrel{def}{=} \boldsymbol{a'} \leq \boldsymbol{a} \wedge \bigvee_{i=1}^{n} \Diamond(\neg a'_i \wedge a_i).$$

Note that, whenever $\boldsymbol{a}$ coincides with the original propositional signature $\mathcal{A}$, $\boldsymbol{a'} \leq \boldsymbol{a}$ amounts to axiom (13) and, as we saw, is used to guarantee that atoms in trace $\mathbf{H}$ are also included in trace $\mathbf{T}$. Expression $\boldsymbol{a'} < \boldsymbol{a}$ further forces $\mathbf{H} < \mathbf{T}$ as stated by the following proposition.

Proposition 12
Let $\mathbf{T}^*$ be a trace over signature $\mathcal{A}^* = \mathcal{A} \cup \{a' \mid a \in \mathcal{A}\}$ and let $\mathbf{H}$ and $\mathbf{T}$ be such that $H_i \stackrel{def}{=} \{a \mid a' \in \mathbf{T}^*_i\}$ and $T_i \stackrel{def}{=} T^*_i \cap \mathcal{A}$. Also, let $\boldsymbol{a}$ be a sequence containing all propositional variables on $\mathcal{A}$ and let $\boldsymbol{a'}$ be of the same length as $\boldsymbol{a}$. Then, we have:

1. $\mathbf{T}^*, 0 \models \boldsymbol{a'} \leq \boldsymbol{a}$ iff $\mathbf{H} \leq \mathbf{T}$,
2. $\mathbf{T}^*, 0 \models \boldsymbol{a'} < \boldsymbol{a}$ iff $\mathbf{H} < \mathbf{T}$. ⊠

We can now define the temporal version of the SM operator from (Ferraris et al. 2007): Let φ be a temporal formula over signature $\mathcal{A}$ and let $\boldsymbol{a}$ be a tuple with all propositions in $\mathcal{A}$. Then, we define the QLTL expression:

$$\text{SM}[\varphi] \stackrel{def}{=} \varphi \wedge \neg\exists \boldsymbol{a'}\ \big(\boldsymbol{a'} < \boldsymbol{a} \wedge \varphi^*\big). \tag{19}$$

The fact that $\text{SM}[\varphi]$ captures the TS-models of φ is established by the following theorem.

Theorem 8
A trace $\mathbf{T}$ is a TS-model of a temporal formula φ iff $\mathbf{T}$ is a QLTL-model of $\text{SM}[\varphi]$. ⊠

Note that for finite traces this amounts to a characterisation in terms of *weak* QLTL.

Example 3 (*Example 2 continued*)
Consider again formula $\varphi = \Box(\neg b \rightarrow a)$ in (15). As mentioned, formula $\boldsymbol{a} \leq \boldsymbol{a'}$ amounts to axiom (13) for all atoms in the signature which, in this case, corresponds to formula (14). Now $\boldsymbol{a} < \boldsymbol{a'}$ is stronger, as it requires not only $\mathbf{H} \leq \mathbf{T}$ but also $\mathbf{H} \neq \mathbf{T}$. In this case, we obtain the expression:

$$(14) \wedge \big(\ \Diamond(\neg a' \wedge a) \vee \Diamond(\neg b' \wedge b)\ \big) \tag{20}$$

whereas φ^* corresponds to (16), as illustrated above. So, as a result, $\text{SM}[\varphi]$ is finally unfolded into:

$$\begin{array}{rcl}\text{SM}[\varphi] & = & \text{SM}[\Box(\neg b \rightarrow a)] \\ & = & \Box(\neg b \rightarrow a) \wedge \neg\exists\ a'\ b'\ \big(\\ & & \quad \Box(a' \rightarrow a) \wedge \Box(b' \rightarrow b) \wedge \big(\ \Diamond(\neg a' \wedge a) \vee \Diamond(\neg b' \wedge b)\ \big) \\ & & \quad \wedge\ \Box(\neg b \rightarrow a) \wedge \Box(\neg b \rightarrow a')\ \ \big)\end{array}$$

The LTL-models of $\Box(\neg b \rightarrow a)$ are traces $\mathbf{T}$ where each state satisfies $T_i \neq \emptyset$. Take any $\mathbf{T}$ where there is some T_i including atom b. Then, we can extend $\mathbf{T}$ to $\mathbf{T}^*$ for signature $\{a, b, a', b'\}$ repeating the truth of a' and b' with respect to a and b in all states, except in state T_i^* where we just leave b' false. It is not difficult to see that this extended interpretation $\mathbf{T}^*$ satisfies $\boldsymbol{a} < \boldsymbol{a'} \wedge \varphi^*$ and so, $\mathbf{T}$ cannot be a model of SM$[\varphi]$. Therefore, $\mathbf{T}$ must make b false at all states and the only remaining model for SM$[\varphi]$ is the trace $\mathbf{T}$ where $T_i = \{a\}$ for all states, which coincides with the only TS-model of φ.

4.2 From QLTL *to automata*

Once the TS-models of φ are captured by a quantified temporal formula, there are several possibilities to obtain the corresponding automaton. For instance, (Cabalar and Demri 2011) used the following approach[5] in the case of TEL$_\omega$: First, we build a Büchi automaton $\mathfrak{A}_1$ that yields the LTL$_\omega$-models of φ as usual. Then, we build a second automaton, $\mathfrak{A}_2$, corresponding to the temporal formula $\boldsymbol{a} < \boldsymbol{a'} \wedge \varphi^*$ for the extended signature $\mathcal{A}^*$. On this last construction, we perform a filter operation to obtain the new automaton $h(\mathfrak{A}_2)$ that produces the same language but removes the primed atoms from the signature. In other words, $h(\mathfrak{A}_2)$ captures the models of the quantified temporal formula $\exists \boldsymbol{a'}(\boldsymbol{a} < \boldsymbol{a'} \wedge \varphi^*)$ and, in terms of THT, this corresponds to the $\mathbf{T}$ traces for which there exists some $\mathbf{H} < \mathbf{T}$. The final step is using the operations of complement and intersection of Büchi automata to obtain $\mathfrak{A}_1 \cap \overline{h(\mathfrak{A}_2)}$ that naturally corresponds to the formula (19) of the SM operator. This method was implemented later on in the tool `abstem` (Cabalar and Diéguez 2014) that allows for both computing TS-models and checking strong equivalence of temporal formulas in TEL$_\omega$.

In what follows, we present an analogous method for TEL$_f$ yet by reducing SM formulas to automata over finite words. More precisely, we consider two different types of finite automata $\mathfrak{A}$, namely, NFAs and AFWs.

A NFA is a structure $(\Sigma, S, s_0, \delta, F)$ where:

1. Σ is the alphabet,
2. S is a set of states,
3. $S_0 \subseteq S$ is a set of initial states,
4. $\delta : \Sigma \times S \mapsto 2^S$ is a transition function and
5. $F \subseteq S$ is a set of final states.

A run of a NFA is a finite sequence of states $s_0, \cdots, s_n$ such that $s_0 \in S_0$ and $\delta(a, s_i) = s_{i+1}$ for all $0 \leq i < n$. The run is said to be accepting if $s_n \in F$. By $\mathcal{L}(\mathfrak{A})$ we denote the set of runs that are accepted by $\mathfrak{A}$. The following theorem shows the relation between LTL$_f$ and NFAs.

Theorem 9 (*Zhu et al. 2017; Camacho et al. 2018*)
A given temporal formula φ over $\mathcal{A}$ can be translated into a NFA $\mathfrak{A}_\varphi$ such that the set of LTL$_f$-models of φ corresponds to $\mathcal{L}(\mathfrak{A}_\varphi)$. ⊠

[5] In fact, in (Cabalar and Demri 2011), the formula SM$[\varphi]$ was never constructed as an intermediate step, but the description we provide here is equivalent.

Note that the alphabet of $\mathfrak{A}_\varphi$ consists of $2^{\mathcal{A}}$.

To give an example of how the corresponding algorithm works, let us consider the formula $\varphi = \Box(\neg a \to \circ a)$ in (2). In LTL$_f$, φ is equivalent to $\Box(a \vee \circ a)$, which means that every LTL$_f$-model of this formula satisfies a at a state i or $i+1$, for all $0 \leq i < \lambda$. The set of LTL$_f$-models of φ is captured by the NFA of Figure 1. We see that every run in which a is false in two consecutive states is disregarded.

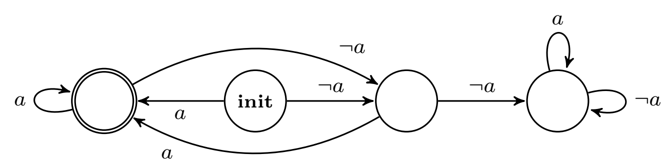

Fig. 1. NFA accepting the LTL$_f$-models of $\Box(\neg a \to \circ a)$.

On the other hand, an AFW is a structure $(\Sigma, S, s_0, \delta, F)$ where:

1. Σ is the alphabet,
2. S is a set of states,
3. $s_0 \subseteq S$ is a set of initial states,
4. $\delta : \Sigma \times S \mapsto \mathbb{B}^+(S)$ is a transition function, where $\mathbb{B}^+(S)$ is built from the elements of S, conjunction, disjunction, *true* and *false*,
5. $F \subseteq S$ is a set of final states.

Given an input word w of length n, a run of an AFW is a tree labelled by states of the AFW such that 1) the root is labelled by s_0, 2) if a node x at level i is labelled by a state q and $\delta(q, a_i) = \Theta$ then either Θ is *true* of some $P \subseteq S$ and x has a child for each element in P, 3) the run is accepting if all leaves at depth n are labeled by states in F. Thus, a branch in an accepting run has to hit the *true* transition or hit an accepting state after reading all the input word w.

The correspondence between LTL$_f$ and AFW is stated in the following theorem

Theorem 10 (*De Giacomo and Vardi 2013*)
A temporal formula φ can be translated into an AFW $\mathfrak{A}_\varphi = (2^{\mathcal{A} \cup \{Last\}}, S, \{s_0\}, \delta, \emptyset)$, where

1. *Last* corresponds a fresh atom representing the last state of the trace.
2. S corresponds to the negation-closed set of subformulas.
3. s_0 is the initial state, which corresponds to the state labeled with the formula φ,
4. F is a set of accepting states and
5. $\delta : S \times 2^{\mathcal{A} \cup \{Last\}} \mapsto \mathbb{B}^+(S)$ is the transition function defined as follows, where $X \subseteq \mathcal{A} \cup \{Last\}$ is a state
   (a) $\delta(\top, X) = \text{true}$
   (b) $\delta(\bot, X) = \text{false}$
   (c) $\delta(a, X) = \begin{cases} \text{true} & \text{if } a \in X; \\ \text{false} & \text{otherwise} \end{cases}$
   (d) $\delta(\neg\varphi, X) = \overline{\delta(\varphi, X)}$, where $\overline{\delta(\varphi, X)}$ is obtained from $\delta(\varphi, X)$ by switching $\wedge$ and $\vee$, by switching true and false and, in addition, by negating subformulas in X.

(e) $\delta(\varphi \wedge \psi, X) = \delta(\varphi, X) \wedge \delta(\psi, X)$
(f) $\delta(\varphi \vee \psi, X) = \delta(\varphi, X) \vee \delta(\psi, X)$
(g) $\delta(\varphi \rightarrow \psi, X) = \delta(\neg\varphi, X) \vee \delta(\psi, X)$
(h) $\delta(\circ\varphi, X) = \begin{cases} \varphi & \text{if } \mathit{Last} \notin X \\ \text{false} & \text{otherwise} \end{cases}$
(i) $\delta(\widehat{\circ}\varphi, X) = \begin{cases} \varphi & \text{if } \mathit{Last} \notin X \\ \text{true} & \text{otherwise} \end{cases}$
(j) $\delta(\varphi \,\mathbf{U}\, \psi, X) = \begin{cases} \delta(\psi, X) & \text{if } \mathit{Last} \in X \\ \delta(\psi, X) \vee (\delta(\varphi, X) \wedge (\varphi \,\mathbf{U}\, \psi)) & \text{otherwise} \end{cases}$
(k) $\delta(\varphi \,\mathbf{R}\, \psi, X) = \begin{cases} \delta(\psi, X) & \text{if } \mathit{Last} \in X \\ \delta(\psi, X) \wedge (\delta(\varphi, X) \vee (\varphi \,\mathbf{R}\, \psi)) & \text{otherwise} \end{cases}$ ⊠

Moreover, as in the case of NFAs, $\mathcal{L}(\mathfrak{A}_\varphi)$ corresponds to the set of LTL$_f$-models of φ.

As above, the formula $\square(\neg a \rightarrow \circ a) \equiv \square(a \vee \circ a)$ can be translated into the AFW of Figure 2.

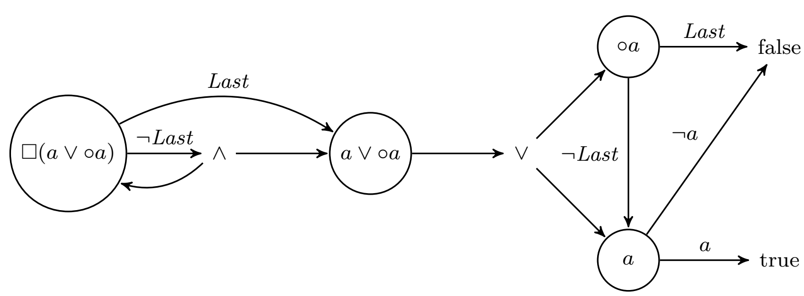

Fig. 2. AFW accepting the LTL$_f$-models of $\square(a \vee \circ a)$.

In order to provide an example of accepted and rejected inputs, let us consider Figure 3. It shows two different accepting runs of the AFW of Figure 2 with the input $\emptyset \cdot \{a\} \cdot \{a\}$. Time is represented from left to right and, for each tree, all the branches hit the true transition. Therefore, $\emptyset \cdot \{a\} \cdot \{a\}$ is accepted.

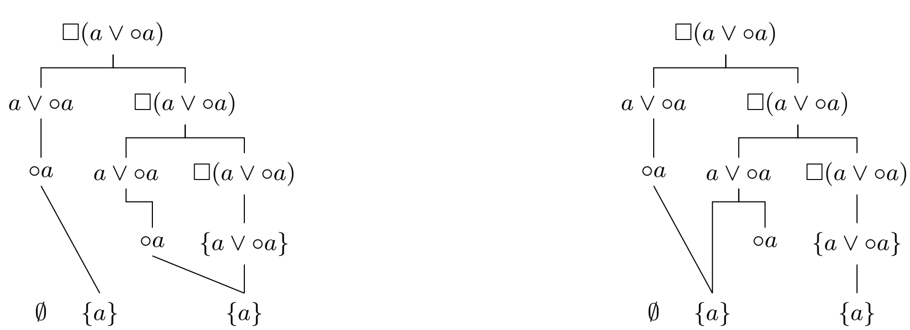

Fig. 3. Two different runs of the AFW of Figure 2 accepting the input $\emptyset \cdot \{a\} \cdot \{a\}$.

On the contrary, Figure 4 shows a run of the AFW of Figure 2 that rejects the input

$\emptyset \cdot \{a\} \cdot \emptyset$. Clearly, it is not possible that the rightmost branch hits the true transition because a is not satisfied at the end of the trace.

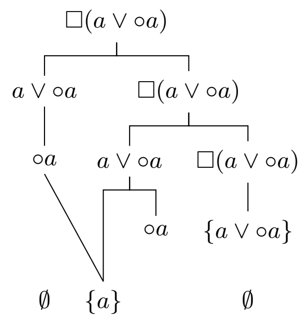

Fig. 4. A run of the AFW of Figure 2 rejecting the input $\emptyset \cdot \{a\} \cdot \emptyset$.

The rest of this section is devoted to show how Theorem 9 and 10 can be used to obtain the following result.

Theorem 11
A temporal formula φ can be translated into an AFW/NFA $\mathfrak{A}_\varphi$ such that $\mathcal{L}(\mathfrak{A}_\varphi)$ corresponds to the set of TEL$_f$-models of φ.

Our construction relies on several intermediate automata transformations corresponding to the ones in (Cabalar and Demri 2011). Each of them accepts the model of one specific part of the SM expression. The involved automata and their correspondence with $\mathrm{SM}[\varphi]$ are depicted in (21).

$$\mathrm{SM}[\varphi] = \underbrace{\underbrace{\varphi}_{\mathfrak{A}_1} \wedge \underbrace{\neg \underbrace{\exists \boldsymbol{a'} \; \underbrace{\boldsymbol{a'} < \boldsymbol{a} \wedge (\varphi)^*}_{\mathfrak{A}_2}}_{h(\mathfrak{A}_2)}}_{\overline{h(\mathfrak{A}_2)}}}_{\mathfrak{A}_1 \cap \overline{\mathfrak{A}_2}} . \tag{21}$$

The computation of the final automaton $\mathfrak{A}_1 \cap \overline{\mathfrak{A}_2}$ accepting the TEL$_f$-models of φ is done in a compositional manner, analogous to the translation of QLTL or *Weak Monadic Second Order Theory of one Successor* into NFA/AFW (Bozzelli et al. 2015; Henriksen et al. 1995). The two initial automata $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are computed by using Lemma 10 or 9, depending on whether we are interested in computing an NFA or an AFW. Note that the formulas recognized by $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are temporal formulas, which justifies the application of these lemmas. $h(\mathfrak{A}_2)$ is obtained from $\mathfrak{A}_2$ by *projecting* the atoms of the type a'. In language-theoretic terms, a *projection operation* $h(\mathcal{L}(\mathfrak{A}))$ is defined as follows.

$$h(\mathcal{L}(\mathfrak{A})) \stackrel{def}{=} \{w \mid \exists\, w' \text{ s.t. } w \text{ is identical to } w' \text{ except for all atoms of the type } a'\}.$$

Since $h(\mathcal{L}(\mathfrak{A})) = \mathcal{L}(h(\mathfrak{A}))$ (Cabalar and Demri 2011), the projection can be applied directly over $\mathfrak{A}_2$ and, broadly speaking, the resulting automaton accepts all LTL$_f$-models $\mathbf{T}$ for which there exists a trace $\mathbf{H}$ satisfying $\mathbf{H} < \mathbf{T}$. $\overline{h(\mathfrak{A}_2)}$ is obtained by complementing $h(\mathfrak{A}_2)$; it accepts all traces $\mathbf{T}$ that are either no LTL$_f$-models of φ or for which exists no trace $\mathbf{H}$ satisfying $\mathbf{H} < \mathbf{T}$. Automata complementation is the usual way to obtain automata accepting the LTL_f models of negated formulas. Finally, $\mathfrak{A}_1 \cap \overline{h(\mathfrak{A}_2)}$ is obtained by intersecting $\mathfrak{A}_1$ and $\overline{h(\mathfrak{A}_2)}$. This operation is equivalent to selecting all traces $\mathbf{T}$ being LTL_f-models of φ for which there is no trace $\mathbf{H} < \mathbf{T}$ satisfying $\langle \mathbf{H}, \mathbf{T} \rangle, 0 \models \varphi$. In this case, $\langle \mathbf{T}, \mathbf{T} \rangle$ is an equilibrium model of φ.

If we analyze $\varphi = \Box(\neg a \rightarrow \circ a)$ in TEL_f, a is false at the initial state since it cannot be provable. Since $\neg a \rightarrow \circ a$ is satisfied in the initial state, a is true at time point 1, so the rule $\neg a \rightarrow \circ a$ cannot be applied to prove a at time point 2 and the cycle starts again. Hence, TEL_f-models of φ are of the form $\emptyset \cdot \{a\} \cdot \emptyset \cdot \{a\} \cdots$. They are captured by the NFA of Figure 5, whose accepting runs correspond to sequences of the aforementioned type.

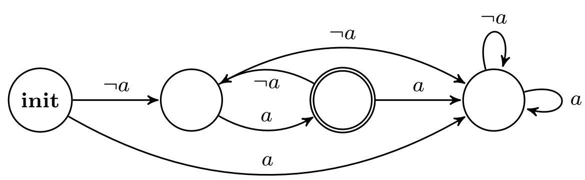

Fig. 5. NFA accepting the TEL_f-models of $\Box(\neg a \rightarrow \circ a)$.

5 Temporal Logic Programs

In the previous section, we have seen a first method to compute TS-models and check TEL-satisfiability, relying first on a quantified LTL expression, and then, deriving the construction of different types of automata: Büchi for TEL_ω and NFA or AFW for TEL_f. Automata-based techniques are interesting for the analysis of *the set of traces* (TS-models) induced by a given temporal formula. This is useful, for instance, to check several types of equivalence between two different temporal representations, or to check the existence of solution for a given planning problem. However, in many practical applications, rather than exploring all possibilities, our interest is more focused on the generation of one, or a few TS-models that encode solutions to a given temporal problem. This is, in fact, the usual model-based problem solving orientation of ASP, where stable models encode solutions to a particular problem, encoded by the rules in the logic program. In this way, if we want to solve a temporal diagnosis scenario, we are interested in finding at least one TS-model that explains the observations. Similarly, for a given planning problem, we look for some plan that reaches the goal in a finite number of steps.

In this section, we show how the generation of TS-models can be done using ASP technology by considering temporal theories with a syntax closer to logic programming. As we see next, both TEL_ω and TEL_f can be reduced to a normal form that we call *temporal logic programs*. This normal form is useful for ASP-based computation. For the finite case, there are two translations of this normal form into ASP. One allows any temporal formula and provides a translation for a given trace length. The other translation imposes some restrictions on the formula but allows for incremental solving. There are tools supporting these translations, most notably `telingo` (Cabalar et al. 2019) for the incremental approach.

We provide a generic normal form for temporal programs and a translation for temporal formulas into this normal form. Both the finite and the infinite case are special cases of this normal form.

Definition 7 (*Temporal literal, rule, and program*)
Given alphabet $\mathcal{A}$, we define the set of *temporal literals* as $\{a, \neg a, \bullet a, \neg\bullet a \mid a \in \mathcal{A}\}$.

A *temporal rule* is either:

- an *initial rule* of the form $B \to A$
- a *dynamic rule* of the form $\widehat{\circ}\,\Box(B \to A)$
- a *fulfillment rule* of the form $\Box(\,\Box\, p \to q\,)$ or $\Box(\,p \to \Diamond\, q\,)$

where $B = b_1 \wedge \cdots \wedge b_n$ with $n \geq 0$, $A = a_1 \vee \cdots \vee a_m$ with $m \geq 0$ and the b_i and a_j are temporal literals for dynamic rules and regular literals $\{a, \neg a \mid a \in \mathcal{A}\}$ for initial rules, and p and q are atoms.

A *temporal logic program* is a set of temporal rules.

Theorem 12 (*Normal form; Aguado et al. 2013; Cabalar et al. 2018*)
Every temporal formula φ can be converted into a temporal program being THT_f-equivalent to φ and into one being THT_ω-equivalent to φ.

We introduce another type of temporal rules, called *final rules*, of the form $\Box(\,\mathbf{F} \to (B \to A)\,)$.

Corollary 4
Every temporal formula can be converted into a THT_f-equivalent temporal logic program that consists of initial, dynamic, and final rules only. ⊠

In other words, in TEL_f, fulfillment rules are obsolete in temporal logic programs.

The reduction of temporal formulas into the normal form of temporal logic programs uses an extended alphabet $\mathcal{A}^+ \supseteq \mathcal{A}$ that additionally contains a new atom ℓ_φ (a.k.a. label) for each formula φ in the original language over $\mathcal{A}$ along with the label $\ell_{\neg\Diamond\mathbf{F}}$. This label represents the value of the formula $\neg\Diamond\mathbf{F}$, which means that it can be used to restrict ourselves to finite traces by including the formula $\Box(\neg\ell_{\neg\Diamond\mathbf{F}})$. For convenience, we use $\ell_\varphi \stackrel{def}{=} \varphi$ if φ is $\top, \bot$ or an atom $a \in \mathcal{A}$. For any non-atomic formula μ over $\mathcal{A}$, we define the translation *df* given in Tables 1 and 2, and call *df* the *definition* of μ. Note that the translation of the *next* operator $\circ$ includes the label $\ell_{\neg\Diamond\mathbf{F}}$. This ensures that if the formula $\Box(\neg\ell_{\neg\Diamond\mathbf{F}})$ is included and therefore only finite traces are considered, then $\circ\varphi$ cannot be satisfied in the final state.

These definitions of temporal formulas are not yet in normal form, but they contain some double implications that can easily be transformed into the format of temporal logic programs by simple, non-modal transformations in the propositional logic of HT. This translation of the definition into normal form can be found in (Cabalar et al. 2018) for the finite case and in (Aguado et al. 2013) for the infinite case. We denote the resulting set of formulas in normal form by $df^\star$. Given a theory Γ, we define $sub(\Gamma)$ as the set of all subformulas of all formulas in Γ.

Table 1. Definition of formulas

μ	$df(\mu)$
$\varphi \wedge \psi$	$\Box(\ell_\mu \leftrightarrow \ell_\varphi \wedge \ell_\psi)$
$\varphi \vee \psi$	$\Box(\ell_\mu \leftrightarrow \ell_\varphi \vee \ell_\psi)$
$\varphi \to \psi$	$\Box(\ell_\mu \leftrightarrow \ell_\varphi \to \ell_\psi)$
$\circ\varphi$	$\widehat{\circ}\Box(\bullet\ell_\mu \leftrightarrow \ell_\varphi)$ $\Box(\Box\ell_\mu \to \ell_{\neg\Diamond\mathbf{F}})$
$\bullet\varphi$	$\neg\ell_\mu$ $\widehat{\circ}\Box(\ell_\mu \leftrightarrow \bullet\ell_\varphi)$

Table 2. Definition of formulas

μ	$df(\mu)$
$\varphi \,\mathbf{U}\, \psi$	$\widehat{\circ}\Box(\bullet\ell_\mu \leftrightarrow \bullet\ell_\psi \vee (\bullet\ell_\varphi \wedge \ell_\mu))$ $\Box(\ell_\mu \to \Diamond\ell_\psi)$ $\Box(\ell_\psi \to \Diamond\ell_\mu)$
$\varphi \,\mathbf{R}\, \psi$	$\widehat{\circ}\Box(\bullet\ell_\mu \leftrightarrow \bullet\ell_\psi \wedge (\bullet\ell_\varphi \vee \ell_\mu))$ $\Box(\Box\ell_\psi \to \ell_\mu)$ $\Box(\Box\ell_\mu \to \ell_\psi)$
$\varphi \,\mathbf{S}\, \psi$	$\ell_\mu \leftrightarrow \ell_\psi$ $\widehat{\circ}\Box(\ell_\mu \leftrightarrow \ell_\psi \vee (\ell_\varphi \wedge \bullet\ell_\mu))$
$\varphi \,\mathbf{T}\, \psi$	$\ell_\mu \leftrightarrow \ell_\psi$ $\widehat{\circ}\Box(\ell_\mu \leftrightarrow \ell_\psi \wedge (\ell_\varphi \vee \bullet\ell_\mu))$

Definition 8 (*Aguado et al. 2013; Cabalar et al. 2018*)
We define the translation σ as the following temporal logic program:

$$\sigma(\Gamma) = \{\ell_\gamma \mid \gamma \in \Gamma\} \cup \{df^\star(\mu) \mid \mu \in sub(\Gamma)\}$$

⊠

Theorem 13 (*Aguado et al. 2013; Cabalar et al. 2018*)
For any theory Γ over $\mathcal{A}$, we have $\{\mathbf{M} \mid \mathbf{M} \models \Gamma\} = \{\mathbf{M}'|_{\mathcal{A}} \mid \mathbf{M}' \models \sigma(\Gamma)\}$. ⊠

This, correspondence between models of Γ and $\sigma(\Gamma)$ is, in fact, one-to-one, since the satisfaction of each label ℓ_γ is equivalent to the satisfaction of its associated formula γ. The next result shows that the computation of $\sigma(\Gamma)$ has a polynomial complexity on the size of Γ.

Theorem 14
Translation σ is linear. ⊠

As an example, consider the theory $\Gamma_1 = \Box(\neg p \to q \,\mathbf{U}\, p)$. Its translation $\sigma(\Gamma_1)$ can be determined using Table 3, independently of the finiteness of the trace. On finite traces, the fulfillment rules (29), (30), (42) and (43) can be replaced by the corresponding final rules. On infinite traces, the initial rules (22), (23), (31), (32), (33), (34) can be combined with dynamic rules (24), (25), (35), (36), (37), (38) into formulas of the form $\Box(r)$ where r is the given initial rule. Rules (30) and (43) are superfluous in the infinite case.

The interest of temporal logic programs is that, as suggested by their name, they have a syntax closer to logic programming. This suggests that, at least for finite traces TEL_f, computing TS-models can be delegated to an ASP solver in some way. In the rest of the section, we explain two different methods to achieve that goal.

5.1 Bounded translation of temporal programs over finite traces to ASP

A first way to encode a temporal program P into standard ASP is by assuming that we know the (finite) trace length λ beforehand. In that case, we can compute all models in $\mathrm{TEL}(P, \lambda)$ by a simple translation of P into a regular program. For this, we let $\mathcal{A}_k = \{a_k \mid a \in \mathcal{A}\}$ be a time stamped copy of alphabet $\mathcal{A}$ for each time point $k \in [0..\lambda)$.

Table 3. Translation of Γ_1 into normal form

μ	$df^\star(\mu)$	
$\neg p$	$\ell_1 \wedge p \to \bot$	(22)
	$\neg p \to \ell_1$	(23)
	$\widehat{\circ}\Box(\ell_1 \wedge p \to \bot)$	(24)
	$\widehat{\circ}\Box(\neg p \to \ell_1)$	(25)
$q \mathbf{U} p$	$\widehat{\circ}\Box(\bullet\ell_2 \to \bullet p \vee \bullet q \wedge \ell_2)$	(26)
	$\widehat{\circ}\Box(\bullet q \wedge \ell_2 \to \bullet\ell_2)$	(27)
	$\widehat{\circ}\Box(\bullet p \to \bullet\ell_2)$	(28)
	$\Box(\ell_2 \to \Diamond p)$	(29)
	$\Box(p \to \Diamond\ell_2)$	(30)

μ	$df^\star(\mu)$	
$\neg p \to q \mathbf{U} p$	$\ell_3 \wedge \ell_1 \to \ell_2$	(31)
	$\neg\ell_1 \to \ell_3$	(32)
	$\ell_2 \to \ell_3$	(33)
	$\ell_1 \vee \neg\ell_2 \vee \ell_3$	(34)
	$\widehat{\circ}\Box(\ell_2 \to \ell_3)$	(35)
	$\widehat{\circ}\Box(\ell_3 \wedge \ell_1 \to \ell_2)$	(36)
	$\widehat{\circ}\Box(\neg\ell_1 \to \ell_3)$	(37)
	$\widehat{\circ}\Box(\top \to \ell_1 \vee \neg\ell_2 \vee \ell_3)$	(38)
$\Box(\neg p \to q \mathbf{U} p)$	$\widehat{\circ}\Box(\bullet\ell_3 \wedge \ell_4 \to \bullet\ell_4)$	(39)
	$\widehat{\circ}\Box(\bullet\ell_4 \to \bullet\ell_3)$	(40)
	$\widehat{\circ}\Box(\bullet\ell_4 \to \ell_4)$	(41)
	$\Box(\Box\ell_3 \to \ell_4)$	(42)
	$\Box(\Box\ell_4 \to \ell_3)$	(43)

Definition 9 (*Bounded translation; Cabalar et al. 2018*)
We define the translation τ of a temporal literal at time point k as:

$$\tau_k(a) \stackrel{def}{=} a_k \qquad \tau_k(\neg a) \stackrel{def}{=} \neg a_k \qquad \text{for } a \in \mathcal{A}$$
$$\tau_k(\bullet a) \stackrel{def}{=} a_{k-1} \qquad \tau_k(\neg\bullet a) \stackrel{def}{=} \neg a_{k-1} \qquad \text{for } a \in \mathcal{A}$$

We define the translation of any temporal rule r in Definition 7 at time point k as:

$$\tau_k(r) \stackrel{def}{=} \tau_k(a_1) \vee \cdots \vee \tau_k(a_m) \leftarrow \tau_k(b_1) \wedge \cdots \wedge \tau_k(b_n)$$

We define the translation of a temporal program P bounded by finite length λ as:

$$\tau_\lambda(P) \stackrel{def}{=} \{\tau_0(r) \mid r \in I(P)\} \ \cup\ \{\tau_k(r) \mid r \in D(P), k \in [1..\lambda)\} \ \cup\ \{\tau_{\lambda-1}(r) \mid r \in F(P)\}$$

⊠

Note that the translation of temporal rules is similar in just considering the implication $B \to A$ in Definition 7; their difference manifests itself in their instantiation in $\tau_\lambda(P)$.

As an example, let program P be the set of temporal rules:

$$\{\ \to a,\quad \widehat{\circ}\,\Box(\bullet a \to b),\quad \Box(\mathbf{F} \to (\neg b \to \bot))\ \} \tag{44}$$

This program has a single finite temporal stable model of length 2, viz. $\{a\} \cdot \{b\}$. Applying translation τ for some bound λ to our temporal program P in (44) yields regular logic programs of the following form.

$$\tau_\lambda(P) = \{a_0 \leftarrow\ \} \ \cup\ \{b_k \leftarrow a_{k-1} \mid k \in [1..\lambda)\} \ \cup\ \{\bot \leftarrow \neg b_{\lambda-1}\}$$

Program $\tau_1(P)$ has the stable model $\{a_0, b_1\}$ but all $\tau_\lambda(P)$ for $\lambda > 2$ are unsatisfiable.

Theorem 15 (*Cabalar et al. 2018*)
Let P be a temporal program over $\mathcal{A}$. Let $\mathbf{T} = (T_i)_{i\in[0..\lambda)}$ be a trace of finite length λ over $\mathcal{A}$ and X a set of atoms over $\bigcup_{i\in[0..\lambda)} \mathcal{A}_i$ such that $a \in T_i$ iff $a_i \in X$ for $i \in [0..\lambda)$.

Then, $\mathbf{T}$ is a temporal stable model of P iff X is a stable model of $\tau_\lambda(P)$. ⊠

Applied to our example, this result confirms that the temporal stable model $\{a\} \cdot \{b\}$ of P corresponds to the stable model $\{a_0, b_1\}$ of $\tau_2(P)$.

Using this translation we have implemented a system, `tel`[6], that takes a propositional theory Γ of arbitrary temporal formulas and a bound λ and returns the regular logic program $\tau_\lambda(P)$, where P is the intermediate normal form of Γ left implicit. The resulting program $\tau_\lambda(P)$ can then be solved by any off-the-shelf ASP system. For illustration, consider the representation of our example temporal program in (44) in `tel`'s input language.

```
a.
#next^ #always+ ( (#previous a) -> b).
#always+ ( #final -> (~ b -> #false)).
```

As expected, passing the result of `tel`'s translation for horizon 2 to `clingo` yields the stable model containing `a(0)` and `b(1)` (suppressing auxiliary atoms).

5.2 Pointwise translation of temporal programs over finite traces to ASP

The bounded translation $\tau_\lambda(P)$ allows us to compute all models in $\mathrm{TEL}(P, \lambda)$ for a fixed bound λ. However, in many practical problems (as in planning, for instance), λ is unknown beforehand and the crucial task consists in finding a representation of $\mathrm{TEL}(P, k)$ that is easily obtained from that of $\mathrm{TEL}(P, k\text{-}1)$. In ASP, this can be accomplished via incremental solving techniques that rely upon the composition of logic program modules (Oikarinen and Janhunen 2006). The idea is then to associate the knowledge at each time point with a module and to successively add modules corresponding to increasing time points (while leaving all previous modules unchanged). A stable model obtained after k compositions then corresponds to a TEL_f-model of length k. This technique of modular computation, however, is only applicable when modules are *compositional* (positive loops cannot be formed across modules), something that cannot always be guaranteed for arbitrary temporal programs. Still, we identify in (Cabalar et al. 2018) a quite general syntactic fragment[7] that implies compositionality. We say that a temporal rule as in Definition 7 is *present-centered*, whenever all the literals $a_1, \dots, a_m$ in its head A are regular. Accordingly, a set of such rules is a present-centered temporal program. In fact, such programs are sufficient to capture common action languages – as an illustration, in Section 6, we show how to encode action language $\mathcal{BC}$ (Lee et al. 2013) into present-centered temporal programs in TEL_f.

Following these ideas, we provide next a "*point-wise*" variant of our translation that allows for defining one module per time point and is compositional for the case of present-centered temporal programs. We begin with some definitions. A *module* $\mathbb{P}$ is a triple (P, I, O) consisting of a logic program P over alphabet $\mathcal{A}_P$ and sets I and O of *input* and *output* atoms such that

1. $I \cap O = \emptyset$,
2. $\mathcal{A}_P \subseteq I \cup O$, and
3. $H(P) \subseteq O$,

where $H(P)$ gives all atoms occurring in rule heads in P. Whenever clear from context,

[6] `https://github.com/potassco/tel`
[7] In order to compute loop formulas for TEL_ω, (Cabalar and Diéguez 2011) used a similar fragment (*splittable programs*) where rules cannot derive information from the future to the past.

we associate $\mathbb{P}$ with (P, I, O). In our setting, a set X of atoms is a stable model of $\mathbb{P}$, if X is a stable model of logic program P.[8] Two modules $\mathbb{P}_1$ and $\mathbb{P}_2$ are *compositional*, if $O_1 \cap O_2 = \emptyset$ and $O_1 \cap C = \emptyset$ or $O_2 \cap C = \emptyset$ for every strongly connected component C of the positive dependency graph of the logic program $P_1 \cup P_2$. In other words, all rules defining an atom must belong to the same module, and no positive recursion is allowed among modules. Whenever $\mathbb{P}_1$ and $\mathbb{P}_2$ are compositional, their *join* is defined as the module $\mathbb{P}_1 \sqcup \mathbb{P}_2 = (P_1 \cup P_2, (I_1 \setminus O_2) \cup (I_2 \setminus O_1), O_1 \cup O_2)$. The module theorem (Oikarinen and Janhunen 2006) ensures that compatible stable models of $\mathbb{P}_1$ and $\mathbb{P}_2$ can be combined to one of $\mathbb{P}_1 \sqcup \mathbb{P}_2$, and vice versa.

For literals and rules, the point-wise translation τ^* coincides with τ up to final rules.

Definition 10 (*Point-wise translation: Temporal rules; Cabalar et al. 2018*)
We define the translation of a final rule r as in Definition 7 at time point k as

$$\tau^*_k(r) \stackrel{def}{=} \tau_k(a_1) \vee \cdots \vee \tau_k(a_m) \leftarrow \tau_k(b_1) \wedge \cdots \wedge \tau_k(b_n) \wedge \neg q_{k+1} \tag{45}$$

for a new atom $q \notin \mathcal{A}$ and of an initial or dynamic rule r as $\tau^*_k(r) \stackrel{def}{=} \tau_k(r)$.

The new atoms q_{k+1} in (45) are used to deactivate instances of final rules. This allows us to implement operator $\mathbf{F}$ by using $\neg q_{k+1}$ and therefore to enable the actual final rule unless q_{k+1} is derivable. The idea is then to make sure that at each horizon k the atom q_{k+1} is false, while $q_1, \dots, q_k$ are true. In this way, only $\tau^*_k(r)$ is potentially applicable, while all rules $\tau^*_i(r)$ are disabled at earlier time points $i \in [1..k)$.

Translation τ^* is then used to define modules for each time point as follows.

Definition 11 (*Point-wise translation: Modules*)
Let P be a present-centered temporal program over $\mathcal{A}$. We define the module $\mathbb{P}_k$ corresponding to P at time point k as:

$$\mathbb{P}_0 \stackrel{def}{=} (P_0, \{q_1\}, \mathcal{A}_0) \qquad \mathbb{P}_k \stackrel{def}{=} (P_k, \mathcal{A}_{k-1} \cup \{q_{k+1}\}, \mathcal{A}_k \cup \{q_k\}) \quad \text{for } k > 0$$

where

$$\begin{aligned} P_0 &\stackrel{def}{=} \{\tau^*_0(r) \mid r \in I(P)\} \cup \{\tau^*_0(r) \mid r \in F(P)\} \\ P_k &\stackrel{def}{=} \{\tau^*_k(r) \mid r \in D(P)\} \cup \{\tau^*_k(r) \mid r \in F(P)\} \cup \{q_k \leftarrow\} \end{aligned}$$

⊠

Each module $\mathbb{P}_k$ defines what holds at time point k. The underlying present-centeredness warrants that modules only incorporate atoms from previous time points, as reflected by $\mathcal{A}_{k-1}$ in the input of $\mathbb{P}_k$. The exception consists of auxiliary atoms like q_{k+1} that belong to the input of each $\mathbb{P}_k$ for $k > 0$ but only get defined in the next module $\mathbb{P}_{k+1}$. This corresponds to the aforementioned idea that q_{k+1} is false when $\mathbb{P}_k$ is the final module, and is set permanently to true once the horizon is incremented by adding $\mathbb{P}_{k+1}$. Note that atoms like q_{k+1} only occur negatively in rule bodies in $\mathbb{P}_k$ and hence cannot invalidate the modularity condition. This technique allows us to capture the transience of final rules.

[8] Note that the default value assigned to input atoms is *false* in multi-shot solving (Gebser et al. 2019); this differs from the original definition (Oikarinen and Janhunen 2006) where a choice rule is used.

The point-wise translation of our present-centered example program P from (44) yields the following modules.

$$\mathbb{P}_0 = (\{a_0 \leftarrow\} \cup \{\leftarrow \neg b_0, \neg q_1\}, \{q_1\}, \{a_0, b_0\})$$
$$\mathbb{P}_i = (\{b_i \leftarrow a_{i-1}\} \cup \{\leftarrow \neg b_i, \neg q_{i+1}\} \cup \{q_i \leftarrow\}, \{a_{i-1}, b_{i-1}, q_{i+1}\}, \{a_i, b_i, q_i\})$$
$$\bigsqcup_{i=0}^{\lambda-1} \mathbb{P}_i = (P_0 \cup \bigcup_{i=1}^{\lambda-1} P_i, \{q_\lambda\}, \{a_i, b_i \mid i \in [0..\lambda)\} \cup \{q_i \mid i \in [1..\lambda)\})$$

As above, only the composed module for $\lambda = 1$ yields a stable model, viz. $\{a_0, b_1, q_1\}$.

Theorem 16
Let P be a present-centered temporal program over $\mathcal{A}$. Let $\mathbf{T} = (T_i)_{i\in[0..\lambda)}$ be a trace of finite length λ over $\mathcal{A}$ and X a set of atoms over $\bigcup_{0\leq i<\lambda} \mathcal{A}_i$ such that $a \in T_i$ iff $a_i \in X$ for $i \in [0..\lambda)$. Then, we have that
$\mathbf{T}$ is a temporal stable model of P iff $X \cup \{q_i \mid i \in [0..\lambda)\}$ is a stable model of $\bigsqcup_{i\in[0..\lambda)} \mathbb{P}_i$.

As with Theorem 15, this result confirms that the temporal stable model $\{a\} \cdot \{b\}$ of P corresponds to the stable model $\{a_0, b_1, q_1\}$ of $\mathbb{P}_0 \sqcup \mathbb{P}_1$.

As one might expect, not any temporal theory is reducible to a present-centered temporal program. Hence, computing models via incremental solving imposes some limitations on the possible combinations of temporal operators. Fortunately, we can identify in (Cabalar et al. 2018) again a quite natural and expressive syntactic fragment that is always reducible to present-centered programs. We say that a temporal formula is a *past-future rule* if it consists of rules as those in Definition 7 where B and A are just temporal formulas with the following restrictions: B and A contain no implications other than negations $(\alpha \to \bot)$, B contains no future operators, and A contains no past operators. An example of a past-future rule is, for instance,

$$\square(\mathit{shoot} \wedge \bullet\blacklozenge \mathit{shoot} \wedge \blacksquare \mathit{unloaded} \to \lozenge \mathit{fail}) \tag{46}$$

capturing the sentence: *"If we make two shots with a gun that was never loaded, then it will eventually fail."* Then, we have the following result.

Theorem 17 (*Past-future reduction*)
Every past-future rule r can be converted into a present-centered temporal program that is TEL_f-equivalent to r. ⊠

This past-future form is aligned with Gabbay's orientation presented in (Gabbay 1987a) where past operators in the body (or antecedent) are used to declaratively check the recorded story, whereas future operators in the head (or consequent) are employed to provide imperative commands to be executed afterwards, as a result of firing the rule.

We have implemented a second system, `telingo`[9] (Cabalar et al. 2019), that deals with present-centered temporal programs that are expressible in the full (non-ground) input language of `clingo` extended with temporal operators. In addition, `telingo` offers several syntactic extensions to facilitate temporal modeling: first, next operators can be used in singular heads and, second, arbitrary temporal formulas can be used in integrity constraints. All syntactic extensions beyond the normal form of Theorem 12 are compiled

[9] `https://github.com/potassco/telingo`

away by means of the translation used in its proof. The resulting present-centered temporal programs are then processed according the point-wise translation.

To facilitate the use of operators $\bullet$ and $\circ$, telingo allows us to express them by adding leading or trailing quotes to the predicate names of atoms, respectively. For instance, the temporal literals $\bullet p(a)$ and $\circ q(b)$ can be expressed by 'p(a) and q'(b), respectively. For another example, consider the representation of the sentence *"A robot cannot lift a box unless its capacity exceeds the box's weight plus that of all held objects"*:

```
:- lift(R,B), robot(R), box(B,W),
   #sum { C : capacity(R,C); -V,O : 'holding(R,O,V) } < W.
```

Atom 'holding(R,O,V) expresses what the robot was holding at the *previous* time point.

The distinction between different types of temporal rules is done in telingo via clingo's #program directives (Gebser et al. 2019), which allow us to partition programs into subprograms. More precisely, each rule in telingo's input language is associated with a temporal rule r of form $(b_1 \wedge \cdots \wedge b_n \rightarrow a_1 \vee \cdots \vee a_m)$ as in Definition 7 and interpreted as r, $\widehat{\circ}\Box r$, or $\Box(\mathbf{F} \rightarrow r)$ depending on whether it occurs in the scope of a program declaration headed by initial, dynamic, or final, respectively. Additionally, telingo offers always for gathering rules preceded by $\Box$ (thus dropping $\widehat{\circ}$ from dynamic rules). A rule outside any such declaration is regarded to be in the scope of initial. This allows us to represent the temporal program in (44) in the two alternative ways shown in Table 4.

```
#program initial.
a.

#program dynamic.
b :- 'a.

#program final.
  :- not b.
```

```
#program always.
a :- &initial.


b :- 'a.


  :- not b, &final.
```

Table 4. Two alternative telingo encodings for the temporal program in (44)

As mentioned, telingo allows us to use nested temporal formulas in integrity constraints as well as in negated form in place of temporal literals within rules. This is accomplished by encapsulating temporal formulas like φ in expressions of the form '&tel { φ }'. To this end, the full spectrum of temporal operators is at our disposal. They are expressed by operators built from < and > depending on whether they refer to the past or the future, respectively. So, </1, <?/2, and <*/2 stand for $\bullet$, $\mathbf{S}$, and $\mathbf{T}$, and >/1, >?/2, >*/2 for $\circ$, $\mathbf{U}$, $\mathbf{R}$. Accordingly, <*/1, <?/1, <:/1 represent $\blacksquare$, $\blacklozenge$, $\widehat{\bullet}$, and analogously their future counterparts. $\mathbf{I}$ and $\mathbf{F}$ are are represented by &initial and &final. This is complemented by Boolean connectives &, |, ~, etc. For example, the integrity constraint [10] '$shoot \wedge \blacksquare unloaded \wedge \bullet\blacklozenge shoot \rightarrow \bot$' can be expressed as follows.

```
:- shoot, &tel { <* unloaded & < <? shoot }.
```

[10] Similarly, formula (46) could be represented by several present-centered rules (including auxiliary atoms).

Listing 1. telingo encoding for the Fox, Goose and Beans Puzzle

```
1  #program always.

3  item(fox;beans;goose).
4  route(river_bank,far_bank). route(far_bank,river_bank).
5  eats(fox,goose). eats(goose,beans).

7  #program initial.

9  at(farmer,river_bank).
10 at(X,river_bank) :- item(X).

12 #program dynamic.

14 move(farmer).
15 0 { move(X) : item(X) } 1.

17 at(X,B) :- 'at(X,A), move(X), route(A,B).
18 :- move(X), item(X), 'at(farmer,A), not 'at(X,A).

20 at(X,A) :- 'at(X,A), not move(X).

22 #program always.

24 :- at(X,A), at(X,B), A<B.
25 :- eats(X,Y), at(X,A), at(Y,A), not at(farmer,A).

27 #program final.

29 :- at(X,river_bank).

31 #show move/1.
32 #show at/2.
```

Once `telingo` has translated an (extended) temporal program into a regular one, it is incrementally solved by `clingo`'s multi-shot solving engine (Gebser et al. 2019).

To conclude this section, we provide a larger example of `telingo` encoding (a more detailed description of the system is included in (Cabalar et al. 2019)). Listing 1 contains an exemplary `telingo` encoding of the *Fox, Goose and Beans Puzzle* available at `https://github.com/potassco/telingo/tree/master/examples/river-crossing`.

> *Once upon a time a farmer went to a market and purchased a fox, a goose, and a bag of beans. On his way home, the farmer came to the bank of a river and rented a boat. But crossing the river by boat, the farmer could carry only himself and a single one of his purchases: the fox, the goose, or the bag of beans. If left unattended together, the fox would eat the goose, or the goose would eat the beans. The farmer's challenge was to carry himself and his purchases to the far bank of the river, leaving each purchase intact. How did he do it?*
>
> (`https://en.wikipedia.org/wiki/Fox,_goose_and_bag_of_beans_puzzle`)

In Listing 1, lines 3-5 and 9-10 provide facts holding in all and the initial states, respectively; this is indicated by the program directives headed by `always` and `initial`. The dynamic rules in lines 14-22 describe the transition function. The `farmer` moves at each time step (Line 14), and may take an item or not (Line 15). Line 17 describes the effect of action `move/1`, Line 18 its precondition, and Line 20 the law of inertia. The second part of the `always` rules give state constraints in Line 24 and 25. The `final` rule in Line 29 gives the goal condition. All in all, we obtain two shortest plans consisting of eight states in about 20 ms. Restricted to the move predicate, `telingo` reports the following solutions:

Time	Solution 1		Solution 2	
1				
2	`move(farmer)`	`move(goose)`	`move(farmer)`	`move(goose)`
3	`move(farmer)`		`move(farmer)`	
4	`move(beans)`	`move(farmer)`	`move(farmer)`	`move(fox)`
5	`move(farmer)`	`move(goose)`	`move(farmer)`	`move(goose)`
6	`move(farmer)`	`move(fox)`	`move(beans)`	`move(farmer)`
7	`move(farmer)`		`move(farmer)`	
8	`move(farmer)`	`move(goose)`	`move(farmer)`	`move(goose)`

This example was also used by (Cabalar and Diéguez 2011) to illustrate the working of `stelp`, a tool for temporal answer set programming with TEL_ω. We note that `stelp` and `telingo` differ syntactically in describing transitions by using next or previous operators, respectively. Since `telingo` extends `clingo`'s input language, it offers a richer input language, as witnessed by the cardinality constraints in Line 15 in Listing 1. Finally, `stelp` uses a model checker and outputs an automaton capturing all infinite traces while `telingo` returns finite traces corresponding to plans.

6 TEL_f and the action language $\mathcal{BC}$

Present-centered temporal programs over finite traces are sufficient to capture common action languages. We show this by providing a translation from action descriptions in action language $\mathcal{BC}$ (Lee et al. 2013) into temporal logic programs in TEL_f. The result is similar to the one in (Lee et al. 2013), where a translation from $\mathcal{BC}$ into a sequence of regular logic programs with nested expressions is presented.

To this end, we need the following definitions describing the major concepts of $\mathcal{BC}$. An *action signature* in the language $\mathcal{BC}$ includes two finite set of symbols, the set $\mathbf{F}$ of fluent constants and the set $\mathbf{A}$ of action constants. A finite set with at least two elements, called domain, is assigned to each fluent constant. The set of all values of all domains is denoted by $\mathbf{V}$. An atom is an expression of the form $f = v$, where f is a fluent constant, and v is an element of its domain. If the domain of f consists of truth values $\mathbf{f}$ and $\mathbf{t}$, we say that f is Boolean. The *transition system* of such an action signature $\langle \mathbf{V}, \mathbf{F}, \mathbf{A} \rangle$ consists of:

- a set S,
- a function V from $\mathbf{F} \times S$ into $\mathbf{V}$, and
- a subset R of $S \times \mathbf{A} \times S$.

An *action description* in $\mathcal{BC}$ consists of a finite set of static and dynamic laws.

- A *static law* is an expression of the form

$$A_0 \textbf{ if } A_1, \dots, A_m \textbf{ ifcons } A_{m+1}, \dots, A_n \tag{47}$$

($n \geq m \geq 0$), where each A_i is an atom.

- A *dynamic law* is an expression of the form

$$A_0 \textbf{ after } A_1, \ldots, A_m \textbf{ ifcons } A_{m+1}, \ldots, A_n \tag{48}$$

($n \geq m \geq 0$), where

— A_0 is an atom containing a regular fluent constant,
— each of $A_1, \ldots, A_m$ is an atom or an action constant, and
— $A_{m+1}, \ldots, A_n$ are atoms.

Theorem 18 (*Translation from $\mathcal{BC}$ into* TEL_f)
For every action description D in $\mathcal{BC}$, there is a temporal logic program $P(D)$ such that the temporal stable models of length λ of $P(D)$ represent paths of length λ in the transition system corresponding to D.

Proof
We prove Theorem 18 by constructing a program $P(D)$ satisfying the desired property. We then define the transition system described by D and use induction over the path and trace length λ to show the result.

- We can construct such a program $P(D)$ consisting of the following rules:

— the translations

$$\{(A_1 \wedge \ldots \wedge A_m \rightarrow A_0 \vee \neg A_{m+1} \vee \ldots \vee A_n),$$
$$\widehat{\circ}\Box(A_1 \wedge \ldots \wedge A_m \rightarrow A_0 \vee \neg A_{m+1} \vee \ldots \vee A_n)\}$$

of all static laws A_0 **if** $A_1, \ldots, A_m$ **ifcons** $A_{m+1}, \ldots, A_n$ of D,

— the translations

$$\widehat{\circ}\Box(\bullet A_1 \wedge \ldots \wedge \bullet A_m \rightarrow A_0 \vee \neg A_{m+1} \vee \ldots \vee A_n)$$

of all dynamic laws A_0 **after** $A_1, \ldots, A_m$ **ifcons** $A_{m+1}, \ldots, A_n$ of D,

— the choice rule $\{A\}$ for every atom A containing a regular fluent constant,
— the choice rules $\{a\}$ and $\widehat{\circ}\Box\{a\}$ for every action constant a
— the existence of value constraints $\top \rightarrow ((f = v_1) \vee \cdots \vee (f = v_k))$ and $\widehat{\circ}\Box(\top \rightarrow ((f = v_1) \vee \cdots \vee (f = v_k)))$ for every fluent constant f, where $v_1, \ldots, v_k$ are all elements of the domain of f,
— the uniqueness of value constraints $(f = v) \wedge (f = w) \rightarrow \bot$ and $\widehat{\circ}\Box((f = v) \wedge (f = w) \rightarrow \bot)$ for every fluent constant f and every pair of distinct elements v, w of its domain.

- The transition system $T(D)$ corresponding to D can be defined as follows:

— For every temporal stable model (X_0) of $P(D)$ of length 1, the set of atoms that belong to X_0 is a state of $T(D)$ ($\Rightarrow$ set S)
— For every temporal stable model $(X_i)_{i\in\{0,1\}}$ of $P(D)$, $T(D)$ includes the transition $\langle s_0, a, s_1\rangle$, where s_i with $i \in \{0,1\}$ is the set of atoms that belong to X_i, and a is the set of action constants that belong to X_0. ($\Rightarrow$ subset R of $S \times \mathbf{A} \times S$)
— Soundness: For every transition $\langle s_0, a, s_1\rangle$, s_0 and s_1 are states.

- For every state s and every fluent constant f there exists exactly one v such that $f = v$ belongs to s (due to the existence of value and uniqueness of value constraints). This v is considered the value of f in state s. ($\Rightarrow$ function V from $\mathbf{F} \times S$ into $\mathbf{V}$)

- For $\lambda = 0$ and for $\lambda = 1$ it follows directly from the definition that paths of length λ in this transition system correspond to the temporal stable models of length λ of $P(D)$.
- For a trace $(X_i)_{i\in(0,\dots,\lambda)}$ and for $n < \lambda$ we define $t(X_n) = \langle s_n, a, s_{n+1}\rangle$, where s_n is the set of atoms that belong to X_n (and analogously for n+1), and a is the set of action constants that belong to X_n. By induction on λ, we see that $(X_i)_{i\in(0,\dots,\lambda)}$ is a temporal stable model for $P(D)$ iff $t(X_n)$ is a transition for all $n < \lambda$.

□

7 Related Work

As explained in the introduction, the field of Temporal Logic Programming has its roots in the mid 1980s, appearing soon after the introduction of modal extensions of Prolog (Fariñas del Cerro 1986; Bieber et al. 1988). Several logic programming languages dealing with LTL operators were proposed (Moszkowski 1986; Fujita et al. 1986; Gabbay 1987b; Abadi and Manna 1989; Orgun and Wadge 1992; Baudinet 1992). The latter, a formalism called TEMPLOG, is perhaps a prominent case from a logical point of view. It provides a logical semantics in terms of a least LTL-model, in the spirit of the well-known least Herbrand model for positive[11] logic programs (van Emden and Kowalski 1976). In (Cabalar and Diéguez 2014) (Theorem 19), it was proved that TEMPLOG is actually subsumed by TEL, that is, the latter can be used as a generalisation of the former for an arbitrary temporal syntax that includes default negation. However, although TEL provides a common underlying semantics to TEMPLOG and our temporal programs (which are the basis of `telingo`), these two languages understand the temporal logic programming paradigm in a substantially different way, analogous to the differences between Prolog and ASP. In particular, TEMPLOG understands rules in a top-down fashion where the head is considered a *goal* and the body is seen as a *method* (list of subgoals) to achieve the goal. As a result, (future-time) LTL operators are used to represent temporal relations affecting to the achievement of these subgoals. As an example, one possible TEMPLOG rule could look like:

$$\Box(\ \mathit{printorder} \leftarrow \mathit{ack}, \circ(\mathit{print}, \Diamond \mathit{finished})\ )$$

meaning that, at any moment, a *printorder* is fulfilled by immediately sending an acknowledgment *ack*, then starting the printer *print* and eventually sending a *finished* message. This TEMPLOG rule naturally corresponds to the TEL implication:

$$\Box(\ \mathit{ack} \wedge \circ(\mathit{print} \wedge \Diamond \mathit{finished}) \rightarrow \mathit{printorder}\ )$$

and has indeed the same semantics in TEL but, as we see, does not fit into the past-future syntactic fragment used in `telingo`, since the *printorder* in the head is at the relative

[11] Note that the different semantics for negation as failure were still in their early steps at that moment.

past of the conditions required in the body, talking about future satisfaction of *print* or *finished*. Following the ASP bottom-up understanding of rules, which is closer to causal laws in action languages, this same example would be represented instead as the past-future formula:

$$\Box(printorder \rightarrow ack \wedge \circ(print \wedge \Diamond finished))$$

which can be reduced afterwards to a present-centered logic program. To sum up, TEL is rich enough to cover both bottom-up and top-down readings of program rules, but for its use for temporal ASP, it suffices with a syntactic fragment where past in checked in the bodies and future is used in the head.

The extension of ASP with temporal operators has been studied in other approaches. A preliminary definition of temporal answer sets for LTL was already presented in (Cabalar 1999), but it relied on a different three-valued semantics that was not a proper extension of Equilibrium Logic, losing some of the interesting properties that the latter has shown as a logical encoding of ASP. Another approach that defined temporal answer sets as the result of a models selection among linear traces was presented in (Giordano et al. 2013). In this case, rather than LTL, the temporal approach used as a basis was *Dynamic Linear Temporal Logic* (DLTL) (Henriksen and Thiagarajan 1999), a linear-time variant of the well-known *Dynamic Logic* (Harel et al. 2000). Besides, that definition of temporal answer sets was only applicable to a syntactic fragment of DLTL (also called temporal logic programs) since the semantics relied on a temporal extension of the classical (syntactic) *reduct* transformation (Gelfond and Lifschitz 1988). In order to facilitate a formal comparison, the approach in (Aguado et al. 2013) introduced a proper extension of TEL that covers the full syntax of DLTL. The main result in that paper proved that, in fact, this extension of TEL for DLTL subsumes the semantics in (Giordano et al. 2013) and, in fact, provides its generalization for DLTL arbitrary formulas, without depending on syntactic transformations.

Thanks to the incorporation of regular expressions, dynamic modalities are very interesting for the specification of *control rules* describing steps that are required to be followed by any solution to a temporal problem. Differently from DLTL, another (perhaps more direct) direction for introducing dynamic logic modalities on top of a linear-time semantics is the so-called *Linear Dynamic Logic* (LDL) (De Giacomo and Vardi 2013). Both DLTL and LDL are more expressive than LTL, while satisfiability in the three logics share the same PSpace complexity. However, DLTL uses regular expressions as modifiers of the *until* and *release* operators, whereas the syntax of LDL is closer to the usual in dynamic logic, where modalities include the standard necessity $[\rho]\varphi$ and possibility $\langle\rho\rangle\varphi$ constructs, being ρ is a regular expression. Besides, DLTL regular expressions are built on a separated signature (a set of atomic *actions*) and do not accept the test construct φ? from dynamic logic. One more advantage of LDL is that a finite trace variant LDL_f has also been proposed and that both LDL and LDL_f properly generalize LTL and LTL_f respectively. All these reasons make LDL a more promising candidate to incorporate dynamic operators in ASP. As a result, the paper (Bosser et al. 2018) followed analogous steps to TEL with LTL and introduced a non-monotonic extension of LDL called *Linear Dynamic Equilibrium Logic* (DEL). Moreover, a first implementation of LDL operators in `telingo` has been recently presented in (Cabalar et al. 2020), although these are only allowed in constraint rules. Although this restriction will be lifted in the future, it already supports an agreeable modeling methodology for dynamic domain separating action and control theories. The idea is to model the actual action theory with temporal rules, fixing static and dynamic laws, while the control theory, enforcing certain (sub)trajectories, is expressed by integrity constraints using dynamic formulas. This is similar to the pairing of action theories in situation calculus and Golog programs (Levesque et al. 1997).

The combination of temporal modalities and non-monotonic reasoning is not exclusive from logic programming approaches and was also explored inside the area of Reasoning about Actions and Change. An early approach using LTL is, for instance, the *Past Temporal Logic for Actions* (PTLA) in (Mendez et al. 1996), an action language incorporating past LTL operators in the conditions of action laws. These action descriptions were then translated into logic programs under the ASP semantics using a similar transformation as the one presented in Section 5.1. Several authors have explored the formalization of action and change using Dynamic Logic (Matteo Baldoni and Patti 1996; Schwind 1997; Castilho et al. 1999), though only the first one relies on a logic programming paradigm.

Another prominent AI field for reasoning about dynamic systems where temporal logic has played an important role is Planning. In this area, the system behavior is specified in terms of some planning-specific language like STRIPS or PDDL, with a carefully limited syntax that avoids the need for an explicit representation of the (non-monotonic) law of inertia. This limited syntax restricts the specification of dynamic systems to a less expressive language than temporal ASP with `telingo`, where defaults, induction or aggregates can be freely combined with temporal operators, but has the advantage of allowing the design and implementation of efficient planning algorithms. Rather than in the description of the transition system, the introduction of temporal formulas in planning has been traditionally related to a richer specification of the *goal*, so that it not only provides a condition about the final state to be reached, but is also extended with temporal formulas (Bacchus and Kabanza 2000; Bertoli et al. 2001; Kvarnström et al. 2008) imposing constraints on the sequence of actions that form the plan, something that can be used to improve the efficiency of the planning algorithm. This strategy can also be extrapolated to temporal ASP, although the effect of temporal constraints in `telingo` has a different impact on efficiency, given that the algorithm is based on incremental solving. A different strategy for temporal ASP closer to classical planning algorithms was explored in the prototype presented in (Cabalar et al. 2019). In this case, a classical graph search algorithm was implemented and multi-shot ASP solving was used to compute successor states during search. This has the advantage of being able to explore the whole state space and deciding whether a given planning problem has no solution, something impossible by the incremental horizon strategy followed in `telingo`. However, this planning based prototype had a rather limited syntax and did not allow the use of temporal expressions to incorporate temporal goals.

Other AI areas where temporal logic has been succesfully applied are, for instance, Multi Agent Systems (MAS) and Ontologies. In the case of MAS, a prominent example is the system Concurrent `MetateM` (Fisher 1994), a language using LTL operators that exploits Gabbay's separation theorem (Gabbay 1987a) asserting that LTL descriptions can be reduced to a set of implications with the form $past \rightarrow future$. Note that we also exploited this past-future form in `telingo` and our Theorem 17. The main difference between `MetateM` and temporal ASP, however, is that the former relies on monotonic LTL, so it does not allow closed world assumption, defaults or induction. On the other hand, MetateM incorporates high level constructs for agents specification and message passing that are not present in telingo.

Finally, in the case of LTL in Ontologies, the combination of Description Logics (DL) (Baader et al. 2003) with temporal patterns is an important field of knowledge representation that has been widely studied in the literature (see, for instance, the surveys (Artale and Franconi 2000; Artale and Franconi 2001; Lutz et al. 2008)). However, as happened in the MAS case, the result of these combinations are also monotonic. Combining ontological reasoning and logic programming has also been extensively studied, being perhaps the ASP extension called *Hybrid Knowledge Bases* (de Bruijn et al. 2010) the closest one to our approach, as it is based on Quantified Equilibrium Logic. In the preliminary work (Cabalar and Schaub 2019), we presented a combination of hybrid knowledge bases with the temporal ontology framework ALC-LTL from (Baader et al. 2008) that allowed the use of temporal ontological axioms in temporal logic programs, but no implementation has been made so far.

8 Discussion

We have provided a wide overview of the main definitions and recent results for the formalism of *Temporal Equilibrium Logic* (TEL), a combination of Equilibrium Logic (a logical characterization of Answer Set Programming, ASP) with Linear-Time Temporal Logic. After more than a decade of study, the knowledge about TEL has achieved a high degree of maturity, both at a fundamental level and also at an incipient practical application. An important breakthrough for the latter has been the introduction of a finite trace variant, TEL_f, more aligned with the usual problem solving orientation followed in ASP. This has opened the possibility of introducing temporal operators in ASP solving with the construction of the tool telingo, a temporal extension of clingo that exploits the incremental solving capabilities of the latter. Moreover, the definition of TEL_f has also provided a methodology for the study of other extensions, like the introduction of dynamic logic operators (Cabalar et al. 2020) or, more recently, the definition of temporal metric constructions (Cabalar et al. 2020) that are planned to be further developed and combined with other features inside the software packages of potassco.org.

Regarding future work, there are many interesting lines to be explored yet. For instance, we still miss an axiomatization that covers the finite trace variant of the monotonic basis, THT_f, or a general axiomatization that is applicable to THT, that is, regardless of the trace length. Also, properties related to inter-definability of operators need to be further explored for the finite trace case. Related to this, in (Aguado et al. 2020) proposed the introduction of an explicit negation operator that allows inter-definability of temporal operators like *until* and *release* using De Morgan laws. The connection of TEL_f to finite automata is also a topic of active current research. Automata construction techniques may become crucial in the future both for formal verification of dynamic systems specified with telingo but also for exploiting temporal expressions in a more compact way during computation, rather than making a full unfolding into regular ASP programs. Another important important line for future study is how to exploit temporal constructions for grounding: right now, telingo does not recognize temporal information at that stage. A first study about temporal grounding was provided in (Aguado et al. 2017), but it was focused on infinite traces, being those results not extrapolable to the case of `telingo`, where the hypothesis of finite traces can be exploited more efficiently.

References

Abadi, M. and Manna, Z. 1989. Temporal logic programming. *Journal of Symbolic Computation 8*, 277–295.

Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., and Vidal, C. 2013. Temporal equilibrium logic: a survey. *Journal of Applied Non-Classical Logics 23,* 1-2, 2–24.

Aguado, F., Cabalar, P., Diéguez, M., Pérez, G., and Vidal, C. 2017. Temporal equilibrium logic with past operators. *Journal of Applied Non-Classical Logics 27,* 3-4, 161–277.

Aguado, F., Cabalar, P., Fandinno, J., Pérez, G., and Vidal, C. 2020. Explicit negation in linear-dynamic equilibrium logic. In *Proc. of the 24th European Conference on Artificial Intelligence (ECAI'20)* (10-12 June). Santiago de Compostela, Spain.

Aguado, F., Cabalar, P., Pearce, D., Pérez, G., and Vidal, C. 2015. A denotational semantics for equilibrium logic. *Theory and Practice of Logic Programming 15,* 4-5, 620–634.

Aguado, F., Cabalar, P., Pérez, G., Vidal, C., and Diéguez, M. 2017. Temporal logic programs with variables. *Theory and Practice of Logic Programming 17,* 2, 226–243.

Aguado, F., Pérez, G., and Vidal, C. 2013. Integrating temporal extensions of answer set programming. In *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*, P. Cabalar and T. Son, Eds. Lecture Notes in Artificial Intelligence, vol. 8148. Springer-Verlag, 23–35.

Artale, A. and Franconi, E. 2000. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence 30,* 1-4, 171–210.

Artale, A. and Franconi, E. 2001. Temporal description logics. In *Handbook of Time and Temporal Reasoning in AI*. The MIT Press.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., Eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

Baader, F., Ghilardi, S., and Lutz, C. 2008. LTL over description logic axioms. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*. KR'08. AAAI Press, 684–694.

Bacchus, F. and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence 116,* 1-2, 123–191.

Balbiani, P., Boudou, J., Diéguez, M., and Fernández-Duque, D. 2020. Intuitionistic linear temporal logics. *ACM Transactions on Computational Logic 21,* 2, 14:1–14:32.

Balbiani, P. and Diéguez, M. 2016. Temporal here and there. In *Proceedings of the Fifteenth European Conference on Logics in Artificial Intelligence (JELIA'16)*, L. Michael and A. Kakas, Eds. Lecture Notes in Artificial Intelligence, vol. 10021. Springer-Verlag, 81–96.

Baudinet, M. 1992. A simple proof of the completeness of temporal logic programming. In *Intensional Logics for Programming*, L. F. del Cerro and M. Penttonen, Eds. Clarendon Press, 51–83.

Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, B. Nebel, Ed. Morgan Kaufmann, 473–478.

Bieber, P., Fariñas del Cerro, L., and Herzig, A. 1988. MOLOG: a modal PROLOG. In *9th International Conference on Automated Deduction, Argonne, Illinois, USA, May 23-26, 1988, Proceedings*. 762–763.

Bosser, A., Cabalar, P., Diéguez, M., and Schaub, T. 2018. Introducing temporal stable models for linear dynamic logic. In *Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'18)*, M. Thielscher, F. Toni, and F. Wolter, Eds. AAAI Press, 12–21.

Bozzelli, L., Maubert, B., and Pinchinat, S. 2015. Unifying hyper and epistemic temporal logics. In *Proceedings of the Eighteenth International Conference on Foundations of Software Science and Computation Structures*, A. M. Pitts, Ed. Vol. 9034. Springer, 167–182.

Bozzelli, L. and Pearce, D. 2015. On the complexity of temporal equilibrium logic. In *Proceedings of the Thirtieth Annual Symposium on Logic in Computer Science (LICS'15)*. IEEE Computer Society Press, 645–656.

Büchi, J. R. 1962. On a Decision Method in Restricted Second Order Arithmetic. In *Proceedings of the 1960 International Congress on Logic, Methodology, and Philosophy of Science*, E. Nagel, P. Suppes, and A. Tarski, Eds. Stanford University Press, 1–11.

Cabalar, P. 1999. Temporal answer sets. In *1999 Joint Conference on Declarative Programming, AGP'99, L'Aquila, Italy, September 6-9, 1999*, M. C. Meo and M. V. Ferro, Eds. 351–366.

Cabalar, P. and Demri, S. 2011. Automata-based computation of temporal equilibrium models. In *Proceedings of the Twenty-first International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR'11)*, G. Vidal, Ed. Lecture Notes in Computer Science, vol. 7225. Springer-Verlag, 57–72.

Cabalar, P. and Diéguez, M. 2011. STELP — a tool for temporal answer set programming. In *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, J. Delgrande and W. Faber, Eds. Lecture Notes in Artificial Intelligence, vol. 6645. Springer-Verlag, 370–375.

Cabalar, P. and Diéguez, M. 2014. Strong equivalence of non-monotonic temporal theories. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)* (July 20-24). Vienna, Austria.

Cabalar, P., Diéguez, M., Laferriere, F., and Schaub, T. 2020. Implementing dynamic answer set programming over finite traces. In *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*, G. De Giacomo, Ed. IOS Press. To appear.

Cabalar, P., Diéguez, M., Schaub, T., and Schuhmann, A. 2020. Towards metric temporal answer set programming. In *Proc. of the 36th International Conference on Logic Programming (ICLP'20)* (September (to appear)).

Cabalar, P., Kaminski, R., Morkisch, P., and Schaub, T. 2019. telingo = ASP + time. In *Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*, M. Balduccini, Y. Lierler, and S. Woltran, Eds. Lecture Notes in Artificial Intelligence, vol. 11481. Springer-Verlag, 256–269.

Cabalar, P., Kaminski, R., Schaub, T., and Schuhmann, A. 2018. Temporal answer set programming on finite traces. *Theory and Practice of Logic Programming 18,* 3-4, 406–420.

Cabalar, P., Rey, M., and Vidal, C. 2019. A complete planner for temporal answer set programming. In *Progress in Artificial Intelligence, 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3-6, 2019, Proceedings, Part II*, P. M. Oliveira, P. Novais, and L. P. Reis, Eds. Lecture Notes in Computer Science, vol. 11805. Springer, 520–525.

Cabalar, P. and Schaub, T. 2019. Temporal logic programs with temporal description logic axioms. C. Lutz, U. Sattler, C. Tinelli, A. Turhan, and F. Wolter, Eds. Lecture Notes in Computer Science, vol. 11560. Springer-Verlag, 174–186.

Cabalar, P. and Vega, G. P. 2007. Temporal equilibrium logic: A first approach. In *Proceedings of the Eleventh International Conference on Computer Aided Systems Theory (EUROCAST'17)*, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds. Lecture Notes in Computer Science, vol. 4739. Springer-Verlag, 241–248.

Camacho, A., Baier, J. A., Muise, C. J., and McIlraith, S. A. 2018. Finite LTL synthesis as planning. In *Proceedings of the Twenty-Eight International Conference on Automated*

Planning and Scheduling, M. de Weerdt, S. Koenig, G. Röger, and M. T. J. Spaan, Eds. AAAI Press, 29–38.

Castilho, M. A., Gasquet, O., and Herzig, A. 1999. Formalizing action and change in modal logic I: the frame problem. *Journal of Logic and Computation 9,* 5, 701–735.

Chandra, A. K., Kozen, D. C., and Stockmeyer, L. J. 1981. Alternation. *Journal of the ACM 28,* 1, 114–133.

de Bruijn, J., Pearce, D., Polleres, A., and Valverde, A. 2010. A semantical framework for hybrid knowledge bases. *Knowledge Informaiton Systems 25,* 1, 81–104.

De Giacomo, G. and Vardi, M. 2013. Linear temporal logic and linear dynamic logic on finite traces. See Rossi (2013), 854–860.

Demri, S., Goranko, V., and Lange, M. 2016. *Temporal Logics in Computer Science: Finite-State Systems.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., and Xu, L. 2016. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16).* Lecture Notes in Computer Science, vol. 9938. Springer, 122–129.

Fages, F. 1994. Consistency of Clark's completion and the existence of stable models. *Journal of Methods of Logic in Computer Science 1*, 51–60.

Fariñas del Cerro, L. 1986. MOLOG: A system that extends PROLOG with modal logic. *New Generation Computing 4,* 1, 35–50.

Ferraris, P., Lee, J., and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, M. Veloso, Ed. AAAI/MIT Press, 372–379.

Fisher, M. 1994. A survey of concurrent METATEM - the language and its applications. In *Temporal Logic, First International Conference, ICTL '94, Bonn, Germany, July 11-14, 1994, Proceedings*, D. M. Gabbay and H. J. Ohlbach, Eds. Lecture Notes in Computer Science, vol. 827. Springer, 480–505.

French, T. and Reynolds, M. 2002. A sound and complete proof system for QPTL. In *Proceedings of the Fourth Conference on Advances in Modal Logic*, P. Balbiani, N. Suzuki, F. Wolter, and M. Zakharyaschev, Eds. King's College Publications, 127–148.

Fujita, M., Kono, S., Tanaka, T., and Moto-oka, T. 1986. Tokio: Logic programming language based on temporal logic and its compilation into prolog. In *Proc. of the 3rd Intl. Conf. on Logic Programming (LNAI 225).* Springer-Verlag.

Gabbay, D. 1987a. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Proceedings of the Conference on Temporal Logic in Specification*, B. Banieqbal, H. Barringer, and A. Pnueli, Eds. Lecture Notes in Computer Science, vol. 398. Springer-Verlag, 409–448.

Gabbay, D. 1987b. Modal and temporal logic programming. In *Temporal Logics and their Applications*, A. Galton, Ed. Academic Press, Chapter 6, 197–237.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming 19,* 1, 27–82.

Gelfond, M. and Lifschitz, V. 1988. Compiling circumscriptive theories into logic programs: Preliminary report. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Morgan Kaufmann Publishers, 455–459.

Gelfond, M. and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming 17,* 2-4, 301–321.

Gerth, R., Peled, D. A., Vardi, M. Y., and Wolper, P. 1995. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV, Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification,* *Testing and Verification, Warsaw, Poland, June 1995*, P. Dembinski and M. Sredniawa, Eds. IFIP Conference Proceedings, vol. 38. Chapman & Hall, 3–18.

Giordano, L., Martelli, A., and Theseider Dupré, D. 2013. Reasoning about actions with temporal answer sets. *Theory and Practice of Logic Programming 13,* 2, 201–225.

Goldblatt, R. 1992. *Logics of Time and Computation*, 2 ed. Number 7 in CSLI Lecture Notes. Center for the Study of Language and Information.

Gödel, K. 1932. Zum intuitionistischen Aussagenkalkül. *Anzeiger der Akademie der Wissenschaften in Wien*, 65–66.

Harel, D., Tiuryn, J., and Kozen, D. 2000. *Dynamic Logic.* MIT Press.

Henriksen, J., Jensen, J., Jørgensen, M., Klarlund, N., Paige, B., Rauhe, T., and Sandholm, A. 1995. Mona: Monadic second-order logic in practice. In *Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, E. Brinksma, R. Cleaveland, K. G. Larsen, T. Margaria, and B. Steffen, Eds. Vol. 1019. Springer, 89–110.

Henriksen, J. and Thiagarajan, P. 1999. Dynamic linear time temporal logic. *Annals Pure and Applied Logic 96,* 1-3, 187–207.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. In *Sitzungsberichte der Preussischen Akademie der Wissenschaften.* Deutsche Akademie der Wissenschaften zu Berlin, 42–56. Reprint in Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik, Akademie-Verlag, 1986.

Hosoi, T. 1966. The Axiomatization of the Intermediate Propositional Systems S_2 of Gödel. *Journal of the Faculty of Science of the University of Tokyo 13,* 2, 183–187.

Kamp, J. 1968. Tense logic and the theory of linear order. Ph.D. thesis, University of California at Los Angeles.

Kvarnström, J., Heintz, F., and Doherty, P. 2008. A temporal logic-based planning and execution monitoring system. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, J. Rintanen, B. Nebel, J. C. Beck, and E. A. Hansen, Eds. AAAI, 198–205.

Lee, J., Lifschitz, V., and Yang, F. 2013. Action language BC: Preliminary report. See Rossi (2013), 983–989.

Levesque, H., Reiter, R., Lespérance, Y., Lin, F., and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming 31,* 1-3, 59–83.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence 138,* 1-2, 39–54.

Lukasiewicz, J. 1941. Die logik und das grundlagenproblem. *Les Entreties de Zürich sur les Fondaments et la Méthode des Sciences Mathématiques 1938 12,* 6-9, 82–100.

Lutz, C., Wolter, F., and Zakharyaschev, M. 2008. Temporal description logics: A survey. In *15th International Symposium on Temporal Representation and Reasoning, TIME 2008, Université du Québec à Montréal, Canada, 16-18 June 2008*, S. Demri and C. S. Jensen, Eds. IEEE Computer Society, 3–14.

Matteo Baldoni, Laura Giordano, A. M. and Patti, V. 1996. An abductive proof procedure for reasoning about actions in modal logic programming. In *Proc. of Non-Monotonic Extensions on Logic Programming (NMELP'96).*

Mendez, G., Llopis, J., Lobo, J., and Baral, C. 1996. Temporal logic and reasoning about actions. In *Proc. of Common Sense 96.*

Mints, G. 2010. Cut-free formulations for a quantified logic of here and there. *Annals of Pure and Applied Logic 162,* 3, 237–242.

Moszkowski, B. 1986. *Executing Temporal Logic Programs.* Cambridge University Press.

Oikarinen, E. and Janhunen, T. 2006. Modular equivalence for normal logic programs. In

Proceedings of the Seventeenth European Conference on Artificial Intelligence (ECAI'06), G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, Eds. IOS Press, 412–416.

Orgun, M. A. and Wadge, W. W. 1992. Theory and practice of temporal logic programming. In *Intensional Logics for Programming*, L. F. del Cerro and M. Penttonen, Eds. Clarendon Press, Oxford, 21–50.

Pearce, D. 1997. A new logical characterisation of stable models and answer sets. In *Proceedings of the Sixth International Workshop on Non-Monotonic Extensions of Logic Programming (NMELP'96)*, J. Dix, L. Pereira, and T. Przymusinski, Eds. Lecture Notes in Computer Science, vol. 1216. Springer-Verlag, 57–70.

Pearce, D., Tompits, H., and Woltran, S. 2001. Encodings for equilibrium logic and logic programs with nested expressions. In *Proceedings of the Tenth Portuguese Conference on Artificial Intelligence (EPIA'01)*, P. Brazdil and A. Jorge, Eds. Lecture Notes in Computer Science, vol. 2258. Springer-Verlag, 306–320.

Pearce, D. and Valverde, A. 2006. Quantified equilibrium logic and the first order logic of here-and-there. Technical Report MA-06-02, University of Málaga.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the Eight-teenth Symposium on Foundations of Computer Science (FOCS'77)*. IEEE Computer Society Press, 46–57.

Rossi, F., Ed. 2013. *Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI'13)*. IJCAI/AAAI Press.

Schwind, C. B. 1997. A logic based framework for action theories. In *The Tbilisi Symposium on Logic, Language and Computation*, J. Ginzburg, Z. Khasidashvili, C. Vogel, J.-J. Lévy, and E. Vallduví, Eds. 275–291.

Simpson, A. K. 1994. The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D. thesis, University of Edinburgh.

Sistla, A. and Clarke, E. 1985. The complexity of propositional linear temporal logic. *Journal of the ACM 32,* 3, 733–749.

Sistla, A. P. 1983. Theoretical issues in the design and verification of distributed systems. Ph.D. thesis, Harvard University.

Sistla, A. P., Vardi, M. Y., and Wolper, P. 1987. The complementation problem for büchi automata with appplications to temporal logic. *Theoretical Computer Science 49*, 217–237.

van Dalen, D. 1986. Intuitionistic logic. In *Handbook of Philosophical Logic*. Vol. 166. Springer Netherlands, 225–339.

van Emden, M. and Kowalski, R. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM 23,* 4, 733–742.

Wolper, P. 1983. Temporal logic can be more expressive. *Information and Control 56,* 1/2, 72–99.

Zhu, S., Pu, G., and Vardi, M. Y. 2019. First-order vs. second-order encodings for LTLf-to-automata translation. In *Proceedings of the fifteenth Conference on Theory and Applications of Models of Computation*, T. V. Gopal and J. Watada, Eds. Springer, 684–705.

Zhu, S., Tabajara, L. M., Li, J., Pu, G., and Vardi, M. Y. 2017. Symbolic ltlf synthesis. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, C. Sierra, Ed. ijcai.org, 1362–1369.