TLP **18** (3–4): 406–420, 2018. © Cambridge University Press 2018
doi:10.1017/S1471068418000297

Temporal Answer Set Programming on Finite Traces

PEDRO CABALAR
University of Corunna, Spain

ROLAND KAMINSKI, TORSTEN SCHAUB and ANNA SCHUHMANN
University of Potsdam, Germany

submitted 30 April 2018; accepted 11 May 2018

Abstract

In this paper, we introduce an alternative approach to Temporal Answer Set Programming that relies on a variation of Temporal Equilibrium Logic (TEL) for finite traces. This approach allows us to even out the expressiveness of TEL over infinite traces with the computational capacity of (incremental) Answer Set Programming (ASP). Also, we argue that finite traces are more natural when reasoning about action and change. As a result, our approach is readily implementable via multi-shot ASP systems and benefits from an extension of ASP's full-fledged input language with temporal operators. This includes future as well as past operators whose combination offers a rich temporal modeling language. For computation, we identify the class of temporal logic programs and prove that it constitutes a normal form for our approach. Finally, we outline two implementations, a generic one and an extension of the ASP system *clingo*.

Under consideration for publication in Theory and Practice of Logic Programming (TPLP)

1 Introduction

Representing and reasoning about dynamic systems is a key problem in Artificial Intelligence and beyond. Accordingly, various formal systems have arisen, including temporal logics (Emerson 1990) and calculi for reasoning about actions and change (Sandewall 1994). In Answer Set Programming (ASP; Lifschitz 1999), this is reflected by temporal extensions of Equilibrium Logic (Aguado et al. 2013), the host logic of ASP, and action languages (Gelfond and Lifschitz 1998). Although both constitute the main directions of non-monotonic temporal systems, their prevalence lags way behind the usage of plain ASP for modeling dynamic domains. Hence, notwithstanding the meticulous modeling of dynamics in ASP due to an explicit representation of time points, it seems that its pragmatic advantages, such as its rich (static) modeling language and readily available solvers, often seem to outweigh the firm logical foundations of both dedicated approaches.

Although the true reasons are arguably inscrutable, let us discuss some possible causes. The appeal of action languages lies in their elegant syntactic and semantic simplicity: they usually consist of static and dynamic laws inducing a unique transition system. Although most of them are implemented in ASP, their simplicity denies the expressive possibilities of ASP. Also, despite some recent reconciliation (Lee et al. 2013), existing action languages lack the universality of ASP as reflected by the variety of variants.

Temporal Equilibrium Logic (TEL; Aguado et al. 2013) builds upon an extension of the logic of Here and There (HT; Heyting 1930) with Linear Temporal Logic (LTL; Pnueli

1977). This results in an expressive non-monotonic modal logic, which extends traditional temporal logic programming approaches (Cabalar et al. 2015) to the general syntax of LTL and possesses a computational complexity beyond LTL (Bozzelli and Pearce 2015). As in LTL, a model in TEL is an *infinite* sequence of states, called a trace. This rules out computation by ASP technology (and necessitates model checking) and is unnatural for applications like planning, where plans amount to finite prefixes of one or more traces.

Unlike this, we address the representation and reasoning about dynamic systems by proposing an alternative combination of the logics of HT and LTL whose semantics rests upon finite traces. On the one hand, this amounts to a restriction of TEL to finite traces. On the other hand, this is similar to the restriction of LTL to LTL_f advocated by De Giacomo and Vardi (2013). Our new approach, dubbed TEL_f, has the following advantages. First, it is readily implementable via ASP. Second, it can be reduced to a normal form which is close to logic programs and much less complex than the one obtained for TEL. Finally, its temporal models are finite and offer a one-to-one correspondence to plans. Interestingly, TEL_f also sheds light on concepts and methodology used in incremental ASP solving when understanding incremental parameters as time points.

Another distinctive feature of TEL_f is the inclusion of future as well as past temporal operators. We associate this with the following benefits. When using the causal reading of program rules, it is generally more natural to draw upon the past in rule bodies and to refer to the future in rule heads. A similar argument was put forward by Gabbay (1987). This format also yields a simpler normal form and lends itself to a systematic modeling methodology which favors the definition of states in terms of the past rather than mixing in future operators. For instance, in reasoning about actions, the idea is to derive action effects for the current state and check their preconditions in the previous one, rather than to represent this as a transition from the current to the next state. This methodology aligns state constraints, effect axioms, etc. to capture the present state. As well, past operators are much easier handled computationally than their future counterparts when it comes to incremental reasoning, since they refer to already computed knowledge.

We make the above arguments more precise once our formal apparatus is set up. In fact, we introduce our approach in a more general semantic setting encompassing not only TEL_f but also its close ancestors TEL, LTL, and LTL_f. This uniform base provides us with immediate insights into their interrelationships. Once we have formally elaborated our approach (in the restricted space), we turn to computational aspects. First, we define the class of temporal logic programs and show that they constitute a normal form for TEL_f. Then, we provide bounded and time point-wise translations of temporal logic programs into regular ones. Finally, we sketch two existing implementations: *tel*, computing bounded temporal models of TEL_f theories, and *telingo*, incrementally computing temporal models of temporal logic programs over the extended full-fledged input language of *clingo*.

2 Temporal Equilibrium Logic on Finite Traces

All logics treated in this paper share the common syntax of LTL with past operators (Emerson 1990). We start from a given set $\mathcal{A}$ of atoms which we call the *alphabet*. Then, a (temporal) *formula* φ is defined by the grammar:

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

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

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

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

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

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

As an example of a temporal formula, take for instance:

$$\Box(\mathit{shoot} \wedge \bullet\blacklozenge \mathit{shoot} \wedge \blacksquare \mathit{unloaded} \rightarrow \Diamond \mathit{fail}) \tag{1}$$

capturing the sentence: *"If we make two shots with a gun that was never loaded, then it will eventually fail."* Intuitively, '$\Box$' is used to assert that the rule is applicable at every time point. An explanation why shooting a loaded gun fails in unloading it, could then be queried as follows (where double negation allows us to express an integrity constraint).

$$\Box(\mathbb{F} \rightarrow \neg\neg(\mathit{shoot} \wedge \bullet \mathit{loaded} \wedge \mathit{loaded})) \tag{2}$$

For the semantics, we start by defining a *trace* of length λ over alphabet $\mathcal{A}$ as a sequence $\langle H_i \rangle_{i=0}^{\lambda}$ of sets $H_i \subseteq \mathcal{A}$. We say that the trace is *infinite* if $\lambda = \omega$ and *finite* otherwise, that is, $\lambda = n$ for some natural number $0 \leq n < \omega$. We let $i = j..k$ stand for $i \in \mathbb{N} \cup \{\omega\}$ and $j \leq i \leq k$. Given traces $\mathbf{H} = \langle H_i \rangle_{i=0}^{\lambda}$ and $\mathbf{H'} = \langle H'_i \rangle_{i=0}^{\lambda}$ both of length λ, we write $\mathbf{H} \leq \mathbf{H'}$ if $H_i \subseteq H'_i$ for each $i = 0..\lambda$; accordingly, $\mathbf{H} < \mathbf{H'}$ iff both $\mathbf{H} \leq \mathbf{H'}$ and $\mathbf{H} \neq \mathbf{H'}$.

A *Here-and-There trace* (for short HT-*trace*) of length λ over alphabet $\mathcal{A}$ is a sequence of pairs $\langle H_i, T_i \rangle_{i=0}^{\lambda}$ such that $H_i \subseteq T_i \subseteq \mathcal{A}$ for any $i = 0..\lambda$. As before, an HT-trace is infinite if $\lambda = \omega$ and finite otherwise. We often represent an HT-trace as a pair of traces $\langle \mathbf{H}, \mathbf{T} \rangle$ of length λ where $\mathbf{H} = \langle H_i \rangle_{i=0}^{\lambda}$ and $\mathbf{T} = \langle T_i \rangle_{i=0}^{\lambda}$ and $\mathbf{H} \leq \mathbf{T}$.

Definition 1 (*Satisfaction*)
An HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ of length λ over alphabet $\mathcal{A}$ *satisfies* a temporal formula φ at time point $k = 0..\lambda$, $k \neq \omega$, written $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$, if the following conditions hold:

1. $\langle \mathbf{H}, \mathbf{T} \rangle, k \not\models \bot$
2. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models a$ iff $a \in H_k$, for any atom $a \in \mathcal{A}$
3. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi \wedge \psi$ iff $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ and $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \psi$
4. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi \vee \psi$ iff $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ or $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \psi$
5. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi \rightarrow \psi$ 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. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \bullet\varphi$ iff $k > 0$ and $\langle \mathbf{H}, \mathbf{T} \rangle, k{-}1 \models \varphi$
7. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi\,\mathbf{S}\,\psi$ iff for some $j = 0..k$, we have $\langle \mathbf{H}, \mathbf{T} \rangle, j \models \psi$ and $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for all $i = j{+}1..k$
8. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi\,\mathbf{T}\,\psi$ iff for all $j = 0..k$, we have $\langle \mathbf{H}, \mathbf{T} \rangle, j \models \psi$ or $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for some $i = j{+}1..k$
9. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \circ\varphi$ iff $k < \lambda$ and $\langle \mathbf{H}, \mathbf{T} \rangle, k{+}1 \models \varphi$

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

10. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi \mathbb{U} \psi$ iff for some $j = k..\lambda$, we have $\langle \mathbf{H}, \mathbf{T} \rangle, j \models \psi$ and $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for all $i = k..j{-}1$
11. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi \mathbb{R} \psi$ iff for all $j = k..\lambda$, we have $\langle \mathbf{H}, \mathbf{T} \rangle, j \models \psi$ or $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for some $i = k..j{-}1$. ⊠

A formula φ is a *tautology*, written $\models \varphi$, iff $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ for any HT-trace and any $k = 0..\lambda$. We call the logic induced by the set of all tautologies *Temporal logic of Here and There* (THT for short). We say that an HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle$ is a *model* of a set of formulas (or *theory*) Γ iff $\langle \mathbf{H}, \mathbf{T} \rangle, 0 \models \varphi$ for any $\varphi \in \Gamma$.

When compared to standard temporal logics, the main peculiarity of Definition 1 is the satisfaction of implication $\varphi \rightarrow \psi$ that requires that both (i) $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ implies $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \psi$ and (ii) $\langle \mathbf{T}, \mathbf{T} \rangle, k \models \varphi$ implies $\langle \mathbf{T}, \mathbf{T} \rangle, k \models \psi$. This interpretation of '$\rightarrow$' is inherited from the (non-temporal) logic of Here and There (HT; Heyting 1930), an intermediate logic dealing with exactly two worlds $\{h, t\}$ with the accessibility relation $h \leq t$ plus the reflexive closure $h \leq h$ and $t \leq t$. This logic is weaker than classical logic and does not satisfy, among others, some classical tautologies such as the law of the *Excluded Middle* $\varphi \vee \neg\varphi$. A particular type of HT-traces are the ones of form $\langle \mathbf{T}, \mathbf{T} \rangle$ (that is, $\mathbf{H} = \mathbf{T}$) which we call *total*. Total models can be forced by adding the following variant of the excluded middle axiom schema:

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

Under total models, implication collapses to material implication and THT satisfaction $\langle \mathbf{T}, \mathbf{T} \rangle, k \models \varphi$ collapses to $\mathbf{T}, k \models \varphi$ in LTL (for possibly infinite traces). This implies that all THT tautologies are LTL tautologies but not vice versa, e.g. (EM).

Another important remark is that the finiteness of $\langle \mathbf{H}, \mathbf{T} \rangle$ only affects the last three items of Definition 1 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 $\mathbb{U}$ and $\mathbb{R}$ (the last two items) j ranges in the finite interval $\{k, \ldots, n\}$. Besides, if $\lambda = n$ the satisfaction of $\bigcirc\varphi$ forces $k < n$ so that it implies that *there does exist a next state* $k{+}1$. As a result, the formula $\bigcirc\top$ is not always satisfied, since it is false when $k = n = \lambda$. On the other hand, when $\lambda = \omega$, time points j in the semantics for $\mathbb{U}$ and $\mathbb{R}$ are just required to be $j \geq k$ without an upper limit. Similarly, the condition $k < \lambda = \omega$ in the satisfaction of $\bigcirc\varphi$ becomes obviously true, and so irrelevant (a state $k{+}1$ is always granted).

The semantics for derived operators can also be easily deduced:

12. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \top$
13. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \blacksquare\varphi$ iff $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for all $i = 0..k$
14. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \blacklozenge\varphi$ iff $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for some $i = 0..k$
15. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \mathbf{I}$ iff $k = 0$
16. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \widehat{\bullet}\varphi$ iff $k = 0$ or $\langle \mathbf{H}, \mathbf{T} \rangle, k{-}1 \models \varphi$
17. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \Box\varphi$ iff $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for any $i = k..\lambda$
18. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \Diamond\varphi$ iff $\langle \mathbf{H}, \mathbf{T} \rangle, i \models \varphi$ for some $i = k..\lambda$
19. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \mathbb{F}$ iff $k = \lambda$
20. $\langle \mathbf{H}, \mathbf{T} \rangle, k \models \widehat{\bigcirc}\varphi$ iff $k = \lambda$ or $\langle \mathbf{H}, \mathbf{T} \rangle, k{+}1 \models \varphi$

We see that operators $\mathbf{I}$ and $\mathbb{F}$ exclusively depend on the value of time point k, so that the valuation for atoms from $\langle \mathbf{H}, \mathbf{T} \rangle$ becomes irrelevant for them. As a result, they behave "classically" and satisfy the excluded middle, that is, $\models \mathbf{I} \vee \neg\mathbf{I}$ and $\models \mathbb{F} \vee \neg\mathbb{F}$ are THT

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

tautologies. Besides, operator $\mathbb{F}$ has the additional peculiarity that it can only be true with finite traces (remember that $k \neq \omega$ in Def. 1). This implies that the inclusion of axiom $\Diamond\mathbb{F}$ in any theory forces its models to be finite traces, while including its negation $\neg\Diamond\mathbb{F}$ instead causes the opposite effect, that is, all models of the theory are infinite traces.

In this paper, we consider several logics that are stronger than THT and that can be obtained by the addition of axioms (or the corresponding restriction on sets of traces). For instance, we define THT_ω as $\text{THT} + \{\neg\Diamond\mathbb{F}\}$, that is, THT where we exclusively consider infinite HT-traces.[1] The finite-trace version, we call THT_f, corresponds to $\text{THT} + \{\Diamond\mathbb{F}\}$ instead. Linear Temporal Logic for possibly infinite traces, LTL, can be obtained as $\text{THT} + \{(\text{EM})\}$, that is, THT with total HT-traces. Accordingly, we can define LTL_ω as $\text{THT}_\omega + \{(\text{EM})\}$, i.e. infinite and total HT-traces, and obtain LTL_f as $\text{THT}_f + \{(\text{EM})\}$, that is, LTL on finite traces (De Giacomo and Vardi 2013).

In the rest of the paper, we study several transformations preserving some kind of THT-equivalence. In this sense, it is important to observe that being equivalent is something generally stronger than simply having the same set of models. We say that two formulas φ, ψ are (globally) *equivalent*, written $\varphi \equiv \psi$, iff $\models \varphi \leftrightarrow \psi$, that is, $\langle \mathbf{H}, \mathbf{T}\rangle, k \models \varphi \leftrightarrow \psi$ for any HT-trace $\langle \mathbf{H}, \mathbf{T}\rangle$ of length λ and any $k = 0..\lambda$, $k \neq \omega$. Whenever φ and ψ are equivalent, they are completely interchangeable when occurring 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, that is, $\langle \mathbf{H}, \mathbf{T}\rangle, 0 \models \varphi$ iff $\langle \mathbf{H}, \mathbf{T}\rangle, 0 \models \psi$, for any HT-trace $\langle \mathbf{H}, \mathbf{T}\rangle$. Obviously, $\varphi \equiv \psi$ implies $\varphi \equiv_0 \psi$ but not vice versa. To put a simple 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. The following are some generally useful properties satisfied in THT.

Proposition 1 (*Persistence*)
Let $\langle \mathbf{H}, \mathbf{T}\rangle$ be an HT-trace of length λ and φ be a temporal formula. Then, for any $k = 0..\lambda$, $k \neq \omega$, 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. As said before, all THT tautologies are LTL tautologies but not vice versa. However, they coincide for some types of equivalences, as stated below.

Proposition 2
Let φ and ψ be 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:

$$\varphi \mathbb{U} \psi \equiv \psi \vee (\varphi \wedge \circ(\varphi \mathbb{U} \psi)) \tag{3}$$

is also valid in THT due to Proposition 2. In fact, we can exploit this result further. By De Morgan laws, LTL satisfies a kind of duality guaranteeing, for instance, that (3) iff:

$$\varphi \mathbb{R} \psi \equiv \psi \wedge (\varphi \vee \widehat{\circ}(\varphi \mathbb{R} \psi)) \tag{4}$$

and, by Proposition 2 again, this is also a valid equivalence in THT. Let us define all the pairs of dual connectives as follows: $\wedge/\vee$, $\top/\bot$, $\mathbb{U}/\mathbb{R}$, $\circ/\widehat{\circ}$, $\Box/\Diamond$, $\mathbf{S}/\mathbf{T}$, $\bullet/\widehat{\bullet}$, $\blacksquare/\blacklozenge$. For a

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

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

formula φ without implications, we define $\delta(\varphi)$ as the result of replacing each connective by its dual operator. Then, we get the following corollary of Proposition 2.

Corollary 1 (*Boolean Duality*)
Let φ and ψ be formulas without implication. Then, THT satisfies: $\varphi \equiv \psi$ iff $\delta(\varphi) \equiv \delta(\psi)$.

In a similar manner, we can also exploit the temporal symmetry in the system so we can switch the temporal direction of operators to conclude, for instance, that (3) iff $\varphi \,\mathbf{S}\, \psi \equiv \psi \vee (\varphi \wedge \bullet(\varphi \,\mathbf{S}\, \psi))$. This second type of duality, however, has some obvious limitations when we allow for infinite traces. In that case, for instance, the past has a beginning $\blacklozenge\mathbf{I} \equiv \top$ but the future may have no end $\Diamond\mathbb{F} \not\equiv \top$. If we restrict ourselves to finite traces, we get the following result. Let $\mathbb{U}/\mathbf{S}$, $\mathbb{R}/\mathbf{T}$, $\circ/\bullet$, $\widehat{\circ}/\widehat{\bullet}$, $\Box/\blacksquare$, 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 1
There exists a mapping ϱ on finite THT-traces of the same length $n \geq 0$ such that for any $k = 0..n$, $\langle \mathbf{H}, \mathbf{T}\rangle, k \models \varphi$ if and only if $\varrho(\langle \mathbf{H}, \mathbf{T}\rangle), n-k \models \sigma(\varphi)$.

Theorem 1 (*Temporal Duality Theorem*)
A temporal formula φ is a THT_f-tautology if and only if $\sigma(\varphi)$ is a THT_f-tautology.

Before introducing non-monotonicity, we begin by providing notation for representing sets of THT models. We write $\mathrm{THT}(\Gamma, \lambda)$ to stand for the set of THT models of length λ of a theory Γ, and define $\mathrm{THT}(\Gamma) \stackrel{def}{=} \bigcup_{\lambda=0}^{\omega} \mathrm{THT}(\Gamma, \lambda)$, that is, the whole set of models of Γ of any length. Given a set of THT models, we define the ones in equilibrium as follows.

Definition 2 (*Temporal Equilibrium 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 $\langle \mathbf{H}, \mathbf{T}\rangle \in \mathfrak{S}$. ⊠

If this is the case, we also say that trace $\mathbf{T}$ is a *temporal stable 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.

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

Proposition 3
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 or finite traces, respectively.

As an example of non-monotonicity, consider the formula

$$\Box(\bullet \mathit{loaded} \wedge \neg \mathit{unloaded} \rightarrow \mathit{loaded}) \tag{5}$$

along with literal *loaded* which combines the inertia for *loaded* with the initial state for that fluent. Without entering into much detail, this formula behaves as the logic program consisting of fact `loaded(0)` and rule '`loaded(T) :- loaded(T-1), not unloaded(T)`'

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

for any time point `T>0`. As expected, for some fixed λ, we get a unique temporal stable model of the form $\mathbf{T}$ such that $T_i = \{loaded\}$ for $i = 0..\lambda$, as there is no reason to become *unloaded*. 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 (5) with $loaded \wedge \bigcirc\bigcirc unloaded$ which amounts to adding the fact `unloaded(2)`. As expected, for each λ, the only temporal stable model now is $T_0 = T_1 = \{loaded\}$, $T_2 = \{unloaded\}$ and $T_i = \emptyset$ for $i = 3..\lambda$. Note that by making $\bigcirc\bigcirc unloaded$ true we are also forcing $\lambda \geq 3$, that is, there are no temporal stable models (nor even THT models) of length smaller than three. Thus, by adding the new information $\bigcirc\bigcirc unloaded$ some conclusions that could be derived before, such as $\Box loaded$, are not derived any more.

As an example emphasizing the behavior of finite traces, take the formula $\Box(\neg a \rightarrow \bigcirc a)$ 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 where the final state λ makes a false, since the rule would force $\bigcirc a$ and this implies the existence of a successor state. As a result, the temporal stable models of this formula have the (regular expression) form $(\emptyset\ \{a\})^+$ for finite traces in TEL_f, or the infinite trace $(\emptyset\ \{a\})^\omega$ in TEL_ω.

Another interesting example is the temporal formula $\Box(\neg\bigcirc a \rightarrow a) \wedge \Box(\bigcirc a \rightarrow a)$. 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 temporal stable model, but in TEL_f, we can use the fact that $\bigcirc a$ is always false in the last state. Then, $\Box(\neg\bigcirc a \rightarrow a)$ supports a in that state and therewith $\Box(\bigcirc a \rightarrow a)$ inductively supports a everywhere.

As an example of a temporal expression not so close to logic programming consider, for instance, the formula $\Box\Diamond a$, 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 $\Box(\mathbb{F} \rightarrow a)$ in LTL_f, that is, a is true at the final state (and 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 for a fixed $\lambda \in \mathbb{N}$, we obtain a unique temporal stable model where a is true at the last state, and false everywhere else, whereas 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 $\mathbf{H}$ where we remove a at one 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 $\Box\Diamond a$ there always exists some model $\langle \mathbf{H}, \mathbf{T} \rangle$ with strictly smaller $\mathbf{H} < \mathbf{T}$.

3 Temporal Logic Programs

Our computational approach to TEL_f relies on a reduction to a normal form suitable for ASP systems. For this, we identified the class of temporal logic programs defined next.

Definition 3 (*Temporal literal, rule, and program*)
Given alphabet $\mathcal{A}$, we define the set of *temporal literals* as $\{a, \neg a, \bullet a, \neg\bullet a \mid a \in \mathcal{A}\}$.

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

A *temporal rule* is either:

- an *initial rule* of the form $B \not\rightarrow A$
- a *dynamic rule* of the form $\widehat{\circ}\,\Box(B \not\rightarrow A)$
- a *final rule* of the form $\Box(\,\mathbb{F} \rightarrow (B \not\rightarrow A)\,)$

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, or regular literals $\{a, \neg a \mid a \in \mathcal{A}\}$ for initial and final rules. A *temporal logic program* (TEL$_f$ *program*, for short) is a set of temporal rules.

We let $I(P)$, $D(P)$, and $F(P)$ stand for the set of all initial, dynamic, and final rules in a TEL$_f$ program P, respectively. A TEL$_f$ program just consisting of initial rules amounts to a regular logic program. Dynamic rules are preceded by the weak version of next $\widehat{\circ}$ rather than $\circ$ since we deal with finite traces and the final state has no subsequent state.

Our earlier examples in (2) and (5) are already close to TEL$_f$ programs, and a minor transformation yields the following temporal rules equivalent to (2) and (5), respectively.

$$\widehat{\circ}\,\Box(\mathit{shoot} \wedge \bullet \mathit{loaded} \wedge \mathit{loaded} \rightarrow \mathit{goal}) \wedge \Box(\mathbb{F} \rightarrow (\neg \mathit{goal} \rightarrow \bot))$$
$$\widehat{\circ}\,\Box(\bullet \mathit{loaded} \wedge \neg \mathit{unloaded} \rightarrow \mathit{loaded})$$

Note that no initial rules are needed since $\bullet \mathit{loaded}$ is false at the initial time point, and *goal* is a new auxiliary atom. In the remainder, for illustration purposes, we use the simple TEL$_f$ program P:

$$\{\ \rightarrow a, \quad \widehat{\circ}\,\Box(\bullet a \rightarrow b), \quad \Box(\mathbb{F} \rightarrow (\neg b \rightarrow \bot))\ \} \tag{6}$$

which has a single finite temporal stable model of length 1, viz. $\langle \{a\}, \{b\} \rangle$.

The following result warrants that TEL$_f$ programs constitute indeed a normal form.

Theorem 2 (*Normal form*)
Every temporal formula φ can be converted into a TEL$_f$ program THT$_f$-equivalent to φ.

For transforming arbitrary temporal formulas into normal form, we use a Tseitin-style reduction (1968) that relies on an alphabet extended by new atoms for each formula in the original language. The equivalence result in Theorem 2 is then obtained after removing auxiliary atoms and, in fact, is still preserved inside the context of a larger theory for the original vocabulary (i.e. we have *strong equivalence* modulo auxiliary atoms).

Now, given a TEL$_f$ program P and a fixed (finite) λ, we can compute all models in TEL(P, λ) by a 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 $k = 0..\lambda$.

Definition 4 (*Bounded translation*)
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 3 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

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

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

Note that the translation of temporal rules is similar in just considering the implication $B \rightarrow A$ in Definition 3; their difference manifests itself in their instantiation in $\tau_\lambda(P)$.

Applying translation τ for some bound λ to our TEL$_f$ program P in (6) yields regular logic programs of the following form.

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

Program $\tau_1(P)$ has the stable model $\{a_0, b_1\}$, but all $\tau_\lambda(P)$ for $\lambda > 1$ are unsatisfiable.

Theorem 3

Let P be a TEL$_f$ program over $\mathcal{A}$. Let $\mathbf{T} = \langle T_i \rangle_{i=0}^{\lambda}$ be a trace of finite length λ over $\mathcal{A}$ and X a set of atoms over $\bigcup_{0 \leq i \leq \lambda} \mathcal{A}_i$ such that $a \in T_i$ iff $a_i \in X$ for $0 \leq i \leq \lambda$.

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 $\langle \{a\}, \{b\} \rangle$ of P corresponds to the stable model $\{a_0, b_1\}$ of $\tau_2(P)$.

Using this translation we have implemented a system, *tel*[2], that takes a propositional theory Γ of arbitrary TEL$_f$ 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 (6) 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 1 to *clingo* yields the stable model containing `a(0)` and `b(1)` (suppressing auxiliary atoms).

The bounded translation $\tau_\lambda(P)$ allows us to compute all models in TEL(P, λ) 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 TEL(P, k) that is easily obtained from that of 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 TEL$_f$ programs. Still, we identify a quite general syntactic fragment[3] that implies compositionality. We say that a temporal rule as in Definition 3 is *present-centered*, whenever all the literals $a_1, \ldots, a_m$ in its head A are regular. Accordingly, a set of such rules is a present-centered TEL$_f$ program. In fact, such programs are sufficient to capture common action languages, as witnessed by the correspondence between dynamic

[2] https://github.com/potassco/tel

[3] In order to compute loop formulas for TEL$_\omega$, (Cabalar and Diéguez 2011) used a similar fragment (*splittable programs*) where rules cannot derive information from the future to the past.

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

temporal rules and static and dynamic laws in action language $\mathcal{BC}$ (Lee et al. 2013):[4]

$$\begin{array}{ll} a \textbf{ if } b_1,\ldots,b_m \textbf{ ifcons } c_1,\ldots,c_n & \widehat{\circ}\,\Box(b_1 \wedge \cdots \wedge b_m \rightarrow a \vee \neg c_1 \vee \cdots \vee \neg c_n) \\ & \quad \wedge\ (b_1 \wedge \cdots \wedge b_m \rightarrow a \vee \neg c_1 \vee \cdots \vee \neg c_n) \\ a \textbf{ after } b_1,\ldots,b_m \textbf{ ifcons } c_1,\ldots,c_n & \widehat{\circ}\,\Box(\bullet b_1 \wedge \cdots \wedge \bullet b_m \rightarrow a \vee \neg c_1 \vee \cdots \vee \neg c_n) \end{array}$$

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 TEL$_f$ 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 (i) $I \cap O = \emptyset$, (ii) $\mathcal{A}_P \subseteq I \cup O$, and (iii) $H(P) \subseteq O$, where $H(P)$ gives all atoms occurring in rule heads in P. Whenever clear from context, 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.[5] 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 5 (*Point-wise translation: Temporal rules*)
We define the translation of a final rule r as in Definition 3 at time point k as

$$\tau_k^*(r) \stackrel{def}{=} \tau_k(a_1) \vee \cdots \vee \tau_k(a_m) \leftarrow \tau_k(b_1) \wedge \cdots \wedge \tau_k(b_n) \wedge \neg q_{k+1} \tag{7}$$

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 (7) are used to deactivate instances of final rules. This allows us to implement operator $\mathbb{F}$ 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 $q_1, \ldots, q_k$ are true. In this way, only $\tau_k^*(r)$ is potentially applicable, while all rules $\tau_i^*(r)$ are disabled at earlier time points $i = 1..k{-}1$.

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

Definition 6 (*Point-wise translation: Modules*)
Let P be a present-centered TEL$_f$ 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

$$\begin{aligned} P_0 &\stackrel{def}{=} \{\tau_0^*(r) \mid r \in\ I(P)\} \cup \{\tau_0^*(r) \mid r \in F(P)\} \\ P_k &\stackrel{def}{=} \{\tau_k^*(r) \mid r \in D(P)\} \cup \{\tau_k^*(r) \mid r \in F(P)\} \cup \{q_k \leftarrow\} \end{aligned}$$

[4] In $\mathcal{BC}$, **ifcons** stands for "if consistent", while **if** and **after** have their literal meaning.
[5] Note that the default value assigned to input atoms is *false* in multi-shot solving (Gebser et al. 2018); this differs from the original definition (Oikarinen and Janhunen 2006) where a choice rule is used.

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

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.

The point-wise translation of our present-centered example program P from (6) yields the following modules.

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

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

Theorem 4
Let P be a present-centered TEL$_f$ program over $\mathcal{A}$. Let $\mathbf{T} = \langle T_i \rangle_{i=0}^{\lambda}$ be a trace of finite length λ over $\mathcal{A}$ and X a set of atoms over $\bigcup_{0 \leq i \leq \lambda} \mathcal{A}_i$ such that $a \in T_i$ iff $a_i \in X$ for $0 \leq i \leq \lambda$. Then,

$\mathbf{T}$ is a temporal stable model of P iff $X \cup \{q_i \mid i = 1..\lambda\}$ is a stable model of $\bigsqcup_{i=0}^{\lambda} \mathbb{P}_i$.

As with Theorem 3, this result confirms that the temporal stable model $\langle \{a\}, \{b\} \rangle$ 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 TEL$_f$ theory is reducible to a present-centered TEL$_f$ program. Hence, computing models via incremental solving imposes some limitations on the possible combinations of temporal operators. Fortunately, we can identify 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 3 where B and A are just temporal formulas with the following restrictions: B and A contain no implications other than negations ($\alpha \rightarrow \bot$), B contains no future operators, and A contains no past operators. An example of a past-future rule is (1). Then, we have the following result.

Theorem 5 (*Past-future reduction*)
Every past-future rule r can be converted into a present-centered TEL$_f$ program that is TEL$_f$-equivalent to r.

We have implemented a second system, *telingo*[6], that deals with present-centered TEL$_f$ 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 2 are compiled away by means of the

[6] https://github.com/potassco/telingo

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

translation used in its proof. The resulting present-centered TEL$_f$ 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. 2018), 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 3 and interpreted as r, $\widehat{\circ}\Box r$, or $\Box(\mathbb{F} \rightarrow r)$ 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 TEL$_f$ program in (6) in the two alternative ways shown in Table 1.

Table 1. *Two alternative telingo encodings for the* TEL$_f$ *program in* (6)

```
#program initial.
a.

#program dynamic.
b :- 'a.

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

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


b :- 'a.


  :- not b, &final.
```

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 $\circ$, $\mathbb{U}$, $\mathbb{R}$. Accordingly, `<*/1`, `<?/1`, `<:/1` represent $\blacksquare$, $\blacklozenge$, $\widehat{\bullet}$, and analogously their future counterparts. $\mathbf{I}$ and $\mathbb{F}$ are are represented by `&initial` and `&final`. This is complemented by Boolean connectives `&`, `|`, `~`, etc. For example, the integrity constraint '$\mathit{shoot} \wedge \blacksquare \mathit{unloaded} \wedge \bullet\blacklozenge \mathit{shoot} \rightarrow \bot$' is expressed as follows.

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

Once *telingo* has translated an (extended) TEL$_f$ program into a regular one, it is incrementally solved by *clingo*'s multi-shot solving engine (Gebser et al. 2018).

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

4 Discussion and conclusions

For incorporating temporal representation and reasoning into existing ASP solving technology, we introduced a variant of Temporal Equilibrium Logic, TEL_f, that deals with finite traces, something better aligned with incremental ASP solving for dynamic domains. The original version of this logic, TEL_ω (Aguado et al. 2013), was exclusively thought for infinite traces and, accordingly, its computation (Cabalar and Diéguez 2011) is done in terms of automata obtained by a model checker. This strategy is more adequate for checking properties of reactive systems but not so convenient when looking for minimal plans, performing temporal explanation, or even for diagnosis on finite executions. To analyze which logical properties may vary depending on the finiteness assumption, we defined a more general (and weaker) version of TEL and its monotonic basis THT, which accepts both finite and infinite traces. This general TEL acts as an umbrella to study the relation of the new finite trace variants, TEL_f and THT_f, with their temporal predecessors TEL_ω, LTL_ω, LTL_f as well as HT and its equilibria. For instance, we may conclude that satisfiability for both THT_f and TEL_f are PSPACE-hard, since LTL_f, proved to be PSPACE-complete (De Giacomo and Vardi 2013), can be easily reduced to THT_f or TEL_f by adding the excluded middle axiom (EM). In fact, THT_f is also PSPACE-complete: its membership can be proved by encoding THT_f into LTL_f as in the reduction from THT_ω to LTL_ω made in (Cabalar and Demri 2011). In the case of TEL_f satisfiability, we conjecture its EXPSPACE membership by using a translation into TEL_ω, which is EXPSPACE-complete (Bozzelli and Pearce 2015), similar to the one from LTL_f into LTL_ω in (De Giacomo and Vardi 2013). A detailed complexity analysis remains future work.

As with TEL_ω (Cabalar 2010), we proved that TEL_f can be reduced to a normal form close to logic programs. Moreover, the one for TEL_f happens to be significantly simpler, since it does not need to resort to nested global operators. We developed two translations of this normal form into ASP: (i) one to obtain temporal stable models of fixed length; and (ii), another based on the composition of logic program modules, allowing for incremental computation. These translations gave rise to two different systems. Our first system, *tel*, accepts an arbitrary propositional TEL_f-theory and a bound and then reduces it to normal form followed by translation (i) into ASP. This allows us to harness the full expressiveness of a temporal language while using any off-the-shelf ASP system. Our second system, *telingo*, extends the ASP system *clingo* to compute the temporal stable models of (non-ground) temporal logic programs. To this end, it extends the full-fledged input language of *clingo* with temporal operators and computes temporal models incrementally by multi-shot solving (Gebser et al. 2018) using translation (ii) into ASP. It is also interesting to observe that TEL_f sheds light on existing concepts used in incremental ASP solving, when interpreting increments as time-points. For instance, operator $\mathbb{F}$ naturally corresponds to the so-called "external query atom" (Gebser et al. 2018) used for progressing goal conditions, while the syntactic form of present-centered programs reflects the modeling methodology (Gebser et al. 2012) put forward for incremental ASP solving that avoids "future atoms" referring to time point `T+1` in rule heads.

All in all, TEL_f offers an expressive, semantically well founded language for modeling dynamic systems in ASP that allows for exploiting existing solving technology and, at the same time, enables a fully logical analysis of temporal properties, either from

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press

plain ASP specifications or from action languages that can be naturally translated into TEL_f. For future work, we plan to investigate some topics already studied for TEL_ω in the case of finite traces, such as characterizing strong equivalence, checking unsatisfiability by automata-based methods, or improving the efficiency of grounding for temporal programs.

Acknowledgments. This work was partially supported by MINECO, Spain, grant TIC2017-84453-P, Xunta de Galicia, Spain (GPC ED431B 2016/035 and 2016-2019 ED431G/01, CITIC), and DFG grant SCHA 550/9.

References

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.

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.

Cabalar, P. 2010. A normal form for linear temporal equilibrium logic. In *Proceedings of the Twelfth European Conference on Logics in Artificial Intelligence (JELIA'10)*, T. Janhunen and I. Niemelä, Eds. Lecture Notes in Artificial Intelligence, vol. 6341. Springer-Verlag, 64–76.

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., Diéguez, M., and Vidal, C. 2015. An infinitary encoding of temporal equilibrium logic. *Theory and Practice of Logic Programming 15,* 4-5, 666–680.

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

Emerson, E. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. MIT Press, 995–1072.

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.

Gabbay, D. 1987. 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.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. 2018. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*. To appear.

Gebser, M., Kaminski, R., and Schaub, T. 2012. Gearing up for effective ASP planning. In *Correct Reasoning: Essays on Logic-Based AI in Honour of Vladimir Lifschitz*, E. Erdem, J. Lee, Y. Lierler, and D. Pearce, Eds. Lecture Notes in Computer Science, vol. 7265. Springer-Verlag, 296–310.

Gelfond, M. and Lifschitz, V. 1998. Action languages. *Electronic Transactions on Artificial Intelligence 3,* 6, 193–210.

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.

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

Lifschitz, V. 1999. Answer set planning. In *Proceedings of the International Conference on Logic Programming (ICLP'99)*, D. de Schreye, Ed. MIT Press, 23–37.

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.

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.

Sandewall, E. 1994. *Features and fluents: the representation of knowledge about dynamical systems*. Vol. 1. Oxford University Press, New York, NY, USA.

Tseitin, G. 1968. On the complexity of derivation in the propositional calculus. *Zapiski nauchnykh seminarov LOMI 8*, 234–259.

https://doi.org/10.1017/S1471068418000297 Published online by Cambridge University Press