Computational Aspects of Progression for Temporal Equilibrium Logic

Thomas Eiter and **Davide Soldà**
Institute for Logic and Computation, TU Wien, Favoritenstraße 9–11, 1040 Vienna, Austria,
eiter@kr.tuwien.ac.at, davide.solda@tuwien.ac.at

Abstract

Temporal logic plays a crucial role in specifying and reasoning about dynamic systems, where temporal constraints and properties to be monitored are essential. Traditional approaches like LTL-monitoring assume monotonicity, which limits their applicability to scenarios involving non-monotonic temporal properties. We delve into complexity aspects of monitoring temporal specifications using non-monotonic Temporal Equilibrium Logic (TEL), a temporal extension of Answer Set Programming defined over Temporal Here and There Logic (THT) with a minimality criterion enforcing stable models. Notably, we study the complexity gap between monitoring properties in THT and TEL semantics, and the complexity of monitoring approximations based on progression, which is widely used in verification and in AI. In that, we pay particular attention to the fragment of temporal logic programs.

1 Introduction

Reasoning about dynamic systems and dealing with temporal data appropriately is an important issue. To this end, a range of temporal logics such as LTL, *CTL*, *LDL* etc. has been developed that allows one to specify and assess the behavior of systems with automated support. Monitoring temporal constraints and properties is an essential task that provides input for decision-making, diagnostics, prediction, and many other tasks. In that, data becomes available in a growing stream, requiring that reasoning proceeds in an online fashion. To this end, monitoring approaches for LTL have been developed, in which a 3-valued semantics is employed [Bauer *et al.*, 2007] that approximates the LTL semantics, by informally telling whether a property is established true, false, or remains open.

As with many logics, LTL is monotonic, which limits its applicability when it comes to model systems involving properties and features such as exceptions, dealing with incomplete information, or belief states. Nonmonotonic logics have been conceived to address such aspects. They e.g. allow for expressing defaults and normative behavior [Cabalar *et al.*, 2023], offer a solution to the frame problem [McCarthy and Hayes, 1981; Kautz, 1986] by concise modeling of inertia, and are more amenable to *elaboration tolerance* [McCarthy, 1998] for developing logical representations. In particular, *Equilibrium Logic* [Pearce, 2006], which is the logical basis of Answer Set Programming (ASP), caters to this possibility.

In ASP, temporal reasoning is usually expressed by encoding time in the language, which has disadvantages compared to handling time as a first-class citizen. This is done in *Temporal Equilibrium Logic* (TEL) [Aguado *et al.*, 2013], which combines Equilibrium Logic with LTL by imposing a stability condition on LTL models resorting to *Temporal Here-and-There Logic* (THT) [Aguado *et al.*, 2013]. Rule-based fragments of TEL like *Temporal ASP (TASP)* [Aguado *et al.*, 2023] allow one to express problems beyond LTL (e.g., conformant planning) [Bozzelli and Pearce, 2016]. The interest in TEL has been increasing in the recent years, fueled by the availability of native solvers such as `telingo` [Cabalar *et al.*, 2019].

As of today, support for online temporal reasoning in ASP is immature. ASP Streaming engines such as `oclingo` [Cerexhe *et al.*, 2014], `ticker` [Beck *et al.*, 2017], `i-dlv-sr` [Calimeri *et al.*, 2021] use incremental ASP evaluation, but support only limited fragments of TASP and are like `telingo` bound to finite traces. `MeTeoR` [Wang *et al.*, 2022; Wałęga *et al.*, 2023a] uses interval semantics and does not address monitoring.

Recent work has addressed this issue with progression for temporal ASP [Soldà *et al.*, 2023]. Progression is a well-known technique by which formulas are partially evaluated along temporal states. It has been widely used in logic-based AI, e.g., in planning [Bacchus and Kabanza, 1998], reasoning about actions [Giacomo *et al.*, 2016], or stream reasoning [de Leng and Heintz, 2018]. Notably, the logical approach to incremental TEL allows one to verify at runtime whether a trace satisfies a TEL specification, and to integrate observations as facts that need no justification. While [Soldà *et al.*, 2023] presented the approach and an algorithm for $TASP$ progression, computational aspects such as complexity and tractability, which in an online setting is desired, were not addressed. We fill this gap with the following contributions:

- We extend the language for progression to full THT and TEL, and we generalize the 3-valued semantics and progression operators P and P_{TEL} in [Soldà *et al.*, 2023] for THT and TEL, respectively. Furthermore, we provide a novel incremental version P^{inc}_{TEL} of TEL progression, which is better suited for online evaluation.
- We characterize the complexity of monitoring under TEL_3

and THT_3 semantics, showing that they are like TEL and THT EXPSPACE-complete and PSPACE-complete, respectively, in general [Bozzelli and Pearce, 2015]; we remind that LTL is PSPACE-complete. In our analysis, we also address observations provided as facts in an online manner; notably, they do not lead to a complexity increase in general.

• We show that THT progression with P is feasible in polynomial time, while TEL progression with P_{TEL} (and likewise P^{inc}_{TEL}) is mildly intractable, viz. D^p-complete. Since stable model checking of (atemporal) ASP programs is co-NP-complete [Eiter and Gottlob, 1995; Pearce, 2006], this is close to what we optimally could expect. The derivation employs suitable structures for storing progression formulas that can be incrementally maintained. We then present normal and headcycle-free temporal ASP programs as tractable classes of TASP for TEL progression with P_{TEL}, while unrestricted TASP harnesses the full complexity of THT resp. TEL.

Our results show that P_{TEL} and P progression are (nearly) tractable approximations of the TEL and THT semantics and their 3-valued versions. Furthermore, P^{inc}_{TEL} is a promising basis for practical implementation, as the syntactic restrictions allows one to express action domains and goals to be achieved.

2 Preliminaries

Both TEL and THT over infinite traces [Aguado *et al.*, 2013] share the same syntax. We are interested in a fragment $\mathcal{L}$ of LTL with past-time operators, generated by the grammar

$$\begin{aligned} F &::= \bot \mid p \mid F \circ F \mid \mathbf{X}F \mid F\,\mathbf{R}\,F \mid F\,\mathbf{U}\,F \mid \mathbf{Y}\,F' \\ F' &::= \bot \mid p \mid \mathbf{Y}\,F' \end{aligned} \tag{1}$$

where $p \in \mathcal{P}$ for a finite set $\mathcal{P}$ of propositional atoms and $\circ \in \{\wedge, \vee, \rightarrow\}$. Negation is defined as $\neg\phi \equiv \phi \rightarrow \bot$ and $\top \equiv \neg\bot$. As usual, $\mathbf{G}$ (globally) is defined by $\mathbf{G}\phi \equiv \bot\mathbf{R}\phi$ and $\mathbf{F}$ (finally) by $\mathbf{F}\phi \equiv \top\mathbf{U}\phi$.

The semantics of THT is defined via sequences of pairs of sets of atoms. A THT-trace (or simply trace, if clear from context) is an infinite sequence $\langle\mathbf{H},\mathbf{T}\rangle$ of pairs $\langle\mathbf{H}_i,\mathbf{T}_i\rangle$, where $\mathbf{H}_i \subseteq \mathbf{T}_i \subseteq \mathcal{P}$ for each $i \geq 0$. Both $\mathbf{H}$ and $\mathbf{T}$ are traces as usual, i.e., infinite sequences $\mathbf{H} = \mathbf{H}_0, \mathbf{H}_1, \ldots$ resp. $\mathbf{T} = \mathbf{T}_0, \mathbf{T}_1, \ldots$ of sets of atoms.

Definition 1 (THT-Satisfaction)**.** *Satisfaction of a THT formula by a THT-trace $\mathbf{I} = \langle\mathbf{H},\mathbf{T}\rangle$ at time k, where $0 \leq k$ is integer, is inductively defined as follows:*

- $\mathbf{I}, k \nvDash \bot$
- $\mathbf{I}, k \vDash p$ *if* $p \in \mathbf{H}_k$*, for any atom* $p \in \mathcal{P}$
- $\mathbf{I}, k \vDash \mathbf{Y}\,\phi$ *if* $\mathbf{I}, k-1 \vDash \phi$ *and* $k > 0$
- $\mathbf{I}, k \vDash \phi \vee \psi$ *if* $\mathbf{I}, k \vDash \phi$ *or* $\mathbf{I}, k \vDash \psi$
- $\mathbf{I}, k \vDash \phi \wedge \psi$ *if* $\mathbf{I}, k \vDash \phi$ *and* $\mathbf{I}, k \vDash \psi$
- $\mathbf{I}, k \vDash \phi \rightarrow \psi$ *if* $\begin{cases} \langle\mathbf{T},\mathbf{T}\rangle, k \nvDash \phi \text{ or } \langle\mathbf{T},\mathbf{T}\rangle, k \vDash \psi, \text{ and} \\ \mathbf{I}, k \nvDash \phi \text{ or } \mathbf{I}, k \vDash \psi \end{cases}$
- $\mathbf{I}, k \vDash \mathbf{X}\,\phi$ *if* $\mathbf{I}, k+1 \vDash \phi$
- $\mathbf{I}, k \vDash \phi\,\mathbf{U}\,\psi$ *if there is* $j \geq k$ *s.t.* $\mathbf{I}, j \vDash \psi$*, and for all* $j' \in [k, j)$*,* $\mathbf{I}, j' \vDash \phi$
- $\mathbf{I}, k \vDash \phi\,\mathbf{R}\,\psi$ *if for all* $j \geq k$ *s.t.* $\mathbf{I}, j \nvDash \psi$*, there exists* $j' \in [k, j)$*,* $\mathbf{I}, j' \vDash \phi$

A THT-trace $\mathbf{I}$ is a model for a formula ϕ if $\mathbf{I}, 0 \vDash \phi$.

A THT-trace $\mathbf{I}$ is *total*, if $\mathbf{H} = \mathbf{T}$; we then simply write $\mathbf{T}$ for $\langle\mathbf{T},\mathbf{T}\rangle$ if no confusion arises. Notably, $\mathbf{T} \vDash \phi$, i.e., $\langle\mathbf{T},\mathbf{T}\rangle \vDash \phi$, iff $\mathbf{T} \vDash_{LTL} \phi$ [Aguado *et al.*, 2011]. The salient difference between THT and LTL is a different evaluation of $\rightarrow$. In particular the *excluded middle* axiom $p \vee \neg p$ does not hold in THT. This is witnessed by e.g., $\mathbf{I} = \langle\mathbf{H},\mathbf{T}\rangle$ where $\mathbf{H} = \varnothing^\omega$ and $\mathbf{T} = \{p\}^\omega$. However, THT collapses to LTL under a *temporal excluded middle* axiom

$$\mathcal{TEM} = \mathbf{G}\mathcal{EM}, \text{ where } \mathcal{EM} = \textstyle\bigwedge_{p \in \mathcal{P}}(p \vee \neg p). \tag{2}$$

Proposition 1 (cf. [Cabalar and Demri, 2011])**.** *For $\phi \in \mathcal{L}$ and $\mathbf{I}$=$\langle\mathbf{H},\mathbf{T}\rangle$, $\mathbf{I} \vDash \phi \wedge \mathcal{TEM}$ iff $\mathbf{T} \vDash_{LTL} \phi$ and $\mathbf{T} = \mathbf{H}$.*

For traces $\mathbf{T}, \mathbf{T}'$, and $\mathbf{O}$, we write $\mathbf{T} \leq_{\mathbf{O}} \mathbf{T}'$ if $\mathbf{O}_i \subseteq \mathbf{T}_i \subseteq \mathbf{T}'_i$ holds for every $i \geq 0$. Furthermore, for THT-traces $\mathbf{I} = \langle\mathbf{H},\mathbf{T}\rangle$, $\mathbf{I}' = \langle\mathbf{H}',\mathbf{T}'\rangle$, and a trace $\mathbf{O}$, we write $\mathbf{I} \leq_{\mathbf{O}} \mathbf{I}'$ if $\mathbf{H} \leq_{\mathbf{O}} \mathbf{H}'$ and $\mathbf{T} = \mathbf{T}'$. Intuitively, $\leq_{\mathbf{O}}$ serves for defining $\mathbf{H}$-minimality modulo observations; they are present in $\mathbf{O}$ online as facts and thus do not need to be proven.

We are now ready to introduce the semantics of TEL.

Definition 2 (TEL-Satisfaction w.r.t. Observations)**.** *Given an observation trace $\mathbf{O}$, a trace $\mathbf{O} \leq \mathbf{T}$ is a temporal equilibrium model of a formula $\phi \in \mathcal{L}$ w.r.t. $\mathbf{O}$, if (i) $\langle\mathbf{T},\mathbf{T}\rangle \vDash \phi$, i.e., $\mathbf{T}$ is a total THT model of ϕ, and (ii) no $\mathbf{H} \neq \mathbf{T}$ exists s.t. $\mathbf{H} \leq_{\mathbf{O}} \mathbf{T}$ and $\langle\mathbf{H},\mathbf{T}\rangle \vDash \phi$, i.e., $\langle\mathbf{T},\mathbf{T}\rangle$ is minimal w.r.t. $\mathbf{O}$.*

If the observation trace $\mathbf{O}$ is empty, i.e., $\mathbf{O} = \varnothing^\omega$, Definition 2 yields classical TEL satisfaction [Aguado *et al.*, 2013]. Given two traces $\mathbf{T}$,$\mathbf{O}$ and a formula $\phi \in \mathcal{L}$, we denote by $\mathbf{T} \vDash^{\mathbf{O}}_{TEL} \phi$ that $\mathbf{T}$ is an equilibrium trace of ϕ modulo observations $\mathbf{O}$, where we may drop $\mathbf{O}$ if clear from context.

We will use temporal equilibrium model and equilibrium/stable traces interchangeably. Furthermore, we shall consider temporal programs [Cabalar, 2010], a fragment of $\mathcal{L}$ that resembles and extends the usual logic programming syntax.

Example 1. *Assume $\mathbf{T}$ and $\mathbf{O}$ are traces, where $switch$, and $power\_failure$ can appear in $\mathbf{O}$ (they need no justification). A model π_{EX} of an action domain has the following axioms:*

$$\begin{aligned}
&r_0: \ switch \ \vee \ \mathbf{X}\,anomaly \\
&r_1: \ \mathbf{G}(switch \ \wedge \ \neg light \ \wedge \ \neg\mathbf{X}\,anomaly \rightarrow \ \mathbf{X}\,light) \\
&r_2: \ \mathbf{G}(switch \wedge \ \neg\mathbf{X}\,anomaly \rightarrow \mathbf{X}\,change_light) \\
&r_3: \ \mathbf{G}(light \wedge \ \mathbf{X}\,anomaly \rightarrow \mathbf{X}\,change_light) \\
&r_4: \ \mathbf{G}(\neg\mathbf{X}\,change_light \ \wedge \ light \rightarrow \ \mathbf{X}\,light) \\
&r_5: \ \mathbf{G}(power_failure \rightarrow anomaly) \\
&r_6: \ \mathbf{G}(switch \ \wedge \ \mathbf{X}\,switch \rightarrow \bot)
\end{aligned}$$

Intuitively, r_0 says that either the light will be turned on, or an anomaly will happen at the second state; r_1 is a defeasible effect axiom; r_2 and r_3 keep track of action executions or anomalies that flip the value of light; r_4 is an inertia rule for light; r_5 defines the $power\_failure$ as an anomaly; r_6 is the property to monitor. Trace $\mathbf{T} = \{s\}, \{c\_l, l\}, \{a, p\_f, c\_l\}, \{a, p\_f\}^\omega$ is an equilibrium trace of π_{EX} w.r.t. $\mathbf{O} = \{s\}, \varnothing, \{p\_f\}^\omega$.

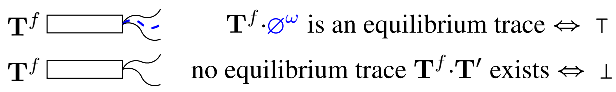

Figure 1: TEL_3 verdict for trace-prefix $\mathbf{T}^f$ under scrutiny.

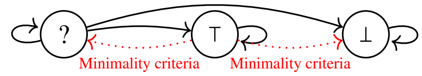

Figure 2: Evolution of the TEL_3 verdict.

We recall that in LTL, satisfiability is PSPACE-complete; the same holds for THT, while in TEL satisfiability is EXPSPACE-complete [Bozzelli and Pearce, 2016], where EXPSPACE-hardness holds for temporal programs. LTL stays in PSPACE if past time operators (including $\mathbf{Y}$) are added, cf. [Lichtenstein *et al.*, 1985]; this extends to the language $\mathcal{L}$.

Lemma 1. *Deciding whether $\phi \in \mathcal{L}$ is THT- (resp. TEL-) satisfiable is PSPACE- (resp. EXPSPACE-)complete.*

3 Online Setting

In online computation, we only have a prefix of a trace at any point in time. This motivates the following THT_3 semantics with truth-values *true* ($\top$), *false* ($\bot$), and *undefined* (?). Given a THT-trace $\mathbf{I} = \langle \mathbf{H}, \mathbf{T} \rangle$, its k-prefix (resp. k-suffix or suffix at k) is the sequence $\langle \mathbf{H}_0, \mathbf{T}_0 \rangle, \ldots, \langle \mathbf{H}_k, \mathbf{T}_k \rangle$ (resp. $\mathbf{I}^k = \langle \mathbf{H}_k, \mathbf{T}_k \rangle, \langle \mathbf{H}_{k+1}, \mathbf{T}_{k+1} \rangle, \ldots$), $0 \leq k$. A prefix of $\mathbf{I}$ is any k-prefix $\mathbf{I}^f$ of $\mathbf{I}$ whose length, denoted by $|\mathbf{I}^f|$, is $k{+}1$. We call $\mathbf{I}$ an extension of $\mathbf{I}^f$ w.r.t. observation trace $\mathbf{O}$, if $\mathbf{H} \leq_{\mathbf{O}} \mathbf{T}$ holds; by $ext(\mathbf{I}^f, \mathbf{O})$ we denote the set of all such $\mathbf{I}$. We denote by Pre_{THT} the set of all possible prefixes. Similarly as above, we may write $\mathbf{T}^f$ instead of $\langle \mathbf{T}^f, \mathbf{T}^f \rangle$ and omit $\mathbf{O}$.

Definition 3 (THT_3 semantics, cf. [Soldà *et al.*, 2023])**.** *The truth value of $\phi \in \mathcal{L}$ w.r.t. a prefix $\mathbf{I}^f$ and an observation trace $\mathbf{O}$ is as follows:*

$$\mathbf{I}^f \vDash^{\mathbf{O}}_{THT_3} \phi = \begin{cases} \top & \text{if } \mathbf{I} \vDash \phi \text{ for every } \mathbf{I} \in ext(\mathbf{I}^f, \mathbf{O}), \\ \bot & \text{if } \mathbf{I} \nvDash \phi \text{ for every } \mathbf{I} \in ext(\mathbf{I}^f, \mathbf{O}), \\ ? & \text{otherwise.} \end{cases}$$

Here $\mathbf{O}$ is a parameter to define a 3-valued logic for TEL, where minimality over traces matters. If $\mathbf{O}$ is empty, Defn. 3 is the THT version of the LTL_3 logic of Bauer *et al.* [2007]; further restriction to total traces yields their LTL_3 semantics.

TEL_3 **semantics** The 3-valued semantics of TEL, depicted in Figure 1, has been defined as follows:

Definition 4 (TEL_3 semantics, cf. [Soldà *et al.*, 2023])**.** *For a total prefix $\mathbf{T}^f$ of length k, $\phi \in \mathcal{L}$, and observation trace $\mathbf{O}$,*

$$\mathbf{T}^f \vDash^{\mathbf{O}}_{TEL_3} \phi = \begin{cases} \top & \text{if } \mathbf{T}^f \cdot \mathbf{O}^k \vDash^{\mathbf{O}}_{TEL} \phi, \\ \bot & \text{if } \mathbf{T} \nvDash^{\mathbf{O}}_{TEL} \phi, \ \forall\, \mathbf{T} \in ext(\mathbf{T}^f, \mathbf{O}), \\ ? & \text{otherwise.} \end{cases}$$

Note that the single minimal LTL model in $ext(\mathbf{T}^f, \mathbf{O})$ is used for the verdict $v = \top$; this is because nonmonotonicity of TEL may compromise the persistence of verdict $\top$ when $\mathbf{T}^f$ is extended. The possible evolution of the verdict over time is illustrated in Figure 2. The dotted arrow would not appear in a monotonic setting like LTL_3, or THT_3, as $\top$ would be valid. For example, for $\phi = \mathbf{G}(q \vee \neg q) \wedge \mathbf{G}(q \rightarrow \mathbf{G}\, p)$ we have $\varnothing \vDash_{TEL_3} \phi = \top$, while $\varnothing \cdot \{q, p\} \vDash_{TEL_3} \phi = ?$. The reading of verdict $\top$ is that if nothing is left to be proven (i.e., empty $\mathbf{T}$-suffix), this extension yields an equilibrium trace.

THT has the so-called persistence property, viz. that $\langle \mathbf{H}, \mathbf{T} \rangle \vDash \phi$ implies $\mathbf{T} \vDash \phi$. As we deal with three possible verdicts and a fixed prefix, only a weaker form holds.

Lemma 2 (Persistence for THT_3)**.** *For every $\phi \in \mathcal{L}$, prefix $\mathbf{I}^f = \langle \mathbf{H}^f, \mathbf{T}^f \rangle$, and observation trace $\mathbf{O}$, $\mathbf{I}^f \vDash^{\mathbf{O}}_{THT_3} \phi \neq \bot$ implies $\mathbf{T}^f \vDash^{\mathbf{O}}_{THT_3} \phi \neq \bot$.*

Note that both (I1) $\mathbf{I}^f \vDash^{\mathbf{O}}_{THT_3} \phi = \bot$ and (T1) $\mathbf{T}^f \vDash^{\mathbf{O}}_{THT_3} \phi = \top$ may hold. E.g., for $\phi = p \vee \neg p$, empty $\mathbf{O}$, $\mathbf{T}^f = \{p\}$, and $\mathbf{I}^f = \langle \varnothing, \{p\} \rangle$, we have $\mathbf{I}^f \nvDash p \vee \neg p$ while $\mathbf{T}^f \vDash p \vee \neg p$. Similarly, both (I2) $\mathbf{I}^f \vDash^{\mathbf{O}}_{THT_3} \phi = \top$ and (T2) $\mathbf{T}^f \vDash^{\mathbf{O}}_{THT_3} \phi =$? may hold, e.g. for $\phi = p \rightarrow \mathbf{X} p$ and $\mathbf{O}$, $\mathbf{I}^f$, $\mathbf{T}^f$ as before.

3.1 Complexity

We now analyze the complexity of the THT_3 and the TEL_3 semantics. We assume that prefixes of traces are explicitly given. We consider here that $\mathbf{O}$ is described by an LTL-formula ψ, such that for each model $\mathbf{T}$ of ψ and position $i \geq 0$, $\mathbf{T}, i \vDash_{LTL} p$ iff $p \in \mathbf{O}_i$.

We start by showing a negative result about the complexity of computing the verdict of THT_3 and of TEL_3 semantics, that justifies the introduction of the progression function P, and, respectively, P_{TEL}, which are more efficient.

Theorem 1. *Given $\phi \in \mathcal{L}$, a prefix $\mathbf{I}^f = \langle \mathbf{H}^f, \mathbf{T}^f \rangle$, and a arbitrary observations trace $\mathbf{O}$ model of an LTL formula ψ, namely $\mathbf{O} \vDash_{LTL} \psi$, deciding $\mathbf{I}^f \vDash^{\mathbf{O}}_{THT_3} \phi = v$ for $v \in \{\top, \bot, ?\}$ is PSPACE-complete, and PSPACE-hardness holds if v is fixed arbitrarily and $\mathbf{O}$ is empty.*

The hardness results are shown by simple reductions from THT-(non)validity, and the membership results by reductions to LTL-(un)satisfiability. For the latter, we exploit the star-transformation of THT into LTL [Cabalar and Demri, 2011]. We encode the prefix $\mathbf{I}^f$ into a formula $\phi_{\mathbf{I}^f}$ and use the following formula $\psi_{\mathbf{O}}$ to constrain the models by the observations from $\mathbf{O}$, which are expressed by the LTL formula ψ:

$$\psi_{\mathbf{O}} = \bar{\psi} \wedge \mathcal{TEM}_{\bar{\mathcal{P}}} \wedge \mathbf{G}(\bar{p} \rightarrow p),$$

where $\bar{\psi}$ is ψ with all occurrences of p replaced by $\bar{p}$ and $\mathcal{TEM}_{\bar{\mathcal{P}}}$ is $\mathcal{TEM}$ restricted to the atoms in $\bar{\mathcal{P}} = \{\bar{p} \mid p \in \mathcal{P}\}$.

Proposition 2. *The THT-models $\mathbf{I}$ of $\psi_{\mathbf{O}}$ coincide on $\mathcal{P}$ with the traces over $\mathcal{P}$ that include $\mathbf{O}$, and each $\mathbf{I}$ restricted to $\bar{\mathcal{P}}$ is total and coincides with the bar-version $\bar{\mathbf{O}}$ of $\mathbf{O}$.*

For the TEL_3-semantics, we obtain a complexity picture analogous to Theorem 1.

Theorem 2. *Given $\phi \in \mathcal{L}$, an LTL-prefix $\mathbf{T}^f$, and a formula ψ describing an observation trace $\mathbf{O}$, deciding $\mathbf{T}^f \vDash^{\mathbf{O}}_{TEL_3} v$ for a given $v \in \{\top, \bot, ?\}$ is EXPSPACE-complete, where EXPSPACE-hardness holds if $v = \bot$ or $v =$?, while for $v = \top$ the problem is PSPACE-complete and co-NP if $\mathbf{O} = \varnothing^\omega$.*

The EXPSPACE membership and hardness results are shown by reductions to resp. from TEL-(un)satisfiability, where the hardness results hold if, in addition, $\mathbf{O}$ is empty. For $v = \top$, the problem is in PSPACE: there are only exponentially many $\mathbf{I}^f$ for which must check $\mathbf{I}^f \cdot \mathbf{O}^k \models \phi$, where k is the length of $\mathbf{T}^f$. For each such $\mathbf{I}^f$, this reduces to an LTL-entailment test, which is in PSPACE, and looping through all $\mathbf{I}^f$ is feasible in PSPACE. Notably, if in addition, $\mathbf{O}$ is empty, the complexity drops to co-NP as the test $\mathbf{I}^f \cdot \varnothing^\omega \models \phi$ reduces to an LTL-model checking problem that is polynomial. The PSPACE- resp. co-NP-hardness is inherited from the complexity of LTL resp. ASP programs [Eiter and Gottlob, 1995].

We note that the hardness proofs in Theorem 2 can be adjusted to temporal programs showing that they harness the full complexity of TEL_3-semantics.

4 Progression for THT

As many monitoring tools have an online setting (see [Cimatti *et al.*, 2022], [Chen and Roşu, 2007] among the others), we are interested in studying a framework where observations are provided on the fly. We thus consider an online computation of the THT_3 semantics that relies on the progression technique for THT in [Soldà *et al.*, 2023], which yields an approximation of the THT_3 semantics.

In the progressive evaluation of a formula, an implication $\phi = p \rightarrow \mathbf{F}q$ may be only partially evaluable in the current state –when p belongs to the current state, but q does not– and we must delegate part of the evaluation to the future, therefore we rewrite ϕ as $\mathbf{F}q$. The main idea is that we propagate $\mathbf{F}q$ until we see q in the trace, when we can finally conclude $\top$.

We denote by $\mathcal{L}_c$ the set of formulas generated by the grammar in (1), where in place of $\rightarrow$ also $\rightarrow_c$ may occur. The $\rightarrow_c$ implication guides the progression by marking implication that must be evaluated in the There-trace only. Note that [Soldà *et al.*, 2023] disregarded full $\mathbf{U}$ and $\mathbf{R}$ (only $\mathbf{F}$ and $\mathbf{G}$ were considered). We now introduce THT progression.

Definition 5 (THT progression on a prefix state)**.** *Progression $P : \mathcal{L}_c \times Pre_{THT} \times \mathbb{N} \rightarrow \mathcal{L}_c$ is the partial function mapping a formula ψ, a THT-prefix $\mathbf{I}^f = \langle \mathbf{H}^f, \mathbf{T}^f \rangle$ of length k, and an integer $i \in [0, k)$, to an $\mathcal{L}_c$ formula as follows:*

1. $P(\bot, \mathbf{I}^f, i) = \bot$
2. $P(p, \mathbf{I}^f, i) = \top$ *if* $p \in \mathbf{H}^f_i$*, and* $p \in \mathcal{P}$
3. $P(p, \mathbf{I}^f, i) = \bot$ *if* $p \notin \mathbf{H}^f_i$*, and* $p \in \mathcal{P}$
4. $P(\phi_1 \vee \phi_2, \mathbf{I}^f, i) = P(\phi_1, \mathbf{I}^f, i) \vee P(\phi_2, \mathbf{I}^f, i)$
5. $P(\phi_1 \wedge \phi_2, \mathbf{I}^f, i) = P(\phi_1, \mathbf{I}^f, i) \wedge P(\phi_2, \mathbf{I}^f, i)$
6. $P(\phi_1 \rightarrow_c \phi_2, \mathbf{I}^f, i) = P(\phi_1, \mathbf{T}^f, i) \rightarrow_c P(\phi_2, \mathbf{T}^f, i)$
7. $P(\phi_1 \rightarrow \phi_2, \mathbf{I}^f, i) = \begin{cases} P(\phi_1, \mathbf{I}^f, i) \rightarrow P(\phi_2, \mathbf{I}^f, i) \wedge \\ (P(\phi_1, \mathbf{T}^f, i) \rightarrow_c P(\phi_2, \mathbf{T}^f, i)) \end{cases}$
8. $P(\mathbf{Y}\phi, \mathbf{I}^f, i) = P(\phi, \mathbf{I}^f, i-1)$ *if* $i > 0$ *else* $\bot$
9. $P(\mathbf{X}\phi, \mathbf{I}^f, i) = \phi$
10. $P(\phi_1 \mathbf{U} \phi_2, \mathbf{I}^f, i) = P(\phi_2, \mathbf{I}^f, i) \vee (P(\phi_1, \mathbf{I}^f, i) \wedge \phi_1 \mathbf{U} \phi_2)$
11. $P(\phi_1 \mathbf{R} \phi_2, \mathbf{I}^f, i) = P(\phi_2, \mathbf{I}^f, i) \wedge (P(\phi_1, \mathbf{I}^f, i) \vee \phi_1 \mathbf{R} \phi_2)$

In addition, $\top \rightarrow^* \bot$ *is replaced by* $\bot$*;* $\bot \rightarrow^* \phi$ *by* $\top$*; and* $\phi \rightarrow^* \top$ *by* $\top$*, for every formula* ϕ *and* $\rightarrow^* \in \{\rightarrow, \rightarrow_c\}$*.Furthermore,* $\top \vee \phi$ *is replaced by* $\top$*;* $\bot \vee \phi$ *by* ϕ*;* $\bot \wedge \phi$ *by* $\bot$*; and* $\top \wedge \phi$ *by* ϕ*.*

Note that $P(\mathbf{G}\phi, \mathbf{I}^f, i) = P(\phi, \mathbf{I}^f, i) \wedge \mathbf{G}\phi$ and $P(\mathbf{F}\phi, \mathbf{I}^f, i) = P(\phi, \mathbf{I}^f, i) \vee \mathbf{F}\phi$. We do not apply the progression recursively on the future states, but we do so on the sub-prefix of the trace, as we assume to have access to the current and past states. Let us progress rule $r_2 \in \pi_{EX}$ of the Example 1 over $\mathbf{T}^f = \{s\}$. The application of progression leads to $P(r_2, \langle \mathbf{T}^f, \mathbf{T}^f \rangle, 0) = \mathbf{G}(s \wedge \neg \mathbf{X}\, a \rightarrow \mathbf{X}\, c\_l) \wedge (\neg a \rightarrow c\_l)$.

Lemma 3. *If ϕ has no temporal operators, then $P(\phi, \mathbf{I}^f, k) = v$, $v \in \{\top, \bot\}$, for every trace $\mathbf{I}^f$ and $k \in [0, |\mathbf{T}^f|)$.*

In an online setting, the prefix is provided incrementally, extending it by one state at each iteration. The progression over a prefix is thus defined as follows.

Definition 6 (THT progression on a prefix)**.** *The progression of $\phi \in \mathcal{L}$ on the THT-prefix $\mathbf{I}^f$ is*

$$P(\phi, \mathbf{I}^f) = \begin{cases} v & \text{if } v \in \{\top, \bot\} \\ ? & \text{otherwise} \end{cases} \tag{3}$$

with $v = P^{|\mathbf{I}^f|}(\phi, \mathbf{I}^f)$, where $P^0(\phi, \mathbf{I}^f) = \phi$ and $P^i(\phi, \mathbf{I}^f) = P(P^{i-1}(\phi, \mathbf{I}^f), \mathbf{I}^f, i)$ for $0 < i \leq |\mathbf{I}^f|$.

An intuitive reading of the progression outcome $P(\phi, \mathbf{I}^f) = v$ is as follows: (i) $v = \top$ means that the property expressed by ϕ is already satisfied within the prefix $\mathbf{I}^f$; (ii) the case $v = \bot$ is the opposite, i.e., the violation of ϕ has been detected within the prefix $\mathbf{I}$; finally, (iii) $v = ?$ means that one abstains given the partially processed formula ϕ. The following result now states that the progression function P faithfully approximates the THT_3 semantics.

Theorem 3 (THT verdict on prefixes)**.** *For every prefix $\mathbf{I}^f$, observation trace $\mathbf{O}$, and formula $\phi \in \mathcal{L}$,*

$$P(\phi, \mathbf{I}^f) = v \text{ implies } \mathbf{I}^f \models^{\mathbf{O}}_{THT_3} \phi = v \text{ for } v \in \{\top, \bot\}.$$

4.1 Computing P Progression

In the next results, we show that a P application to a formula $\phi \in \mathcal{L}_c$ on a prefix $\mathbf{I}^f$ is feasible in polynomial time. By Theorem 3, we thus may justify P progression as a tractable approximation of the THT_3 semantics. We first show that computing a single progression step on a state that is tractable. Let us denote by $\overset{\rightarrow}{n_\psi}$ the number of $\rightarrow$ and $\rightarrow_c$ symbols in ψ, by $\overset{\text{temp}}{n_\psi}$ the number of $\mathbf{U}$ and $\mathbf{R}$ symbols in ψ.

Theorem 4. *Given $\psi \in \mathcal{L}_c$, a prefix $\mathbf{I}^f$ of length k, and $s \in [0, k)$, obtaining some $\psi' \equiv_{THT} P(\psi, \mathbf{I}^f, s) = \psi''$ is feasible in polynomial time where (i) $\psi'' \in \{\top, \bot\}$ implies $\psi' = \psi''$ and (ii) $|\psi'| \leq (\overset{\rightarrow}{n_\psi} + 1)(\overset{\text{temp}}{n_\psi} + 2)|\psi|$.*

Proof (Sketch). The result is obtained by using two look-up tables $\mathcal{T}$ and $\mathcal{H}$, where each sub-formula φ_i of ψ is progressed for $\mathbf{T}^f$ resp. $\mathbf{I}^f$ at every position $t \in [0, k)$ to a formula φ_i' that is memorized in $\mathcal{T}[\varphi_i, t]$ resp. in $\mathcal{H}[\varphi_i, t]$. One can then assemble ϕ_i' from the progression results of its subformulas ϕ_j' stored in $\mathcal{T}$ and $\mathcal{H}$. The key to having a polynomial-size

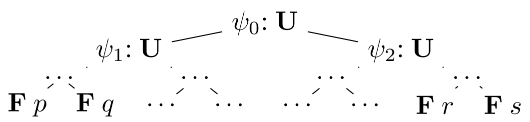

Figure 3: Abstract syntax tree of ψ, where $p, q, r, s \in \mathcal{P}$.

output is the fact that for progression over a total trace, we need to consider only one of $\rightarrow$ and $\rightarrow_c$ in the assembly as they yield equivalent results. Without this optimization, the application of P may lead to exponential outcomes. $\square$

Note that whenever $\varphi_i' \notin \{\top, \bot\}$, we can simply write $\mathcal{T}[\varphi_i, t] = ?$ (or $\mathcal{T}[\varphi_i, t-1] = ?$ for the $\mathbf{Y}$-case) if we are interested in the decision problem only.

For computing $P(\phi, \mathbf{I}^f)$, repeated application of $P(\phi, \mathbf{I}^f, k)$ does not allow us to conclude tractability, as ϕ may grow exponentially when progression is applied due to $\mathbf{U}$ and $\mathbf{R}$ operators. As an extreme example, consider the case of nested $\mathbf{U}$: $\psi_i = \psi_{2i+1}\ \mathbf{U}\ \psi_{2i+2}$ for $i = 0, \ldots, 2^{n-1} - 1$ and $\psi_i = \mathbf{F}\ p$ for some $p \in \mathcal{P}$ for all $i = 2^{n-1}, \ldots, 2^n - 1$, whose abstract syntax tree is depicted in Figure 3. By repeated application of P as in Theorem 4, $P(\psi_0, \mathbf{T}^f)$ grows exponentially in the size of $\mathbf{T}^f = \varnothing^{k \geq 1}$.

Fortunately, we can avoid exponential explosion by sharing subformulas. Indeed, progressing in the example ψ_0 over $\varnothing$ will yield two copies of ψ_1 in the output formula for progression at state 1; these copies can be shared.

To represent formulas for the THT progression in a succinct way, we use the following concept and notation. A DAG representation of a formula ψ is any directed acyclic graph resembling the syntax tree of ϕ with possible sharing of subtrees. Let us denote by $P_{temp}(\phi, \mathbf{I}^f, i)$ the formula as in Definition 5 but with result p in cases 2 and 3; i.e., p is not evaluated over $\mathbf{I}^f$ and thus independent of $\mathbf{I}^f$.

Definition 7. *Given a DAG-representation of $\psi \in \mathcal{L}$ and $\phi \in sub(\psi)$, we denote by $N_i^j\ \phi$ the DAG representing the formula $P^j_{temp}(\phi, \mathbf{I}^f, i)$ for any $\mathbf{I}^f$ of length $\leq i$.*

Notably, $P_{temp}(\phi, \mathbf{I}^f, i)$ can be viewed as a template to obtain $P(\phi, \mathbf{I}^f, i)$ by evaluating it over prefix $\mathbf{I}^f$, and accordingly $N_i^j \phi$ represents a template $P^j_{temp}(\phi, \mathbf{I}^f, i)$ whose evaluation over $\mathbf{I}^f$ yields $P^j(\phi, \mathbf{I}^{f^i})$, i.e., the result of j-fold progression of ϕ over the i-suffix of $\mathbf{I}^f$.

Lemma 4. *Given $\psi \in \mathcal{L}$ and $k > 0$ written in unary, we can compute $N_0^k\ \psi$ in polynomial time (more precisely in $O(k^2|\psi|)$ time).*

Proof. We provide an algorithm that computes a DAG with the root node labeled by $N_0^k\ \psi$, saving memory by means of sharing subformulas. Given the abstract syntax tree of the formula, we label each node representing the subformula ϕ by $N_0^0\ \phi$. When applying progression to $N_i^j\ \phi$, we check whether already a node for $N_i^{j+1} \phi$ exists; if so, the reference to $N_i^{j+1}\ \phi$ is used; otherwise a new node $N_i^{j+1}\ \phi$ is created, and we rewrite ϕ, applying Definition 5, with references to the progressions of its subformulas, by cases as follows:

- $N_i^{j+1} \phi$ points to $N_i^j\ \phi$, if ϕ has no temporal operators;
- $N_i^{j+1} \mathbf{X} \phi$ points to $N_{i+1}^j \phi$;
- $N_i^{j+1} \mathbf{Y} \phi$ points to $N_{i-1}^{j+1} \phi$;
- $N_i^{j+1} \phi_1\ \mathbf{U}\ \phi_2$ points to a formula for $N_i^{j+1} \phi_2 \vee N_i^{j+1} \phi_1 \wedge N_{i+1}^j (\ \phi_1 \mathbf{U} \phi_2)$; to avoid introducing nodes for new subformulas, we may view $\mathbf{U}$ progression as a ternary connective, where $\phi_1 \mathbf{U} \phi_2 = \bot \vee \top \wedge (\phi_1 \mathbf{U} \phi_2)$. The case $N_i^{j+1}(\phi_1\ \mathbf{R}\ \phi_2)$ is analogous.

We iterate the process from $j = 0, \ldots, k$ until the root of the DAG is labeled by $N_0^k\ \psi$; importantly, each N_i^j of $k^2|\psi|$ nodes occurs at any point in time at most once in the DAG.

Also in this case, we first run the algorithm for the $\mathbf{T}$- part, successively for the $\mathbf{H}$-part. The Here-DAG may share some nodes from the There-DAG if there are implications. $\square$

Thus, we can tractably compute the verdict of the progression on a formula ψ when a new state is provided:

Theorem 5. *For any $\phi \in \mathcal{L}$, prefix $\mathbf{I}^f$, such that $|\mathbf{I}^f| = k$, a DAG representation of $P(\phi, \mathbf{I}^f)$ is computable in polynomial time (more precisely, in $O(k^2|\psi|)$ time).*

Proof. By Lemma 4, the DAG $N_0^k \psi$ for the template formula $P^k_{temp}(\phi, \mathbf{I}^f, 0)$ is computable in $O(k^2|\phi|)$ time and by following the lookup technique of Theorem 4 we can compute the verdict, where the Here-DAG and the There-DAG allow us to efficiently navigate to sub-formulas; this is also feasible, including simplifications, in $O(k^2|\phi|)$ time. $\square$

5 Progression for TEL

TEL progression resorts to THT progression as follows.

Definition 8 (TEL progression on a prefix, cf. [Soldà *et al.*, 2023]). *The TEL progression of $\phi \in \mathcal{L}$ on a total prefix $\mathbf{T}^f$ is*

$$P_{TEL}(\phi, \mathbf{T}^f) = \begin{cases} \top & \text{if } \phi' = \top \wedge \forall\, \mathbf{H}^f \subset \mathbf{T}^f : \phi''(\mathbf{H}^f) = \bot \\ \bot & \text{if } \phi' = \bot \vee \exists\, \mathbf{H}^f \subset \mathbf{T}^f : \phi''(\mathbf{H}^f) = \top \\ ? & \text{otherwise,} \end{cases}$$

where $\phi' = P(\phi, \langle \mathbf{T}^f, \mathbf{T}^f \rangle)$, $\phi''(\mathbf{H}^f) = P(\phi, \langle \mathbf{H}^f, \mathbf{T}^f \rangle)$.

Definition 8 faithfully approximates TEL_3 semantics.

Theorem 6 (TEL verdict on prefixes). *For any formula ϕ, prefix $\mathbf{T}^f$, and observation trace $\mathbf{O}$, P_{TEL} progression is sound w.r.t. the TEL_3-verdict, i.e.,*

$$P_{TEL}(\phi, \mathbf{T}^f) = v \text{ implies } \mathbf{T}^f \vDash^{\mathbf{O}}_{TEL_3} \phi = v, \text{ for } v \in \{\top, \bot\}.$$

We note that while both THT_3 and TEL_3 in Defn. 3 resp. 4 take an observation trace into account, the progressions P and P_{TEL} do not. This is because we assume the observations are already encoded in the prefix of the trace to progress. If we replace $\mathbf{H}^f \subset \mathbf{T}^f$ in Defn. 8 with $\mathbf{H}^f <_{\mathbf{O}} \mathbf{T}^f$ defined by $\mathbf{O}_i \subseteq \mathbf{H}^f_i \subseteq \mathbf{T}^f_i$ for $i \in [0, |\mathbf{T}^f|)$ and $\mathbf{H}^f \neq \mathbf{T}^f$, Theorem 6 still holds.

Since we focus on online computations, we introduce an incremental P_{TEL} progression, where we resort to the $\mathbf{Y}$ temporal operator to rewrite the formula, unfolding it using the inductive definitions of the temporal operators.

Definition 9. *The incremental version of TEL progression of $\phi \in \mathcal{L}$ on a total prefix $\mathbf{T}^f$ for position $i \in [0, |\mathbf{T}^f|)$ is*

$$P^{inc}_{TEL}(\phi, \mathbf{T}^f, i) = \begin{cases} \top & \text{if } \phi'_i = \top \wedge \forall\, \mathbf{H}^f \subset \mathbf{T}^f : \phi''_i(\mathbf{H}^f) = \bot \\ \bot & \text{if } \phi'_i = \bot \vee \exists\, \mathbf{H}^f \subset \mathbf{T}^f : \phi''_i(\mathbf{H}^f) = \top \\ \phi^{\mathbf{Y}} & \text{otherwise,} \end{cases}$$

where $\phi'_i = P(\phi, \langle \mathbf{T}^f, \mathbf{T}^f \rangle, i)$, $\phi''_i(\mathbf{H}^f) = P(\phi, \langle \mathbf{H}^f, \mathbf{T}^f \rangle, i)$, $\phi^{\mathbf{Y}}$ is defined inductively by

- $\bot^{\mathbf{Y}} = \bot$;
- $p^{\mathbf{Y}} = \mathbf{Y}p$ *if* $p \in \mathcal{P}$;
- $(\mathbf{X}\,\phi_1)^{\mathbf{Y}} = \phi_1$;
- $(\mathbf{Y}\,\phi_1)^{\mathbf{Y}} = \mathbf{YY}\phi_1$;
- $(\phi_1 \circ \phi_2)^{\mathbf{Y}} = \phi_1^{\mathbf{Y}} \circ \phi_2^{\mathbf{Y}}$ *for* $\circ \in \{\rightarrow, \wedge, \vee\}$;
- $(\phi_1 \mathbf{U}\,\phi_2)^{\mathbf{Y}} = \phi_2^{\mathbf{Y}} \vee (\phi_1^{\mathbf{Y}} \wedge (\phi_1 \mathbf{U}\,\phi_2))$;
- $(\phi_1 \mathbf{R}\,\phi_2)^{\mathbf{Y}} = \phi_2^{\mathbf{Y}} \wedge (\phi_1^{\mathbf{Y}} \vee (\phi_1 \mathbf{R}\,\phi_2))$.

We define $P^{inc,i}_{TEL}(\phi, \mathbf{T}^f)$ as the P^{inc}_{TEL} progression reached at state i, i.e., $P^{inc,0}_{TEL}(\phi, \mathbf{T}^f, 0) = \phi$ and $P^{inc,i}_{TEL}(\phi, \mathbf{T}^f) = P^{inc}_{TEL}(P^{inc,i-1}_{TEL}(\phi, \mathbf{T}^f), \mathbf{T}^f, i)$ for $0 < i \leq |\mathbf{T}^f|$. Then we obtain at the end of $\mathbf{T}^f$ the P_{TEL}-result. Formally, let $P^{inc}_{TEL}(\phi, \mathbf{T}^f) = P^{inc,|\mathbf{T}^f|}_{TEL}(\phi, \mathbf{T}^f)$. Then

Theorem 7. *For every $\phi \in \mathcal{L}$ and prefix $\mathbf{T}^f$, $P_{TEL}(\phi, \mathbf{T}^f) = v$ iff $P^{inc}_{TEL}(\phi, \mathbf{T}^f) = v$.*

Note that the possible evolution of the verdict over time is still captured by Figure 2 even under the 3-valued semantics induced by P_{TEL} (or, equivalently P^{inc}_{TEL}).

5.1 Computing P_{TEL} Progression

The results above show an exponential complexity gap between THT_3-satisfiability (PSPACE-complete) and TEL-satisfiability (EXPSPACE-complete). We find a complexity gap between the P and the P_{TEL} approximation, but it is much smaller: while intractable, P_{TEL} is in the Boolean Hierarchy and close to NP and co-NP, being not harder than the conjunction of an NP and a co-NP problem.

Theorem 8. *Given $\phi \in \mathcal{L}$ and a prefix $\mathbf{T}^f$, deciding $P_{TEL}(\psi, \mathbf{T}^f) = v$ is (i) for $v = \top$ co-NP-complete, (ii) for $v = \bot$ NP-complete, and (iii) for $v =?$ D^p-complete.*

This result is shown by exploiting Theorem 5 for the upper bounds, and by providing reductions from stable model checking problems for logic programs for the hardness parts.

Thus, while by Theorem 8 P_{TEL} progression requires NP/co-NP oracle calls, it may be as a near-tractable approximation of the TEL_3 semantics.

Note that the complexity proofs for P_{TEL}-progression exploit Theorem 5 for checks of varying prefixes $\mathbf{I}^f$. The progression DAGs N^k_0 of $P(\psi, \mathbf{I}^f)$ in Lemma 4 can be instantiated for them on the fly, and with P^{inc}_{TEL} we can easily extend N^k_0 incrementally. It is possible to see that the DAG representation leads to an equivalent formula of the P^{inc}_{TEL}. Taking the $N^k_{i=0}$ as a starting point we proceed top-down adding the proper connectives and placing $k - i$ many $\mathbf{Y}$ operators in front of the sub-formula associated with the leaves of the DAG.

5.2 Progressing Temporal Programs

As mentioned, temporal ASP programs are an important fragment of TEL, for which efficient P_{TEL} progression is desirable. We achieve this by suitable syntactic restrictions that resemble well-known classes of logic programs.

Definition 10 (Temporal Programs, cf. [Cabalar, 2010; Aguado *et al.*, 2023])**.** *A* temporal program *is any set of* temporal rules *of one of the following forms:*

- *(*initial *rule)*
$$r : b_1 \wedge \ldots b_k \rightarrow c_1 \vee \cdots \vee c_l \text{ with } k, l \geq 0 \qquad (4)$$
where all b_i, c_j are in $\{p, \mathbf{X}p, \neg p, \neg \mathbf{X}p \mid p \in \mathcal{P}\}$;
- *(*dynamic *rule) $\mathbf{G}r$, where r is an initial rule;*
- *(*fulfillment *rule) either $\mathbf{G}(\mathbf{G}p \rightarrow q)$ or $\mathbf{G}(p \rightarrow \mathbf{F}q)$, where p, q are atoms.*

An initial or dynamic rule r is a constraint*, if its head is $\bot$ (i.e., l=0), and is a* fact *if its body is empty (k=0) and l=1. We call $\{p, \mathbf{X}\,p \mid\ p \in \mathcal{P}\}$ temporal atoms, and we denote by $B^+(r)$ (resp. $B^-(r)$) the set of (resp. negated) temporal atoms in the body of r, and by $H(r)$ the set $\{c_1, \ldots, c_l\}$.*

Using auxiliary atoms, temporal programs are as expressive as full TEL [Cabalar, 2010] and harness the full complexity of TEL. This is paralleled by the fact that the complexity results in Theorem 8 carry over to temporal programs; in particular that P_{TEL} progression is D^p-complete.

As a syntactic restriction, we first consider *normal programs*, namely rules r, where $H(r) = \{p\}$ or $\{\mathbf{X}\,p\}$, i.e. $l \leq 1$ w.r.t. (4) and fulfillment rules.

Theorem 9. *Given a normal temporal program π and a prefix $\mathbf{T}^f$, deciding whether $P_{TEL}(\mathbf{T}^f, \pi) = v$ for $v \in \{\bot, \top, ?\}$ can be done in polynomial time.*

Proof (Sketch). Let $k = |\mathbf{T}^f| - 1$. By Lemma 4 and Theorem 5, we can compute N^k_0 in polynomial time, and evaluate it on $\mathbf{T}^f$ in polynomial time, obtaining as outcome $v_{\mathbf{T}^f}$. According to the conditions of P_{TEL}(Defn 8), we may have to consider some (all) $\mathbf{H}^f \subset \mathbf{T}^f$ and evaluate N^k_0 over $\mathbf{I}^f = \langle \mathbf{H}^f, \mathbf{T}^f \rangle$. N^k_0 represents a set of implications, and after evaluating in N^k_0 the negated atoms over $\mathbf{T}^f$, the atemporal rules (not involving any temporal operator) in N^k_0 amount to a set of Horn clauses. We can compute the least model $\mathbf{M}^f$ of these clauses in polynomial time resp. find they have no model. As for the remaining implications in N^k_0, sub-formulas within the scope of an $\mathbf{X}, \mathbf{G}, \mathbf{F}$ operator cannot lead to $\top$ or $\bot$, and thus either formulas are removed (when $\phi''(\mathbf{H}^f)$ should yield $\bot$) resp. subformulas (when $\phi''(\mathbf{H}^f)$ should yield $\top$) before computing $\mathbf{M}^f$. From $v_{\mathbf{T}^f}$ and $\mathbf{M}^f$, we then easily obtain $P_{TEL}(\mathbf{T}^f, \pi)$. □

Unfortunately, the temporal program π_{EX} is not a normal temporal program. In what follows we extend tractability to the larger head-cycle-free class, which includes π_{EX}.

Given a temporal program π, its ω-unfolded version π^ω contains every initial rule $r \in \pi$ and for each dynamic or fulfillment rule $\mathbf{G}(r) \in \pi$ it contains $\mathbf{X}^i(r)$ for all $i \geq 0$.

For the sake of readability given an unfolded rule $\mathbf{X}^k(r)$, we denote $p, \mathbf{X}\,q \in B^+(r) \cup B^-(r) \cup H(r)$ as p^k and q^{k+1}.

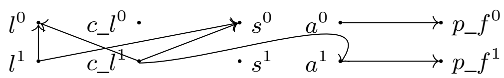

Figure 4: Temporal dependency graph of the program in Example 1.

Definition 11 (Dependency Graph)**.** *The* dependency graph *of a temporal unfolded program π^ω is the directed graph $DG_\pi = \langle V, E\rangle$ where $V = \{p^i \mid p \in \mathcal{P}$ and $i \geq 0\}$ and (i) $(a,b) \in E$ if $a \in H(r)$ and $b \in B^+(r)$ for some rule $\mathbf{X}^k r$ in π^ω, (ii) $(q^k, p^{k'}) \in E$ for $k' \geq k$ if $\mathbf{X}^k(\mathbf{G}\, p \to q) \in \pi^\omega$, and and (iii) $(q^{k'}, p^k) \in E$ for $k' \geq k$ if $\mathbf{X}^k(p \to \mathbf{F}\, q) \in \pi$.*

Intuitively, (ii) and (iii) are added because we can read $\mathbf{G}p$ and $\mathbf{F}p$ as a conjunction resp. disjunction of $\mathbf{X}^i p$, $i \geq 0$.

We call a program π *head-cycle free (hcf)*, if all the heads contain only positive temporal atoms, and its ω-unfolded version π^ω has the hcf property defined as follows: whenever distinct $a, b \in V$ are on a cycle of DG_π, then (a) no rule $\mathbf{X}^k\, r \in \pi^\omega$ satisfies $\{a, b\} \subseteq H(\mathbf{X}^k\, r)$ and (b) if $a = q^k$ and $b = q^{k'}$ with $k \neq k'$ then no fulfillment rule $\mathbf{X}^k\,(p \to \mathbf{F}\, q) \in \pi^\omega$ exists. E.g., the program in Example 1 is hcf.

This notion of hcf for temporal programs conservatively extends hcf for logic programs in [Ben-Eliyahu and Dechter, 1994]. As well known, hcf logic programs can be rewritten into normal logic programs by shifting atoms from the head to the rule body. This property generalizes to temporal programs.

Formally, denote for an initial rule r by $r^\leftharpoonup$ the set of all rules r' such that $B^+(r') = B^-(r)$, $B^-(r') = B^-(r) \cup (H(r) \setminus \{a\})$, and $H(r') = \{a\}$, where $a \in H(r)$, and by $\pi^\leftharpoonup$ the result of replacing in program π each initial rule r and dynamic rule $\mathbf{G}r$ by $r^\leftharpoonup$ resp. $\mathbf{G}r'$, for all $r' \in r^\leftharpoonup$. We then establish the following result, which is of interest in its own right.

Proposition 3. *For any hcf temporal program π, π and $\pi^\leftharpoonup$ have the same equilibrium models.*

As the program $\pi^\leftharpoonup$ is easily constructed from program π, we can generalize Theorem 9 to a larger class of $TASP$:

Theorem 10. *Given a hcf temporal program π and a prefix $\mathbf{T}^f$, deciding whether $P_{TEL}(\mathbf{T}^f, \pi) = v$ for $v \in \{\bot, \top, ?\}$ can be done in polynomial time.*

The notion of hcf can, in particular, be fruitfully applied to $TASP$ classes from the literature, such as the STLP fragment [Cabalar and Diéguez, 2011]. STLP admits two types of initial rules: (i_0) $B \wedge N \to H$, or (i_1) r: $B \wedge \mathbf{X}B' \wedge N \wedge \mathbf{X}N' \to \mathbf{X}H$, and dynamic rules $\mathbf{G}(r)$ with r an initial rule of type r_1, where B, B' are conjunctions of atoms, N, N' are conjunctions of negative literals $\neg p$ with $p \in \mathcal{P}$, and H, H', are disjunctions of atoms. Notably, program π_{EX} is of this form. We show that for STLP programs, the hcf property can be efficiently verified. Let DG^i_π, $i \geq 0$, be the subgraph of DG_π induced by all nodes p^j, where $p \in \mathcal{P}$ and $j \leq i$.

Proposition 4. *A STLP program π is hcf iff DG^1_π satisfies the hcf-condition.*

Figure 4 shows the clipped dependency graph $DG^1_{\pi_{EX}}$, from which the hcf property of π_{EX} is immediately verified.

Notably, we can use DG^i_π to efficiently progress programs π whose unfolding on the prefix $\mathbf{I}^f$ is hcf. Let us say program π is k-hcf if DG^k_π satisfies the hcf-condition.

Corollary 1. *Given a temporal program π and a prefix $\mathbf{T}^f$, deciding whether $P_{TEL}(\mathbf{T}^f, \pi) = v$ for $v \in \{\bot, \top, ?\}$ can be done in polynomial time if π is $(|\mathbf{I}^f| - 1)$-hcf.*

The k-hcf condition can be efficiently checked along the progression, as DG^k_π can be incrementally built.

6 Related Work

Progression has been widely used in CPS, but also in KR, e.g., in [Zhou and Zhang, 2017], to generalize the GL reduct for ASP with variables; in [Belle and Levesque, 2020] and [Liu and Feng, 2023] to reason about actions in the presence of belief; in [Lin and Reiter, 1997] and [Vassos and Levesque, 2013] to update a database by the execution of an action.

Bozzelli and Pearce [2016] made a detailed complexity study of TEL, but considered merely 2-valued semantics.

Tools for stable traces are `telingo` [Cabalar *et al.*, 2019], which builds on the `clingo` solver and handles finite traces, and `STeLP` [Cabalar and Diéguez, 2011], which is based on automata. Progression in temporal ASP is a novel approach in [Soldà *et al.*, 2023], where recently a framework for temporal skeptical reasoning was proposed. However, no complexity study was provided for the monitoring problem.

Furthermore, several stream reasoners and incremental solvers for ASP semantics exist. One of the first incremental ASP-solvers was `oclingo` [Cerexhe *et al.*, 2014], which extends `clingo` to handle dynamic events. `TICKER` [Beck *et al.*, 2017] is another incremental stream reasoner for finite streams, which implements a fragment of LARS [Beck *et al.*, 2018]. Calimeri *et al.* [2021] present an efficient stream reasoner, which however is limited to stratified programs.

`MeTeoR` [Wang *et al.*, 2022] builds on DatalogMTL, which is an extension of Datalog with MTL operators. Wałęga *et al.* [2023a] recently proposed an incremental approach. The main differences between the setting we consider are that they (1) consider an interval-based semantics, (2) use Horn rules precluding backward propagation of information, and (3) define a reduct-based semantics. Furthermore, Walega *et al.* [2021; 2023b] introduced a stable semantics for negation akin to temporal ASP. They studied the complexity of stable model existence, but neither monitoring nor progression.

7 Conclusion

We have characterized the complexity of THT_3 and TEL_3, as well as of progression for THT and P_{TEL} by the P and the P_{TEL} operator, respectively. Our results show that progression can be viewed as a (nearly) tractable approximation of THT_3 and TEL_3 semantics, and that tractability holds for significant fragments covering classes of temporal logic programs.

Our work can be extended in various directions. One could allow for past-time operators without any restriction, but a compact representation of progression formulas like a DAG would be barely needed. More expressive observation traces, represented by automata, and sets of observation traces can also be taken into account, modeling possible nondeterminism in the system. Our future agenda includes this and developing efficient implementations based on the algorithms and results presented, as well as refining the progression technique.

Ethical Statement

There are no ethical issues.

Acknowledgments

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101034440. This work has been partially supported by the WWTF project ICT22-023.

References

[Aguado *et al.*, 2011] Felicidad Aguado, Pedro Cabalar, Gilberto Pérez, and Concepción Vidal. Loop formulas for splitable temporal logic programs. In James P. Delgrande and Wolfgang Faber, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 80–92, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Aguado *et al.*, 2013] Felicidad Aguado, Pedro Cabalar, Martín Diéguez, Gilberto Pérez, and Concepción Vidal. Temporal equilibrium logic: a survey. *Journal of Applied Non-Classical Logics*, 23(1-2):2–24, 2013.

[Aguado *et al.*, 2023] Felicidad Aguado, Pedro Cabalar, Martín Diéguez, Gilberto Pérez, Torsten Schaub, Anna Schuhmann, and Concepción Vidal. Linear-time temporal answer set programming. *Theory Pract. Log. Program.*, 23(1):2–56, 2023.

[Bacchus and Kabanza, 1998] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22:5–27, 1998.

[Bauer *et al.*, 2007] Andreas Bauer, Martin Leucker, and Christian Schallhart. The good, the bad, and the ugly, but how ugly is ugly? In *Proc. of RV 2007: the 7th International Workshop on Runtime Verification*, pages 126–138. Springer, 2007.

[Beck *et al.*, 2017] Harald Beck, Thomas Eiter, and Christian Folie. Ticker: A system for incremental asp-based stream reasoning. *Theory and Practice of Logic Programming*, 17(5-6):744–763, 2017.

[Beck *et al.*, 2018] Harald Beck, Minh Dao-Tran, and Thomas Eiter. Lars: A logic-based framework for analytic reasoning over streams. *Artificial Intelligence*, 261:16–70, 2018.

[Belle and Levesque, 2020] Vaishak Belle and Hector J Levesque. Regression and progression in stochastic domains. *Artificial Intelligence*, 281:103247, 2020.

[Ben-Eliyahu and Dechter, 1994] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for disjunctive logic programs. *Ann. Math. Artif. Intell.*, 12(1-2):53–87, 1994.

[Bozzelli and Pearce, 2015] Laura Bozzelli and David Pearce. On the complexity of temporal equilibrium logic. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 645–656. IEEE Computer Society, 2015.

[Bozzelli and Pearce, 2016] Laura Bozzelli and David Pearce. On the expressiveness of temporal equilibrium logic. In Loizos Michael and Antonis C. Kakas, editors, *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9-11, 2016, Proceedings*, volume 10021 of *Lecture Notes in Computer Science*, pages 159–173, 2016.

[Cabalar and Demri, 2011] Pedro Cabalar and Stéphane Demri. Automata-based computation of temporal equilibrium models. In *International Symposium on Logic-Based Program Synthesis and Transformation*, pages 57–72. Springer, 2011.

[Cabalar and Diéguez, 2011] Pedro Cabalar and Martín Diéguez. STeLP–a tool for temporal answer set programming. In *Proc. of LPNMR 2011: the 11th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 370–375. Springer, 2011.

[Cabalar *et al.*, 2019] Pedro Cabalar, Roland Kaminski, Philip Morkisch, and Torsten Schaub. telingo= ASP + time. In *Proc. of LPNMR 2019: the 15th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 256–269. Springer, 2019.

[Cabalar *et al.*, 2023] Pedro Cabalar, Agata Ciabattoni, and Leendert van der Torre. Deontic equilibrium logic with explicit negation. In Sarah Alice Gaggl, Maria Vanina Martinez, and Magdalena Ortiz, editors, *Logics in Artificial Intelligence - 18th European Conference, JELIA 2023, Dresden, Germany, September 20-22, 2023, Proceedings*, volume 14281 of *Lecture Notes in Computer Science*, pages 498–514. Springer, 2023.

[Cabalar, 2010] Pedro Cabalar. A normal form for linear temporal equilibrium logic. In *Proc. of JELIA 2010: the 12th European Conference on Logics in Artificial Intelligence*, pages 64–76. Springer, 2010.

[Calimeri *et al.*, 2021] Francesco Calimeri, Marco Manna, Elena Mastria, Maria Concetta Morelli, Simona Perri, and Jessica Zangari. I-dlv-sr: a stream reasoning system based on i-dlv. *Theory and Practice of Logic Programming*, 21(5):610–628, 2021.

[Cerexhe *et al.*, 2014] Timothy Cerexhe, Martin Gebser, and Michael Thielscher. Online agent logic programming with oclingo. In *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings 13*, pages 945–957. Springer, 2014.

[Chen and Roşu, 2007] Feng Chen and Grigore Roşu. Mop: an efficient and generic runtime verification framework. In *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications*, pages 569–588, 2007.

[Cimatti *et al.*, 2022] Alessandro Cimatti, Chun Tian, and Stefano Tonetta. Assumption-based runtime verification. *Formal Methods in System Design*, 60(2):277–324, 2022.

[de Leng and Heintz, 2018] Daniel de Leng and Fredrik Heintz. Partial-state progression for stream reasoning with metric temporal logic. In Michael Thielscher, Francesca Toni, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, pages 633–634. AAAI Press, 2018.

[Eiter and Gottlob, 1995] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15:289–323, 1995.

[Giacomo *et al.*, 2016] Giuseppe De Giacomo, Yves Lespérance, Fabio Patrizi, and Stavros Vassos. Progression and verification of situation calculus agents with bounded beliefs. *Stud Logica*, 104(4):705–739, 2016.

[Kautz, 1986] Henry A Kautz. The logic of persistence. In *AAAI*, volume 86, pages 401–405, 1986.

[Lichtenstein *et al.*, 1985] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In Rohit Parikh, editor, *Logics of Programs, Conference, Brooklyn College, New York, NY, USA, June 17-19, 1985, Proceedings*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.

[Lin and Reiter, 1997] Fangzhen Lin and Ray Reiter. How to progress a database. *Artificial Intelligence*, 92(1-2):131–167, 1997.

[Liu and Feng, 2023] Daxin Liu and Qihui Feng. On the progression of belief. *Artificial Intelligence*, 322:103947, 2023.

[McCarthy and Hayes, 1981] John McCarthy and Patrick J Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*, pages 431–450. Elsevier, 1981.

[McCarthy, 1998] John McCarthy. Elaboration tolerance. In *Common sense*, volume 98, page 2, 1998.

[Pearce, 2006] David Pearce. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):3, 2006.

[Soldà *et al.*, 2023] Davide Soldà, Ignacio D. Lopez-Miguel, Ezio Bartocci, and Thomas Eiter. Progression for monitoring in temporal ASP. In *ECAI 2023 - 27th European Conference on Artificial Intelligence, 30 September, 2023 - 5 October, 2023 Krakow, Poland, October 3, 2023 - Including 12th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2023)*, 2023. In press.

[Vassos and Levesque, 2013] Stavros Vassos and Hector J Levesque. How to progress a database iii. *Artificial Intelligence*, 195:203–221, 2013.

[Walega *et al.*, 2021] Przemyslaw Andrzej Walega, David J. Tena Cucala, Egor V. Kostylev, and Bernardo Cuenca Grau. Datalogmtl with negation under stable models semantics. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 609–618, 2021.

[Wałęga *et al.*, 2023a] Przemysław A Wałęga, Mark Kaminski, Dingmin Wang, and Bernardo Cuenca Grau. Stream reasoning with DatalogMTL. *Journal of Web Semantics*, 76:100776, 2023.

[Walega *et al.*, 2023b] Przemyslaw Andrzej Walega, David J. Tena Cucala, Bernardo Cuenca Grau, and Egor V. Kostylev. The stable model semantics of datalog with metric temporal operators. *Theory and Practice of Logic Programming*, page 1–35, 2023.

[Wang *et al.*, 2022] Dingmin Wang, Pan Hu, Przemysław Andrzej Wałęga, and Bernardo Cuenca Grau. Meteor: Practical reasoning in datalog with metric temporal operators. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pages 5906–5913, 2022.

[Zhou and Zhang, 2017] Yi Zhou and Yan Zhang. A progression semantics for first-order logic programs. *Artificial Intelligence*, 250:58–79, 2017.