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) [Bozzielli 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 [Cereize et al., 2014], ticker [Beck et al., 2017], 1-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; Walega 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 $P_{TEL}^{inc}$ of TEL progression, which is better suited for online evaluation.
- We characterize the complexity of monitoring under $TEL_3$.
and $THT_t$ semantics, showing that they are like $TEL$ and $THT$ EXPSPACE-complete and PSPACE-complete, respectively, in general [Bozelli 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 $P_{TEL}^{inc}$) 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 head-cycle-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, $P_{TEL}^{inc}$ 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 $L$ of $LTL$ with past-time operators, generated by the grammar

$$F ::= \perp \mid p \mid F \circ F \mid XF \mid FRF \mid FUF \mid YF'$$

$$F' ::= \perp \mid p \mid YF'$$

where $p \in P$ for a finite set $P$ of propositional atoms and $\circ \in \{\land, \lor, \rightarrow\}$. Negation is defined as $\neg \phi \equiv \phi \rightarrow \perp$ and $\top \equiv \perp$. As usual, $G$ (globally) is defined by $G\phi \equiv 1R\phi$ and $F$ (finally) by $F\phi \equiv \top U \phi$.

The semantics of $THT$ is defined via sequences of pairs of sets of atoms. A $TEL$-trace (or simply trace, if clear from context) is an infinite sequence $(H, T)$ of pairs $(H_i, T_i)$, where $H_i \subseteq T_i \subseteq P$ for each $i \geq 0$. Both $H$ and $T$ are traces as usual, i.e., infinite sequences $H = H_0, H_1, \ldots$ resp. $T = T_0, T_1, \ldots$ of sets of atoms.

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

- $I, k \not\equiv \perp$
- $I, k \equiv p$ if $p \in H_k$, for any atom $p \in P$
- $I, k \equiv Y \phi$ if $I, k − 1 \equiv \phi$ and $k > 0$
- $I, k \equiv \phi \lor \psi$ if $I, k \equiv \phi$ or $I, k \equiv \psi$
- $I, k \equiv \phi \land \psi$ if $I, k \equiv \phi$ and $I, k \equiv \psi$
- $I, k \equiv \phi \rightarrow \psi$ if $(T, T), k \not\equiv \phi$ or $(T, T), k \equiv \psi$, and $(I, k \not\equiv \phi) \land (I, k \equiv \psi)$
- $I, k \equiv X \phi$ if $I, k + 1 \equiv \phi$
- $I, k \equiv \phi \land \psi$ if there is $j \geq k$ s.t. $I, j \equiv \psi$, and for all $j' \in [k, j)$, $I, j' \equiv \phi$
- $I, k \equiv \phi$ if for all $j \geq k$ s.t. $I, j \not\equiv \psi$, there exists $j' \in [k, j)$, $I, j' \equiv \phi$

A $THT$-trace $I$ is a model for a formula $\phi$ if $I, 0 \not\equiv \phi$.

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

$$T \equiv M \equiv GEM, \text{ where } EM = \land_{p \in P}(p \lor \neg p).$$

Proposition 1 (cf. [Cabalar and Demri, 2011]). For $\phi \in L$ and $I = (H, T), I \equiv \phi \land T \equiv M$ iff $T \equiv_{LTL} \phi$ and $T = H$.

For traces $T, T'$, and $O$, we write $T \preceq_O T'$ if $O_i \subseteq T_i \subseteq T'_i$ holds for every $i \geq 0$. Furthermore, for $THT$-traces $I = (H, T)$, $I' = (H', T')$, and a trace $O$, we write $I \preceq_O I'$ if $H \subseteq H'$ and $T = T'$. Intuitively, $\preceq_O$ serves for defining $H$-minimality modulo observations; they are present in $O$ 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 $O$, a trace $O \preceq T$ is a temporal equilibrium model of a formula $\phi \in L$ w.r.t. $O$, if (i) $(T, T) \not\equiv \phi$, i.e., $T$ is a total $THT$ model of $\phi$, and (ii) no $H \preceq T$ exists s.t. $H \preceq_O T$ and $(H, T) \not\equiv \phi$, i.e., $(T, T)$ is minimal w.r.t. $O$.

If the observation trace $O$ is empty, i.e., $O = \emptyset^ω$, Definition 2 yields classical $TEL$ satisfaction [Aguado et al., 2013].

Given two traces $T, O$, and a formula $\phi \in L$, we denote by $T \equiv^O_{TEL} \phi$ that $T$ is an equilibrium trace of $\phi$ modulo observations $O$, where we may drop $O$ 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 $L$ that resembles and extends the usual logic programming syntax.

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

- $r_0$: $\text{switch} \lor X \text{anomaly}$
- $r_1$: $G(\text{switch} \land \lnot \text{light} \land \lnot X \text{anomaly} \rightarrow X \text{light})$
- $r_2$: $G(\text{switch} \land \lnot X \text{anomaly} \rightarrow X \text{change\_light})$
- $r_3$: $G(\text{light} \land X \text{anomaly} \rightarrow X \text{change\_light})$
- $r_4$: $G(\lnot X \text{change\_light} \land \lnot \rightarrow X \text{light})$
- $r_5$: $G(\text{power\_failure} \rightarrow \text{anomaly})$
- $r_6$: $G(\text{switch} \land X \text{switch} \rightarrow 1)$

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 $T = \{s\}, \{c, l\}, \{a, p, f, c\}, \{a, p\}^ω$ is an equilibrium trace of $π_{EX}$ w.r.t. $O = \{s\}, \emptyset, \{p\}^ω$. 3343
T^f \models \varnothing^\omega \text{ is an equilibrium trace } \iff \top \\
T^f \nmodels \text{ no equilibrium trace } T^f, T^f \text{ exists } \iff \bot

Figure 1: TEL₃ verdict for trace-prefix T^f under scrutiny.

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 Y) are added, cf. [Lichtenstein et al., 1985]; this extends to the language L.

Lemma 1. Deciding whether \( \varphi \in L \) is THT- (resp. TEL-)

is 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 THT₃ semantics with truth-values true (\( \top \)), false (\( \bot \)), and undefined (\( ? \)).

Given a THT-trace \( I = (H, T) \), its k-prefix (resp. k-suffix or suffix at k) is the sequence \((H_0, T_0), \ldots, (H_k, T_k)\) (resp. \(I^k = (H_k, T_k), (H_{k+1}, T_{k+1}), \ldots, 0 \leq k \). A prefix of I is any k-prefix \( I^f \) of I whose length, denoted by \(|I^f|\), is k+1. We call I an extension of \( I^f \) w.r.t. observation trace O, if \( H \preceq O \) holds; by \( ext(I^f, O) \), we denote the set of all such I. We denote by \( Pre_{THT} \) the set of all possible prefixes. Similarly as above, we may write \( TEL \) instead of \( (T^f, \bot) \) and omit O.

Definition 3 (THT₃ semantics, cf. [Soldà et al., 2023]). The truth value of \( \varphi \in L \) w.r.t. a prefix \( I^f \) and an observation trace O is as follows:

\[
I^f \models_{THT}^O \varphi = \begin{cases} 
\top & \text{if } I \models \varphi \text{ for every } I \in ext(I^f, O), \\
\bot & \text{if } I \nmodels \varphi \text{ for every } I \in ext(I^f, O), \\
? & \text{otherwise.}
\end{cases}
\]

Here O is a parameter to define a 3-valued logic for TEL, where minimality over traces matters. If O is empty, Defn. 3 is the THT version of the LTL₃ logic of Bauer et al. [2007]; further restriction to total traces yields their LTL₃ semantics.

TEL₃ semantics The 3-valued semantics of TEL, depicted in Figure 1, has been defined as follows:

Definition 4 (TEL₃ semantics, cf. [Soldà et al., 2023]). For a total prefix \( T^f \) of length k, \( \varphi \in L \), and observation trace O,

\[
T^f \models_{TEL}^O \varphi = \begin{cases} 
\top & \text{if } T^f \cdot O^k \models_{TEL}^O \varphi, \\
\bot & \text{if } T^f \nmodels_{TEL}^O \varphi, \forall T \in ext(T^f, O), \\
? & \text{otherwise.}
\end{cases}
\]

Note that the single minimal LTL model in \( ext(T^f, O) \) is used for the verdict \( v = \top \); this is because nonmonotonicity of TEL may compromise the persistence of verdict \( v \) when \( T^f \) 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 LTL₃, or THT₃, as \( v = \top \) would be valid.

For example, for \( \varphi = G(q \lor \neg q) \land G(q \rightarrow p) \) we have \( \varnothing \models_{TEL}^O \varphi = \top \), while \( \varnothing \models (q, p) \models_{TEL}^O \varphi = ? \). The reading of verdict \( \top \) is that if nothing is left to be proven (i.e., empty T-suffix), this extension yields an equilibrium trace.

THT has the so-called persistence property, viz. that \( \langle H, T \rangle \models \varphi \implies T \models \varphi \). As we deal with three possible verdicts and a fixed prefix, only a weaker form holds.

Lemma 2 (Persistence for THT₃). For every \( \varphi \in L \), prefix \( I^f = \langle H^f, T^f \rangle \), and observation trace O, \( I^f \models_{THT}^O \varphi \neq \bot \) implies \( I^f \models_{THT}^O \varphi \neq \bot \).

Note that both (I1) \( I^f \models_{THT}^O \varphi = \bot \) and (T1) \( T^f \models_{THT}^O \varphi = \top \) may hold. E.g., for \( \varphi = p \lor \neg p \), empty O, \( T^f = \{ p \} \), and \( I^f = \{ \} \), we have \( I^f \models \neg p \lor \neg p \) while \( T^f \models p \lor \neg p \). Similarly, both (I2) \( I^f \models_{THT}^O \varphi = \top \) and (T2) \( T^f \models_{THT}^O \varphi = ? \) may hold, e.g. for \( \varphi = p \rightarrow Xp \) and O, \( I^f, T^f \) as before.

3.1 Complexity

We now analyze the complexity of the THT₃ and the TEL₃ semantics. We assume that prefixes of traces are explicitly given. We consider here that O is described by an LTL-formula \( \psi \), such that for each model T of \( \psi \) and position \( i \geq 0, T, i \models_{LTL} p \text{ iff } p \in O_i \).

We start by showing a negative result about the complexity of the verdict of THT₃ and of TEL₃ semantics, that justifies the introduction of the progression function P, and, respectively, \( P_{TEL} \), which are more efficient.

Theorem 1. Given \( \varphi \in L \), a prefix \( I^f = \langle H^f, T^f \rangle \), and an arbitrary observation trace O of an LTL₃ formula \( \psi \), namely O \( =_{LTL} \psi \), deciding \( I^f \models_{THT}^O \varphi \) for \( \varphi \in \{ \top, \bot \} \) is PSPACE-complete, and PSPACE-hardness holds if \( v = \top \) is fixed arbitrarily and 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 \( I^f \) into a formula \( \psi_{I^f} \) and use the following formula \( \psi_O \) to constrain the models by the observations from O, which are expressed by the LTL formula \( \psi_O \):

\[ \psi_O = \psi \land T \models_{\mathcal{M}_P} \land G(\overline{p} \rightarrow p), \]

where \( \psi \) is \( \psi \) with all occurrences of p replaced by \( \overline{p} \) and \( T \models_{\mathcal{M}_P} \) is \( T \models_{\mathcal{M}} \) restricted to the atoms in \( P = \{ \overline{p} : p \in \mathcal{P} \} \).

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

For the TEL₃-semantics, we obtain a complexity picture analogous to Theorem 1.

Theorem 2. Given \( \varphi \in L \), an LTL-prefix \( T^f \), and a formula \( \psi \) describing an observation trace O, deciding \( T^f \models_{TEL}^O \varphi \) for a given \( \varphi \in \{ \top, \bot, ? \} \) 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 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, \( O \) is empty. For \( v = \tau \), the problem is in PSPACE: there are only exponentially many \( I^f \) for which must check \( I^f \cdot O^k = \phi \), where \( k \) is the length of \( T^f \). For each such \( I^f \), this reduces to an LTL entailment test, which is in PSPACE, and looping through all \( I^f \) is feasible in PSPACE. Notably, if in addition, \( O \) is empty, the complexity drops to co-NP as the test \( I^f \cdot O^k = \phi \) 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\( ^P \) 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 Roșu, 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 \( \phi = p \rightarrow Fq \) 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 \( \phi \) as \( Fq \). The main idea is that we propagate \( Fq \) until we see \( q \) in the trace, when we can finally conclude \( \tau \).

We denote by \( L_c \) 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 \( U \) and \( R \) (only \( F \) and \( G \) were considered). We now introduce THT progression.

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

1. \( P(\bot, I^f, i) = \bot \)
2. \( P(p, I^f, i) = \top \) if \( p \in H^f_i \) and \( p \in \mathcal{P} \)
3. \( P(p, I^f, i) = \bot \) if \( p \notin H^f_i \), and \( p \in \mathcal{P} \)
4. \( P(\phi_1 \lor \phi_2, I^f, i) = P(\phi_1, I^f, i) \lor P(\phi_2, I^f, i) \)
5. \( P(\phi_1 \land \phi_2, I^f, i) = P(\phi_1, I^f, i) \land P(\phi_2, I^f, i) \)
6. \( P(\phi_1 \rightarrow \phi_2, I^f, i) = P(\phi_1, I^f, i) \rightarrow P(\phi_2, I^f, i) \)
7. \( P(\phi_1 \rightarrow \phi_2, I^f, i) = \{ P(\phi_1, I^f, i) \rightarrow P(\phi_2, I^f, i) \} \land ( P(\phi_1, T^f, i) \rightarrow P(\phi_2, T^f, i) ) \)
8. \( P(\forall \phi, I^f, i) = P(\phi, I^f, i-1) \) if \( i > 0 \) else \( \bot \)
9. \( P(\exists \phi, I^f, i) = \phi \)
10. \( P(\phi_1 \lor \phi_2, I^f, i) = P(\phi_1, I^f, i) \lor P(\phi_2, I^f, i) \)
11. \( P(\phi_1 \land \phi_2, I^f, i) = P(\phi_1, I^f, i) \land P(\phi_2, I^f, i) \)

In addition, \( \tau \rightarrow \top \) and \( \top \rightarrow \phi \) are replaced by \( \top \) and \( \top \rightarrow \phi \) by \( \tau \), for every formula \( \phi \) and \( \rightarrow \in \{ \rightarrow, \rightarrow c \} \).

**Theorem 3 (THT verdict on prefixes).** For every prefix \( I^f \), observation trace \( O \), and formula \( \phi \in L_c \):

\[ P(\phi, I^f) = \psi \text{ implies } I^f \models^O_{\text{THT}_3} \phi \text{ for } v \in \{ \tau, \top \} . \]

4.1 Computing P Progression

In the next section, we show that a \( P \) application to a formula \( \phi \in L_c \) on a prefix \( I^f \) 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 \( n_\psi \) the number of \( \rightarrow \) and \( \rightarrow c \) symbols in \( \psi \), by \( n_\psi \) the number of \( U \) and \( R \) symbols in \( \psi \).

**Theorem 4.** Given \( \psi \in L_c \), a prefix \( I^f \) of length \( k \), and \( s \in [0, k) \), obtaining some \( \psi^s =_{\text{THT}} P(\psi, I^f, s) = \psi^{s+1} \) is feasible in polynomial time where (i) \( \psi^s \in \{ \tau, \top \} \) implies \( \psi^s = \psi^{s+1} \) and (ii) \( |\psi^s| \leq (n_\psi + 1)(n_\psi + 2)|\psi| \).

**Proof (Sketch).** The result is obtained by using two look-up tables \( T \) and \( H \), where each sub-formula \( \phi_i \) of \( \psi \) is progressed for \( T^f \) resp. \( I^f \) at every position \( t \in [0, k) \) to a formula \( \phi'_i \) that is memORIZED in \( T[\phi_i, t] \) resp. in \( H[\phi_i, t] \). One can then assemble \( \phi'_j \) from the progression results of its subformulas \( \phi_j \) stored in \( T \) and \( H \). The key to having a polynomial-size
output is the fact that for progression over a total trace, we
need to consider only one of \( \rightarrow \) and \( \rightarrow^{-} \) in the assembly as they yield equivalent results. Without this optimization, the application of \( P \) may lead to exponential outcomes. \( \square \)

Note that whenever \( \varphi_i \notin \{ T, \bot \} \), we can simply write \( T[\varphi_i, t] = t \) (or \( T[\varphi_i, t - 1] = t \) for the Y-case) if we are interested in the decision problem only.

For computing \( P(\phi, I') \), repeated application of \( P(\phi, I', k) \) does not allow us to conclude tractability, as \( \phi \) may grow exponentially when progression is applied due to \( U \) and \( R \) operators. As an extreme example, consider the case of nested \( U: \psi_i = \psi_{2i+1} U \psi_{2i+2} \) for \( i = 0, \ldots, 2^{n-1} - 1 \) and \( \psi_i = F \) for some \( p \in P \) 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, \( P(\psi_0, I') \) grows exponentially in the size of \( I' \).

Fortunately, we can avoid exponential explosion by sharing subformulas. Indeed, progressing in the example \( \psi_0 \) over \( \varnothing \) will yield two copies of \( \psi_i \) 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 \( \psi \) is any directed acyclic graph resembling the syntax tree of \( \psi \) with possible sharing of subtrees. Let us denote by \( P_{temp}(\phi, I', i) \) the formula as in Definition 5 but with result \( p \) in cases 2 and 3; i.e., \( p \) is not evaluated over \( I' \) and thus independent of \( I' \).

**Definition 7.** Given a DAG-representation of \( \psi \in L \) and \( \phi \in sub(\psi) \), we denote by \( N_{I'}^k \phi \) the DAG representing the formula \( P_{temp}(\phi, I', i) \) for any \( I' \) of length \( i \).

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

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

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

- \( N_{I'}^{k+1} \phi \) points to \( N_{I'}^k \phi \), if \( \phi \) has no temporal operators;
- \( N_{I'}^{k+1} X \phi \) points to \( N_{I'}^{k+1} \phi \);
- \( N_{I'}^{k+1} Y \phi \) points to \( N_{I'}^{k+1} \phi \);
- \( N_{I'}^{k+1} \psi \) points to a formula for \( N_{I'}^{k+1}(\phi_2 \lor N_{I'}^{k+1} \phi_1 \land N_{I'}^{k+1}(\phi_1 \cup \phi_2)); \) to avoid introducing nodes for new subformulas, we may view \( U \) progression as a ternary connective, where \( \phi_1 U \phi_2 = \bot \lor \top \land (\phi_1 \cup \phi_2) \). The case \( N_{I'}^{k+1}(\phi_1 R \phi_2) \) 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 T- part, successively for the 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 \( \psi \) when a new state is provided:

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

**Proof.** By Lemma 4, the DAG \( N_{I'}^k \psi \) for the template formula \( P_{temp}(\phi, I', 0) \) 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 subformulas; 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 \( \phi \in L \) on a total prefix \( I' \) is

\[
P_{TEL}(\phi, I') = \begin{cases} \top & \text{if } \phi' = \top \land \forall I' \in T' : \phi''(I') = \bot \\ \bot & \text{if } \phi' = \bot \lor \exists I' \in T' : \phi''(I') = \top \\ \bot & \text{otherwise}, \end{cases}
\]

where \( \phi' = P(\phi, (T', I')) \), \( \phi''(I') = P(\phi, (H', T')) \).

Definition 8 faithfully approximates TEL₃ semantics.

**Theorem 6 (TEL verdict on prefixes).** For any formula \( \phi \), prefix \( T' \), and observation trace \( O \), \( P_{TEL} \) progression is sound w.r.t. \( TEL_3 \)-verdict, i.e.,

\[
P_{TEL}(\phi, T') = v \text{ implies } T' = O_{TEL_3} \phi = v, \text{ for } v \in \{ \top, \bot \}.
\]

We note that while both THT₃ and TEL₃ 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 \( H' \in T' \) in Defn. 8 with \( H' \in O \) defined by \( O_i \subseteq H_i \wedge T_i \) for \( i \in [0, |T'|) \) and \( H_0 \neq T' \), Theorem 6 still holds.

Since we focus on online computations, we introduce an incremental \( P_{TEL} \) progression, where we resort to the \( Y \) 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 \( T^i \) for position \( i \in [0, |T^i|) \) is

\[
P_{\text{TEL}}^{\text{inc}}(\phi, T^i, i) = \begin{cases} \top & \text{if } \phi^i = \top \land \forall H^f \in T^i: \phi^i_0(H^f) = \bot \\ \bot & \text{if } \phi^i = \bot \lor \exists H^f \in T^i: \phi^i_0(H^f) = \top \\ \phi^Y & \text{otherwise} \end{cases}
\]

where \( \phi^i_0 = P(\phi, (T^f, T^i), i) \), \( \phi^i_0(H^f) = P(\phi, (H^f, T^i), i) \), \( \phi^Y \) is defined inductively by

- \( \cdot \gamma^Y = \cdot \gamma; \)
- \( (X \cdot \phi_1)^Y = \cdot \phi_1; \)
- \( (\cdot \phi_1)^Y = \cdot \phi_1; \)
- \( (\cdot \phi_2)^Y = \cdot \phi_2 \) for \( \cdot \in \{\neg, \land, \lor\}; \)
- \( (\cdot \phi_1 \cdot \phi_2)^Y = \cdot \phi_2 \lor (\cdot \phi_1 \cdot \phi_2); \)
- \( (\cdot \phi.R \cdot \phi_2)^Y = \cdot \phi_2 \land (\cdot \phi_1 \cdot \phi.R \cdot \phi_2). \)

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

Theorem 7. For every \( \phi \in \mathcal{L} \) and prefix \( T^f \), \( P_{\text{TEL}}(\phi, T^f) = v \) if \( P_{\text{TEL}}^{\text{inc}}(\phi, T^f) = v \).

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

5.1 Computing \( P_{\text{TEL}} \) Progression

The results above show an exponential complexity gap between \( \text{THT}_{3} \)-satisfiability (PSpace-complete) and \( \text{TEL} \)-satisfiability (EXPSpace-complete). We find a complexity gap between the \( P \) and the \( P_{\text{TEL}} \) approximation, but it is much smaller: while intractable, \( P_{\text{TEL}} \) 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 \( T^f \), deciding \( P_{\text{TEL}}(\phi, T^f) = v \) is (i) for \( v = \top \) co-NP-complete, (ii) for \( v = \bot \) NP-complete, and (iii) for \( v = \? \) DSpace-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 \( P_{\text{TEL}} \) progression requires NP/co-NP oracle calls, it may be as near-tractable approximation of the \( \text{TEL}_{3} \) semantics.

Note that the complexity proofs for \( P_{\text{TEL}} \)-progression exploit Theorem 5 for checks of valid prefixes \( \overline{T}^f \). The progression DAGs \( N_0^k \) of \( P(\cdot, \overline{T}^f) \) in Lemma 4 can be instantiated for them on the fly, and with \( P_{\text{TEL}}^{\text{inc}} \) we can easily extend \( N_0^k \) incrementally. It is possible to see that the DAG representation leads to an equivalent formula of the \( P_{\text{TEL}}^{\text{inc}} \). Taking the \( N_{i=0}^{k} \) as a starting point we proceed top-down adding the proper connectives and placing \( k-i \) many \( Y \) 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 \( \text{TEL} \), for which efficient \( P_{\text{TEL}} \) 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)
  
  \[ r : b_1 \land \ldots b_k \rightarrow c_1 \lor \cdots \lor c_l \quad \text{with } k, l \geq 0 \]

where all \( b_i, c_j \) are in \( \{p, Xp, \neg p, \neg Xp \mid p \in P\} \);
- (dynamic rule) \( \text{Gr} \), where \( r \) is an initial rule;
- (fulfillment rule) either \( G(Gr \rightarrow q) \) or \( G(p \rightarrow Fq) \), where \( p, q \) are atoms.

An initial or dynamic rule \( r \) is a constraint, if its head is \( \top \) (i.e., \( l=0 \)), and is a fact if its body is empty (\( k=0 \)) and \( l=1 \). We call \( \{p, Xp \mid p \in P\} \) 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 \( \text{TEL} \) [Cabalar, 2010] and harness the full complexity of \( \text{TEL} \). This is paralleled by the fact that the complexity results in Theorem 8 carry over to temporal programs; in particular that \( P_{\text{TEL}} \) progression is Dim-complete.

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

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

Proof (Sketch). Let \( k = |T^f| - 1 \). By Lemma 4 and Theorem 5, we can compute \( N_0^k \) in polynomial time, and evaluate it on \( T^f \) in polynomial time, obtaining as outcome \( v_{\overline{T}^f} \). According to the conditions of \( P_{\text{TEL}} \) (Defn 8), we may have to consider some (all) \( H^f \in T^f \) and evaluate \( N_0^k \) over \( T^f \). \( N_0^k \) represents a set of implications, and after evaluating in \( N_0^k \) the negated atoms over \( T^f \), the atemporal rules (not involving any temporal operator) in \( N_0^k \) amount to a set of Horn clauses. We can compute the least model \( M^k \) of these clauses in polynomial time resp. find they have no model. As for the remaining implications in \( N_0^k \), sub-formulas within the scope of an \( X, G, F \) operator cannot lead to \( \top \) or \( \bot \), and thus either formulas are removed (when \( \phi^0(H^f) \) should yield \( \bot \) resp. subformulas (when \( \phi^0(H^f) \) should yield \( \top \)) before computing \( M^k \). From \( v_{\overline{T}^f} \) and \( M^k \), we then easily obtain \( P_{\text{TEL}}(T^f, \pi) \).
classes from the literature, such as the STLP fragment.

Proposition 4. Let $DG$ be a temporal program.

For any program $G$, we can generalize Theorem 9 to a larger class of programs.

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 $DG_π = (V, E)$ where $V = \{p^i \mid p \in P$ and $i \geq 0\}$ and (i) $(a, b) \in E$ if $a \in H(r)$ and $b \in B^*(r)$ for some rule $X^k \in π^ω$, (ii) $(q^i, p^j) \in E$ for $k \geq 0$ if $X^i(G \rightarrow q) \in π^ω$, and (iii) $(q^k, p^k) \in E$ for $k \geq 0$ if $X^k(p \rightarrow F q) \in π$.

Intuitively, (ii) and (iii) are added because we can generate $G_p$ and $F_p$ as a conjunction respect disjunction of $X^i_p$, $i \geq 0$.

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 $DG$-nodes, then (a) no rule $X^r \in π^ω$ satisfies $(a, b) \in E$, and (b) if $a = q^k$ and $b = q^l$ with $k \neq l$, then no fulfillment rule $X^k(p \rightarrow F q) \in π^ω$ 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 body rule. This property generalizes to temporal programs.

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

Proposition 3. For any hcf temporal program $π$, $π^r$ and $π^ω$ have the same equilibrium models.

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

Theorem 10. Given a temporal program $π$ and a prefix $T^f$, deciding whether $P_{TEL}(T^f, π) = v$ for $v \in \{1, \top, \bot\}$ 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) $B \land N \rightarrow H$, or (ii) $r$: $B \land X B' \land N \land X N' \rightarrow X H$, and dynamic rules $G(r)$ 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 $p \in P$, and $H, H'$ are disjunctions of atoms. Notably, program $π^ω$ is of this form. We show that for STLP programs, the hcf property can be efficiently verified. Let $DG_π^ω, i \geq 0$, be the subgraph of $DG_π^ω$ induced by all nodes $p^i$, where $p \in P$ and $j \leq i$.

Proposition 4. A STLP program $π$ is hcf iff $DG_π^ω$ satisfies the hcf-condition.

Figure 4 shows the clipped dependency graph $DG_π^ω$, from which the hcf property of $π^ω$ is immediately verified.

Notably, we can use $DG_π^ω$ to efficiently progress programs $π$ whose unfolding on the prefix $T^f$ is hcf. Let us say program $π$ is $k$-hcf if $DG_π^ω$ satisfies the hcf-condition.

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

The $k$-hcf condition can be efficiently checked along the progression, as $DG_π^ω$ 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.

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

Tools for stable traces are clingo [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 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 clingo [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. Walega 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 a kin 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$Ω$ and TEL$Ω$, 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$Ω$ and TEL$Ω$ 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


[de Leng and Heintz, 2018] Daniel de Leng and Fredrik Heintz. Partial-state progression for stream reasoning with


