24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Implementing Dynamic Answer Set Programming over finite traces

Pedro Cabalar[1] and **Martín Diéguez**[2] and **Torsten Schaub**[3] and **Francois Laferriere**[3]

Abstract. We introduce an implementation of an extension of Answer Set Programming (ASP) with language constructs from dynamic (and temporal) logic that provides an expressive computational framework for modeling dynamic applications. Starting from logical foundations, provided by dynamic and temporal equilibrium logics over finite linear traces, we develop a translation of dynamic formulas into temporal logic programs. This provides us with a normal form result establishing the strong equivalence of formulas in different logics. Our translation relies on the introduction of auxiliary atoms to guarantee polynomial space complexity and to provide an embedding that is doomed to be impossible over the same language. Finally, the reduction of dynamic formulas to temporal logic programs allows us to extend ASP with both approaches in a uniform way and to implement both extensions via temporal ASP solvers such as *telingo*.

1 Introduction

Humans are not bothered at all when it comes to choosing a way home among a plethora of alternatives. Similarly, in reasoning about action or planning, the underlying specifications admit an abundance of feasible plans. Among them, we usually find only a few reasonable alternatives while the majority bear an increasing number of redundancies, leading to infinite solutions in the worst case. Popular ways to counterbalance this are to extend the specification by objectives, like shortest plans, and/or imposing limits on the plan length. Both are often implemented with optimization procedures and/or incremental reasoning methods (extending insufficient plan lengths). However, computing a valid plan with such techniques usually involves solving numerous sub-problems of nearly the same scale as the actual problem — not to mention that solving somehow optimal plans is often particularly hard (due to phase transition phenomena). This rules out such techniques when it comes to highly demanding dynamic problems, as we witnessed in applications to robotic intra-logistics [14].

This motivates our approach to pair action theories with control theories in order to restrict our attention to plans selected by the control theory among all feasible ones induced by the action theory. This approach was pioneered by Levesque et al. in [20] by combining action theories in the Situation Calculus with control programs expressed in Golog. The rough idea is that a plan induced by an action theory must be compatible with a run of the associated Golog program. Although Golog's design was inspired by Dynamic Logic [17], its semantics is given by a reduction to first-order logic. Unlike this, we developed in [3, 6] the foundations of an approach integrating Answer Set Programming (ASP [21]) with (linear) Dynamic Logic. More precisely, we are interested in the combination of the logic of Here-and-There (HT [19]) with linear Dynamic Logic over finite traces (LDL_f [11]), called Dynamic logic of Here-and-There (DHT_f) and particularly its non-monotonic extension, Dynamic Equilibrium Logic (DEL_f [6]). We review both logics in Section 2. ASP as such is known to constitute a fragment of Equilibrium Logic [22].

In what follows, our focus lies on implementing temporal and dynamic ASP via a reduction to regular ASP. This aligns with the above discussion insofar as temporal logic programs, featuring one step operators, are well-suited for providing action theories, while dynamic formulas allow for imposing compatibilities with path expressions, that amount to regular expressions over primitive actions. [4] To this end, in this paper, we present the following contributions:

(i) we develop a three-valued characterization of DHT_f;
(ii) we establish a normal form for DHT_f showing that dynamic formulas can be reduced to (so-called) temporal logic programs;
(iii) we use this reduction to implement a solver accepting dynamic formulas in DEL_f.

For (i), we extend the three-valued characterization from [16] to dynamic formulas, something that allows for greatly simplifying proofs and has thus benefits well beyond this paper. Also, it is, to the best of our knowledge, the first time this type of construction is used to capture dynamic logics and thus path expressions. This three-valued definition also allows for establishing (ii), which shows that any dynamic formula can be equivalently reduced to a syntactic fragment called temporal logic programs. This translation relies on the introduction of auxiliary atoms (in a Tseitin-style [25]) for, first, guaranteeing that its result is of polynomial size wrt the input formula, and, second, surmounting the fact that translations of dynamic into temporal formulas are usually impossible without extending the language. We explain both issues in more detail in Section 3. Finally, the great benefit of this reduction is that we can use it for (iii), that is, implementing dynamic formulas in DEL_f, since temporal logic programs can be processed by an existing solver for temporal ASP, viz. *telingo* [8]. We describe the resulting implementation in Section 4 and show the potential impact of pairing action and control theories via an empirical analysis of an elevator scenario borrowed from [20].

2 Linear Dynamic Equilibrium Logic

We start from the syntax of *Linear Dynamic Logic* (LDL) defined in [11]. Given a set $\mathcal{A}$ of propositional variables (called *alphabet*), *dynamic formulas* φ and *path expressions* ρ are mutually defined by

[1] University of Corunna, Spain
[2] University of Pau, France
[3] University of Potsdam, Germany

[4] Temporal formulas constitute a proper fragment of DHT_f (cf. Section 2).

the pair of grammar rules:

$$\varphi ::= a \mid \bot \mid \top \mid \; [\rho]\,\varphi \;\mid\; \langle\rho\rangle\,\varphi$$
$$\rho ::= \tau \mid \varphi? \mid \rho+\rho \mid \rho\,;\,\rho \mid \rho^* \mid \rho^-$$

This syntax is similar to the one of Dynamic Logic (DL [17]) but differs in the construction of atomic path expressions: while DL uses a different alphabet for *atomic actions*, in LDL there is a unique alphabet $\mathcal{A}$ (atomic propositions) and the only atomic path expression is the constant $\tau \notin \mathcal{A}$ (read as "step") that we also write as $\top$ (see below), overloading the constant truth symbol. As we show further below, the above language allows us to capture several derived operators, like the Boolean and temporal ones:

$$\begin{array}{rcl@{\qquad}rcl}
\varphi \land \psi & \overset{def}{=} & \langle\varphi?\rangle\,\psi & \varphi \lor \psi & \overset{def}{=} & \langle\varphi? + \psi?\rangle\,\top \\
\varphi \to \psi & \overset{def}{=} & [\varphi?]\,\psi & \neg\varphi & \overset{def}{=} & \varphi \to \bot \\
\mathbf{F} & \overset{def}{=} & [\top]\bot & \mathbf{I} & \overset{def}{=} & [\top^-]\bot \\
\circ\varphi & \overset{def}{=} & \langle\top\rangle\,\varphi & \bullet\varphi & \overset{def}{=} & \langle\top^-\rangle\,\varphi \\
\widehat{\circ}\varphi & \overset{def}{=} & [\top]\,\varphi & \widehat{\bullet}\varphi & \overset{def}{=} & [\top^-]\,\varphi \\
\Diamond\varphi & \overset{def}{=} & \langle\top^*\rangle\,\varphi & \blacklozenge\varphi & \overset{def}{=} & \langle\top^{*-}\rangle\,\varphi \\
\Box\varphi & \overset{def}{=} & [\top^*]\,\varphi & \blacksquare\varphi & \overset{def}{=} & [\top^{*-}]\,\varphi \\
\varphi\,\mathbf{U}\,\psi & \overset{def}{=} & \langle(\varphi?;\top)^*\rangle\,\psi & \varphi\,\mathbf{S}\,\psi & \overset{def}{=} & \langle(\varphi?;\top)^{*-}\rangle\,\psi \\
\varphi\,\mathbf{R}\,\psi & \overset{def}{=} & (\psi\,\mathbf{U}\,(\varphi\land\psi)) \lor \Box\psi & \varphi\,\mathbf{T}\,\psi & \overset{def}{=} & (\psi\,\mathbf{S}\,(\varphi\land\psi)) \lor \blacksquare\psi
\end{array}$$

All connectives are defined in terms of the dynamic operators $\langle\cdot\rangle$ and $[\cdot]$. This involves the Booleans' $\land$, $\lor$, and $\to$, among which the definition of $\to$ is most noteworthy since it hints at the implicative nature of $[\cdot]$. Negation $\neg$ is then expressed via implication, as usual in HT. Then, $\langle\cdot\rangle$ and $[\cdot]$ also allow defining the future temporal operators $\mathbf{F}$, $\circ$, $\widehat{\circ}$, $\Diamond$, $\Box$, $\mathbf{U}$, $\mathbf{R}$, standing for *final, next, weak next, eventually, always, until,* and *release*, and their past-oriented counterparts: $\mathbf{I}$, $\bullet$, $\widehat{\bullet}$, $\blacklozenge$, $\blacksquare$, $\mathbf{S}$, $\mathbf{T}$. The weak one-step operators, $\widehat{\circ}$ and $\widehat{\bullet}$, are of particular interest when dealing with finite traces, since their behavior differs from their genuine counterparts only at the ends of a trace. In fact, $\widehat{\circ}\varphi$ can also be expressed as $\circ\varphi \lor \mathbf{F}$ (and $\widehat{\bullet}$ as $\bullet\varphi \lor \mathbf{I}$). A formula is *propositional*, if all its connectives are Boolean, and *temporal*, if it includes only Boolean and temporal ones. As usual, a *(dynamic) theory* is a set of (dynamic) formulas. Following the definition of *linear* DL (LDL) in [11], we sometimes use a propositional formula ϕ as a path expression actually standing for $(\phi?;\tau)$. This means that the reading of $\top$ as a path expression amounts to $(\top?;\tau)$ which is just equivalent to τ, as we see below. Another abbreviation is the sequence of n repetitions of some expression ρ defined as $\rho^0 \overset{def}{=} \top?$ and $\rho^{n+1} \overset{def}{=} \rho;\rho^n$. For instance, $\rho^3 = \rho;\rho;\rho;\top?$ which amounts to $\rho;\rho;\rho$, as we see below.

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

Although DHT shares the same syntax as LDL, its semantics relies on traces whose states are pairs of sets of atoms. A *Here-and-There trace* (for short HT-*trace*) of length λ over alphabet $\mathcal{A}$ is a sequence of pairs $\langle H_i, T_i\rangle_{i\in[0..\lambda)}$ such that $H_i \subseteq T_i \subseteq \mathcal{A}$ for any $i \in [0..\lambda)$. As before, an HT-trace is infinite if $\lambda = \omega$ and finite otherwise. The intuition of using these two sets stems from HT and Equilibrium Logic: atoms in H_i are those that can be proved; atoms not in T_i are those for which there is no proof; and, finally, atoms in $T_i \setminus H_i$ are assumed to hold, but have not been proved. We often represent an HT-trace as a pair of traces $\langle\mathbf{H},\mathbf{T}\rangle$ of length λ where $\mathbf{H} = \langle H_i\rangle_{i\in[0..\lambda)}$ and $\mathbf{T} = \langle T_i\rangle_{i\in[0..\lambda)}$ and $\mathbf{H} \leq \mathbf{T}$. The particular type of HT-traces that satisfy $\mathbf{H} = \mathbf{T}$ are called *total*. Given any HT-trace $\mathbf{M} = \langle\mathbf{H},\mathbf{T}\rangle$, we define DHT satisfaction of formulas, namely, $\mathbf{M}, k \models \varphi$, in terms of an accessibility relation for path expressions $\|\rho\|^{\mathbf{M}} \subseteq \mathbb{N}^2$ whose extent depends again on $\models$ by double, structural induction.

Definition 1 (DHT **satisfaction [6])** *An* HT-*trace* $\mathbf{M} = \langle\mathbf{H},\mathbf{T}\rangle$ *of length λ over alphabet $\mathcal{A}$* satisfies *a dynamic formula φ at time point $k \in [0..\lambda)$, written $\mathbf{M}, k \models \varphi$, if the following conditions hold:*

1. $\mathbf{M}, k \models \top$ *and* $\mathbf{M}, k \not\models \bot$
2. $\mathbf{M}, k \models a$ *if* $a \in H_k$ *for any atom* $a \in \mathcal{A}$
3. $\mathbf{M}, k \models \langle\rho\rangle\,\varphi$ *if* $\mathbf{M}, i \models \varphi$ *for some* i *with* $(k,i) \in \|\rho\|^{\mathbf{M}}$
4. $\mathbf{M}, k \models [\rho]\,\varphi$ *if* $\mathbf{M}', i \models \varphi$ *for all* i *with* $(k,i) \in \|\rho\|^{\mathbf{M}'}$ *for both* $\mathbf{M}' = \mathbf{M}$ *and* $\mathbf{M}' = \langle\mathbf{T},\mathbf{T}\rangle$

where, for any HT-*trace* $\mathbf{M}$, $\|\rho\|^{\mathbf{M}} \subseteq \mathbb{N}^2$ *is a relation on pairs of time points inductively defined as follows.*

5. $\|\tau\|^{\mathbf{M}} \overset{def}{=} \{(k,k+1) \mid k, k+1 \in [0..\lambda)\}$
6. $\|\varphi?\|^{\mathbf{M}} \overset{def}{=} \{(k,k) \mid \mathbf{M}, k \models \varphi\}$
7. $\|\rho_1 + \rho_2\|^{\mathbf{M}} \overset{def}{=} \|\rho_2\|^{\mathbf{M}} \cup \|\rho_2\|^{\mathbf{M}}$
8. $\|\rho_1\,;\,\rho_2\|^{\mathbf{M}} \overset{def}{=} \{\,(k,i) \mid (k,j) \in \|\rho_1\|^{\mathbf{M}}$ *and* $(j,i) \in \|\rho_2\|^{\mathbf{M}}$ *for some* $k\,\}$
9. $\|\rho^*\|^{\mathbf{M}} \overset{def}{=} \bigcup_{n\geq 0} \|\rho^n\|^{\mathbf{M}}$
10. $\|\rho^-\|^{\mathbf{M}} \overset{def}{=} \{(k,i) \mid (i,k) \in \|\rho\|^{\mathbf{M}}\}$

We see that $\langle\rho\rangle\,\varphi$ and $[\rho]\,\varphi$ quantify over time points i that are reachable via path expression ρ from the current point k, that is, $(k,i) \in \|\rho\|^{\mathbf{M}} \subseteq [0..\lambda) \times [0..\lambda)$. An HT-trace $\mathbf{M}$ is a *model* of a dynamic theory Γ if $\mathbf{M}, 0 \models \varphi$ for all $\varphi \in \Gamma$. We write $\mathrm{DHT}(\Gamma,\lambda)$ to stand for the set of DHT models of length λ of a theory Γ, and define $\mathrm{DHT}(\Gamma) \overset{def}{=} \bigcup_{\lambda=0}^{\omega} \mathrm{DHT}(\Gamma,\lambda)$, that is, the whole set of models of Γ of any length. When $\Gamma = \{\varphi\}$ we just write $\mathrm{DHT}(\varphi,\lambda)$ and $\mathrm{DHT}(\varphi)$.

A formula φ is a *tautology* (or is *valid*), written $\models \varphi$, iff $\mathbf{M}, k \models \varphi$ for any HT-trace $\mathbf{M}$ and any $k \in [0..\lambda)$. We call the logic induced by the set of all tautologies *(Linear) Dynamic logic of Here-and-There* (DHT for short). Two formulas φ, ψ are said to be *equivalent*, written $\varphi \equiv \psi$, whenever $\mathbf{M}, k \models \varphi$ iff $\mathbf{M}, k \models \psi$ for any HT-trace $\mathbf{M}$ and any $k \in [0..\lambda)$. This allows us to replace φ by ψ and vice versa in any context, and is the same as requiring that $\varphi \leftrightarrow \psi$ is a tautology. Note that this relation, $\varphi \equiv \psi$, is stronger than coincidence of models, viz. $\mathrm{DHT}(\varphi) = \mathrm{DHT}(\psi)$. For instance, $\mathrm{DHT}(\bullet\top) = \mathrm{DHT}(\langle\top^-\rangle\,\top) = \emptyset$ because models are checked at the initial situation $k = 0$ and there is no previous situation at that point, so $\mathrm{DHT}(\bullet\top) = \mathrm{DHT}(\bot)$. However, in general, $\bullet\top \not\equiv \bot$ since $\bullet\top$ is satisfied for any $k > 0$ (for instance $\circ\bullet\top \not\equiv \circ\bot$ but $\circ\bullet\top \equiv \top$ instead). As with formulas, we say that path expressions ρ_1, ρ_2 are *equivalent*, written $\rho_1 = \rho_2$, whenever $\|\rho_1\|^{\mathbf{M}} = \|\rho_2\|^{\mathbf{M}}$ for any HT-trace $\mathbf{M}$.

The following equivalences of path expressions allow us to push the converse operator inside, until it is only applied to τ.

Proposition 1 ([6]) *For all path expressions ρ_1, ρ_2 and ρ and for all formulas φ, the following equivalences hold:*

$$(\rho^-)^- = \rho \qquad (\varphi?)^- = \varphi? \qquad (\rho^*)^- = (\rho^-)^*$$
$$(\rho_1 + \rho_2)^- = \rho_1^- + \rho_2^- \qquad (\rho_1;\rho_2)^- = \rho_2^-\,;\rho_1^-$$

We say that φ is in *converse normal form* if all occurrences of the converse operator in φ are applied to τ.

We now introduce non-monotonicity by selecting a particular set of traces that we call *temporal equilibrium models*. First, given an arbitrary set $\mathfrak{S}$ of HT-traces, we define the ones in equilibrium as follows.

Definition 2 (Temporal Equilibrium/Stable models [6]) *Let $\mathfrak{S}$ be some set of* HT*-traces. A total* HT*-trace $\langle \mathbf{T}, \mathbf{T} \rangle \in \mathfrak{S}$ is an* equilibrium trace *of $\mathfrak{S}$ iff there is no other $\langle \mathbf{H}, \mathbf{T} \rangle \in \mathfrak{S}$ such that $\mathbf{H} < \mathbf{T}$.*

If $\langle \mathbf{T}, \mathbf{T} \rangle$ is such an equilibrium trace, we also say that trace $\mathbf{T}$ is a *stable trace* of $\mathfrak{S}$. We further talk about *temporal equilibrium* or *temporal stable models* of a theory Γ when $\mathfrak{S} = \mathrm{DHT}(\Gamma)$, respectively.

We write $\mathrm{DEL}(\Gamma, \lambda)$ and $\mathrm{DEL}(\Gamma)$ to stand for the temporal equilibrium models of $\mathrm{DHT}(\Gamma, \lambda)$ and $\mathrm{DHT}(\Gamma)$ respectively. Note that stable traces in $\mathrm{DEL}(\Gamma)$ are also LDL-models of Γ and, thus, DEL is stronger than LDL. Besides, as the ordering relation among traces is only defined for a fixed λ, it is easy to see the following result:

Proposition 2 ([6]) *The set of temporal equilibrium models of Γ can be partitioned by the trace length λ, that is, $\bigcup_{\lambda=0}^{\omega} \mathrm{DEL}(\Gamma, \lambda) = \mathrm{DEL}(\Gamma)$.*

(Linear) *Dynamic Equilibrium Logic* (DEL) is the non-monotonic logic induced by temporal equilibrium models of dynamic theories. We obtain the variants DEL_ω and DEL_f by applying the corresponding restriction to infinite or finite traces, respectively.

To illustrate non-monotonicity, take the formula:

$$[(\neg h)^*]\,(\neg h \to s) \tag{1}$$

whose reading is "keep sending an sos (s) while no help (h) is perceived." Intuitively, $[(\neg h)^*]$ behaves as a conditional referring to any future state after $n \geq 0$ repetitions of $(\neg h?; \top)$. Then, $\neg h \to s$ checks whether h fails one more time at $k = n$: if so, it makes s true again. Without additional information, this formula has a unique temporal stable model per each length λ satisfying $\Box(\neg h \wedge s)$, that is, h is never concluded, and so, we repeat s all over the trace. Suppose we add now the formula $\langle \top^5 \rangle\, h$, that is, h becomes true after five transitions. Then, there is a unique temporal stable model for each $\lambda > 5$ satisfying:

$$\langle (\neg h \wedge s)^5; h \wedge \neg s; (\neg h \wedge \neg s)^* \rangle\, \top$$

Clearly, $\Box(\neg h \wedge s)$ is not entailed any more (under temporal equilibrium models) showing that DEL is non-monotonic.

To conclude this section, we provide an alternative three-valued characterization of DHT that is particularly useful for formal elaborations involving auxiliary atoms. This alternative characterization relies on the idea of temporal three-valued interpretation in [4] for the case of TEL and is inspired, in its turn, in the characterization of HT in terms of Gödel's G_3 logic [16]. Under this orientation, we deal with three truth values $\{0, 1, 2\}$ standing for: 2 (or proved true) meaning satisfaction "here"; 0 (or assumed false) meaning falsity "there"; and 1 (potentially true) that are formulas assumed true but not proved. Given an HT-trace $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ we define its associated *truth valuations* as a pair of mutually recursive functions $\boldsymbol{m}(k, \varphi)$ and $\boldsymbol{m}(k, i, \rho)$ that assign a truth value in the set $\{0, 1, 2\}$ to formula φ at time point $k \in [0..\lambda)$ or to the pair (k, i) for path expression ρ, respectively. The valuation of formulas follows the next rules:

$$\begin{aligned}
\boldsymbol{m}(k, \bot) &\overset{def}{=} 0\\
\boldsymbol{m}(k, \top) &\overset{def}{=} 2\\
\boldsymbol{m}(k, a) &\overset{def}{=} \begin{cases} 0 & \text{if } a \notin T_k\\ 1 & \text{if } a \in T_k \setminus H_k\\ 2 & \text{if } a \in H_k \end{cases} \qquad \text{for any atom } a \in \mathcal{A}\\
\boldsymbol{m}(k, [\rho]\,\psi) &\overset{def}{=} \min\{\, \mathrm{imp}(\boldsymbol{m}(k, i, \rho), \boldsymbol{m}(i, \psi)) \mid i \in [0..\lambda) \,\}\\
&\quad \text{where}\\
&\quad \mathrm{imp}(x, y) \overset{def}{=} \begin{cases} 2 & \text{if } x \leq y\\ y & \text{otherwise} \end{cases}\\
\boldsymbol{m}(k, \langle \rho \rangle\, \psi) &\overset{def}{=} \max\{\, \min(\boldsymbol{m}(k, i, \rho), \boldsymbol{m}(i, \psi)) \mid i \in [0..\lambda) \,\}
\end{aligned}$$

whereas the function for path expressions is defined as follows:

$$\begin{aligned}
\boldsymbol{m}(k, j, \tau) &\overset{def}{=} \begin{cases} 2 & \text{if } j = k+1\\ 0 & \text{otherwise.} \end{cases}\\
\boldsymbol{m}(k, j, \varphi?) &\overset{def}{=} \begin{cases} \boldsymbol{m}(k, \varphi) & \text{if } j = k\\ 0 & \text{otherwise.} \end{cases}\\
\boldsymbol{m}(k, j, \rho_1 + \rho_2) &\overset{def}{=} \max(\boldsymbol{m}(k, j, \rho_1), \boldsymbol{m}(k, j, \rho_2))\\
\boldsymbol{m}(k, j, \rho_1\,;\,\rho_2) &\overset{def}{=} \max\{\min(\boldsymbol{m}(k, i, \rho_1),\\
&\qquad\qquad \boldsymbol{m}(i, j, \rho_2)) \mid i \in [0..\lambda)\,\}\\
\boldsymbol{m}(k, j, \rho^*) &\overset{def}{=} \max\{\boldsymbol{m}(k, j, \rho^n) \mid \text{for all } n \geq 0\}\\
\boldsymbol{m}(k, j, \rho^-) &\overset{def}{=} \boldsymbol{m}(j, k, \rho)
\end{aligned}$$

This results in the following three-valued characterisation of HT-traces in DEL_f. [5]

Theorem 1 *Let $\langle \mathbf{H}, \mathbf{T} \rangle$ be a* HT*-trace of length λ, $\boldsymbol{m}$ its associated valuation and $k \in [0..\lambda)$:*

1. *$\langle \mathbf{H}, \mathbf{T} \rangle, k \models \varphi$ iff $\boldsymbol{m}(k, \varphi) = 2$*
2. *$\langle \mathbf{T}, \mathbf{T} \rangle, k \models \varphi$ iff $\boldsymbol{m}(k, \varphi) \neq 0$*
3. *$(k, j) \in \|\rho\|^{\langle \mathbf{H}, \mathbf{T} \rangle}$ iff $\boldsymbol{m}(k, j, \rho) = 2$*
4. *$(k, j) \in \|\rho\|^{\langle \mathbf{T}, \mathbf{T} \rangle}$ iff $\boldsymbol{m}(k, j, \rho) \neq 0$*

3 Reduction to temporal logic programs

In this section, we elaborate upon a reduction of arbitrary dynamic formulas [6] into a syntactic subclass called *temporal logic programs* [9]. A temporal logic program is a conjunction of temporal formulas with a restricted syntax that, when interpreted under temporal stable models, have a close relation to rules from disjunctive logic programming. Temporal logic programs were proved in [9] to constitute a normal form for TEL_f (if we allow for auxiliary atoms) and used later on as a basic syntax for the temporal ASP system *telingo* [8]. We proceed next to describe their syntax.

Given a set of propositional variables $\mathcal{A}$, we define the set of *temporal literals* as $\{a, \neg a, \bullet a, \neg\bullet a, \mid\, a \in \mathcal{A}\}$. A temporal logic program is a set formed by three different types of rules:

1. an *initial rule* is of the form $B \to A$
2. a *dynamic rule* is of the form $\widehat{\circ}\Box\,(B \to A)$

[5] An extended version of the paper with the proofs is available at `http://arxiv.org/abs/2002.06916`

[6] This covers finite dynamic theories, understood as the conjunction of their formulas.

3. a *final rule* is of the form $\Box\,(\mathbf{F} \rightarrow (B \rightarrow A))$

where $B = b_1 \wedge \cdots \wedge b_n$ (with $n \geq 0$) and $A = a_1 \vee \cdots \vee a_m$ (with $m \geq 0$) and b_i and a_j are temporal literals in the case of dynamic rules and regular literals (i.e. $\{a, \neg a \mid a \in \mathcal{A}\}$) in the case of initial and final rules. We also allow for *global rules* of the form $\Box\,(B \rightarrow A)$ that stand for the conjunction of an initial rule $B \rightarrow A$ and a dynamic rule $\widehat{\circ}\Box\,(B \rightarrow A)$.

Theorem 2 (Normal form) *Every dynamic formula γ can be converted into a temporal program being* DHT_f*-equivalent to* γ.

To prove the above theorem, we provide a sound transformation from any dynamic formula γ into a temporal logic program $\pi(\gamma)$. As a first step, we assume that γ is already in converse normal form: this can be achieved by repeatedly applying the equivalences in Proposition 1. The reduction of γ into temporal program $\pi(\gamma)$ uses an extended alphabet $\mathcal{A}^+ \supseteq \mathcal{A}$ that additionally contains new atoms ℓ_φ (aka label) for formulas φ over $\mathcal{A}$ that are either subformulas of γ or elaborations of them. This set of formulas is called the Fisher-Ladner closure [12] of γ and formally defined below.

Definition 3 (Fisher-Ladner closure [12]) *The Fisher-Ladner closure $FL(\gamma)$ of a dynamic formula γ (in converse normal form) is a set of dynamic formulas inductively defined as follows:*

1. $\gamma \in FL(\gamma)$
2. $(\varphi \otimes \psi) \in FL(\gamma)$ *implies* $\varphi \in FL(\gamma)$ *and* $\psi \in FL(\gamma)$, *where* $\otimes \in \{\wedge, \vee, \rightarrow\}$
3. *If* $\langle\rho\rangle\,\varphi \in FL(\gamma)$ *then* $\varphi \in FL(\gamma)$
4. *If* $[\rho]\,\varphi \in FL(\gamma)$ *then* $\varphi \in FL(\gamma)$
5. *If* $\langle\psi?\rangle\,\varphi \in FL(\gamma)$ *then* $\psi \in FL(\gamma)$ *and* $\varphi \in FL(\gamma)$
6. *If* $[\psi?]\,\varphi \in FL(\gamma)$ *then* $\psi \in FL(\gamma)$ *and* $\varphi \in FL(\gamma)$
7. *If* $\langle\rho_1\,;\,\rho_2\rangle\,\varphi \in FL(\gamma)$ *then* $\langle\rho_1\rangle\,\langle\rho_2\rangle\,\varphi \in FL(\gamma)$
8. *If* $[\rho_1\,;\,\rho_2]\,\varphi \in FL(\gamma)$ *then* $[\rho_1]\,[\rho_2]\,\varphi \in FL(\gamma)$
9. *If* $\langle\rho_1 + \rho_2\rangle\,\varphi \in FL(\gamma)$ *then* $\langle\rho_1\rangle\,\varphi \in FL(\gamma)$ *and* $\langle\rho_2\rangle\,\varphi \in FL(\gamma)$
10. *If* $[\rho_1 + \rho_2]\,\varphi \in FL(\gamma)$ *then* $[\rho_1]\,\varphi \in FL(\gamma)$ *and* $[\rho_2]\,\varphi \in FL(\gamma)$
11. *If* $\langle\rho^*\rangle\,\varphi \in FL(\gamma)$ *then* $\langle\rho\rangle\,\langle\rho^*\rangle\,\varphi \in FL(\gamma)$
12. *If* $[\rho^*]\,\varphi \in FL(\gamma)$ *then* $[\rho]\,\langle\rho^*\rangle\,\varphi \in FL(\gamma)$

Any set satisfying these conditions is called closed.

Proposition 3 *For any dynamic formula γ, its closure $FL(\gamma)$ is finite.*

Thus, given the dynamic formula γ on alphabet $\mathcal{A}$ to be translated, we define the extended alphabet $\mathcal{A}^+ \overset{def}{=} \mathcal{A} \cup \{\ell_\mu \mid \mu \in FL(\gamma)\}$. For convenience, we simply use $\ell_\varphi \overset{def}{=} \varphi$ if φ is $\top, \bot$ or an atom $a \in \mathcal{A}$.

As happened with the normal form reduction for TEL_f in [9], the translation is done in two phases: we first obtain a temporal theory containing double implications, and then we unfold them into temporal rules. We start by defining the temporal theory $\sigma(\gamma)$ that introduces new labels ℓ_μ for each formula $\mu \in FL(\gamma)$. This theory contains the formula ℓ_γ and, per each label ℓ_μ, a set of formulas $\eta(\mu)$ fixing the label's truth value. Formally:

$$\sigma(\gamma) = \{\ell_\gamma\} \cup \{\eta(\mu) \mid \mu \in FL(\Gamma)\}$$

Table 1 shows the definitions $\eta(\mu)$ for each μ in the closure $FL(\gamma)$ depending on the outer modality in the formula.

As an example, take the dynamic formula $\gamma = \langle(p?;\top)^*\rangle\,q$ (which corresponds to the temporal formula $p\,\mathbf{U}\,q$). In the first step, we get

$$\Box(\ell_\gamma \leftrightarrow q \vee \ell_\alpha) \tag{2}$$
$$\Box(\mathbf{F} \rightarrow (\ell_\gamma \leftrightarrow q)) \tag{3}$$

$\mu \in FL(\gamma)$	$\eta(\mu)$
$\langle\tau\rangle\,\varphi$	$\widehat{\circ}\Box(\bullet\ell_\mu \leftrightarrow \ell_\varphi)$ $\quad \Box(\mathbf{F} \rightarrow \neg\ell_\mu)$
$[\tau]\,\varphi$	$\widehat{\circ}\Box(\bullet\ell_\mu \leftrightarrow \ell_\varphi)$ $\quad \Box(\mathbf{F} \rightarrow \ell_\mu)$
$\langle\tau^-\rangle\,\varphi$	$\widehat{\circ}\Box(\ell_\mu \leftrightarrow \bullet\ell_\varphi)$ $\quad \neg\ell_\mu$
$[\tau^-]\,\varphi$	$\widehat{\circ}\Box(\ell_\mu \leftrightarrow \bullet\ell_\varphi)$ $\quad \ell_\mu$
$\langle\varphi?\rangle\,\psi$	$\Box(\ell_\mu \leftrightarrow \ell_\varphi \wedge \ell_\psi)$
$[\varphi?]\,\psi$	$\Box(\ell_\mu \leftrightarrow (\ell_\varphi \rightarrow \ell_\psi))$
$\langle\rho + \rho'\rangle\,\varphi$	$\Box(\ell_\mu \leftrightarrow \ell_\alpha \vee \ell_\beta)$ $\quad$ with $\alpha = \langle\rho\rangle\,\varphi, \beta = \langle\rho'\rangle\,\varphi$
$[\rho + \rho']\,\varphi$	$\Box(\ell_\mu \leftrightarrow \ell_\alpha \wedge \ell_\beta)$ $\quad$ with $\alpha = [\rho]\,\varphi, \beta = [\rho']\,\varphi$
$\langle\rho;\rho'\rangle\,\varphi$	$\eta(\,\langle\rho\rangle\,\langle\rho'\rangle\,\varphi\,)$
$[\rho;\rho']\,\varphi$	$\eta(\,[\rho]\,[\rho']\,\varphi\,)$
$\langle\rho^*\rangle\,\varphi$	$\Box(\ell_\mu \leftrightarrow \ell_\varphi \vee \ell_\alpha)$ $\quad$ with $\alpha = \langle\rho\rangle\,\langle\rho^*\rangle\,\varphi$ $\Box(\mathbf{F} \rightarrow (\ell_\mu \leftrightarrow \ell_\varphi))$
$[\rho^*]\,\varphi$	$\Box(\ell_\mu \leftrightarrow \ell_\varphi \wedge \ell_\alpha)$ $\quad$ with $\alpha = [\rho]\,[\rho^*]\,\varphi$ $\Box(\mathbf{F} \rightarrow (\ell_\mu \leftrightarrow \ell_\varphi))$

Table 1. Normal form translation

where we just used q as its label ℓ_q, and α stands for $\langle p?;\top\rangle\,\langle(p?;\top)^*\rangle\,q$ which belongs to $FL(\gamma)$. The truth of ℓ_α is determined by $\eta(\alpha)$ which, in the table, is first unfolded into $\eta(\langle p?\rangle\ \langle\top\rangle\ \langle(p?;\top)^*\rangle\,q)$ leading to:

$$\Box(\ell_\alpha \leftrightarrow p \wedge \ell_\beta) \tag{4}$$

with $\beta = \langle\top\rangle\ \langle(p?;\top)^*\rangle\,q$ also in $FL(\gamma)$. Notice now that β contains γ as a subformula, so it can be written as: $\beta = \langle\top\rangle\ \gamma$. Then $\eta(\beta)$ just corresponds to the pair of formulas:

$$\widehat{\circ}\Box(\bullet\ell_\beta \leftrightarrow \ell_\gamma) \tag{5}$$
$$\Box(\mathbf{F} \rightarrow \neg\ell_\beta) \tag{6}$$

and the whole translation amounts to $\sigma(\gamma) = \{\ell_\gamma\} \cup \{(2) - (6)\}$.

As a second example, consider the formula $\gamma = [(\top;\top)^*]\,p$ meaning that p holds in all even time points (this formula is well-known not to be LTL representable). The formulas obtained for $\eta(\gamma)$ are:

$$\Box(\ell_\gamma \leftrightarrow p \wedge \ell_\alpha) \tag{7}$$
$$\Box(\mathbf{F} \rightarrow (\ell_\gamma \leftrightarrow p)) \tag{8}$$

with $\alpha = [\top;\top]\,[(\top;\top)^*]\,p$ and $\eta(\alpha) = \eta([\top]\,[\top]\,[(\top;\top)^*)]\,p)$ is:

$$\widehat{\circ}\Box(\bullet\ell_\alpha \leftrightarrow \ell_\beta) \tag{9}$$
$$\Box(\mathbf{F} \rightarrow \ell_\alpha) \tag{10}$$

with $\beta = [\top]\,[(\top;\top)^*)]\,p$. Since β is actually $[\top]\,\gamma$, we get $\eta(\beta)$:

$$\widehat{\circ}\Box(\bullet\ell_\beta \leftrightarrow \ell_\gamma) \tag{11}$$
$$\Box(\mathbf{F} \rightarrow \ell_\beta) \tag{12}$$

As we have seen, in the general case, formulas in $\eta(\mu)$ are not temporal rules, since they sometimes contain double implications. However, they all have the forms φ, $\Box\varphi$, $\widehat{\circ}\Box\varphi$ or $\Box(\mathbf{F} \rightarrow \varphi)$, for some inner propositional formula φ formed with temporal literals. For instance, in (2), the inner φ corresponds to the propositional formula $\ell_\gamma \leftrightarrow p \vee \ell_\alpha$. As first shown in [7] propositional formulas in HT can be reduced to conjunctions of disjunctive rules. In this way, we can apply HT transformations (as those in [10]) and THT axioms [1] to eventually obtain a temporal logic program. In the case of (2), the inner double implication is unfolded into three rules that, after applying property $\Box(\alpha \wedge \beta) \leftrightarrow \Box\alpha \wedge \Box\beta$, eventually lead to:

$$\Box(\ell_\gamma \to q \vee \ell_\alpha) \qquad \Box(q \to \ell_\gamma) \qquad \Box(\ell_\alpha \to \ell_\gamma)$$

Given an HT-trace $\langle \mathbf{H}, \mathbf{T} \rangle \overset{def}{=} \langle H_i, T_i \rangle_{i=0}^{\lambda}$, we define its restriction to alphabet $\mathcal{A}$ as $\langle \mathbf{H}, \mathbf{T} \rangle|_\mathcal{A} \overset{def}{=} \langle H_i \cap \mathcal{A}, T_i \cap \mathcal{A} \rangle_{i=0}^{\lambda}$. Similarly, for any set S of HT-traces we write $S|_\mathcal{A}$ to stand for $\{\langle \mathbf{H}, \mathbf{T} \rangle|_\mathcal{A} \mid \langle \mathbf{H}, \mathbf{T} \rangle \in S\}$ as expected.

The following lemma shows that ℓ_μ and μ are equivalent:

Lemma 1 *Let γ be a dynamic formula over $\mathcal{A}$ and let $\langle \mathbf{H}, \mathbf{T} \rangle$ be a* DHT_f *model of $\sigma(\gamma)$ being associated with the three-valuation $\boldsymbol{m}$.*

Then, for any $\mu \in FL(\Gamma)$ and any $k \in [0..\lambda)$, we have $\boldsymbol{m}(k, \ell_\mu) = \boldsymbol{m}(k, \mu)$.

Theorem 3 *For any dynamic formula γ and any length λ, we have*

$$\mathrm{DHT}(\gamma, \lambda) = \mathrm{DHT}(\sigma(\gamma), \lambda)|_\mathcal{A}.$$

Corollary 1 *Let γ be a dynamic formula over $\mathcal{A}$.*

Then, translation $\sigma(\gamma)$ is strongly faithful*, that is:*

$$\mathrm{DEL}(\gamma \wedge \gamma') = \mathrm{DEL}(\sigma(\gamma) \wedge \gamma')|_\mathcal{A}.$$

for any arbitrary dynamic formula γ' over $\mathcal{A}$.

Proposition 4 *Translation $\sigma(\gamma)$ has a polynomial size with respect to the size of γ.*

4 Extending *telingo* with dynamic formulas

We have extended the temporal ASP solver *telingo*[7] with dynamic formulas over finite traces. More precisely, the current version supports negated occurrences of dynamic formula, as in integrity constraints; it is available at `https://github.com/potassco/telingo/releases/tag/v2.0.0`.

To this end, *telingo* provides (theory) atoms of form '`&del{`φ`}`' that encapsulate arbitrary dynamic formulas φ. Their syntax is given in Table 2; it is supplied as a theory grammar to the underlying ASP system *clingo* [13]. [8] The one of the dynamic operators $[\rho]$ and $\langle\rho\rangle$ follows that of $\Box$ and $\Diamond$, respectively, by extending '`>*`' and '`>?`' by prepending path expressions ρ separated by a dot, viz. 'ρ `. >*`' and 'ρ `. >?`' . For instance, the dynamic formula $\langle (p?; \top)^* \rangle\, q$ from Section 3 is expressed as

```
&del{ * (?p ;; &true ) . >? q }
```

The current restriction to negated dynamic formulas allows for an easier algorithmic treatment since formulas in the scope of negation are interpreted as in LDL_f (just as negated formulas in HT can be treated as in classical logic). Although this restriction will be lifted in a next release, it already supports an agreeable modeling methodology for dynamic domain separating action and control theories. The idea is to model the actual action theory with temporal rules, fixing static and dynamic laws, while the control theory, enforcing certain

```
#program always.                                     1

{wait; up; down; serve} = 1 :- not &final.           3
:- up,   at(X), not floor(X+1).                      4
:- down, at(X), not floor(X-1).                      5

at(X+1):- 'up,   'at(X).                             7
at(X-1):- 'down, 'at(X).                             8
at(X)  :- 'at(X), not 'up, not 'down.                9
called(X):- 'called(X), #false:'at(X), 'serve.       10

:- called(X), &final.                                12

ready :- called(X), at(X).                           14

#program initial.                                    16

:- not &del{ *( (*up + *down) ;; ?ready ;; serve)    18
        ;; *wait .>? &final }.                       19
```

Listing 1. *telingo* encoding for the elevator problem

(sub)trajectories, is expressed by integrity constraints using dynamic formulas. This is similar to the pairing of action theories in situation calculus and Golog programs [20].

Let us illustrate this with the example in Listing 1 that aims at modeling a simple elevator. This example is borrowed from [20]. The temporal program in Lines 1-14 constitutes the action theory; it is expressed in TEL_f. All its rules in the scope of the program declaration headed by `always` are thought of as being preceded by $\Box$, that is, they are added as global rules (see beginning of Section 3). Line 3 tells us that exactly one of the four actions wait, up, down, or serve, occurs at any time before the end of the trace. The next two lines check the preconditions of action up and down. Lines 7-9 provide effect and inertia axioms for the fluent at, which reflects the current floor of the elevator. Line 10 expresses that a call at a floor persists unless the floor was served. Line 12 gives the actual goal condition, requiring that no call remains unserved in the final state. Line 14 indicates that the elevator is ready to serve, whenever it is at a floor that it was called to. [9]

The integrity constraint in Line 18-19 represents the following dynamic formula:

$$\bot \leftarrow \neg\langle ((\mathit{up}^* + \mathit{down}^*); \mathit{ready}?; \mathit{serve})^*; \mathit{wait}^* \rangle\, \mathsf{F} \qquad (13)$$

The purpose of this constraint is to eliminate fruitless wandering of the elevator. More precisely, it provides a simple control theory stipulating that the elevator must pick one direction, either up or down, and move in this direction until it reaches a floor to which it was called, and serve this floor; this process is repeated an arbitrary number of times; finally, the elevator may have to wait until the end of the trace. Note that (13) is posed as an initial temporal rule, as indicated by the program directive in Line 16. Hence, its path expression must be matched by a trajectory from the initial to the final state, enforced by F in (13) and `&final` in Line 19 in Listing 1, respectively.

For computation, programs as in Listing 1 are treated according to the translation introduced in Section 3. That is, at the outset, all dynamic formulas are transformed into temporal rules. Each dynamic formula γ is recursively rewritten using translation η until the formula is free of any dynamic constructs. This is accomplished via *clingo*'s functionalities for manipulating abstract syntax trees. Then, *telingo*'s API allows us to turn the obtained equivalences among temporal formulas directly into a regular logic program. This involves the

[7] Details on the functioning of *telingo* are given in [8]; its implementation is available at `https://github.com/potassco/telingo`.

[8] Note that `#` and `&` indicate basic ASP and customizable theory concepts.

[9] The rule in Line 14 only provides an auxiliary atom used in Line 17 below, and may thus be regarded as not belonging the actual action theory.

	`#true`	$\top$	*true*	`#false`	$\bot$	*false*
TEL_f	`&initial`	$\mathbf{I}$	*initial*	`&final`	$\mathbf{F}$	*final*
	`'p`	$\bullet p$	*previous*	`p'`	$\circ p$	*next*
	$<\varphi$	$\bullet\varphi$	*previous*	$>\varphi$	$\circ\varphi$	*next*
	$\phi<?\varphi$	$\phi\ \mathbf{S}\ \varphi$	*since*	$\phi>?\varphi$	$\phi\ \mathbf{U}\ \varphi$	*until*
	$\phi<\star\varphi$	$\phi\ \mathbf{T}\ \varphi$	*trigger*	$\phi>\star\varphi$	$\phi\ \mathbf{R}\ \varphi$	*release*
	$<?\varphi$	$\blacklozenge\varphi$	*eventually before*	$>?\varphi$	$\Diamond\varphi$	*eventually afterward*
	$<\star\varphi$	$\blacksquare\varphi$	*always before*	$>\star\varphi$	$\Box\varphi$	*always afterward*
	$<:\varphi$	$\widehat{\bullet}\varphi$	*weak previous*	$>:\varphi$	$\widehat{\circ}\varphi$	*weak next*
DEL_f				$\rho.>\star\varphi$	$[\rho]\,\varphi$	*always*
				$\rho.>?\varphi$	$\langle\rho\rangle\,\varphi$	*eventually*
	$?\varphi$	$\varphi?$	*test*	$\star\rho$	ρ^*	*star*
	$\rho_1;;\rho_2$	$\rho_1\ ;\ \rho_2$	*sequence*	$\rho_1+\rho_2$	$\rho_1 + \rho_2$	*choice*

Table 2. Past and future temporal operators in *telingo* and their DEL_f and TEL_f counterparts

extension of predicates with time variables as well as the introduction of variables and rules reflecting the successive lengthening of finite traces (cf. [8]; temporal rules are treated analogously). This is needed to be able to solve temporal logic programs incrementally. That is, traces of increasing length are investigated by incrementally extending the underlying logic program. Once a model is found, the search stops and the corresponding traces are provided as output. This amounts to computing a nonempty set $\mathrm{DEL}(P, \lambda)$ of stable traces for the smallest $\lambda \geq 0$ and some temporal program P at hand.

Finally, let us examine the impact of the dynamic formula in Listing 1 on the trajectories induced by the action theory as well as solver performance. Although we believe that (13) allows us to significantly reduce the number of trajectories induced by the action theory in lines 1-14, its impact on search is less clear cut because it comes with an augmentation of the number of constraints in the solver.

To analyze this, we consider the following simple elevator problems: We look at $\mathtt{n} = 5, 7, 9, 11$ floors, respectively, and initially place a call at the ground and top floor while the elevator sits on the middle floor. The goal is to have all floors served at the end of a trace (cf. Line 12). This can be expressed by the following facts.

```
#program always.   floor(1..n).

#program initial.  at((n+1)/2). called(1;n).
```

We run each instance with different horizons, beginning with the minimum lengths of satisfiable traces, viz. $\lfloor 3\mathtt{n} + 1\rfloor/2$, and gradually extending this by 1 to 4 to introduce more room for redundancies. Our experiments were run with *telingo* 2-α (based on *clingo* 5.4.1) and obtained without imposing any time or memory restrictions. Our results are summarized in Table 3. Each entry, a/c, contrasts statistics obtained from the pure action theory in Line 1-14, viz. a, with the combination of action and control theory in Line 1-19, c.

For each setting, we give the number of traces, number of choices during search, and the number of constraints in the solver (after all translations, preprocessing etc.). First of all, we notice that the addition of (13) yields exactly two valid traces, no matter what setting is considered. In the first trace, the elevator goes first *up* all the way and then straight *down*, and vice versa in the second trace. Both traces are actually minimal as witnessed throughout the first column. This is because the elevator is placed at the mid floor, otherwise one would be shorter than the other. Next, we observe how drastically the number of trajectories and the underlying search increases for the action theories with each extension of the horizon. For instance, for 11 floors and an horizon of 19 (11+5+4) the mere action theory admits 200900 valid traces, among which only two are conformant with (13). Looking at the underlying search, it is amazing how radically (13) trims the search, in that only nine choices are needed to find the two traces. This is also astonishing since its addition led to an increase from 7042 to 9026 constraints in the solver. Similar yet less extreme observations can be made in all remaining settings.

To get an idea on runtime, we conducted the same experiment on the larger instance with 71 floors (λ=107), since the ones obtained for 5 to 11 floors are negligible. Finding the first model without the dynamic constraint takes 19.4 sec, including 17.8 sec of solving, while adding the dynamic constraint yields a runtime of only 2.2 sec with 0.01 sec of solving. As with filtering traces, it seems that the dynamic constraint greatly contributes to guiding the solver.

Clearly, our empirical analysis is rather limited and can only indicate the potential impact of dynamic formulas on reducing search efforts for finite traces. Nonetheless, we observe that adding the dynamic formula in (13) not only (sometimes drastically) reduces the number of feasible traces but also significantly cuts down the number of choices despite its non-negligible increase of the resulting problem.

5 Discussion

We have elaborated upon the computational foundations of the Dynamic logic of Here-and-There and its equilibrium traces, viz. DHT_f and DEL_f [6], in order to design and implement an expressive ASP system for modeling and solving dynamic domains. Our approach was motivated by the methodology of separating action and control theories, similar to what is done in Situation Calculus and Golog [20].

To this end, we carved out a normal form for dynamic formulas in DEL_f that consists of its fragment corresponding to temporal logic programs. The translation of dynamic formulas into normal form heavily relies on the introduction of auxiliary variables. This allows us to keep the size of the resulting temporal program polynomial in that of the original formula. And moreover it allows us to overcome the common intranslatability of dynamic into temporal formulas when keeping the same language. Our proof of the normal form result relies on a novel characterization of DEL_f in terms of a three-valued logic.

The reduction of dynamic formulas to temporal logic programs enabled us to implement dynamic expression on top of the temporal

λ (horizon)	$\lfloor 3\mathtt{n}+1 \rfloor/2$	$\cdot + 1$	$\cdot + 2$	$\cdot + 3$	$\cdot + 4$	indicators
$\mathtt{n}$ (floors)						
5	2/2	34/2	340/2	2618/2	17204/2	models
	141/2	295/7	660/7	3183/8	19209/11	choices
	1119/1929	1402/2306	1717/2715	2064/3156	2443/3629	constraints
7	2/2	46/2	598/2	5796/2	46690/2	models
	453/2	842/5	1758/6	7917/7	49982/8	choices
	2016/3092	2391/3561	2798/4062	3237/4595	3708/5160	constraints
9	2/2	58/2	928/2	10846/2	103530/2	models
	1560/2	2206/7	3437/7	15171/7	112964/9	choices
	3181/4523	3648/5084	4147/5677	4678/6302	5241/6959	constraints
11	2/2	70/2	1330/2	18200/2	200900/2	models
	5057/2	7896/6	7043/7	26276/8	219391/9	choices
	4614/6222	5173/6875	5764/7560	6387/8277	7042/9026	constraints

Table 3. Summary of experimental results in the elevator domain

ASP solver *telingo*. Since it constitutes a true extension of the ASP system *clingo*, we obtain a full-fledged modeling language extended by temporal and dynamic constructs. We provided a limited empirical analysis demonstrating the potential impact of using dynamic formulas to select traces among the ones induced by an associated temporal logic program. This is how we see the interaction of control and action theories in our framework.

To the best of our knowledge, *telingo* provides the first ASP system augmented with constructs from dynamic and temporal logics. Encodings for bounded model checking in LTL over infinite traces were given in [18]. This was extended to certain action theories expressed with dynamic operators in [15]. A key feature of these encodings is to capture loops inducing infinite traces. This is avoided in [5], where infinite traces in TEL are captured by Büchi automata via model checking. Encodings of Golog in ASP were proposed in [24, 23]. This amounts to directly implementing a filter on traces, as done in Listing 1, without any logical underpinnings. In [6], we provided a different translation from converse-free dynamic formulas in DEL_f to propositional formulas in HT, which themselves can be translated into an equivalent disjunctive logic program (cf. [10]). More precisely, a dynamic formula γ is translated in [6] into a logic program $(\gamma)_i$. This translation differs from the current one, $\sigma(\gamma)$, in several aspects. First, the target language is different: while $\sigma(\gamma)$ produces a temporal logic program, $(\gamma)_i$ directly obtains a propositional logic program corresponding to some fixed time point i. This is because $\sigma(\gamma)$ is thought for using *telingo* as a backend, along with its incremental solving mode, whereas $(\gamma)_i$ was thought for the direct use of a specific trace length. Thus, the advantage of $(\gamma)_i$ is that it does not pass through temporal expressions from *telingo* as an intermediate step. On the other hand, its disadvantages are that γ cannot contain the converse operator and that $(\gamma)_i$ can be exponential, since it makes use of distributivity for normalization both of path expressions and of formulas into logic programs. Note that σ is applicable to any arbitrary dynamic formula γ and takes polynomial time and space. Its main disadvantage is that, in the general case, the resulting temporal logic program may not be amenable for incremental ASP computation. To do so, an extra condition is required: (non-constraint) temporal rules must additionally be *present-centered*, that is, conditions may refer to the past or present of head expressions, but not to their future. In the general case, $\sigma(\gamma)$ may not satisfy this requirement: for instance, one of the directions of (10) yields the rule $\widehat{\circ}\Box(\bullet\ell_\alpha \leftarrow \ell_\beta)$, so the head $\bullet\ell_\alpha$ depends on a condition ℓ_β in its future. Fortunately, in the case of negated formulas $\sigma(\neg\gamma) = \sigma(\bot \leftarrow \gamma)$, as the ones currently implemented in *telingo*, this limitation does not apply, since we can exclusively use constraints for the translation. Our future work aims at a full integration of dynamic formulas into ASP and thus *telingo*. In particular, we will study the more general case in which dynamic expressions can be used in non-constraint rules. As we did for temporal theories and the so-called *past-future* form (see [6]), we plan to identify a similar syntactic condition for a dynamic formula γ so the resulting temporal program $\sigma(\gamma)$ is guaranteed to be present-centered. It will be interesting to experiment with encodings deriving dynamic formulas rather than merely testing them.

ACKNOWLEDGEMENTS

This work was partially supported by Ministry of Science and Innovation, Spain (grant TIC2017-84453-P), Xunta de Galicia, Spain (grants GPC ED431B 2019/03 and 2016-2019 ED431G/01, CITIC Research Center).

REFERENCES

[1] P. Balbiani and M. Diéguez, 'Temporal here and there', in *Proceedings of the Fifteenth European Conference on Logics in Artificial Intelligence (JELIA'16)*, pp. 81–96. Springer, (2016).

[2] *Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*. Springer, 2019.

[3] A. Bosser, P. Cabalar, M. Diéguez, and T. Schaub, 'Introducing temporal stable models for linear dynamic logic', in *Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'18)*, pp. 12–21. AAAI Press, (2018).

[4] P. Cabalar, 'A normal form for linear temporal equilibrium logic', in *Proceedings of the Twelfth European Conference on Logics in Artificial Intelligence (JELIA'10)*, pp. 64–76. Springer, (2010).

[5] P. Cabalar and M. Diéguez, 'STELP — a tool for temporal answer set programming', in *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, pp. 370–375. Springer, (2011).

[6] P. Cabalar, M. Diéguez, and T. Schaub, 'Towards dynamic answer set programming over finite traces', In [2], pp. 148–162.

[7] P. Cabalar and P. Ferraris, 'Propositional theories are strongly equivalent to logic programs', *Theory and Practice of Logic Programming*, **7**(6), 745–759, (2007).

[8] P. Cabalar, R. Kaminski, P. Morkisch, and T. Schaub, 'telingo = ASP + time', In [2], pp. 256–269.

[9] P. Cabalar, R. Kaminski, T. Schaub, and A. Schuhmann, ‘Temporal answer set programming on finite traces’, *Theory and Practice of Logic Programming*, **18**(3-4), 406–420, (2018).

[10] P. Cabalar, D. Pearce, and A. Valverde, ‘Reducing propositional theories in equilibrium logic to logic programs’, in *Proceedings of the Twelfth Portuguese Conference on Artificial Intelligence (EPIA’05)*, pp. 4–17. Springer, (2005).

[11] G. De Giacomo and M. Vardi, ‘Linear temporal logic and linear dynamic logic on finite traces’, in *Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI’13)*, pp. 854–860. IJCAI/AAAI Press, (2013).

[12] M. Fischer and R. Ladner, ‘Propositional dynamic logic of regular programs’, *Journal of Computer and System Sciences*, **18**(2), 194–211, (1979).

[13] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and P. Wanko, ‘Theory solving made easy with clingo 5’, in *Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP’16)*, OASIcs, pp. 2:1–2:15. (2016).

[14] M. Gebser, P. Obermeier, T. Otto, T. Schaub, O. Sabuncu, V. Nguyen, and T. Son, ‘Experimenting with robotic intra-logistics domains’, *Theory and Practice of Logic Programming*, **18**(3-4), 502–519, (2018).

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

[16] K. Gödel, ‘Zum intuitionistischen Aussagenkalkül’, *Anzeiger der Akademie der Wissenschaften in Wien*, 65–66, (1932).

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

[18] K. Heljanko and I. Niemelä, ‘Bounded LTL model checking with stable models’, *Theory and Practice of Logic Programming*, **3**(4-5), 519–550, (2003).

[19] A. Heyting, ‘Die formalen Regeln der intuitionistischen Logik’, in *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, 42–56, Deutsche Akademie der Wissenschaften zu Berlin, (1930).

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

[21] V. Lifschitz, *Answer Set Programming*, Springer, 2019.

[22] D. Pearce, ‘Equilibrium logic’, *Annals of Mathematics and Artificial Intelligence*, **47**(1-2), 3–41, (2006).

[23] M. Ryan, ‘Efficiently implementing GOLOG with answer set programming’, in *Proceedings of the Twenty-Eighth National Conference on Artificial Intelligence (AAAI’14)*, pp. 2352–2357. AAAI Press, (2014).

[24] T. Son, C. Baral, T. Nam, and S. McIlraith, ‘Domain-dependent knowledge in answer set planning’, *ACM Transactions on Computational Logic*, **7**(4), 613–657, (2006).

[25] G. Tseitin, ‘On the complexity of derivation in the propositional calculus’, *Zapiski nauchnykh seminarov LOMI*, **8**, 234–259, (1968).