A Logic for Specifying Metric Temporal Constraints for Golog Programs

Till Hofmann
hofmann@kbsg.rwth-aachen.de

Gerhard Lakemeyer
gerhard@kbsg.rwth-aachen.de

Knowledge-Based Systems Group
RWTH Aachen University, Germany

Abstract

Executing a Golog program on an actual robot typically requires additional platform constraints to be satisfied. Such constraints are often temporal, refer to metric time, and require modifications to the abstract Golog program. Based on $\mathcal{ES}$ and $\mathcal{ESG}$, modal variants of the Situation Calculus, we propose the logic t-$\mathcal{ESG}$, a logic which allows the specification of metric temporal constraints for Golog programs. We provide a comparison to $\mathcal{ESG}$ and show that Metric Temporal Logic (MTL) can be embedded into t-$\mathcal{ESG}$. We show how to formulate constraints using a model of the robot platform, and we sketch a procedure that solves those constraints using Simple Temporal Networks (STNs).

1 Introduction

While Golog [14], an agent programming language based on the Situation Calculus [15, 20], allows a clear and abstract specification of an agent's behavior, executing a Golog program on a real robot often creates additional issues. Typically, the robot's platform requires additional constraints that are ignored when designing a Golog program. As an example, a robot needs to calibrate its arm before it can use it, and it needs to enable its perception module before it interacts with its environment. Developers typically face two options: First, they can directly model all the platform details in the basic action theory (BAT) and add respective actions to their programs, e.g., they can enable the perception before a *pick* action and disable it afterwards again. However, this makes the behavior specification much more complex and error-prone. If the program involves some form of planning or search, the platform details may cause performance degradation, as the search space's size is increased by the additional actions. Additionally, most of these platform details are not specific to one particular program, but to the platform. Any change to the platform needs to be reflected in the behavior specification. As an alternative, the developers may decide to use Golog for an abstract behavior specification and deal with the platform details on the lower levels. However, this is often not possible, as platform constraints require changes to the high-level program, and thus the program cannot be clearly separated from the platform it is running on.

In this paper, we propose a different approach: As before, the agent's behavior is specified with an abstract program. However, based on platform models as shown in Figure 1, we construct a maintenance BAT that allows to specify a set of additional constraints on the platform. The abstract plan from the original Golog program is then transformed into an executable action sequence that satisfies the platform constraints.

The main ideas of this approach were sketched in [10]. In the following, we focus on the logical foundations that allow the specification of metric temporal constraints. Our starting point is the logic $\mathcal{ESG}$ [6], a temporal variant of $\mathcal{ES}$ [13], which in turn is a modal variant of the Situation Calculus. We extend $\mathcal{ESG}$ by metric time, quantitative temporal operators similar to Metric Temporal Logic (MTL) [12], and additional temporal operators *previous* and *since*, which refer to past states in the same way as *next* and *until* refer to future states. We show the relationship between $\mathcal{ESG}$ and t-$\mathcal{ESG}$ and we provide an embedding of MTL into t-$\mathcal{ESG}$. Next, we show how to describe a BAT and an

In: G. Steinbauer, A. Ferrein (eds.): Proceedings of the 11th International Workshop on Cognitive Robotics, Tempe, AZ, USA, 27-Oct-2018, published at http://ceur-ws.org

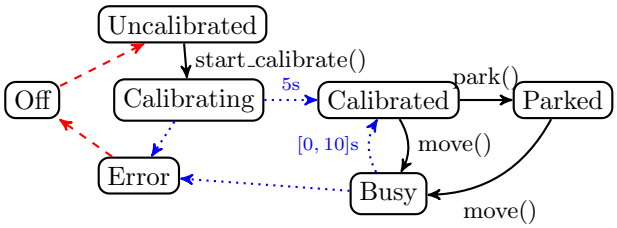

Figure 1: A finite state machine as a platform model for the arm. The edges are annotated with their action and time bounds. Adapted from [10].

abstract GOLOG program in *t-ESG* and how to formulate metric temporal platform constraints with *t-ESG*. Finally, we sketch how to resolve those constraints with a Simple Temporal Network (STN) to transform the abstract program into an executable action sequence.

2 Foundations & Related Work

The Situation Calculus [15, 20] is a first-order logic for representing and reasoning about actions. Action preconditions and effects are axiomatized with basic action theories and situations are histories of actions, e.g., the situation term $do(pick(o1), S_0)$ refers to the situation after doing action $pick(o1)$ in the initial situation. The programming language GOLOG [14] is based on the Situation Calculus and offers imperative programming constructs such as sequences of actions and iteration as well as non-deterministic branching and non-deterministic choice. The semantics of GOLOG and its on-line variant INDIGOLOG can be specified in terms of transitions [7].

The logic *ES* [13] is a modal variant of the Situation Calculus which gets rid of explicit action terms and uses modal operators instead. As an example, $[pick(o1)]Holding(o1)$ is satisfied iff $Holding(o1)$ is true after doing $pick(o1)$. The logic *ESG* [6, 5] is a temporal extension of *ES* for the verification of GOLOG programs. It specifies program transition semantics similar to the transition semantics of INDIGOLOG and extends *ES* with the temporal operators $\mathbf{X}$ (*next*) and $\mathbf{U}$ (*until*).

The Situation Calculus has also been extended by a notion of time. Reiter requires each action A to have an explicit time argument, i.e., each action is of the form $A(\vec{x}, t)$ [19]. Durative actions are modeled with *start* and *stop* actions, and the time of the last occurrence of an action can be referred by $time(A(\vec{x}, t))$. Reiter also extends the Situation Calculus with concurrency, e.g., $do(\{end\_pick(o1), start\_goto(kitchen)\}, s)$ refers to the situation after ending the action $pick(o1)$ and at the same time start moving to the *kitchen*. A similar approach is described by [18], which also allows temporal reasoning about situations. In particular, they differentiate between *possible* and *actual* evolvements of the current situation by adding axioms for a distinct predicate *actual*, where $actual(s)$ is true iff s is an actual evolvement of the world. This differentiation is important, because otherwise, we could not express much about the future; as long as there is a possible action that causes a formula α to be false, we cannot state that α is true in the next situation, even if the agent has no intention (e.g., it is not the next action in the GOLOG program) to execute the action. As we will see, this problem does not occur in *ESG* or *t-ESG*.

MTL [12] is an extension of Linear Time Logic (LTL) with metric time, which allows expressions such as $\mathbf{F}_{\leq c}$, meaning *eventually within time c*. In MTL, formulas are interpreted over *timed words* or *timed state sequences*, where each state specifies which propositions are true, and each state has an associated time value. Depending on the choice of the state and time theory, the satisfiability problem for MTL becomes undecidable [2]. However, for finite words, it has been shown to be decidable [16]. MTL has been restricted to Metric Interval Temporal Logic (MITL) [3], which prohibits singular intervals and is interpreted over time intervals instead of time points, which makes the satisfiability problem for MITL decidable. MTL^{P} [4] is an extension of MTL with temporal operators referring to the past, e.g., $\mathbf{V}$ (*preVious*) and $\mathbf{S}$ (*Since*).

Simple Temporal Networks (STNs) [8] provide an efficient procedure to solve a constraint problem given by a set of time-points $\{t_i\}$ and a set of binary temporal constraints of the form $t_j - t_i \leq \delta$. Disjunctive Linear Relations (DLRs) [11] allow more expressive temporal constraints; the restricted Horn DLRs can be solved in polynomial time and still subsume STNs.

Similar to the proposed approach, Schiffer, Wortmann, and Lakemeyer extend GOLOG for self-maintenance [21] by allowing temporal constraints using Allen's Interval Algebra [1]. Those constraints are resolved on-line by interleaving the original program with maintenance actions. Closely related is also the work by [9], who propose a hybrid approach of temporal constraint reasoning and reasoning about actions based on the Situation Calculus. Similar to [21], they allow constraints based on Allen's Interval Algebra, which are translated into a temporal constraint network.

3 Timed *ESG*

In this section, we present the syntax and semantics of *t-ESG*. As *t-ESG* is based on *ESG*, its syntax and semantics are also based on *ESG* and *ES*. We refer to [13, 6, 5] for more details on the original syntax and semantics.

3.1 Syntax

Definition 1 (Language of t-$\mathcal{ESG}$)**.** The language consists of formulas over symbols from the following vocabulary:

1. object variables $x_1, x_2, x_3, \ldots, y_1, \ldots$
2. action variables $a, a_1, a_2, a_3, \ldots$
3. number variables $t_1, t_2, t_3, \ldots$
4. object standard names $\mathcal{N}_O = \{o_1, o_2, o_3, \ldots\}$
5. action standard names $\mathcal{N}_A = \{p_1, p_2, p_3, \ldots\}$
6. number standard names $\mathcal{N}_N = \{0, 1, 2, \frac{1}{2}, \ldots\}$; here, we assume $\mathcal{N}_N = \mathbb{Q}$
7. fluent predicates of arity k: $\mathcal{F}^k : \{f_1^k, f_2^k, \ldots\}$, e.g., *Holding*; we assume this list contains the distinguished predicates *Poss* and $<$
8. rigid functions of arity k: $\mathcal{G}^k = \{g_1^k, g_2^k, \ldots\}$, e.g., *goto*; including distinguished functions $+, \cdot \in G^2$
9. fluent object functions of arity k: $\mathcal{H}^k = \{h_1^k, h_2^k, \ldots\}$, e.g., *parent*
10. fluent number functions of arity k: $\mathcal{I}^k = \{i_1^k, i_2^k, \ldots\}$, e.g., *battery*; including distinguished functions *time* $\in \mathcal{I}^1$ and *now* $\in \mathcal{I}^0$
11. open, closed, and half-closed intervals, e.g., $[1, 2]$, with constants of sort *number* as interval endpoints
12. connectives and other symbols: $=$, $\wedge$, $\vee$, $\neg$, $\forall$, $\mathbf{X}_I$,$\mathbf{V}_I$, $\mathbf{U}_I$, $\mathbf{S}_I$ (with interval I), $\Box$, $[\cdot]$, $[\![\cdot]\!]$

We also write $\mathcal{F}$ for the set $\bigcup_{k \in \mathbb{N}_0} \mathcal{F}^k$, similarly for $\mathcal{G}, \mathcal{H}, \mathcal{I}$. We also denote the set of standard names as $\mathcal{N} = \mathcal{N}_O \cup \mathcal{N}_A \cup \mathcal{N}_N$. Furthermore, we call a term *primitive* if it is of the form $f(n_1, \ldots, n_k)$, with f, n_i being standard names. We denote the set of primitive terms as $\mathcal{P}_O$ (objects), $\mathcal{P}_A$ (actions), and $\mathcal{P}_N$ (numbers), and $\mathcal{P} = \mathcal{P}_O \cup \mathcal{P}_A \cup \mathcal{P}_N$.

We read $\mathbf{X}_I$ as *next (within interval I)*, $\mathbf{V}_I$ as *previously (within interval I)*, $\mathbf{U}_I$ as *until (within interval I)*, and $\mathbf{S}_I$ as *since (within interval I)*.

Definition 2 (Terms of t-$\mathcal{ESG}$)**.** The set of terms of t-$\mathcal{ESG}$ is the least set such that

- every variable is a term of the corresponding sort,
- every standard name is a term of the corresponding sort,
- if $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \ldots, t_k)$ is a term of the same sort as f.

Definition 3 (Programs)**.**

$$\delta ::= t \mid \alpha? \mid \delta_1; \delta_2 \mid \delta_1 | \delta_2 \mid \pi x.\, \delta \mid \delta_1 \| \delta_2 \mid \delta^*$$

where t is an action term and α is a static situation formula. A program consists of primitive actions t, tests $\alpha?$, sequences $\delta_1; \delta_2$, nondeterministic branching $\delta_1 | \delta_2$, nondeterministic choice of argument $\pi x.\, \delta$, interleaved concurrency $\delta_1 \| \delta_2$, and nondeterministic iteration δ^*.

Definition 4 (Situation Formulas)**.** The *situation formulas* are the least set such that

1. if $t_1, \ldots, t_k$ are terms and P is a k-ary predicate symbol, then $P(t_1, \ldots, t_k)$ is a situation formula,
2. if t_1 and t_2 are terms, then $(t_1 = t_2)$ is a situation formula,
3. if α and β are situation formulas, x is a variable, P is a predicate symbol, δ is a program, and ϕ is a trace formula, then $\alpha \wedge \beta$, $\neg\alpha$, $\forall x.\, \alpha$, $\Box\alpha$, $[\delta]\alpha$, and $[\![\delta]\!]\phi$ are situation formulas.

We call a situation formula of the form $P(n_1, \ldots, n_k)$ with $n_i \in \mathcal{N}$ a *primitive formula* and denote the set of primitive formulas as $\mathcal{P}_F$.

Definition 5 (Trace Formulas)**.** The *trace formulas* are the least set such that

1. if α is a situation formula, then it is also a trace formula,
2. if ϕ and ψ are trace formulas, x is a variable, and I is an interval, then $\phi \wedge \psi$, $\neg\phi$, $\forall x.\, \phi$, $\mathbf{X}_I\, \phi$, $\mathbf{V}_I\, \phi$, $\phi\, \mathbf{S}_I\, \psi$, and $\phi\, \mathbf{U}_I\, \psi$ are also trace formulas.

We also write $< c$, $\leq c$, $= c$, $> c$, and $\geq c$ for the respective intervals $[0, c)$, $[0, c]$, $[c, c]$, (c, ∞), and $[c, \infty)$. We use the short-hand notation $\mathbf{F}_I\, \phi \stackrel{def}{=} (\top\, \mathbf{U}_I\, \phi)$ (*future*) and $\mathbf{G}_I\, \phi \stackrel{def}{=} \neg\mathbf{F}_I\, \neg\phi$ (*globally*), as well as $\mathbf{P}_I\, \phi \stackrel{def}{=} (\top\, \mathbf{S}_I\, \phi)$ (*past*) and $\mathbf{H}_I\, \phi \stackrel{def}{=} \neg\mathbf{P}_I\, \neg\phi$ (*historically*). For intervals, $c + [s, e]$ denotes the interval $[s + c, e + c]$, similarly for $c + (s, e)$, $c + [s, e)$, and $c + (s, e]$. We also omit the interval I if $I = [0, \infty)$, e.g., $\phi\, \mathbf{U}\, \psi$ is short for $\phi\, \mathbf{U}_{[0,\infty)}\, \psi$.

Definition 6 (Static Formulas)**.** A situation formula α is *static* if it contains no $[\cdot]$, $\Box$, or $[\![\cdot]\!]$ operators.

Definition 7 (Bounded Formulas)**.** A situation formula α is bounded if it contains no $\Box$ or $[\![\cdot]\!]$ operators, and $[t]$ operators only in case the argument is an atomic action t.

Definition 8 (Fluent Formulas)**.** A situation formula α is *fluent* if it is static and contains no predicate *Poss*.

3.1.1 Comparison to $\mathcal{ESG}$

The language of t-$\mathcal{ESG}$ extends the language of $\mathcal{ESG}$ by the constrained temporal operators $\mathbf{X}_I$ and $\mathbf{U}_I$. Also, $\mathcal{ESG}$ has no operators referring to the past; i.e., no $\mathbf{V}$ or $\mathbf{S}$ (or their constrained variants $\mathbf{V}_I$ and $\mathbf{S}_I$). Finally, as we understand $\mathbf{X}\,\phi$ ($\phi\,\mathbf{U}\,\psi$) as short-hand notation for $\mathbf{X}_{[0,\infty)}\,\phi$ ($\phi\,\mathbf{U}_{[0,\infty)}\,\psi$), the language of $\mathcal{ESG}$ is a subset of t-$\mathcal{ESG}$.

3.2 Semantics

Definition 9 (Timed Traces)**.** A *timed trace* is a possibly infinite timed sequence of action standard names with monotonically non-decreasing time. Formally, a trace π is a mapping $\pi : \mathbb{N} \to \mathcal{P}_A \times \mathcal{N}_N$, and for any $i, j \in \mathbb{N}$ with $\pi(i) = (\sigma_i, t_i)$, $\pi(j) = (\sigma_j, t_j)$: If $i < j$, then $t_i \leq t_j$. Also, we require the sequence $(t_i)_{i\in\mathbb{N}}$ to be *non-Zeno*, i.e., it is either finite or unbounded.

For a finite timed trace $z = \langle (a_1, t_1) \cdot (a_2, t_2) \cdot \ldots \cdot (a_k, t_k)\rangle$, we define $time(z) \stackrel{def}{=} t_k$, i.e., $time(z)$ is the time value of the last action in z. We denote the set of finite timed traces as $\mathcal{Z}$, the set of infinite timed traces as Π, and the set of all traces as $\mathcal{T} = \mathcal{Z} \cup \Pi$.

Definition 10 (World)**.** Intuitively, a world w determines the truth of fluent predicates as well as the value of fluent functions, not just initially, but after any (timed) sequence of actions. Formally, a world is a mapping w that maps (1) $\mathcal{P}_F \times \mathcal{Z} \to \{0, 1\}$, (2) $\mathcal{P}_O \times \mathcal{Z} \to \mathcal{N}_O$, and (3) $\mathcal{P}_N \times \mathcal{Z} \to \mathcal{N}_N$.

Similar to $\mathcal{ES}$ and $\mathcal{ESG}$, the truth of a fluent after any sequence of actions is determined by a world w. Different to $\mathcal{ES}$ and $\mathcal{ESG}$, we require all traces referred by a world to contain time values for each action. This also means that in the same world, a fluent predicate $F(\vec{n})$ may have a different value after the same sequence of actions if the actions were executed at different times, i.e., $w[F(\vec{n}, \langle (a_1, 1)\rangle)]$ may have a different value than $w[F(\vec{n}, \langle (a_1, 2)\rangle)]$.

Definition 11 (Denotation of terms)**.** Given a ground term t, a world w, and a timed trace $z \in \mathcal{Z}$, we define $|t|^z_w$ by:

1. if $t \in \mathcal{N}$, then $|t|^z_w = t$,
2. if $t = now$, then $|t|^z_w = time(z)$,
3. if $t = time(a(t_1, \ldots, t_k))$, then $|t|^z_w = \max\{t_a \mid (a(n_1, \ldots, n_k), t_a) \in z\} \cup \{0\}$, where $n_i = |t_i|^z_w$,
4. if $t = f(t_1, \ldots, t_k)$ then $|t|^z_w = w[f(n_1, \ldots, n_k), z]$, where $n_i = |t_i|^z_w$

Note the special denotation for the function symbols $now \in \mathcal{I}_0$ and $time \in \mathcal{I}_1$. The term now always refers to the current time, while $time(A(\vec{x}))$ refers to the time of the last occurrence of $A(\vec{x})$, or 0 if the action has never occurred.

Definition 12 (Program Transition Semantics)**.** The transition relation $\stackrel{w}{\to}$ among configurations, given a world w, is the least set satisfying

1. $\langle z, a\rangle \stackrel{w}{\to} \langle z \cdot (p, t), nil\rangle$, if $p = |a|^z_w$, $t \geq time(z)$, and $w, z \cdot (nil, t) \models Poss(p)$
2. $\langle z, \delta_1; \delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \gamma; \delta_2\rangle$, if $\langle z, \delta_1\rangle \stackrel{w}{\to} \langle z \cdot p, \gamma\rangle$,
3. $\langle z, \delta_1; \delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$ if $\langle z, \delta_1\rangle \in \mathcal{F}^w$ and $\langle z, \delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$
4. $\langle z, \delta_1|\delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$ if $\langle z, \delta_1\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$ or $\langle z, \delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$
5. $\langle z, \pi x.\, \delta\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$, if $\langle z, \delta^x_n\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$ for some $n \in \mathcal{N}_x$
6. $\langle z, \delta^*\rangle \stackrel{w}{\to} \langle z \cdot p, \gamma; \delta^*\rangle$ if $\langle z, \delta\rangle \stackrel{w}{\to} \langle z \cdot p, \gamma\rangle$
7. $\langle z, \delta_1\|\delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \delta'\|\delta_2\rangle$ if $z, \delta_1 \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$
8. $\langle z, \delta_1\|\delta_2\rangle \stackrel{w}{\to} \langle z \cdot p, \delta_1\|\delta'\rangle$ if $z, \delta_2 \stackrel{w}{\to} \langle z \cdot p, \delta'\rangle$

The set of final configurations $\mathcal{F}^w$ is the smallest set such that

1. $\langle z, \alpha?\rangle \in \mathcal{F}^w$ if $w, z \models \alpha$,
2. $\langle z, \delta_1; \delta_2\rangle \in \mathcal{F}^w$ if $\langle z, \delta_1\rangle \in \mathcal{F}^w$ and $\langle z, \delta_2\rangle \in \mathcal{F}^w$
3. $\langle z, \delta_1|\delta_2\rangle \in \mathcal{F}^w$ if $\langle z, \delta_1\rangle \in \mathcal{F}^w$, or $\langle z, \delta_2\rangle \in \mathcal{F}^w$
4. $\langle z, \pi x.\, \delta\rangle \in \mathcal{F}^w$ if $\langle z, \delta^x_n\rangle \in \mathcal{F}^w$ for some $n \in \mathcal{N}_x$
5. $\langle z, \delta^*\rangle \in \mathcal{F}^w$
6. $\langle z, \delta_1\|\delta_2\rangle \in \mathcal{F}^w$ if $\langle z, \delta_1\rangle \in \mathcal{F}^w$ and $\langle z, \delta_2\rangle \in \mathcal{F}^w$

The program transition semantics is very similar to the semantics of $\mathcal{ESG}$. The only difference is in Rule 1, which has an additional constraint on the time, and which requires the action to be executable.

Definition 13 (Program Traces)**.** Given a world w and a finite sequence of action standard names z, the set $\|\delta\|^z_w$ of *(timed) traces of a program* δ is

$$\|\delta\|^z_w =$$
$$\{z' \in \mathcal{Z} \mid \langle z, \delta\rangle \stackrel{w}{\to}{}^* \langle z \cdot z', \delta'\rangle \text{ and } \langle z \cdot z', \delta'\rangle \in \mathcal{F}^w\} \cup$$
$$\{\pi \in \Pi \mid \langle z, \delta\rangle \stackrel{w}{\to} \langle z \cdot \pi^{(1)}, \delta_1\rangle \stackrel{w}{\to} \langle z \cdot \pi^{(2)}, \delta_2\rangle \stackrel{w}{\to} \ldots$$
$$\text{where for all } i \leq 0,\ \langle z \cdot \pi^{(i)}, \delta_i\rangle \notin \mathcal{F}^w\}$$

Definition 14 (Truth of Situation and Trace Formulas)**.** Given a world $w \in \mathcal{W}$ and a situation formula α, we define $w \models \alpha$ as $w, \langle\rangle \models \alpha$, where for any $z \in \mathcal{Z}$:

1. $w, z \models F(t_1, \ldots, t_k)$ iff $w[F(n_1, \ldots, n_k), z] = 1$, where $n_i = |t_i|^z_w$,
2. $w, z \models (t_1 = t_2)$ iff n_1 and n_2 are identical, where $n_i = |t_i|^z_w$,
3. $w, z \models \alpha \wedge \beta$ iff $w, z \models \alpha$ and $w, z \models \beta$
4. $w, z \models \neg\alpha$ iff $w, z \not\models \alpha$
5. $w, z \models \forall x.\, \alpha$ iff $w, z \models \alpha^x_n$ for all $n \in \mathcal{N}_x$
6. $w, z \models \Box\alpha$ iff $w, z \cdot z' \models \alpha$ for all $z' \in \mathcal{Z}$
7. $w, z \models [\delta]\alpha$ iff for all finite $z' \in \|\delta\|^z_w$, $w, z \cdot z' \models \alpha$
8. $w, z \models [\![\delta]\!]\phi$ iff for all $\tau \in \|\delta\|^z_w$, $w, z, \tau \models \phi$

Intuitively, $[\delta]\alpha$ means that *after every execution* of δ, the situation formula α is true. $[\![\delta]\!]\phi$ means that *during every execution* of δ, the trace formula ϕ is true.

The truth of trace formulas ϕ is defined as follows for $w \in \mathcal{W}$, $z \in \mathcal{Z}$, and $\tau \in \mathcal{T}$:

1. $w, z, \tau \models \alpha$ iff $w, z \models \alpha$ and α is a situation formula,
2. $w, z, \tau \models \phi \wedge \psi$ iff $w, z, \tau \models \phi$ and $w, z, \tau \models \psi$,
3. $w, z, \tau \models \neg\phi$ iff $w, z, \tau \not\models \phi$,
4. $w, z, \tau \models \forall x.\, \phi$ iff $w, z, \tau \models \phi^x_n$ for all $n \in \mathcal{N}_x$,
5. $w, z, \tau \models \mathbf{X}_I\, \phi$ iff there is $p \in \mathcal{P}_A$ with $\tau = p \cdot \tau'$, $w, z \cdot p, \tau' \models \phi$, and $time(p) \in time(z) + I$,
6. $w, z, \tau \models \mathbf{V}_I\, \phi$ iff $z = z' \cdot p$ for some $p \in \mathcal{P}_A$, $w, z', p \cdot \tau \models \phi$, and $time(z) \in time(z') + I$,
7. $w, z, \tau \models \phi\, \mathbf{U}_I\, \psi$ iff there is a z_1 such that
   (a) $\tau = z_1 \cdot \tau'$,
   (b) $time(z_1) \in time(z) + I$,
   (c) $w, z \cdot z_1, \tau' \models \psi$,
   (d) for all $z_2 \neq z_1$ with $z_1 = z_2 \cdot z_3$: $w, z \cdot z_2, z_3 \cdot \tau' \models \phi$
8. $w, z, \tau \models \phi\, \mathbf{S}_I\, \psi$ iff there is a z_1 such that
   (a) $z = z_1 \cdot z_2$,
   (b) $time(z) \in time(z_1) + I$,
   (c) $w, z_1, z_2 \cdot \tau \models \psi$,
   (d) for all $z_3 \neq \langle\rangle$ with $z_2 = z_3 \cdot z_4$: $w, z_1 \cdot z_3, z_4 \cdot \tau \models \phi$

We shortly explain the intuitive meaning of the temporal operators:

- $\mathbf{X}_I\, \phi$ is true iff ϕ holds at the next state (i.e., after the next action), which must be within interval I.
- $\mathbf{V}_I\, \phi$ is true iff ϕ holds at the previous state (i.e., before the last action), which must be within interval I.
- $\phi\, \mathbf{U}_I\, \psi$ is true iff there is a state in the future within interval I where ψ is true, and in all states before that, ϕ must be true.
- $\phi\, \mathbf{S}_I\, \psi$ is true iff there is a state in the past within interval I where ψ is true, and in all states between that state and the current state, ϕ must be true.

Note that trace formulas are only interpreted over actual traces of the program δ. Thus, a formula $\mathbf{X}\, \alpha$ is true even if there is a possible action a that renders α to be false, as long as a does not occur in δ. Therefore, there is no need for a distinct predicate $actual(s)$ as used by [18].

Definition 15 (Satisfiability)**.** A situation formula α is *satisfiable* iff there is a world w such that

$$w \models \alpha$$

Definition 16 (Validity)**.** A situation formula α is *valid* iff for any world w: $w \models \alpha$. We also write $\models \alpha$ for a valid situation formula α. A trace formula ϕ is *valid* iff for any world w and any trace τ: $w, \langle\rangle, \tau \models \phi$. We also write $\models \phi$ for a valid trace formula ϕ.

4 *t-ESG* and *ESG*

In the following, we compare the first-order fragment of *ESG* to *t-ESG*. To distinguish the semantics of *ESG* and *t-ESG*, we denote the respective semantics with a subscript whenever necessary, e.g., $\mathcal{Z}_{t\text{-}ESG}$ denotes the finite traces of *t-ESG*. We assume a slightly different program transition semantics for *ESG*: For an action term a, we only allow the transition $\langle z, a\rangle \xrightarrow{w} \langle z \cdot p, nil\rangle$ if $w, z \models Poss(p)$; the program can only transition from action a to nil if action a is currently possible. Formally, we replace Rule 1 of the *ESG* program transition semantics by the following rule:

1. $\langle z, a\rangle \xrightarrow{w} \langle z \cdot p, nil\rangle$ if $p = |a|^z_w$ and $w, z \models Poss(p)$

This is similar to replacing each primitive action a in a program δ by $Poss(a)?; a$ as done by [6]. Additionally, as *time* and *now* have a special meaning in *t-ESG*, we assume wlog that α does not mention either of them. Apart from that, syntax and semantics of *ESG* are similar to *t-ESG*, but without constrained temporal operators (e.g., no $\mathbf{X}_I$, only the unconstrained variant $\mathbf{X}$), and without any past temporal operators ($\mathbf{V}$ and $\mathbf{S}$). We refer to [6, 5] for the full *ESG* syntax and semantics.

Theorem 1. *Let α be a first-order sentence of ESG that does not mention time or now. If $\models_{t\text{-}ESG} \alpha$, then also $\models_{ESG} \alpha$.*

Proof. We show that if there is an $\mathcal{ESG}$ world w and an $\mathcal{ESG}$ trace $z \in \mathcal{Z}_{\mathcal{ESG}}$ with $w, z \not\models \alpha$, then there is a t-$\mathcal{ESG}$ world w_t and a t-$\mathcal{ESG}$ trace $z_t \in \mathcal{Z}_{t\text{-}\mathcal{ESG}}$ with $w_t, z_t \not\models \alpha$.

Let w be an $\mathcal{ESG}$ world and $z = \langle a_1, \ldots, a_k \rangle \in \mathcal{Z}_{\mathcal{ESG}}$ such that $w, z \not\models \alpha$. We construct w_t and z_t with $w_t, z_t \models \alpha$ iff $w, z \models \alpha$ as follows: Let w_t be a t-$\mathcal{ESG}$ world such that for every $\rho \in \mathcal{P}$ and every $z'_t = \langle (a'_1, t'_1), (a'_2, t'_2), \ldots, (a'_j, t'_j) \rangle$:

$$w_t[\rho, \langle (a'_1, t'_1), (a'_2, t'_2), \ldots, (a'_j, t'_j) \rangle] = w[\rho, \langle a'_1, a'_2, \ldots, a'_j \rangle]$$

In other words, w_t agrees with w on any primitive fluent after any sequence of actions irrespective of the time. Additionally, we set $z_t = \langle (a_1, 1), (a_2, 2), \ldots (a_k, k) \rangle$. First, we need to show that $|p|^z_w = |p|^{z_t}_{w_t}$ for any term p occurring in α. We do this by induction over p:

- Let $p \in \mathcal{N}$. Then $|p|^z_w = |p|^{z_t}_{w_t} = t$.
- Let $p = f(p_1, \ldots, p_l)$.
  Then, by definition of $|\cdot|$, $|p|^z_w = w[f(n_1, \ldots, n_l)z]$ with $n_i = |p_i|^z_w$. By induction, $|p_i|^{z_t}_{w_t} = |p_i|^z_w$, and thus $|p_i|^{z_t}_{w_t} = n_i$. Also, by definition of w_t, $w_t[f(n_1, \ldots, n_l), z_t] = w[f(n_1, \ldots, n_l), z]$, and thus $|p|^{z_t}_{w_t} = |p|^z_w$.

Note that p cannot mention *now* because α does not mention *now*.

Now, we show by induction over α that $w_t, z_t \models \alpha$ iff $w, z \models \alpha$.

- Let $\alpha = F(p_1, \ldots, p_l)$. As shown: $|p_i|^{z_t}_{w_t} = |p_i|^z_w$. Therefore, by definition of w_t, $w_t[F(p_1, \ldots, p_l), z_t] = w[F(p_1, \ldots, p_l), z]$, and thus $w_t, z_t \models \alpha$ iff $w, z \models \alpha$.
- Let $\alpha = (p_1 = p_2)$. With $|p_i|^{z_t}_{w_t} = |p_i|^z_w$, it follows that $|p_1|^{z_t}_{w_t} = |p_2|^{z_t}_{w_t}$ iff $|p_1|^z_w = |p_2|^z_w$, and thus $w_t, z_t \models \alpha$ iff $w, z \models \alpha$.
- Let $\alpha = \beta \wedge \gamma$. By induction, $w_t, z_t \models \beta$ iff $w, z \models \beta$ and $w_t, z_t \models \gamma$ iff $w, z \models \gamma$. As the semantics of the connective $\wedge$ do not differ between $\mathcal{ESG}$ and t-$\mathcal{ESG}$, it follows that $w_t, z_t \models \alpha$ iff $w, z \models \alpha$.
- Let $\alpha = \neg\beta$. By induction: $w_t, z_t \models \beta$ iff $w, z \models \beta$.
- Let $\alpha = \forall x.\, \alpha$. By induction, for each $n \in \mathcal{N}_x$: $w_t, z_t \models \alpha^x_n$ iff $w, z \models \alpha^x_n$. As $\mathcal{ESG}$ and t-$\mathcal{ESG}$ use the same standard names $\mathcal{N}_x$, it follows that $w_t, z_t \models \alpha$ iff $w, z \models \alpha$.
- Let $\alpha = \Box\beta$. Assume $w, z \not\models \alpha$. Then there is a $z' \in \mathcal{Z}_{\mathcal{ESG}}$ with $w, z \cdot z' \not\models \beta$. Let $z' = \langle a'_1, \ldots, a'_j \rangle$. Then, we set $z'_t = \langle (a'_1, k+1), \ldots, (a'_j, k+j) \rangle \in \mathcal{Z}_{t\text{-}\mathcal{ESG}}$. By induction, $w_t, z_t \cdot z'_t \not\models \beta$.
  For the other direction, assume $w_t, z_t \not\models \alpha$. Then there is $z'_t = \langle (a'_1, k+1), \ldots, (a'_j, k+j) \rangle \in \mathcal{Z}_{t\text{-}\mathcal{ESG}}$ with $w_t, z_t \cdot z'_t \not\models \beta$. By induction, it follows that for $z' = \langle a'_1, \ldots, a'_j \rangle$: $w, z \cdot z' \not\models \beta$.
  Therefore, $w, z \models \Box\beta$ iff $w_t, z_t \models \Box\beta$.
- Let $\alpha = [\delta]\beta$. First, note that

$$\|\delta\|^{z_t}_{w_t} = \{\langle (a'_1, t'_1), (a'_2, t'_2), \ldots, (a'_j, t'_j) \rangle \mid \langle a'_1, a'_2, \ldots, a'_j \rangle \in \|\delta\|^z_w, t_i \in \mathcal{N}_N, t_1 \geq time(z_t), \text{for } i < j : t_i \leq t_j\} \quad (1)$$

  This follows from the fact that the program transition semantics $\|\cdot\|$ differs from $\mathcal{ESG}$ to t-$\mathcal{ESG}$ only in Rule 1, where we add a constraint on time. As α does not mention *now* or *time*, the additional constraint only enforces monotonically non-decreasing time.
  Now, assume there is $z' = \langle a'_1, \ldots, a'_l \rangle \in \|\delta\|^z_w$ with $w, z \cdot z' \not\models \beta$. Then $z'_t = \langle (a'_1, t'), \ldots, (a'_l, t') \rangle \in \|\delta\|^{z_t}_{w_t}$ with $t' = time(z)$. By induction, $w_t, z_t \cdot z'_t \not\models \beta$.
  For the other direction, assume there is $z'_t = \langle (a'_1, t'_1), \ldots, (a'_l, t'_l) \rangle \in \|\delta\|^{z_t}_{w_t}$ with $w_t, z_t \cdot z'_t \not\models \beta$. Then $z' = \langle a_1, \ldots, a_l \rangle \in \|\delta\|^z_w$ and by induction: $w, z \cdot z' \not\models \beta$. Thus, $w, z \models [\delta]\beta$ iff $w_t, z_t \models [\delta]\beta$.
- Let $\alpha = [\![\delta]\!]\beta$. First, note that Equation 1 also holds for infinite traces.
  Let $\tau \in \|\delta\|^z_w$ be an arbitrary trace with $\tau = \langle a'_1, a'_2, \ldots \rangle$, then $\tau_t = \langle (a'_1, t'_1), (a'_2, t'_2), \ldots \rangle \in \|\delta\|^{z_t}_{w_t}$ for arbitrary t_i with the constraints from Equation 1. We show by sub-induction over ϕ: $w, z, \tau \models \phi$ iff $w_t, z_t, \tau_t \models \phi$.
  - Let $\phi = \beta$ be a situation formula. Then by induction: $w, z \models \beta$ iff $w_t, z_t \models \beta$.
  - Let $\phi = \phi_1 \wedge \phi_2$. As in the case for the situation formula, the semantics of the connective $\wedge$ do not differ between $\mathcal{ESG}$ and t-$\mathcal{ESG}$.
  - Let $\phi = \neg\psi$. By sub-induction: $w, z, \tau \models \psi$ iff $w_t, z_t, \tau_t \models \psi$.
  - Let $\phi = \forall x.\, \psi$. Again, as $\mathcal{ESG}$ and t-$\mathcal{ESG}$ use the same standard names, we can follow by sub-induction: $\mathcal{N}_x$, $w, z, \tau \models \psi^x_n$ iff $w_t, z_t, \tau_t \models \psi^x_n$ for every $n \in \mathcal{N}_x$.
  - Let $\phi = \mathbf{X}\,\psi$. Assume $w, z, \tau \models \mathbf{X}\,\psi$. There is $p \in \mathcal{P}_A$ with $\tau = p \cdot \tau'$ such that $w, z \cdot p, \tau' \models \psi$. Then there is a $t_p \geq time(z_t)$ such that $\tau_t = (p, t_p) \cdot \tau'_t$. By induction, $w_t, z_t \cdot (p, t_p), \tau' \models \psi$. Furthermore, in t-$\mathcal{ESG}$, $\mathbf{X}$ is short-hand for $\mathbf{X}_{[0,\infty)}$. As $t_p \in time(z_t) + [0, \infty)$, it follows that $w_t, z_t, \tau_t \models \mathbf{X}\,\psi$.

Now, assume $w_t, z_t, \tau_t \models \mathbf{X}\,\psi$. There is $p \in \mathcal{N}_A$ with $\tau_t = (p, t_p) \cdot \tau'_t$ such that $t_p \geq time(z_t)$ and $w_t, z_t \cdot (p, t_p), \tau'_t \models \psi$. By Equation 1, there is a $\tau' = \langle a'_1, \ldots \rangle$ such that $\tau = p \cdot \tau'$ and $\tau'_t = \langle (a'_1, t'_1), \ldots \rangle$. Then, by induction: $w, z \cdot p, \tau' \models \psi$.

– Let $\phi = \psi_1 \,\mathbf{U}\, \psi_2$. First, note that in t-$\mathcal{ESG}$, $\mathbf{U}$ stands for $\mathbf{U}_{[0,\infty)}$, thus there are no constraints on the time. Similar to the previous case, we can construct pairs $(z_i, z_{t,i})$, such that $w, z \cdot z_i, \tau' \models \psi$ iff $w_t, z_t \cdot z_{t,i}, \tau'_t \models \psi$, same for ϕ. Thus, by induction, $w, z, \tau \models \phi \,\mathbf{U}\, \psi$ iff $w_t, z_t, \tau_t \models \phi \,\mathbf{U}\, \psi$. □

Note that the other direction of the theorem is not true; a valid sentence in $\mathcal{ESG}$ is not necessarily valid in t-$\mathcal{ESG}$. As an example, consider the sentence

$$\alpha = [A]F \vee [A]\neg F$$

In $\mathcal{ESG}$, α is valid, because for each w, either $w \models [A]F$ or $w \models [A]\neg F$, as $w[F, \langle A \rangle]$ is either 0 or 1. However, in t-$\mathcal{ESG}$, α is not valid. Consider a world w_t of t-$\mathcal{ESG}$ with $w[F, \langle (A, 0) \rangle] = 0$ and $w[F, \langle (A, 1) \rangle] = 1$, i.e., if A is performed at time 0, then F is false, but if it is performed at time 1, then F is true. Thus, $w \not\models [A]F$, but also $w \not\models [A]\neg F$, and therefore $w \not\models \alpha$.

5 t-$\mathcal{ESG}$ and MTL

We show that MTL is part of t-$\mathcal{ESG}$. We do this by translating a timed word ρ of MTL into an t-$\mathcal{ESG}$ world w. First, we summarize MTL and its pointwise semantics following the notation by [17].

Definition 17 (Formulas of MTL)**.** Given a set P of atomic propositions, the formulas of MTL are built as follows:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \,\mathbf{U}_I\, \phi$$

Definition 18 (Pointwise semantics of MTL)**.** Given an alphabet of events Σ, a timed word ρ is a finite or infinite sequence $(\sigma_0, \tau_0)(\sigma_1, \tau_1)\ldots$ where $\sigma_i \in \Sigma$ and $\tau_i \in \mathbb{R}_+$ such that the sequence (τ_i) is monotonically non-decreasing[1] and non-Zeno.

Given a timed word $\rho = (\sigma, \tau)$ over alphabet 2^P and an MTL formula ϕ, $\rho, i \models \phi$ is defined as usual for the boolean operators, and with the following rule for $\mathbf{U}_I$: $\rho, i \models \phi_1 \,\mathbf{U}_I\, \phi_2$ iff there exists j such that (1) $i < j < |\rho|$, (2) $\rho, j \models \phi_2$, (3) $\tau_j - \tau_i \in I$, and (4) $\rho, k \models \phi_1$ for all k with $i < k < j$.

Theorem 2. *Let ϕ be a sentence of MTL. Then $\models_{t\text{-}\mathcal{ESG}} \phi$ iff $\models_{MTL} \phi$.*

[1]Some variants of MTL require (τ_i) to be strictly increasing, which can be represented in t-$\mathcal{ESG}$ by requiring strictly increasing traces.

Proof. We assume wlog that $P \subseteq F^0$, i.e., atomic propositions of MTL are 0-ary fluents of t-$\mathcal{ESG}$. Given a t-$\mathcal{ESG}$ world w and a trace π, we construct a timed word ρ such that $w, \langle\rangle, \pi \models_{t\text{-}\mathcal{ESG}} \phi$ iff $\rho \models_{\text{MTL}} \phi$, and vice versa.

⇒: Let w be a t-$\mathcal{ESG}$ world and π a trace. We construct the timed word $\rho = (\sigma, \tau)$ as follows:

1. Set $\tau_0 = 0$ and $\sigma_0[p] = w[p, \langle\rangle]$ for every $p \in P$.
2. For every $i \in \mathcal{N}$: For $\pi(i) = (a_i, t_i)$, set $\tau_i = t_i$.
3. For every finite prefix $z_i = \langle (a_1, t_1), \ldots, (a_i, t_i) \rangle$ of π and every $p \in P$, set $\sigma_i[p] = w[p, z_i]$.

We show that $w, z_i, \pi \models_{t\text{-}\mathcal{ESG}} \phi$ iff $\rho, i \models \phi$ by induction over ϕ.

- Let $\phi = p_i$ be an atomic formula. The claim follows by definition of ρ.
- Let $\phi = \neg\psi$. By induction, $w, z_i, \pi \models \psi$ iff $\rho, i \models \psi$, so the claim follows.
- Let $\phi = \phi_1 \wedge \phi_2$. By induction, $w, z_i, \pi \models \phi_1$ iff $\rho, i \models \phi_1$, same for ϕ_2.
- Let $\phi = \phi_1 \,\mathbf{U}_I\, \phi_2$.

  First, assume $w, z_i, \pi \models \phi_1 \,\mathbf{U}_I\, \phi_2$. Then there is a z_j with $\pi = z_j \cdot \pi'$ such that $time(z_j) \in time(z_i) + I$, $w, z_i \cdot z_j, \pi' \models \phi_2$, and for all $z_k \neq z_j$ with $z_j = z_k \cdot z_l$: $w, z_i \cdot z_k, z_l \cdot \pi' \models \phi_1$. Then, by definition of ρ: $\rho, i + j \models \phi_2$, $\tau_j \in I$, and for all k with $i < i + k < i + j$, $\rho, i + k \models \phi_1$. Thus, by definition of MTL's $\mathbf{U}_I$: $\rho, i \models \phi_1 \,\mathbf{U}_I\, \phi_2$.

  Now, assume $\rho, i \models \phi_1 \,\mathbf{U}_I\, \phi_2$. Similar to the previous case: There is a j with $i < j < |\rho|$, $\rho, j \models \phi_2$, $\tau_j - \tau_i \in I$, and for all k with $i < k < j$: $\rho, k \models \phi_1$. Then, by definition of ρ, there is a z_j with $\pi = z_j \cdot \pi'$ and: $w, z_i \cdot z_j, \pi' \models \phi_1$. Furthermore, $time(z_i \cdot z_j) \in time(z_i) + I$, and for all $z_k \neq z_j$ with $z_j = z_k \cdot z_l$: $w, z_i \cdot z_k, z_l \cdot \pi' \models \phi_2$. Thus, $w, z_i, \pi \models \phi_1 \,\mathbf{U}_I\, \phi_2$.

⇐: Given a timed word $\rho = (\sigma, \tau)$, we construct the world w and trace π as follows:

1. $\pi(i) = (a, \tau_i)$
2. $w[p, \langle\rangle] = \sigma_0[p]$ for every $p \in P$
3. $w[p, \langle (a, \tau_1), (a, \tau_2), \ldots, (a, \tau_i) \rangle] = \sigma_i[p]$ for every $i \in \mathbb{N}$ and every $p \in P$

Analogously to above, we show that $w, z_i, \pi \models_{t\text{-}\mathcal{ESG}} \phi$ iff $\rho, i \models \phi$ by induction over ϕ. □

6 Basic Action Theories

Similarly to $\mathcal{ES}$ and $\mathcal{ESG}$, a t-$\mathcal{ESG}$ BAT is a set of sentences describing the initial situation and the agent's actions with their preconditions and effects.

Definition 19 (Basic Action Theory)**.** Given a set of fluent predicates $\mathcal{F}$, a set $\Sigma \subseteq t\text{-}\mathcal{ESG}$ of sentences is called a *basic action theory* over $\mathcal{F}$ iff $\Sigma = \Sigma_0 \cup \Sigma_{\text{pre}} \cup \Sigma_{\text{post}}$, where Σ mentions only fluents in $\mathcal{F}$ and

1. Σ_0 is any set of fluent sentences,
2. Σ_{pre} is a set of fluent formulas with free variable a,
3. Σ_{post} is a set of sentences, with one sentence of the form $\Box[a]f(\vec{x}) \equiv \gamma_f$ for each fluent predicate $f \in \mathcal{F}$, and one sentence of the form $\Box[a]f(\vec{x}) = v \equiv \gamma_f$ for each $f \in \mathcal{H} \cup \mathcal{I}$, and where γ_f is a fluent formula.

In a BAT, Σ_0 describes the initial situation, Σ_{post} is a set of successor state axioms, and Σ_{pre} is a set of precondition axiom for all actions in the domain that is understood disjunctively, i.e., from Σ_{pre}, we construct a single precondition axiom of the form $\Box Poss(a) \equiv \bigvee \Sigma_{\text{pre}}$. This is slightly different to $\mathcal{ES}$ and $\mathcal{ESG}$ and allows us to combine multiple basic action theories.

6.1 A Simple Carrier Robot

We present a BAT of a simple robot that can move around and pick up and put down objects. All actions are modeled as durative actions with *start* and *stop* actions.

$$
\begin{aligned}
\Sigma_{\text{pre}} = \{ & \\
& \exists s, g.\, a = start\_goto(s,g) && (2)\\
& \quad \wedge \neg\exists a'.\, Performing(a'), \\
& \exists s, g.\, a = end\_goto(s,g) \wedge && (3)\\
& \quad Performing(goto(s,g)) \wedge \\
& \quad now \geq time(goto(s,g)) + d(s,g), \\
& \exists o, l.\, a = start\_pick(o) \wedge && (4)\\
& \quad RobotAt(l) \wedge At(o,l), \\
& \exists o.\, a = end\_pick(o) \wedge && (5)\\
& \quad Performing(pick(o)) \wedge \\
& \quad now \geq time(pick(o)) + 3, \\
& \exists o, l.\, a = start\_put(o,l) \wedge && (6)\\
& \quad Holding(o) \wedge RobotAt(l), \\
& \exists o, l.\, a = end\_put(o,l) \wedge && (7)\\
& \quad Performing(put(o,l)) \wedge \\
& \quad now \geq time(put(o,l)) + 2 \\
\} &
\end{aligned}
$$

The precondition axiom states that the robot can (2) start *goto* if it is currently not performing any action, (3) stop *goto* (and reach its destination) if it was moving at least $d(s,g)$ time units, (4) start picking an object if it as the same location as the object, (5) end picking an object if it started picking at least 3 time units ago, (6) start putting an object to a location if it is holding the object and it is at the location, (7) end putting an object if it started putting at least 2 time units ago. Additionally, ending any action is only possible if the robot is currently *performing* the action.

The robot's actions have effects on the following predicates:

$$
\begin{aligned}
& \Box[a] RobotAt(l) \equiv && (8)\\
& \quad \exists s.\, a = end\_goto(s,l)) \vee \\
& \quad\quad RobotAt(l) \wedge \neg\exists s', g'.\, a = start\_goto(s', g') \\
& \Box[a] At(o,l) \equiv && (9)\\
& \quad a = end\_put(o,l) \vee At(o,l) \wedge a \neq start\_pick(o) \\
& \Box[a] Holding(o) \equiv && (10)\\
& \quad a = end\_pick(o) \vee \\
& \quad\quad Holding(o) \wedge a \neq start\_pick(o) \\
& \Box[a] Performing(a') \equiv && (11)\\
& \quad a = a' \wedge \big(\\
& \quad\quad \exists s, g\, [a = start\_goto(s,g)] \vee \\
& \quad\quad \exists o\, [a = start\_pick(o)] \vee \\
& \quad\quad \exists o\, [a = start\_put(o)] \\
& \quad \big) \vee Performing(a') \wedge \\
& \quad\quad \neg\exists s, g\, [a = end\_goto(s,g)] \wedge \\
& \quad\quad \neg\exists o\, [a = end\_pick(o)] \wedge \\
& \quad\quad \neg\exists o\, [a = end\_put(o)]
\end{aligned}
$$

The successor state axioms state that (8) the robot is at location l if it stops doing a *goto* with l as goal location, or if it was at that location and did not start moving anywhere else, (9) an object o is at location l if the robot ends putting o to l or if the object was there before and is not being picked up, (10) the robot is holding object o if ends picking up o or if it was holding the object before and does not start putting it down, (11) the robot is performing action a' if it starts a' or it has been performing a' and does not end a'.

As usual in Golog, we define **while** and **if then else** as macros:

$$
\begin{aligned}
\textbf{while } \phi \textbf{ do } \delta \textbf{ done} &\stackrel{def}{=} (\phi?;\delta)^*;\neg\phi? \\
\textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2 \textbf{ fi} &\stackrel{def}{=} [\phi?;\delta_1] \mid [\neg\phi?;\delta_2] \\
\textbf{if } \phi \textbf{ then } \delta_1 \textbf{ fi} &\stackrel{def}{=} \textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } nil \textbf{ fi}
\end{aligned}
$$

The BAT already allows us to define simple Golog

if $\neg RobotAt(table)$ **then**
  $\pi s.\ start\_goto(s, table); end\_goto(s, table)$
fi
while $\exists o At(o, table)$ **do**
  $\pi o.\ At(o, table)?;$
    $start\_pick(o); end\_pick(o);$
    $start\_goto(table, shelf); end\_goto(table, shelf);$
    $start\_put(o, shelf); end\_put(o, shelf);$
  $start\_goto(shelf, table); end\_goto(shelf, table);$
done

Listing 1: An abstract program to clear all objects from the table.

programs. The program shown in Listing 1 picks up all objects from the table and puts them onto the shelf.

Given the following initial situation:

$$\begin{aligned}\Sigma_0 = \{&RobotAt(shelf),\\ &At(obj1, table),\\ &At(obj3, shelf),\\ &d(shelf, table) = 20,\ d(table, shelf) = 20,\\ &\forall o.\,(o \neq obj1 \wedge o \neq obj3) \supset \neg\exists l.\, At(o, l)\}\end{aligned}$$

We can infer that:

$$\Sigma \models [\delta]\neg\exists o.\, At(o, table)$$
$$\Sigma \models [\delta] RobotAt(table)$$

Following the program transition semantics, one possible successful trace of δ is:

$$\begin{aligned}z = \langle &(start\_goto(table), 0), (end\_goto(table), 20),\\ &(start\_pick(obj1), 20), (end\_pick(obj1), 23),\\ &(start\_goto(shelf), 23), (end\_goto(shelf), 43),\\ &(start\_put(obj1), 43), (end\_put(obj1), 45)\rangle\end{aligned}$$

7 Timed Temporal Constraints

Listing 1 already shows an abstract program that clears the table. However, to execute the program on a real robot, additional constraints must be satisfied, e.g., the robot's perception module needs to be enabled before an object is picked up or put down. Instead of encoding this in the abstract BAT, we provide an additional maintenance BAT which takes care of these kinds of constraints. Such a maintenance BAT may be generated from platform models as shown in Figure 1, a simplified version may look as follows:

$$\begin{aligned}\Sigma^{\mathrm{M}}_{\mathrm{pre}} = \{&\\ &a = start\_calibration \wedge && (12)\\ &\quad state(arm) \neq calibrating \wedge\\ &\quad state(arm) \neq calibrated,\\ &a = end\_calibration \wedge && (13)\\ &\quad time(start\_calibration) \geq 5 \wedge\\ &\quad state(arm) = calibrating,\\ &a = start\_perception \wedge && (14)\\ &\quad state(perception) \neq running\\ &a = stop\_perception \wedge && (15)\\ &\quad state(perception) = running\}\end{aligned}$$

$$\begin{aligned}&\Box[a] state(arm) = s \equiv && (16)\\ &\quad a = start\_calib \wedge s = calibrating \vee\\ &\quad a = end\_calib \wedge s = calibrated \vee\\ &\quad s = state(arm) \wedge\\ &\qquad a \neq start\_calib \wedge a \neq end\_calib\\ &\Box[a] state(perception) = s \equiv && (17)\\ &\quad a = start\_perception \wedge s = running \vee\\ &\quad a = stop\_perception \wedge s = paused \vee\\ &\quad s = state(perception) \wedge\\ &\qquad a \neq start\_perception \wedge a \neq stop\_perception\end{aligned}$$

$$\begin{aligned}\Sigma^{M}_0 = \{&state(arm) = uncalibrated\\ &state(perception) = paused\}\end{aligned}$$

Using the maintenance BAT, we can formulate a set of additional constraints that must be satisfied during the execution of any program δ:

$$\begin{aligned}&[\![\delta]\!]\mathbf{G}\,\big[\exists o.\, Performing(pick(o)) && (18)\\ &\qquad \supset state(arm) = calibrated\big]\\ &[\![\delta]\!]\mathbf{G}\,\big[\exists o, l.\, Performing(put(o, l)) && (19)\\ &\qquad \supset state(arm) = calibrated\big]\\ &[\![\delta]\!]\mathbf{G}\,\big[\exists o.\, Performing(pick(o)) && (20)\\ &\qquad \supset \mathbf{H}_{\leq 1}\, state(perception) = running\big]\\ &[\![\delta]\!]\mathbf{G}\,\big[\neg\mathbf{F}_{\leq 1}\,\exists o, l.(Performing(pick(o)) && (21)\\ &\qquad\qquad \vee Performing(put(o, l)))\\ &\qquad \supset state(perception) = paused)\big]\end{aligned}$$

The constraints state that (18) if the robot is performing a *pick* action, then the arm must be calibrated, (19) if the robot is performing a *put* action, then the

arm must be calibrated, (20) if the robot is performing a *pick* action, then the perception must have been *running* for at least 1 second, (21) if the robot is not performing a *pick* or *put* action in the next second, then the perception should be *paused*.

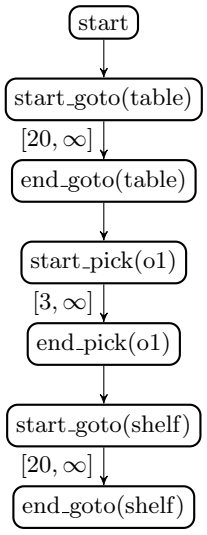

Figure 2: The initial STN.

7.1 The Constraint Language

As constraint language, we allow a subset of *t-ESG*.

Definition 20 (Constraint)**.** Given a BAT Σ and a maintenance BAT Σ^M. Let ϕ be a fluent trace formula only mentioning terms from Σ. Let ψ be a fluent trace formula only mentioning terms from Σ^M. Then the formula $\theta = \phi \supset \psi$ is a constraint formula.

Intuitively, we interpret a constraint $\phi \supset \psi$ as follows: As ϕ is from the abstract BAT and we can only add actions from Σ^M to our action sequence, ϕ is given. Therefore, if ϕ is true, we need to add maintenance actions such that ψ is also satisfied. In other words, we cannot satisfy a constraint by making ϕ false.

Definition 21 (Constraint Satisfaction)**.** Given a program δ and a BAT Σ, we say that a constraint θ is satisfied by δ iff

$$\Sigma \models [\![\delta]\!]\mathbf{G}\,\theta$$

7.2 Solving Timed Temporal Constraints

Using the example above, we now sketch a procedure that modifies a given action sequence of an abstract program to also satisfy the platform constraints. First, we convert the action sequence into a STN, where each action of the sequence is a node in the STN. Such a STN for the program in Listing 1 is shown in Figure 2.

For each node starting at the *start* node, we now check whether each constraint is satisfied. As an example, in the node *start_pick*(*o*1), Constraint 18 is not satisfied. From the platform models, we can generate an action sequence that satisfies the constraint. In our example, by searching the state automaton in Figure 1, we can generate the sequence $\langle$*start_calibrate*, *end_calibrate*$\rangle$, which is then added to the STN. Similarly, the perception needs to be running before doing a *pick*. Thus, the action *start_perception* is added to the STN. By Constraint 20, the perception must have been running for at least 1 second. Thus, the edge from *start_perception* to *start_pick*(*o*1) has the time constraint $[1, \infty]$. The resulting STN after checking all STN nodes and all constraints is shown in Figure 3.

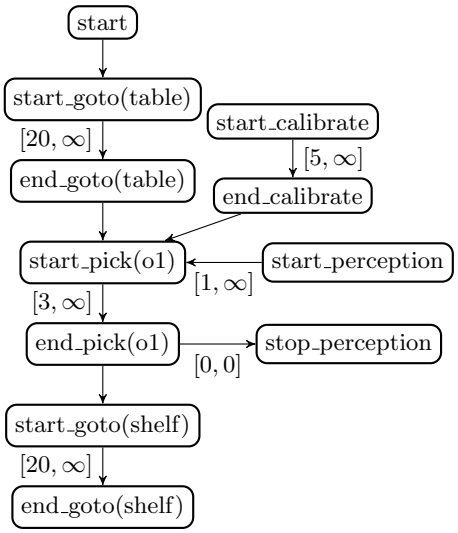

Figure 3: The STN after all constraints have been checked and respective maintenance actions have been added.

8 Conclusion

We presented *t-ESG*, a logic for reasoning about actions that allows the specification of metric temporal constraints. *t-ESG* extends *ESG* with metric time and temporal operators referring to the past. We showed that Metric Temporal Logic (MTL) can be embedded into *t-ESG*, while *ESG* cannot be fully embedded into *t-ESG*.We presented a constraint language for specifying metric temporal constraints on a platform model, and we sketched a translation of those constraints into Simple Temporal Networks (STNs). Using the solution of the STN, the abstract Golog program can be transformed into an action sequence that is executable on the robot platform, thereby allowing the separation of the abstract agent behavior specification and the details of the platform, even if they are inter-dependent.

Acknowledgements

T. Hofmann was supported by the German National Science Foundation (DFG) under grant number GL-747/23-1.

References

[1] James F Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] R. Alur and T.A. Henzinger. Real-Time Logics: Complexity and Expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[3] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The Benefits of Relaxing Punctuality. *Journal of the ACM*, 43(1), 1996.

[4] Rajeev Alur and Thomas A. Henzinger. Back to the Future: Towards a Theory of Timed Regular Languages. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 177–186, 1992.

[5] Jens Claßen. *Planning and Verification in the Agent Language Golog*. PhD thesis, RWTH Aachen University, 2013.

[6] Jens Claßen and Gerhard Lakemeyer. A Logic for Non-Terminating Golog Programs. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 589–599, 2008.

[7] Giuseppe De Giacomo, Yves Lespérance, Hector J Levesque, and Sebastian Sardina. IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents. In *Multi-Agent Programming*. Springer, 2009.

[8] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.

[9] Alberto Finzi and Fiora Pirri. Representing Flexible Temporal Behaviors in the Situation Calculus. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 436–441, 2005.

[10] Till Hofmann, Victor Mataré, Stefan Schiffer, Alexander Ferrein, and Gerhard Lakemeyer. Constraint-Based Online Transformation of Abstract Plans into Executable Robot Actions. In *AAAI Spring Symposium: Integrating Representation, Reasoning, Learning, and Execution for Goal Directed Autonomy*, 2018.

[11] Peter Jonsson and Christer Bäckström. A Unifying Approach to Temporal Constraint Reasoning. *Artificial Intelligence*, 102(1):143–155, 1998.

[12] Ron Koymans. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255–299, 1990.

[13] Gerhard Lakemeyer and Hector J Levesque. Situations, Si! Situation Terms, No! In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 516–526, 2004.

[14] Hector J. Levesque, Raymond Reiter, Yves Lesperance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31(1-3), 1997.

[15] John McCarthy. Situations, Actions, and Causal Laws. Technical report, Stanford University, 1963.

[16] J. Ouaknine and J. Worrell. On the Decidability of Metric Temporal Logic. *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 188–197, 2005.

[17] Joël Ouaknine and James Worrell. Some Recent Results in Metric Temporal Logic. *Lecture Notes in Computer Science*, 5215 LNCS:1–13, 2008.

[18] Javier Pinto and Raymond Reiter. Reasoning About Time in the Situation Calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, 1995.

[19] Raymond Reiter. Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 2–13, 1996.

[20] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[21] Stefan Schiffer, Andreas Wortmann, and Gerhard Lakemeyer. Self-Maintenance for Autonomous Robots controlled by ReadyLog. In *Proceedings of the 7th IARP Workshop on Technical Challenges for Dependable Robots*, 2010.