LTCS–Report

On the Complexity of Verifying Timed Golog Programs over Description Logic Actions (Extended Version)

Patrick Koopmann Benjamin Zarieß

LTCS-Report 18-06

On the Complexity of Verifying Timed Golog Programs over Description Logic Actions (Extended Version)[*]

Patrick Koopmann Benjamin Zarrieß

Abstract

Golog programs allow to model complex behaviour of agents by combining primitive actions defined in a Situation Calculus theory using imperative and non-deterministic programming language constructs. In general, verifying temporal properties of Golog programs is undecidable. One way to establish decidability is to restrict the logic used by the program to a Description Logic (DL), for which recently some complexity upper bounds for verification problem have been established. However, so far it was open whether these results are tight, and lightweight DLs such as $\mathcal{EL}$ have not been studied at all. Furthermore, these results only apply to a setting where actions do not consume time, and the properties to be verified only refer to the timeline in a qualitative way. In a lot of applications, this is an unrealistic assumption. In this work, we study the verification problem for timed Golog programs, in which actions can be assigned differing durations, and temporal properties are specified in a metric branching time logic. This allows to annotate temporal properties with time intervals over which they are evaluated, to specify for example that some property should hold for at least n time units, or should become specified within some specified time window. We establish tight complexity bounds of the verification problem for both expressive and lightweight DLs. Our lower bounds already apply to a very limited fragment of the verification problem, and close open complexity bounds for the non-metrical cases studied before.

Contents

[*]Supported by the DFG within the collaborative research center SFB 912 (HAEC) and the research unit FOR 1513 (HYBRIS).

1 Introduction

Golog [12, 7] is a family of *high-level* programming languages to describe the behaviour of autonomous agents. For example, one can use the following Golog program to describe the behaviour of an author that can non-deterministically choose a venue for paper or continue writing it as long as paper has not been submitted.

$$\begin{array}{l}\textbf{while } \neg\exists x.\,(\mathit{Venue}(x) \land \mathit{submitted\text{-}to}(\mathsf{paper}, x)) \textbf{ do}\\ \quad \mathtt{write}(\mathsf{paper}) \mid \mathtt{submit}(\mathsf{paper}, \mathsf{lpar}) \mid \mathtt{submit}(\mathsf{paper}, \mathsf{ijcar})\\ \textbf{end}\end{array} \tag{1}$$

The primitives of the language are user-defined actions like $\mathtt{write}(\mathsf{paper})$ or $\mathtt{submit}(\mathsf{paper}, \mathsf{lpar})$ whose preconditions and effects are defined in an action theory of the Situation Calculus [14, 16] by abstracting away the details of how they are actually realised. The action theory also provides a first-order logic knowledge base that incompletely represents the initial situation. Timed Golog programs allow to attach durations and effects to each action. A basic reasoning task for Golog programs is *formal verification*, where the aim is to verify whether a given program satisfies a set of given temporal constraints, which in the domain of the example given could regard submission deadlines, and in more advanced scenarios would describe temporal constraints of a complex multi-agent environment.

Due to the non-deterministic nature, the use of imperative constructs and the expressive power of the underlying action formalism, Golog is a Turing-complete language and formal verification is undecidable. Instead of studying sound but incomplete verification methods for Golog as in [6, 13], one recently followed approach is to obtain decidable fragments of the problem by using action formalisms based on Description Logics (DLs) [4, 1, 9]. The work in [3, 5, 19] provides decidability results for verifying qualitative temporal properties of Golog programs over DL actions where the tests (like the loop condition in (1)) and the propositions in the temporal properties are restricted to be DL axioms. For instance, for our example program, one would like to verify whether an execution exists that eventually submits the paper and that respects global constraints on the names *Venue* and *submitted-to*, formalised in a DL ontology. While decision procedures have recently been developed [3, 5, 19], the results do not yield tight complexity bounds for the verification problem. Furthermore, only qualitative temporal properties are considered, while quantitative temporal properties that can address specific time windows, as well as programs in which actions are connected with temporal information such as their durations, have not been investigated in this context yet.

In this paper, we show that the abstraction techniques from [5, 19] are worst-case optimal. The hardness result for the verification problem that we obtain already holds for a surprisingly small fragment of the language, in which only an comparatively inexpressive PTIME-decidable DL is used, and only programs without iteration are considered. Second, as a more positive result, we show that, without increasing the worst-case complexity of the verification problem, the abstraction technique can be extended to a setting where actions have durations (given as natural numbers), and the temporal properties to be verified use metric (numerical) constraints. For example, for specifying correctly timed executions of the example program above, it would be useful to take into account that write(paper) is a time consuming action and that conferences accept submissions only within a certain time frame.

The remainder of the paper is structured as follows. In the next section, we recap basic notions of expressive DLs from the $\mathcal{ALC}$-family and the lightweight DL $\mathcal{ELO}_\bot$. In the subsequent section, we introduce our underlying action formalisms with durative actions and define the syntax and semantics of the Golog-like programming language, which is based on ConGolog. In Section 4, the verification problem is defined for a DL-based extension of the quantitative temporal logic TCTL* [11]. In Section 5, the hardness results for the verification problem are presented and in Section 6 we report on matching upper bounds for the verification of TCTL* specifications.

2 Description Logics

We recall the DLs $\mathcal{ELO}_\bot$, $\mathcal{ALCO}$, $\mathcal{ALCIO}$, $\mathcal{ALCQO}$ and $\mathcal{ALCQIO}$. Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be three pairwise disjoint sets of respectively *concept names*, *role names* and *individual names*. A *role* R is an expression of the form r or r^- (*inverse role*), where $r \in \mathsf{N_R}$. An *$\mathcal{ALCQIO}$-concept* C is an expression built according to the following syntax rule:

$$C ::= \bot \mid \top \mid A \mid \{a\} \mid C \sqcap C \mid \neg C \mid \exists R.C \mid {\geq} nR.C$$

where $A \in \mathsf{N_C}$, $a \in \mathsf{N_I}$, R is a role and $n \in \mathbb{N}$.

If a concept only uses the constructs $\top$, $\bot$, A, $\{a\}$ and $C \sqcap C$ and no inverse roles, it is an $\mathcal{ELO}_\bot$-concept. If it additionally uses the construct $\neg C$, it is an $\mathcal{ALCO}$-concept. From $\mathcal{ALCO}$-concepts, we get to concepts in the more expressive DLs $\mathcal{ALCIO}$, $\mathcal{ALCQO}$ and $\mathcal{ALCQIO}$ by additionally allowing inverse roles ($\mathcal{ALCIO}$, $\mathcal{ALCQIO}$) and concepts of the form ${\geq}nR.C$ ($\mathcal{ALCQO}$, $\mathcal{ALCQIO}$). We define further concept expressions as abbreviations: $C \sqcup D = \neg(\neg C \sqcap \neg D)$, $\forall R.C = \neg(\exists R.\neg C)$ and ${\leq}nR.C = \neg{\geq}(n-1)R.C$.

Let $\mathcal{L} \in \{\mathcal{ELO}_\bot, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQIO}\}$. An $\mathcal{L}$-KB is a set $\mathcal{K}$ of *$\mathcal{L}$-axioms*, which are expressions of the forms $C \sqsubseteq D$, $C(a)$ and $r(a,b)$, with C and D being $\mathcal{L}$-concepts, $a, b \in \mathsf{N_I}$ and $r \in \mathsf{N_R}$. A *Boolean $\mathcal{L}$-KB* is a Boolean combination of axioms. KBs are a special case of Boolean KBs, which we interpret as conjunctions of axioms.

The semantics of DLs is defined in terms of *interpretations*. An interpretation is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a set of *domain elements* and $\cdot^\mathcal{I}$ is the *interpretation function* that maps individual names $a \in \mathsf{N_I}$ to elements $a^\mathcal{I} \in \Delta^\mathcal{I}$, concept names $A \in \mathsf{N_C}$ to sets $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and role names $r \in \mathsf{N_R}$ to relations $r^\mathcal{I} \in \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation function is extended to roles by setting $(r^-)^\mathcal{I} := (r^\mathcal{I})^-$, and to concepts by setting

$$\bot^\mathcal{I} := \emptyset \qquad \top^\mathcal{I} := \Delta^\mathcal{I} \qquad \{a\}^\mathcal{I} := \{a^\mathcal{I}\} \qquad (C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I} \qquad (\neg C)^\mathcal{I} := \Delta^\mathcal{I} \setminus C^\mathcal{I}$$
$$(\exists R.C)^\mathcal{I} := \left\{d \in \Delta^\mathcal{I} \mid \exists e : (d,e) \in R^\mathcal{I} \wedge e \in C^\mathcal{I}\right\}$$
$$({\geq}nR.C)^\mathcal{I} := \left\{d \in \Delta^\mathcal{I} \mid \#\{(d,e) \in R^\mathcal{I} \mid e \in C\} \geq n\right\}.$$

Satisfaction of a Boolean KB φ in an interpretation $\mathcal{I}$, denoted by $\mathcal{I} \models \varphi$, is defined by induction on the structure of φ:

$\mathcal{I} \models C \sqsubseteq D$	iff $C^\mathcal{I} \subseteq D^\mathcal{I}$	$\mathcal{I} \models C(a)$	iff $a^\mathcal{I} \in C^\mathcal{I}$
$\mathcal{I} \models r(a,b)$	iff $\left(a^\mathcal{I}, b^\mathcal{I}\right) \in r^\mathcal{I}$	$\mathcal{I} \models \neg\varphi_1$	iff $\mathcal{I} \not\models \varphi_1$
$\mathcal{I} \models \varphi_1 \wedge \varphi_2$	iff $\mathcal{I} \models \varphi_1$ and $\mathcal{I} \models \varphi_2$	$\mathcal{I} \models \varphi_1 \vee \varphi_2$	iff $\mathcal{I} \models \varphi_1$ or $\mathcal{I} \models \varphi_2$.

If $\mathcal{I} \models \varphi$, we also say that $\mathcal{I}$ is a *model of* φ. A (Boolean) KB is *satisfiable* iff it has a model. A (Boolean) KB ψ is entailed by another (Boolean) KB φ, in symbols $\varphi \models \psi$, iff

$\mathcal{I} \models \psi$ for all models $\mathcal{I}$ of φ. Satisfiability of and entailment from $\mathcal{L}$ KBs is P-complete for $\mathcal{L} = \mathcal{ELO}_\bot$ [2], ExpTime-complete for $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and NExpTime-complete for $\mathcal{ALCQIO}$ [17, 18], even if numbers are encoded in binary.

Example 1. For the example domain considered in the introduction, we would model papers and conferences using the concept names *Paper* and *Conf*, respectively. Conferences currently accepting paper submissions are described by the concept name *Open*. The role name *sub-to* relates papers to conferences and the concept name *Writing* describes papers that are in the writing process. As a general domain constraint, we formulate that papers can be submitted to at most one conference and only papers can be submitted to conferences:

$$\mathit{Paper} \sqsubseteq {\leq}1\, \mathit{sub\text{-}to}.\mathit{Conf} \qquad \exists \mathit{sub\text{-}to}.\mathit{Conf} \sqsubseteq \mathit{Paper}.$$

Using a Boolean KB, we can describe a situation where lpar is a conference that is currently not open for submissions, and where p (individual name) is a paper that has not been submitted yet to lpar:

$$(\mathit{Conf} \sqcap \neg \mathit{Open})(\mathsf{lpar}) \wedge \mathit{Paper}(\mathsf{p}) \wedge \neg\left(\mathit{sub\text{-}to}(\mathsf{p}, \mathsf{lpar})\right). \qquad \blacktriangle$$

Remark. The letter $\mathcal{O}$ in $\mathcal{ELO}_\bot$, $\mathcal{ALCO}$, $\mathcal{ALCIO}$, etc. denotes that a DL supports *nominal expressions* of the form $\{a\}$. As nominals can be straight-forwardly simulated using actions (see also Section 5.2) they do not form a special case in our setting, and complexity results directly transfer.

3 Timed Actions and Programs

Before we introduce programs, we introduce *action theories* as our underlying representation of *primitive actions* and the changes obtained by them. In this section, we first introduce *action theories* as our underlying representation of the changes that are caused by the execution of *primitive actions*. We do not follow an axiomatic approach as it is done in dynamic logic or in the Situation Calculus, but use a semantics based on a meta-theoretic *state transition model*. Here, the *state* of the world is viewed as a (first-order) interpretation, and an *action* is viewed as an operator that modifies interpretation. In an $\mathcal{L}$-action theory, the domain designer provides an incomplete representation of the initial state in terms of a Boolean $\mathcal{L}$-KB and a specification of a list of actions, their effects and their durations.

3.1 Primitive Actions

We start with defining *effect descriptions* to describe atomic changes.

Definition 2. Let $A \in \mathsf{N_C}$ be a concept name, $r \in \mathsf{N_R}$ a role name, $o, o' \in \mathsf{N_I}$ individual names and φ a Boolean $\mathcal{L}$-KB. An $\mathcal{L}$*-effect description* ($\mathcal{L}$*-effect* for short) has one of the following forms

$$\varphi \rhd \langle A(a)\rangle^+, \varphi \rhd \langle r(a,b)\rangle^+, \varphi \rhd \langle A(a)\rangle^-, \varphi \rhd \langle r(a,b)\rangle^-,$$

where φ is called *effect condition*. In case the effect condition φ is a tautology like for example $\top \sqsubseteq \top$, then the effect description is called *unconditional* and is written without the effect condition. $\blacktriangle$

We define the *update of an interpretation* given a set of effects.

Definition 3. Let $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation and L a set of unconditional effects. The *update of* $\mathcal{I}$ *with* L is an interpretation denoted by $\mathcal{I}^{\mathsf{L}}$ and is defined as follows

- $\Delta_{\mathcal{I}^{\mathsf{L}}} = \Delta_{\mathcal{I}}$;
- $A^{\mathcal{I}^{\mathsf{L}}} := A^{\mathcal{I}} \setminus \{a^{\mathcal{I}} \mid \langle A(a)\rangle^{-} \in \mathsf{L}\} \cup \{b^{\mathcal{I}} \mid \langle A(b)\rangle^{+} \in \mathsf{L}\}$ for all $A \in \mathsf{N_C}$;
- $r^{\mathcal{I}^{\mathsf{L}}} := r^{\mathcal{I}} \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \langle r(a,b)\rangle^{-} \in \mathsf{L}\} \cup \{(c^{\mathcal{I}}, d^{\mathcal{I}}) \mid \langle r(c,d)\rangle^{+} \in \mathsf{L}\}$ for all $r \in \mathsf{N_R}$;
- $a^{\mathcal{I}^{\mathsf{L}}} := a^{\mathcal{I}}$ for all $a \in \mathsf{N_I}$.

Let E be a set of (possibly conditional) effects. The *update of* $\mathcal{I}$ *with* E, denoted by $\mathcal{I}^{\mathsf{E}}$, is given by the update $\mathcal{I}^{\mathsf{E}(\mathcal{I})}$ with $\mathsf{E}(\mathcal{I}) := \{\mathsf{l} \mid (\varphi \rhd \mathsf{l}) \in \mathsf{E}, \mathcal{I} \models \varphi\}$. ▲

We define an $\mathcal{L}$-action theory with an initial KB and a finite set of primitive actions where each of them is equipped with a finite set of $\mathcal{L}$-effect descriptions and a duration.

Definition 4. An $\mathcal{L}$-*action theory* is a tuple of the form

$$\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur}),$$

where

- $\mathcal{K}$ is a Boolean $\mathcal{L}$-KB,
- Act is a finite set of *primitive action names* (*actions* for short),
- Eff maps each primitive action name $\alpha \in \mathsf{Act}$ to a set of $\mathcal{L}$-effects $\mathsf{Eff}(\alpha)$,
- Dur maps each primitive action name to a natural number.

▲

In the following, the symbols (possibly indexed or primed) α, β stand for primitive action names, the symbol σ for a (possibly empty) sequence of actions and $\langle\rangle$ for the empty sequence. We often just write "effect" or "action theory" in the following and do not mention the underlying logic $\mathcal{L}$, if it is not relevant.

The *size of an action theory* $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ is given by the sum of

- size of $\mathcal{K}$,
- the sum of the number of symbols needed to write down the effects in $\mathsf{Eff}(\alpha)$ for each primitive action name $\alpha \in \mathsf{A_P}$,
- the sum of the number of bits needed to encode the duration for each $\alpha \in \mathsf{Act}$.

Example 5. We consider a domain with papers and conferences using the concept names *Paper* and *Conf*, respectively. Conferences currently accepting paper submissions are described by the concept name *Open*, the role name *sub-to* relates papers to conferences and the concept name *Writing* describes papers that are in the writing process. As a general domain constraint we formulate that papers can be submitted to at most one conference and only papers can be submitted to conferences:

$$(\mathit{Paper} \sqsubseteq {\leq}1\, \mathit{sub\text{-}to}.\mathit{Conf}) \wedge (\exists \mathit{sub\text{-}to}.\mathit{Conf} \sqsubseteq \mathit{Paper})\,. \tag{2}$$

Facts about the initial situation:

$$((\mathit{Conf} \sqcap \neg \mathit{Open})(\mathsf{lpar})) \wedge (\mathit{Paper}(\mathsf{p})) \wedge (\neg \mathit{sub\text{-}to}(\mathsf{p}, \mathsf{lpar})) \tag{3}$$

express that lpar is a conference that is currently not open for submissions. p (individual name) is a paper that has not been submitted yet to lpar.

We consider a set of primitive actions

$$\mathtt{open}(\mathsf{lpar}), \mathtt{close}(\mathsf{lpar}), \mathtt{start\text{-}w}(\mathsf{p}), \mathtt{end\text{-}w}(\mathsf{p}), \mathtt{submit}(\mathsf{p}, \mathsf{lpar}), \mathtt{tick} \tag{4}$$

with the following effects:

$$\mathsf{Eff}(\mathtt{open}(\mathsf{lpar})) := \{\langle \mathit{Open}(\mathsf{lpar})\rangle^{+}\}, \quad \mathsf{Eff}(\mathtt{close}(\mathsf{lpar})) := \{\langle \mathit{Open}(\mathsf{lpar})\rangle^{-}\},$$
$$\mathsf{Eff}(\mathtt{start\text{-}w}(\mathsf{p})) := \{\langle \mathit{Writing}(\mathsf{p})\rangle^{+}\}, \quad \mathsf{Eff}(\mathtt{end\text{-}w}(\mathsf{p})) := \{\langle \mathit{Writing}(\mathsf{p})\rangle^{-}\}$$

that is the action $\mathtt{open}(\mathsf{lpar})$ stands for the starting point of a submission phase of lpar, $\mathtt{close}(\mathsf{lpar})$ for the corresponding end point and $\mathtt{start\text{-}w}(\mathsf{p})$ and $\mathtt{end\text{-}w}(\mathsf{p})$ mark the start and end points of a writing phase of paper p. For the submission action we have:

$$\mathsf{Eff}(\mathtt{submit}(\mathsf{p}, \mathsf{lpar})(\mathsf{p}, \mathsf{lpar})) := \{((\neg \mathit{Writing} \sqcap \neg \exists \mathit{sub\text{-}to}.\mathit{Conf})(\mathsf{p})) \triangleright \langle \mathit{sub\text{-}to}(\mathsf{p}, \mathsf{lpar})\rangle^{+}\}.$$

that is, if p has not been submitted to some conference before executing $\mathtt{submit}(\mathsf{p}, \mathsf{lpar})$, then $\mathit{sub\text{-}to}(\mathsf{p}, \mathsf{lpar})$ is made true afterwards and nothing else is changing.

In our example domain, we consider all user-definable actions α to be instantaneous with duration zero ($\mathsf{Dur}(\alpha) = 0$). In order to model the passage of one time unit, we use a single action $\mathtt{tick}$ that has a duration of one. We have $\mathsf{Dur}(\mathtt{tick}) = 1$ and $\mathsf{Eff}(\mathtt{tick}) = \emptyset$. ▲

3.2 ConGolog-like Programs

In this section, we define syntax and semantics of a ConGolog-like action programming language [12, 7]. It allows to describe agent behaviour through program expressions that are constructed from primitive actions, programming constructs and tests formulated as Boolean KBs.

Definition 6. Let $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ be an $\mathcal{L}$-action theory. A *program expression* δ over Σ is built according to the following syntax rule

$$\delta ::= \langle\rangle \mid \alpha \mid \psi? \mid \delta;\delta \mid (\delta|\delta) \mid \delta^* \mid \delta \| \delta,$$

where $\langle\rangle$ is the empty program, $\alpha \in \mathsf{Act}$ is a primitive action name and ψ is a Boolean $\mathcal{L}$-KB. ▲

A program can thus be the empty program $\langle\rangle$, a primitive action α, a test $\psi?$, where ψ is a Boolean KB, or constructed from subprograms by means of *sequence* $\delta;\delta$, *non-deterministic choice* $\delta|\delta$, *non-deterministic iteration* δ^* (meaning execute δ zero or more times) and *interleaving* $\delta\|\delta$.

To handle terminating, non-terminating and failing runs of a program uniformly two dummy actions for termination and failure are introduced that are executed indefinitely in a final state and failure state, respectively.

Definition 7. Let $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ be an $\mathcal{L}$-action theory and δ a program expression over Σ. An *$\mathcal{L}$-ConGolog program* (*program* for short) is a tuple $\mathcal{P} = (\Sigma, \delta)$ such that there are two actions in Act namely $\boldsymbol{\epsilon}$ (termination action) and $\mathfrak{f}$ (failure action) that do not occur in δ and we have

$$\mathsf{Eff}(\boldsymbol{\epsilon}) := \{\langle \mathit{Final}(\mathsf{prog})\rangle^{+}\} \quad \mathsf{Eff}(\mathfrak{f}) := \{\langle \mathit{Fail}(\mathsf{prog})\rangle^{+}\}$$

1. $\langle \mathcal{I}, \sigma, \langle\rangle \rangle \in \mathsf{Final}()$;
2. $\langle \mathcal{I}, \sigma, \psi? \rangle \in \mathsf{Final}(\Sigma)$, if $\mathcal{I} \models \psi$;
3. $\langle \mathcal{I}, \sigma, \delta^* \rangle \in \mathsf{Final}(\Sigma)$;
4. $\langle \mathcal{I}, \sigma, \delta_1; \delta_2 \rangle \in \mathsf{Final}(\Sigma)$, if $\langle \mathcal{I}, \sigma, \delta_1 \rangle \in \mathsf{Final}(\Sigma)$ and $\langle \mathcal{I}, \sigma, \delta_2 \rangle \in \mathsf{Final}(\Sigma)$;
5. $\langle \mathcal{I}, \sigma, \delta_1 | \delta_2 \rangle \in \mathsf{Final}(\Sigma)$, if $\langle \mathcal{I}, \sigma, \delta_1 \rangle \in \mathsf{Final}(\Sigma)$ or $\langle \mathcal{I}, \sigma, \delta_2 \rangle \in \mathsf{Final}(\Sigma)$;
6. $\langle \mathcal{I}, \sigma, \delta_1 \| \delta_2 \rangle \in \mathsf{Final}(\Sigma)$, if $\langle \mathcal{I}, \sigma, \delta_1 \rangle \in \mathsf{Final}(\Sigma)$ and $\langle \mathcal{I}, \sigma, \delta_2 \rangle \in \mathsf{Final}(\Sigma)$.

Figure 1: Rules for final program states

and $\mathsf{Dur}(\boldsymbol{\epsilon}) = \mathsf{Dur}(\mathfrak{f}) = 0$, where *Final* and *Fail* are concept names and prog an individual name not occurring in any test in δ and not altered by any effect of any other primitive action in Act. Furthermore, we require $\mathcal{K} \models \neg \mathit{Final}(\mathsf{prog}) \land \neg \mathit{Fail}(\mathsf{prog})$. ▲

The execution of a program in some model of the initial KB $\mathcal{K}$ yields an infinite tree, where states are labeled with interpretations and the edges with natural numbers.

Definition 8. Let $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ be an action theory. A *program state over* Σ is a tuple of the form

$$\langle \mathcal{I}, \sigma, \delta \rangle,$$

where

- $\mathcal{I}$ is an interpretation,
- $\sigma \in \mathsf{Act}^*$ an action sequence and
- δ a program expression over Σ.

The *set of all program states over* Σ is denoted by $\mathsf{States}(\Sigma)$ (for an arbitrary but fixed signature of concept names $\mathsf{N_C}$, role names $\mathsf{N_R}$ and individual names $\mathsf{N_I}$).

The set of all *final program states* over Σ, denoted by $\mathsf{Final}(\Sigma)$, is defined as the smallest set satisfying the conditions in Figure 1.

A *partial transition relation*

$$\longrightarrow_\Sigma \subseteq \mathsf{States}(\Sigma) \times \mathbb{N} \times \mathsf{States}(\Sigma)$$

is defined as the smallest set satisfying the rules in Figure 2.

The set of all *failure program states over* Σ, denoted by $\mathsf{Fail}(\Sigma)$, is defined as follows

$$\mathsf{Fail}(\Sigma) := \{ s \in \mathsf{States}(\Sigma) \mid s \notin \mathsf{Final}(\Sigma) \text{ and}$$
$$\text{there is no } s' \in \mathsf{States}(\Sigma) \text{ and no } d \in \mathbb{N} \text{ with } s \xrightarrow{d}_\Sigma s' \}.$$

▲

The semantics of programs is now defined in terms of a *first-order timed transition system* (*first-order TTS* for short). First, we provide the general definition and then define the first-oder TTS induced by a program.

Definition 9. Let $\mathfrak{L}_\mathfrak{T}$ be a set of state labels. A *transition system* over $\mathfrak{L}_\mathfrak{T}$ is a tuple

$$\mathfrak{T} = (Q_\mathfrak{T}, I_\mathfrak{T}, \hookrightarrow_\mathfrak{T}, \lambda_\mathfrak{T}),$$

where

1. $\langle \mathcal{I}, \sigma, \alpha\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \langle\rangle\rangle$, if $\mathcal{I}' = \mathcal{I}^{\mathsf{Eff}(\mathcal{I},\alpha)}$ and $d \in \mathsf{Dur}(\alpha)$;
2. $\langle \mathcal{I}, \sigma, \delta^*\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta'; \delta^*\rangle$, if $\langle \mathcal{I}, \sigma, \delta\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta'\rangle$;
3. $\langle \mathcal{I}, \sigma, \delta_1; \delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_1'; \delta_2\rangle$, if $\langle \mathcal{I}, \sigma, \delta_1\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_1'\rangle$;
4. $\langle \mathcal{I}, \sigma, \delta_1; \delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_2'\rangle$, if $\langle \mathcal{I}, \sigma, \delta_1\rangle \in \mathsf{Final}(\Sigma)$ and $\langle \mathcal{I}, \sigma, \delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_2'\rangle$;
5. $\langle \mathcal{I}, \sigma, \delta_1|\delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta'\rangle$, if $\langle \mathcal{I}, \sigma, \delta_1\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma, \delta'\rangle$ or $\langle \mathcal{I}, \sigma, \delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma, \delta'\rangle$
6. $\langle \mathcal{I}, \sigma, \delta_1\|\delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_1'\|\delta_2\rangle$, if $\langle \mathcal{I}, \sigma, \delta_1\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_1'\rangle$;
7. $\langle \mathcal{I}, \sigma, \delta_1\|\delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_1\|\delta_2'\rangle$, if $\langle \mathcal{I}, \sigma, \delta_2\rangle \xrightarrow{d}_{\Sigma} \langle \mathcal{I}', \sigma\cdot\alpha, \delta_2'\rangle$.

Figure 2: Transition rules

- $Q_{\mathfrak{T}}$ is a *set of states* and $I_{\mathfrak{T}} \subseteq Q_{\mathfrak{T}}$ a *set of initial states*;
- $\hookrightarrow_{\mathfrak{T}} \subseteq Q_{\mathfrak{T}} \times \mathbb{N} \times Q_{\mathfrak{T}}$ is the *transition relation* such that for each state $q \in Q_{\mathfrak{T}}$ there is at least one state $q' \in Q_{\mathfrak{T}}$ and $d \in \mathbb{N}$ such that $(q, d, q') \in \hookrightarrow_{\mathfrak{T}}$;
- $\lambda_{\mathfrak{T}} : Q_{\mathfrak{T}} \to \mathfrak{L}_{\mathfrak{T}}$ is a total *labeling function* that maps each state to a single element of the *label set* $\mathfrak{L}_{\mathfrak{T}}$.

$\mathfrak{T}$ is called a *first-order TTS* iff $\mathfrak{L}_{\mathfrak{T}}$ is a set of first-order interpretations and the labeling function $\lambda_{\mathfrak{T}} : q \mapsto \mathcal{I}_q$ maps each state $q \in Q_{\mathfrak{T}}$ to a first-order interpretation $\mathcal{I}_q$.

Let AP be a finite set of atomic propositions (propositional letters). $\mathfrak{T}$ is called a *propositional TTS over* AP iff $\mathfrak{L}_{\mathfrak{T}} = 2^{\mathsf{AP}}$, that is the labelling function $\lambda_{\mathfrak{T}}$ maps each state to a set of atomic propositions from AP.

Instead of

$$(q, d, q') \in \hookrightarrow_{\mathfrak{T}} \quad \text{we often write } q \overset{d}{\hookrightarrow}_{\mathfrak{T}} q'.$$

A *path* π in $\mathfrak{T}$ is an infinite sequence of the form

$$\pi = q_0 d_0 q_1 d_1 q_2 d_2 q_3 \cdots, \text{ where } q_i \overset{d_i}{\hookrightarrow}_{\mathfrak{T}} q_{i+1} \text{ holds for all } i \geq 0.$$

Given a path $\pi = q_0 d_0 q_1 d_1 q_2 d_2 q_3 \cdots$ and an index $j \in \{0, 1, 2, \ldots\}$

the infinite suffix path $q_j d_j q_{j+1} d_{j+1} q_{j+2} d_{j+2} q_{j+3} \cdots$ is denoted by $\pi[j..]$ and
the initial path fragment $q_0 d_0 q_1 d_1 q_2 \cdots q_j$ is denoted by $\pi[..j]$.

The jth state in π is denoted by $\pi[j]$. The *duration* of an initial path fragment $\pi[..j]$, denoted by $\mathsf{time}(\pi[..j])$, is the sum of the transition durations:

$$\mathsf{time}(\pi[..j]) := d_0 + \cdots + d_{j-1}.$$

The *set of all paths* in $\mathfrak{T}$ starting in a state $q \in Q_{\mathfrak{T}}$ is denoted by $\mathsf{paths}(\mathfrak{T}, q)$. ▲

Now, we are ready to define the first-order TTS induced by a program $\mathcal{P}$.

Definition 10. Let $\mathcal{P} = (\Sigma, \delta)$ be a program consisting of an action theory $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ and a program expression δ over Σ. The *induced first-order TTS*

$$\mathfrak{T}(\mathcal{P}) = (Q_{\mathcal{P}}, I_{\mathcal{P}}, \hookrightarrow_{\mathcal{P}}, \lambda_{\mathcal{P}})$$

is defined as follows:

- $Q_{\mathcal{P}} := \mathsf{States}(\Sigma)$;
- $I_{\mathcal{P}} := \{\langle \mathcal{I}, \langle\rangle, \delta' \rangle \in \mathsf{States}(\Sigma) \mid \mathcal{I} \models \mathcal{K}, \delta' = \delta\}$;
- $$\hookrightarrow_{\mathcal{P}} := \longrightarrow_{\Sigma} \cup \left\{ (\langle \mathcal{I}, \sigma, \delta' \rangle, 0, \langle \mathcal{J}, \sigma \cdot \boldsymbol{\epsilon}, \langle\rangle \rangle) \,\middle|\, \langle \mathcal{I}, \sigma, \delta' \rangle \in \mathsf{Final}(\Sigma), \mathcal{J} = \mathcal{I}^{\mathsf{Eff}(\boldsymbol{\epsilon})} \right\}$$
  $$\cup \left\{ (\langle \mathcal{I}, \sigma, \delta' \rangle, 0, \langle \mathcal{J}, \sigma \cdot \mathfrak{f}, \delta' \rangle) \,\middle|\, \langle \mathcal{I}, \sigma, \delta' \rangle \in \mathsf{Fail}(\Sigma), \mathcal{J} = \mathcal{I}^{\mathsf{Eff}(\mathfrak{f})} \right\};$$
- $\lambda_{\mathcal{P}} : \langle \mathcal{I}, \sigma, \delta' \rangle \mapsto \mathcal{I}$ for all $\langle \mathcal{I}, \sigma, \delta' \rangle \in \mathsf{States}(\Sigma)$.

▲

Example 11. We continue Example 5. In our domain, we have three agents: the conference organiser opens and closes lpar for submissions an arbitrary number of times, the author of p can do several writing phases as long as p is not submitted and can do several attempts to submit to lpar, and the clock is ticking forever. There is one program expression for each agent:

$$\begin{aligned}
\delta_{\mathsf{conf}} &:= (\texttt{open}(\mathsf{lpar}); \texttt{close}(\mathsf{lpar}))^*, \\
\delta_{\mathsf{author}} &:= ([\neg\exists \mathit{sub\text{-}to}.\top(\mathsf{p})]?; \texttt{start-w}(\mathsf{p}); \texttt{end-w}(\mathsf{p}))^* \parallel \texttt{submit}(\mathsf{p}, \mathsf{lpar})^*, \\
\delta_{\mathsf{clock}} &:= (\texttt{tick})^*; (\neg(\top \sqsubseteq \top))?.
\end{aligned}$$

It might happen that δ_{conf} and δ_{author} terminate but the clock runs forever due to the unsatisfiable test $(\neg(\top \sqsubseteq \top))?$. All three programs run in parallel:

$$\delta_{\mathsf{domain}} := \delta_{\mathsf{conf}} \parallel \delta_{\mathsf{author}} \parallel \delta_{\mathsf{clock}}$$

In δ_{domain} the submission phase and the writing phase might overlap. The length of each phase is determined by the number of `tick` actions that occur in between the corresponding start and end points. Note that the program expression itself does not impose any timing constraints. ▲

4 Temporal Properties and the Verification Problem

For specifying correctly timed executions of the program, we use the metric temporal logics $\mathcal{L}$-TCTL*, $\mathcal{L}$-TCTL, $\mathcal{L}$-CTL* and $\mathcal{L}$-CTL, where $\mathcal{L}$ is some DL. The verification problem for a program and a specification in these logics is formalised as a model checking problem.

Syntactically, $\mathcal{L}$-TCTL* (as well as the other logics) is defined like propositional TCTL* in [11], but it allows Boolean $\mathcal{L}$-KBs instead of propositional letters.

Definition 12. Formulas describing properties of a state and of a path are distinguished. *$\mathcal{L}$-TCTL* state formulas* Φ and *$\mathcal{L}$-TCTL* path formulas* Ψ are built according to the following syntax rules:

$$\begin{aligned}
\Phi &::= \varrho \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{E}\Psi \mid \mathsf{A}\Psi; \\
\Psi &::= \Phi \mid \neg\Psi \mid \Psi \wedge \Psi \mid \mathsf{X}\Psi \mid \Psi\, \mathsf{U}_{\sim c}\, \Psi,
\end{aligned} \tag{5}$$

where ϱ stands for an $\mathcal{L}$-axiom, $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. For path formulas we use the following common abbreviations: $\mathsf{F}_{\sim c}\Psi := \mathsf{true}\ \mathsf{U}_{\sim c}\ \Psi$ (eventually) and $\mathsf{G}_{\sim c}\Psi := \neg\mathsf{F}_{\sim c}\neg\Psi$ (always), where true stands for $\top \sqsubseteq \top$.

Propositional TCTL* state and path formulas over some set of atomic proposition AP are defined using the same syntax rules as above, but requiring that $\varrho \in \mathsf{AP}$.

$\mathcal{L}$-TCTL formulas (without star) are $\mathcal{L}$-TCTL* state formulas that are built according to the syntax rule

$$\Phi ::= \varrho \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{EX}\Phi \mid \mathsf{E}\Phi\ \mathsf{U}_{\sim c}\ \Phi \mid \mathsf{A}\Phi\ \mathsf{U}_{\sim c}\ \Phi.$$

We use a measure for the size of a formula where the numbers attached to the until operator are coded in binary. ▲

If all until operators are indexed with ≥ 0, we may omit the indices and speak of $\mathcal{L}$-CTL*- and $\mathcal{L}$-CTL-formulas respectively.

The semantics is defined in terms of first-order TTS, as defined in the last section.

Definition 13. Let Φ be an $\mathcal{L}$-TCTL* state formula, $\mathfrak{T} = (Q_\mathfrak{T}, I_\mathfrak{T}, \hookrightarrow_\mathfrak{T}, \lambda_\mathfrak{T})$ a first-order TTS and $q \in Q$ a state. *Satisfaction of* Φ *in* $\mathfrak{T}, q$, denoted by $\mathfrak{T}, q \models \Phi$, is defined inductively by:

$$\begin{array}{ll}
\mathfrak{T}, q \models \varrho & \text{iff } \mathcal{I}_q \models \varrho, \text{ where } \varrho \text{ is an } \mathcal{L}\text{-axiom and } \mathcal{I}_q = \lambda(q), \\
\mathfrak{T}, q \models \neg\Phi' & \text{iff } \mathfrak{T}, q \not\models \Phi' \\
\mathfrak{T}, q \models \Phi_1 \wedge \Phi_2 & \text{iff } \mathfrak{T}, q \models \Phi_1 \text{ and } \mathfrak{T}, q \models \Phi_2 \\
\mathfrak{T}, q \models \mathsf{E}\Psi & \text{iff } \mathfrak{T}, \pi \models \Psi \text{ for some } \pi \in \mathsf{paths}(\mathfrak{T}, q), \\
\mathfrak{T}, q \models \mathsf{A}\Psi & \text{iff } \mathfrak{T}, \pi \models \Psi \text{ for all } \pi \in \mathsf{paths}(\mathfrak{T}, q),
\end{array}$$

satisfaction of an $\mathcal{L}$-TCTL* path formula Ψ *in a path* π *in* $\mathfrak{T}$, denoted by $\mathfrak{T}, \pi \models \Psi$, is defined similar for formulas $\neg\Psi'$ and $\Psi_1 \wedge \Psi_2$, and otherwise by

$$\begin{array}{ll}
\mathfrak{T}, \pi \models \Phi & \text{iff } \mathfrak{T}, \pi[0] \models \Phi \\
\mathfrak{T}, \pi \models \mathsf{X}\Psi' & \text{iff } \mathfrak{T}, \pi[1..] \models \Psi' \\
\mathfrak{T}, \pi \models \Psi_1 \mathbin{\mathsf{U}_{\sim c}} \Psi_2 & \text{iff there exists a } j \text{ with } j \geq 0 \text{ such that } \mathfrak{T}, \pi[j..] \models \Psi_2,\ \mathsf{time}(\pi[..j]) \sim c, \\
& \quad \text{and for all } k \text{ with } 0 \leq k < j, \text{ we have } \mathfrak{T}, \pi[k..] \models \Psi_1.
\end{array}$$
▲

Now, we are ready to define the verification problem.

Definition 14 (verification problem)**.** Let $\mathcal{P} = (\Sigma, \delta)$ be a program, Φ a $\mathcal{L}$-TCTL* state formula, and $\mathfrak{T}(\mathcal{P}) = (Q_\mathcal{P}, I_\mathcal{P}, \hookrightarrow_\mathcal{P}, \lambda_\mathcal{P})$ the induced first-order TTS. We write

$$\mathcal{P} \models \Phi$$

iff $\mathfrak{T}(\mathcal{P}), s \models \Phi$ for all $s \in I_\mathcal{P}$. The *verification problem* is the problem of deciding whether $\mathcal{P} \models \Phi$ is true. ▲

For a purely *qualitative (untimed) instance of the verification problem*, we assume that the temporal property is an $\mathcal{L}$-CTL* or $\mathcal{L}$-CTL formula (that is, only ≥ 0 is used as a time constraint) and in the action theory, each primitive action has duration 1.

Example 15. We continue our running example. Assume the action `tick` counts days. One can then specify that the submission phase of lpar lasts exactly 60 days (Ψ_1), and that lpar opens at least every 365 days (Ψ_2):

$$\begin{aligned}
\Psi_1 &:= \mathsf{G}_{>0} \left[(\neg \mathit{Open}(\mathsf{lpar}) \wedge \mathsf{X}\, \mathit{Open}(\mathsf{lpar})) \rightarrow \mathsf{X} \left(\mathit{Open}(\mathsf{lpar}) \mathbin{\mathsf{U}_{=60}} (\neg \mathit{Open}(\mathsf{lpar})) \right) \right], \\
\Psi_2 &:= \mathsf{G}_{>0} \left[\neg \mathit{Open}(\mathsf{lpar}) \rightarrow \mathsf{F}_{<365}\, \mathit{Open}(\mathsf{lpar}) \right].
\end{aligned}$$

We can use this additional constraint to define the property that eventually, p is submitted to lpar.

$$\Phi := \mathsf{E} \left(\Psi_1 \wedge \Psi_2 \wedge \mathsf{F}_{\geq 0} \left(\mathit{sub\text{-}to}(\mathsf{p}, \mathsf{lpar}) \right) \right)$$

This state formula is not entailed by the program. In contrast, the formula $(\forall \mathit{sub\text{-}to}.\bot(\mathsf{p})) \rightarrow \Phi$, which additionally requires p not to be submitted anywhere else initially, is. ▲

5 Hardness

We first analyse complexity lower bounds of the verification problem, before we provide matching upper bounds in the next section. Our lower bounds already apply to the qualitative (untimed) instance of the verification problem, which is why we only consider this setting here.

In this and the following section, it is convenient to focus on the complementary problem of formula *satisfiability* rather than on verification. A TCTL* formula Φ is *satisfiable* in a program $\mathcal{P}$ iff there exists an initial state $s \in I_{\mathcal{P}}$ s.t. $\mathfrak{T}(\mathcal{P}), s \models \Phi$. One easily establishes that Φ is satisfiable in $\mathcal{P}$ iff $\mathcal{P} \not\models \neg\Phi$.

5.1 Expressive Description Logics

We start with programs and verification formulae over expressive DLs. Specifically, we show hardness for the expressive DLs by a reduction from the satisfiability problem for DLs with *nominal schemas* [10]. Nominal schemas allow variables to be shared between different concepts in an axiom. To preserve decidability however, they only quantify over a specified finite set of individual names. Let $\mathsf{N_V}$ be a set of *variable names* disjoint from $\mathsf{N_I}$,$\mathsf{N_C}$ and $\mathsf{N_R}$. For a DL $\mathcal{L}$ introduced in Section 2, we denote by $\mathcal{LV}$ the DL obtained from $\mathcal{L}$ by additionally allowing concepts of the form $\{x\}$, where $x \in \mathsf{N_V}$, which are called nominal schemas.

For simplicity, we assume that every axiom in an $\mathcal{LV}$-KB is of the form $C \sqsubseteq D$. All other types of axioms can be brought into this form, since $C(a)$ is equivalent to $\{a\} \sqsubseteq C$ and $r(a, b)$ is equivalent to $\{a\} \sqsubseteq \exists r.\{b\}$. Given a concept C with nominal schemas, and a function $\nu : \mathsf{N_V} \to \mathsf{N_I}$, we denote by C^ν the result of replacing every variable $v \in \mathsf{N_V}$ by $\nu(v)$. The semantics of nominal schemas is then provided by the following definition.

Definition 16. Let $\mathcal{K}$ be an $\mathcal{LV}$-KB with nominal schemas, Var the finite set of all variable names occurring in $\mathcal{K}$, and let Obj be a finite set of object names. We define the *set of all possible groundings of $\mathcal{K}$ w.r.t.* Obj as follows:

$$\mathsf{ground}(\mathcal{K}, \mathsf{Obj}) := \bigcup_{C \sqsubseteq D \in \mathcal{K}} \{C^\nu \sqsubseteq D^\nu \mid \nu \text{ with } \nu(x) \in \mathsf{Obj} \text{ for all } x \in \mathsf{Var}\}.$$

We say that $\mathcal{K}$ *is satisfiable w.r.t.* Obj iff the $\mathcal{L}$-TBox $\mathsf{ground}(\mathcal{K}, \mathsf{Obj})$ is satisfiable. ▲

If the given set of object names has cardinality m, then an axiom with n different variable names has m^n possible groundings. Therefore, the set of all possible groundings of a KB with nominal schemas w.r.t. a finite set of object names can be exponentially large. In fact, for the logics considered in this paper, the complexity of satisfiability checking increases by an exponential if we add nominal schemas.

Theorem 17 ([10])**.** *Satisfiability of $\mathcal{LV}$-KBs is*

1. ExpTime-*complete for $\mathcal{L} = \mathcal{ELO}_\bot$,*
2. 2ExpTime-*complete for $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and*
3. N2ExpTime-*complete for $\mathcal{L} = \mathcal{ALCQIO}$.*

We transfer these bounds to the verification problem, by reducing satisfiability of $\mathcal{LV}$-KBs to satisfiability of $\mathcal{L}$-CTL-formulas in untimed $\mathcal{L}$-ConGolog-programs, that is, all actions have a duration of 1. For a given $\mathcal{L}$-KB $\mathcal{K}$ with nominal schemas and a finite set Obj of object names, we construct an $\mathcal{L}$-action theory $\Sigma_{\mathcal{S},\mathsf{Obj}} = (\mathcal{K}_{\mathcal{S},\mathsf{Obj}}, \mathsf{Act}_{\mathcal{S},\mathsf{Obj}}, \mathsf{Eff}, \mathsf{Dur})$, a program expression $\delta_{\mathcal{S},\mathsf{Obj}}$

and a temporal property $\Phi_{\mathcal{S},\mathsf{Obj}}$ of polynomial size such that $\mathcal{K}$ is consistent w.r.t. Obj iff $\Phi_{\mathcal{S},\mathsf{Obj}}$ is satisfiable in $\mathcal{P} = (\Sigma_{\mathcal{S},\mathsf{Obj}}, \delta_{\mathcal{S},\mathsf{Obj}})$. Intuitively, the program generates all groundings of the variables and checks the satisfaction of the axioms in $\mathcal{K}$ against each of them.

Let $\mathsf{Var} = \{x_1, \dots, x_n\}$ be the set of all variable names occurring in $\mathcal{K}$ and let $\mathsf{Obj} = \{o_1, \dots, o_m\}$. The initial KB $\mathcal{K}_{\mathcal{S},\mathsf{Obj}}$ is given by

$$\mathcal{K}_{\mathcal{S},\mathsf{Obj}} = \{A_x \sqsubseteq \bot \mid x \in \mathsf{Var}\} \cup \{(\neg \mathit{AllGrounded})(s)\}.$$

The concepts A_x will store the current grounding of the variable x, while we will use the assertion $\mathit{AllGrounded}(s)$ to state whether the grounding of variables is complete. For every pair $x \in \mathsf{Var}, o \in \mathsf{Obj}$, we use the action $\mathtt{ground}_{x,o}$ with $\mathsf{Eff}(\mathtt{ground}_{x,o}) = \{\langle A_x(o)\rangle^+\}$ to instantiate x with o. The action $\mathtt{finished}$ is used to indicate that all variables are instantiated, for which we set $\mathsf{Eff}(\mathtt{finished}) = \{\langle \mathit{AllGrounded}(s)\rangle^+\}$.

We define the program sequence $\delta_{\mathcal{S},\mathsf{Obj}}$. To non-deterministically assign an instantiation to a variable $x \in \mathsf{Var}$, we use the program expression δ^x defined as

$$\delta^x := (\mathtt{ground}_{x,o_1} \mid \mathtt{ground}_{x_i,o_2} \mid \cdots \mid \mathtt{ground}_{x_i,o_m}).$$

The final program sequence $\delta_{\mathcal{S},\mathsf{Obj}}$ that generates all possible groundings is then defined as

$$\delta_{\mathcal{S},\mathsf{Obj}} := \delta^{x_1}; \delta^{x_2}; \cdots; \delta^{x_n}; \mathtt{finished}.$$

It remains the define the temporal property to be checked. For an $C \sqsubseteq D \in \mathcal{K}$, the axiom $\widehat{C} \sqsubseteq \widehat{D}$ is obtained from $C \sqsubseteq D$ by simultaneously replacing all occurrences of the concepts $\{x\}$, $x \in \mathsf{Var}$, in C and D by the corresponding concept name A_x. The temporal property is then defined as

$$\Phi_{\mathcal{S},\mathsf{Obj}} := \mathsf{AG}\left((\mathit{AllGrounded}(s)) \rightarrow \left(\bigwedge_{C \sqsubseteq D \in \mathcal{K}} \widehat{C} \sqsubseteq \widehat{D} \right) \right).$$

The size of $\Sigma_{\mathcal{S},\mathsf{Obj}}$, $\delta_{\mathcal{S},\mathsf{Obj}}$ and $\Phi_{\mathcal{S},\mathsf{Obj}}$ is quadratic in the size of $\mathcal{K}$ and Obj, and indeed, we can show that our reduction captures satisfiability of $\mathcal{K}$ w.r.t. Obj.

We first show that the groundings are performed properly.

Lemma 18. *Let $\mathcal{P} = (\mathfrak{D}(\Sigma_{\mathcal{S},\mathsf{Obj}}), \delta_{\mathcal{S},\mathsf{Obj}})$ be as defined above and $\langle \mathcal{I}, \sigma, \rho\rangle$ a state in the first-order transition system $\mathfrak{T}(\mathcal{P}) = (Q_\mathcal{P}, I_\mathcal{P}, \hookrightarrow_\mathcal{P}, \lambda_\mathcal{P})$ induced by $\mathcal{P}$ that satisfies the following conditions*

- *there exists an initial state $\langle \mathcal{I}_0, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}}\rangle \in I_\mathcal{P}$ with $\langle \mathcal{I}_0, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}}\rangle \hookrightarrow_\mathcal{P}{}^* \langle \mathcal{I}, \sigma, \rho\rangle$;*
- *$\mathcal{I} \models \mathit{AllGrounded}(s)$.*

Then, it holds that for all $x \in \mathsf{Var}$ there exists an object name $o \in \mathsf{Obj}$ such that $(A_x)^\mathcal{I} = \{o^\mathcal{I}\}$.

Proof. Let $\langle \mathcal{I}, \sigma, \rho\rangle$ be a state in the transition system $\mathfrak{T}(\mathcal{P})$ satisfying the conditions described above. Let

$$\mathfrak{D}(\Sigma_{\mathcal{S},\mathsf{Obj}}) = (\mathcal{F}, \mathcal{K}_{\mathcal{S},\mathsf{Obj}}, \mathsf{Act}_{\mathcal{S},\mathsf{Obj}}, \mathcal{E}, \succ_{\mathsf{poss}})$$

be the FO-DS induced by $\Sigma_{\mathcal{S},\mathsf{Obj}}$. Since $\langle \mathcal{I}, \sigma, \rho\rangle$ is reachable from an initial state, there is a model $\mathcal{I}_0 \models \mathcal{K}_{\mathcal{S},\mathsf{Obj}}$ such that $\mathcal{I}_0 \Rightarrow^\sigma_\mathfrak{D} \mathcal{I}$.

Note that since $\mathcal{I}_0 \models \mathcal{K}_{\mathcal{S},\mathsf{Obj}}$, we have $(A_x)^{\mathcal{I}_0} = \emptyset$ for all $x \in \mathsf{Var}$.

Since the ABox assertion $\mathit{AllGrounded}(s)$ is true in $\mathcal{I}$, the state $\langle \mathcal{I}, \sigma, \rho\rangle$ was reached by executing `finished`. It follows that σ is of the form

$$\sigma = \mathtt{ground}_{x_1,o_{j_1}} \cdot \cdots \cdot \mathtt{ground}_{x_n,o_{j_n}} \cdot \mathtt{finished}$$

for some objects $o_{j_1}, \ldots, o_{j_n} \in \mathsf{Obj}$. Now, $\mathcal{I}_0 \Rightarrow^{\sigma}_{\mathfrak{D}} \mathcal{I}$ implies that

$$\mathcal{I}_0 \Rightarrow_{\mathfrak{D}}^{\mathtt{ground}_{x_1,o_{j_1}}} \mathcal{I}_1 \Rightarrow_{\mathfrak{D}}^{\mathtt{ground}_{x_2,o_{j_2}}} \cdots \Rightarrow_{\mathfrak{D}}^{\mathtt{ground}_{x_n,o_{j_n}}} \mathcal{I}_n \Rightarrow_{\mathfrak{D}}^{\mathtt{finished}} \mathcal{I}.$$

It follows that $(A_{x_i})^{\mathcal{I}} = \{o_{j_i}^{\mathcal{I}}\}$ for all $i = 1, \ldots, n$. □

We are now ready to prove that consistency of $\mathcal{K}$ w.r.t. Obj implies satisfiability of $\Phi_{\mathcal{S},\mathsf{Obj}}$ in $\mathcal{P}$.

Lemma 19. *If $\mathcal{K}$ is satisfiable w.r.t.* Obj*, then $\Phi_{\mathcal{S},\mathsf{Obj}}$ is satisfiable by $\mathcal{P} = (\mathfrak{D}(\Sigma_{\mathcal{S},\mathsf{Obj}}), \delta_{\mathcal{S},\mathsf{Obj}})$.*

Proof. Assume that $\mathcal{K}$ is satisfiable w.r.t. Obj. Let

$$\mathfrak{D}(\Sigma_{\mathcal{S},\mathsf{Obj}}) = (\mathcal{F}, \mathcal{K}_{\mathcal{S},\mathsf{Obj}}, \mathsf{Act}_{\mathcal{S},\mathsf{Obj}}, \mathcal{E}, \succ_{\mathsf{poss}})$$

be the FO-DS induced by $\Sigma_{\mathcal{S},\mathsf{Obj}}$. The newly introduced concept names $A_{x_1}, \ldots, A_{x_n}$ and *AllGrounded* do not occur in $\mathcal{K}$. Therefore, there exists an interpretation $\mathcal{I}$ with

$$\mathcal{I} \models \mathsf{ground}(\mathcal{K}, \mathsf{Obj}) \text{ and } \mathcal{I} \models \mathcal{K}_{\mathcal{S},\mathsf{Obj}}.$$

Consequently, $\langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}}\rangle$ is an initial state in $\mathfrak{T}(\mathcal{P})$. We show that

$$\mathfrak{T}(\mathcal{P}), \langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}}\rangle \models \mathsf{AG}\big(\mathit{AllGrounded}(s) \rightarrow \big(\bigwedge_{C \sqsubseteq D \in \mathcal{K}} \widehat{C} \sqsubseteq \widehat{D}\big)\big).$$

Let $\pi \in \mathsf{paths}(\mathfrak{T}(\mathcal{P}), \langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}}\rangle)$ and $j \geq 0$ a natural number such that

$$\mathfrak{T}(\mathcal{P}), \pi[j] \models \mathit{AllGrounded}(s).$$

Let $\pi[j] = \langle \mathcal{J}, \sigma, \rho\rangle$. It follows that $\mathcal{I} \Rightarrow^{\sigma}_{\mathfrak{D}} \mathcal{J}$. Since all the concept and role names occurring in $\mathcal{K}$ are not affected by the actions in $\mathsf{Act}_{\mathcal{S},\mathsf{Obj}}$, we have that $\mathcal{I} \models \mathsf{ground}(\mathcal{K}, \mathsf{Obj})$ implies also $\mathcal{J} \models \mathsf{ground}(\mathcal{K}, \mathsf{Obj})$. Furthermore, due to Lemma 18 it follows that there are object names $o_{j_1}, \ldots, o_{j_n} \in \mathsf{Obj}$ such that

$$(A_{x_i})^{\mathcal{J}} = \{o_{j_i}^{\mathcal{J}}\} \text{ for all } i = 1, \ldots, n.$$

There exists a variable mapping ν_σ such that $\nu_\sigma(x_i) = o_{j_i}$ for all $i = 1, \ldots, n$. With $\mathcal{J} \models \mathsf{ground}(\mathcal{K}, \mathsf{Obj})$ it follows that

$$\mathcal{J} \models \bigwedge_{C \sqsubseteq D \in \mathcal{K}} C^{\nu_\sigma} \sqsubseteq D^{\nu_\sigma}.$$

Due to $(A_{x_i})^{\mathcal{J}} = \{\nu_\sigma(x_i)\}^{\mathcal{J}}$ for all $i = 1, \ldots, n$ it follows that

$$\mathcal{J} \models \bigwedge_{C \sqsubseteq D \in \mathcal{K}} \widehat{C} \sqsubseteq \widehat{D}.$$

Consequently, we have

$$\mathfrak{T}(\mathcal{P}), \pi[j] \models \mathit{AllGrounded}(s) \rightarrow \big(\bigwedge_{C \sqsubseteq D \in \mathcal{K}} \widehat{C} \sqsubseteq \widehat{D}\big).$$

Note that $\pi \in \mathsf{paths}(\mathfrak{T}(\mathcal{P}), \langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}}\rangle)$ and $j \geq 0$ are arbitrarily chosen. The claim follows directly. □

Lemma 20. *If $\Phi_{\mathcal{S},\mathsf{Obj}}$ is satisfiable in $\mathcal{P} = (\mathfrak{D}(\Sigma_{\mathcal{S},\mathsf{Obj}}), \delta_{\mathcal{S},\mathsf{Obj}})$, then $\mathcal{K}$ is satisfiable w.r.t.* Obj.

Proof. Let

$$\mathfrak{D}(\Sigma_{\mathcal{S},\mathsf{Obj}}) = (\mathcal{F}, \mathcal{K}_{\mathcal{S},\mathsf{Obj}}, \mathsf{Act}_{\mathcal{S},\mathsf{Obj}}, \mathcal{E}, \succ_{\mathsf{poss}})$$

be the FO-DS induced by $\Sigma_{\mathcal{S},\mathsf{Obj}}$. Assume there is an initial state $\langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}} \rangle$ in $\mathfrak{T}(\mathcal{P})$ such that $\mathfrak{T}(\mathcal{P}), \langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}} \rangle \models \Phi_{\mathcal{S},\mathsf{Obj}}$. We show that for an arbitrary variable mapping ν with $\nu(x_i) \in \mathsf{Obj}$ for all $i = 1, \ldots, n$ it holds that

$$\mathcal{I} \models \bigwedge_{C \sqsubseteq D \in \mathcal{K}} C^\nu \sqsubseteq D^\nu.$$

This implies $\mathcal{I} \models \mathsf{ground}(\mathcal{K}, \mathsf{Obj})$. For an arbitrary but fixed ν with $\nu(x_i) \in \mathsf{Obj}$ for all $i = 1, \ldots, n$ we consider the following action sequence:

$$\sigma^\nu = \mathtt{ground}_{x_1,\nu(x_1)} \cdot \cdots \cdot \mathtt{ground}_{x_n,\nu(x_n)} \cdot \mathtt{finished}.$$

Obviously, σ^ν is executable in $\mathcal{I}$ and is admitted in $\delta_{\mathcal{S},\mathsf{Obj}}$. There exists an interpretation $\mathcal{J}^\nu$ with $\mathcal{I} \Rightarrow_{\mathfrak{D}}^{\sigma^\nu} \mathcal{J}^\nu$ and subprogram $\rho \in \mathsf{sub}(\delta_{\mathcal{S},\mathsf{Obj}})$ such that $\langle \mathcal{J}^\nu, \sigma^\nu, \rho \rangle$ is a state in $\mathfrak{T}(\mathcal{P})$ and is reachable from $\langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}} \rangle$. Initially, we have

$$\mathcal{I} \models \{A_x \sqsubseteq \bot \mid x \in \mathsf{Var}\}.$$

It follows that after executing σ in $\mathcal{I}$ the resulting interpretation $\mathcal{J}^\nu$ satisfies:

$$(A_{x_i})^{\mathcal{J}^\nu} = \{\nu(x_i)^{\mathcal{J}^\nu}\} \text{ for all } i = 1, \ldots, n \text{ and } \mathcal{J}^\nu \models \mathit{AllGrounded}(s).$$

The assumption $\mathfrak{T}(\mathcal{P}), \langle \mathcal{I}, \langle\rangle, \delta_{\mathcal{S},\mathsf{Obj}} \rangle \models \Phi_{\mathcal{S},\mathsf{Obj}}$ implies that

$$\mathcal{J}^\nu \models \bigwedge_{C \sqsubseteq D \in \mathcal{K}} \widehat{C} \sqsubseteq \widehat{D}.$$

With $(A_{x_i})^{\mathcal{J}^\nu} = \{\nu(x_i)^{\mathcal{J}^\nu}\}$ for all $i = 1, \ldots, n$ it follows that

$$\mathcal{J}^\nu \models \bigwedge_{C \sqsubseteq D \in \mathcal{K}} C^\nu \sqsubseteq D^\nu.$$

For all concept, role and object names X mentioned in $\mathcal{K}$ it holds that $X^{\mathcal{J}^\nu} = X^{\mathcal{I}}$, because the execution of σ^ν in $\mathcal{I}$ changes only the interpretation of the names $A_{x_1}, \ldots, A_{x_n}$ and *AllGrounded*, which are not occurring in $\mathcal{K}$. It follows that

$$\mathcal{I} \models \bigwedge_{C \sqsubseteq D \in \mathcal{K}} C^\nu \sqsubseteq D^\nu.$$

Since the grounding ν with $\nu(x_i) \in \mathsf{Obj}$ for all $i = 1, \ldots, n$ was arbitrarily chosen, we obtain $\mathcal{I} \models \mathsf{ground}(\mathcal{K}, \mathsf{Obj})$. □

Together with Theorem 17, we thus obtain the following lower bounds regarding verification for expressive DLs.

Theorem 21. *Verifying $\mathcal{L}$-CTL properties of an $\mathcal{L}$-ConGolog program is*

1. 2ExpTime*-hard for $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and*
2. coN2ExpTime*-hard for $\mathcal{L} = \mathcal{ALCQIO}$.*

Proof. Checking whether an $\mathcal{L}$-CTL state formula Φ is satisfiable in an $\mathcal{L}$-ConGolog program over local effect actions of the form $\mathcal{P} = (\mathfrak{D}(\Sigma), \delta)$, where Σ is a local effect $\mathcal{L}$-action theory is 2ExpTime-hard if $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and N2ExpTime-hard if $\mathcal{L} = \mathcal{ALCQIO}$. This is a consequence of Lemma 19 and 20 and Theorem 17. It holds that Φ is valid in $\mathcal{P}$ iff $\neg\Phi$ is not satisfiable. Therefore, the verification problem has the same complexity as the complementary problem of the satisfiability problem. Hence, verification is 2ExpTime-hard if $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and co-N2ExpTime-hard if $\mathcal{L} = \mathcal{ALCQIO}$. $\square$

Note that, since our reduction uses only a simple CTL property, only unconditional actions and no loops or tests in the program, this hardness result already applies to verification of CTL formulae over programs in a quite restricted fragment of $\mathcal{L}$-ConGolog.

5.2 The Lightweight Description Logic $\mathcal{ELO}_\bot$

Lemma 19 also gives us an ExpTime lower bound for the lightweight DL $\mathcal{ELO}_\bot$, since by Theorem 17, satisfiability of $\mathcal{ELOV}_\bot$-KBs is ExpTime-complete. As a lightweight DL, $\mathcal{ELO}_\bot$ admits tractable complexity for common reasoning tasks, which are between ExpTime and NExpTime for the other DLs considered in this paper. It might therefore be reasonable to believe that also the complexity of verification decreases when we restrict ourselves to $\mathcal{ELO}_\bot$. However, it turns out that this is not the case, and that already for $\mathcal{ELO}_\bot$, verification becomes 2ExpTime-hard, as it is already for $\mathcal{ALCO}$, $\mathcal{ALCIO}$ and $\mathcal{ALCQO}$.

Without loss of generality, we assume that the KB of the program is empty: if it is not, we can simply add it as conjunct to the verification formula, and obtain a formula that is satisfiable by the program without KB iff the original formula is satisfiable by the original program.

For technical reasons, it is furthermore convenient to assume that our formula is satisfiable by interpretations that have at least two domain elements. Note that may not be the case if the program or the formula contains axioms of the form $\top \sqsubseteq \{a\}$. The following lemma however shows that our simplifying assumption is sufficient.

Lemma 22. *Let Φ be a formula and $\mathcal{P}$ be a program. Then, there is a formula Φ' and a program $\mathcal{P}'$ that can be constructed in polynomial time from Φ and $\mathcal{P}$ s.t. Φ is satisfiable by $\mathcal{P}$ iff Φ' is satisfiable by $\mathcal{P}'$ by a state $q_0 \in I_{\mathcal{P}} = (\mathcal{I}, \langle\rangle, \delta)$ s.t. $|\Delta^{\mathcal{I}}| > 1$.*

Proof. We first note that nominals can be simulated using actions: for this, we use a fresh concept name A_a for every named individual a and replace every occurrence of $\{a\}$ by A_a. For every named individual, we add the axiom $A_a \sqsubseteq \bot$ to the initial KB, and prepend the program with a sequence of actions that places a into the extension of a, so that a is the only named individual in the extension of A_a. It is easy to see that A_a now behaves exactly like the concept $\{a\}$. For this reason, we can assume without loss of generality that neither $\mathcal{P}$ nor Φ contain any concepts of the form $\{a\}$. The idea is now to duplicate every named individual a into two named individuals a_1 and a_2, and replace the effects of every action α in $\mathcal{P}$ so that all changes apply two both individuals synchronously. Specifically, we replace

- every positive effect $A(a)$ by $A(a_1)$ and $A(a_2)$,
- every positive effect $r(a, b)$ by $r(a_1, b_1)$, $r(a_1, b_2)$, $r(a_2, b_1)$ and $r(a_2, b_2)$,
- every negative effect $\neg A(a)$ by $\neg A(a_1)$ and $\neg A(a_2)$, and
- every negative effect $\neg r(a, b)$ by $\neg r(a_1, b_1)$, $\neg r(a_1, b_2)$, $\neg r(a_2, b_1)$ and $\neg r(a_2, b_2)$.

Denote the resulting program by $\mathcal{P}'$. Now assume that Φ is only satisfiable by $\mathcal{P}$ via interpretations $\mathcal{I}$ s.t. $|\Delta^{\mathcal{I}}| = 1$. Specifically, let π be an execution path in $\mathfrak{T}(\mathcal{P})$ s.t. for some i, $|\Delta^{\lambda_{\mathcal{P}}(\pi[i])}| = 1$. There must then exists a named individual a s.t. $\lambda_{\mathcal{P}}(\pi[i]) \models \top \sqsubseteq \{a\}$. Clearly, there is a corresponding execution path π' in $\mathfrak{T}(\mathcal{P}')$ s.t. $\lambda_{\mathcal{P}'}(\pi[i]) \models \top \sqsubseteq \{a_1, a_2\}$, and therefore, $|\Delta^{\lambda_{\mathcal{P}'}(\pi'[i])}| = 2$. It follows that Φ is satisfiable in $\mathcal{P}$ iff it is satisfiable in $\mathcal{P}'$ via interpretations that have at most two domain elements. $\square$

We present our reduction. The main idea is to use the non-determinism inherent to expressions of the form $\exists r.C$, in which C might refer to a specific named individual whose interpretation we can change using an action. Depending on whether, in the current interpretation, the role successor of some domain element points to that specific individual or not, we interpret it as a concept or its complement.

In the following, let $\mathcal{P} = (\Sigma, \delta)$ be a fixed Golog program and Φ be a fixed CTL state formula to be verified against $\mathcal{P}$. We define a polynomially-sized program $\mathcal{P}'$ and a polynomiall-sized CTL state formulae Φ' such that both use only $\mathcal{ELO}_\bot$ concepts, and such that Φ is satisfiable by $\mathcal{P}$ iff Φ' is satisfiable by $\mathcal{P}'$. The idea is to use a special concept name $A_{\overline{D}}$ to represent the negation of a concept D, so that any domain element in a relevant interpretation in the transition system of $\mathcal{P}'$ satisfies $A_{\overline{D}}$ iff it satisfies $\neg D$. We first define an operation that replaces $\mathcal{ALCO}$ concepts by $\mathcal{ELO}_\bot$ concepts by using these special concept names. For a given $\mathcal{ALCO}$ concept C, we define the operation $C^\dagger$ as the result of replacing from C every outermost concept of the form $\neg D$ by $A_{\overline{D}}$, where $A_{\overline{D}}$ is a concept fresh to both $\mathcal{P}$ and Φ, and uniquely determined by D.

To ensure that every domain element in an interpretation satisfies either C or $A_{\overline{D}}$, we define the KB ψ_1 which contains for every introduced concept name $A_{\overline{D}}$ the following $\mathcal{ELO}_\bot$-axioms, where S_D, a_D, $a_{\overline{D}}$ and r_a are fresh.

- $D^\dagger \sqcap A_{\overline{D}} \sqsubseteq \bot$,
- $\top \sqsubseteq \exists r_a.S_D$,
- $\exists r_a.\{a_D\} \sqsubseteq D^\dagger$, and
- $\exists r_a.\{a_{\overline{D}}\} \sqsubseteq A_{\overline{D}}$.

Note that we may need to introduce further concept names $A_{D'}$ due to the concept $D^\dagger$ occurring in the axioms, for which we inductively add the corresponding axioms as well. The first axiom ensures that every domain element can only satisfy one of $D^\dagger$ or $A_{\overline{D}}$, thus expressing that the concepts D and $\neg D$ must be disjoint. The next axioms ensure for the extended program $\mathcal{P}'$ defined below that it also satisfies at least one of $D^\dagger$ and $A_{\overline{D}}$, thus completing the behaviour of concept negation. Specifically, we demand that every domain element d has an r_a-successor satisfying S_D, for which we in the program will make sure that it contains exactly two named individuals, namely a_D and $a_{\overline{D}}$. Depending on which is the case, d will then satisfy either $D^\dagger$ or $A_{\overline{D}}$.

We are now ready to define the $\mathcal{ELO}_\bot$ program sequence δ', which is of the form $\psi_2?\alpha_s;\delta^\dagger$, with the components ψ_2, α_s and $\delta^\dagger$ defined one after the other in the following. The role of the prefix $\psi_2?\alpha_s$ is to ensure that in execution paths of the program, S_D contains only the two named individuals a_D and $a_{\overline{D}}$, so that the axioms in ψ_1 have their intended effect.

ψ_2 contains for every introduced concept name S_D the axiom $S_D \sqsubseteq \bot$ to make sure that the interpretation of S_D is empty. α_s now assigns the required named individuals, which is achieved by setting the effects

$$\mathsf{Eff}(\alpha_s) = \{\langle S_D(a_D)\rangle^+, \langle S_D(a_{\overline{D}})\rangle^+ \mid \ S_D \text{ was introduced } \}.$$

Note that actions may change which concept names a named individual satisfies, but any such change has to be specified explicitly by the action. Therefore, our program has to deal explicitly with the interpretation of the introduced names. For each introduced concept name $A_{\overline{D}}$ and each named individual a occurring in $\mathcal{P}$ or Φ, we use the actions $\alpha_{D,d}$ and $\alpha_{\overline{D},a}$ for this, which are defined by

$$\mathsf{Eff}(\alpha_{D,d}) = \{\langle A_{\overline{D}}(d)\rangle^-, \langle r_a(a, a_D)\rangle^+ \langle r_a(d, a_{\overline{D}})\rangle^-\}$$

and

$$\mathsf{Eff}(\alpha_{\overline{D},d}) = \{\langle A_{\overline{D}}(d)\rangle^+, \langle r_a(d, a_D)\rangle^-, \langle r_a(d, a_{\overline{D}})\rangle^-\},$$

and are then non-deterministically applied after each action. We further use two special actions α_e^+ and α_e^- to mark "exceptional states" in which the interpretation of $A_{\overline{D}}$ might not have been repaired yet, which are defined by $\mathsf{Eff}(\alpha_e^+) = \{\langle A_e(a_e)\rangle^+\}$ and $\mathsf{Eff}(\alpha_e^-) = \{\langle A_e(a_e)\rangle^-\}$. Intuitively, $A_e(a_e)$ marks states in which we are currently fixing the interpretation of the introduced symbols.

From the original program sequence δ, we obtain $\delta^\dagger$ by replacing every axiom of the form $C \sqsubseteq D$ by $C^\dagger \sqsubseteq D^\dagger$, every axiom of the form $C(a)$ by $C^\dagger(a)$, and every action α with

$$\alpha; \alpha_e^+; (\alpha_{C_1,a_1} \mid \alpha_{\overline{C_1},a_1}); \ldots; (\alpha_{C_n,a_n} \mid \alpha_{\overline{C_n},a_n}); \alpha_e^-,$$

where C_i, a_i range over all (quadratically many) pairs of named individuals a_i occurring in the input and concepts C_i for which we introduced the concept name $A_{\overline{C_i}}$. Note that this explores all possibilities of choosing some assignment in the program. The CTL* formula will later make sure that we pick the right one in each case.

This formula we construct next. For this, we define the operator $\cdot^\circ$ on CTL*-formulae, which is recursively defined as follows.

- $(C \sqsubseteq D)^\circ = (C^\dagger \sqsubseteq D^\dagger)$,
- $(C(a))^\circ = C^\dagger(a)$,
- $(\neg\Psi)^\circ = \neg\Psi^\circ$,
- $(\Psi_1 \wedge \Psi_2)^\circ = \Psi_1^\circ \wedge \Psi_2^\circ$,
- $(\mathsf{X}\Psi)^\circ = \mathsf{X}(A_e(a_e) \,\mathsf{U}\, \Psi^\circ)$,
- $(\Psi_1 \,\mathsf{U}\, \Psi_2)^\circ = (A_e(a_e) \vee \Psi_1^\circ) \,\mathsf{U}\, (\neg A_e(a_e) \wedge \Psi_2^\circ)$
- $(\mathsf{E}\Psi)^\circ = \mathsf{E}(\mathsf{G}(A_e(a_e) \vee \Psi_1) \wedge (A_e(a_e) \,\mathsf{U}\, \Psi^\circ))$

We first translate all $\mathcal{ALCO}$-concepts into $\mathcal{ELO}_\bot$ concepts using the operator $\cdot^\circ$ defined above. Since we appended to each action in the original program a sequence of non-deterministic actions, we have to translate formulae of the form $\mathsf{X}\Psi$ so that the states in which the interpretation is still adapted, which are those that satisfy $A_e(a_e)$, are followed first, for which we use the until-operator. $A_e(a_e)$ has also to be taken into consideration for formulae of the form $\psi_1 \,\mathsf{U}\, \Psi_2$. When we choose a path that for formulae of the form $\mathsf{E}\Psi$, we now have to additionally make sure we pick a path in which the introduced names are repaired correctly, which we check with the formula $\mathsf{G}(A_e(a_e) \vee \psi_1)$. Again, we skip states which satisfy $A_e(a_e)$, before we check the translated formula Ψ°.

The final CTL-formula Φ' is now of the form $\mathsf{EX}(\Phi^\circ \wedge \mathsf{G}(\psi_1 \vee A_e(a_e))$. This formula picks a path in which the introduced names are always repaired in accordance with ψ_1, and skips the first action α_s using the next operator, before the translated input formula Φ° is verified.

Lemma 23. *If Φ is satisfiable by $\mathcal{P}$, then Φ' is satisfiable by $\mathcal{P}'$.*

Proof. Assume Φ is satisfiable in $\mathcal{P}$, and let $q_0 = (\mathcal{I}_0, \langle\rangle, \delta) \in I_\mathcal{P}$ be an initial state in $\mathfrak{T}(\mathcal{P})$ s.t. $\mathfrak{T}(\mathcal{P}), q_0 \models \Phi$.

We first define a transformation on interpretations and states reachable from q_0. Given an interpretation $\mathcal{I}$ occurring in a state reachable from $\mathcal{I}_0$, we extend $\mathcal{I}$ to a model $\mathcal{I}^\circ$ of ψ_1 s.t. all concept and role names that occur in the input are interpreted the same by $\mathcal{I}$, and additionally we have $A_{\overline{D}}^{\mathcal{I}^\circ} = (\neg C)^{\mathcal{I}^\circ}$ for all introduced concept names $A_{\overline{D}}$. For every such introduced concept name, we proceed as follows. The interpretation of $A_{\overline{D}}$ is straightforward:

- $A_{\overline{D}}^{\mathcal{I}^\circ} = (\neg D)^{\mathcal{I}}$.

The other introduced symbols are interpreted as follows. We select two individuals $e_D, e_{\overline{d}} \in \Delta^{\mathcal{I}_0}$ s.t. $e_D \neq e_{\overline{D}}$ and set

- $a_D^{\mathcal{I}^\circ} = e_D$,
- $a_{\overline{D}}^{\mathcal{I}} = e_{\overline{D}}$,
- $S_D^{\mathcal{I}} = \{e_d, e_{\overline{d}}\}$, and
- $(r_a)^{\mathcal{I}^\circ} = \{(d, e_D) \mid d \in D^{\mathcal{I}}\} \cup \{(d, e_{\overline{D}}) \mid d \in (\neg D)^{\mathcal{I}}\}$

The construction makes sure that for every introduced concept name $A_{\overline{D}}$, we have $A_{\overline{D}}^{\mathcal{I}^\circ} = (\neg D)^{\mathcal{I}}$ and $(D^\dagger)^{\mathcal{I}^\circ} = D^{\mathcal{I}}$.

The initial state $q_{-1}^\circ \in I_{\mathcal{P}'}$ is defined as $q_{-1}^\circ = (\mathcal{I}_{-1}^\circ, \langle\rangle, \delta^\dagger)$, where $\mathcal{I}_{-1}^\circ$ is identical to $\mathcal{I}_0^\circ$ except that $S_{\overline{D}} = \emptyset$ for all introduced concept names $S_{\overline{D}}$. Note that we have $\mathcal{I}_{-1}^\circ \models \psi_2$, and also

$$(\mathcal{I}_{-1}^\circ, \langle\rangle, \phi_2?; \alpha_s; \delta^\dagger) \hookrightarrow_{\mathcal{P}'} (\mathcal{I}_0^\circ, \alpha_s, \delta^\dagger).$$

For all other states $q \in Q_\mathcal{P}$, we define q° by induction on the execution paths in q°. Specifically, if $q_1 = (\mathcal{I}_1, \sigma, \delta_1) \in Q_\mathcal{P}$, $q_1^\circ = (\mathcal{I}_1^\circ, \sigma^\circ, \delta_1^\circ) \in Q_{\mathcal{P}'}$ and $q_2 = (\mathcal{I}_2, \sigma \cdot \alpha, \delta_2)$ s.t. $q_1 \hookrightarrow_\mathcal{P} q_2$, then q_2° is defined as

$$(\mathcal{I}_2^\circ, \sigma^\circ \cdot \alpha \cdot \alpha_e^+ \cdot \alpha_1 \cdot \ldots \cdot \alpha_n \alpha_e^-, \delta_2^\circ),$$

where $\alpha_1 \cdot \ldots \cdot \alpha_n$ is a sequence of actions chosen from the program expression

$$(\alpha_{C_1,a_1} \mid \alpha_{\overline{C_1},a_1}); \ldots; (\alpha_{C_n,a_n} \mid \alpha_{\overline{C_n},a_n});$$

used when constructing $\mathcal{P}'$, so that q_2° is a state in $Q_{\mathcal{P}'}$. Such a state always exists, and clearly we have $q_1^\circ \hookrightarrow_{\mathcal{P}'}{}^* q_2^\circ$. Note that this associates to every state $q \in Q_\mathcal{P}$ s.t. $q_0 \hookrightarrow_\mathcal{P} q$ a state q° s.t. $q_0^\circ \hookrightarrow_{\mathcal{P}'}{}^* q^\circ$. Furthermore, for any two states q_1, q_2 s.t. $q_1 \hookrightarrow_\mathcal{P} q_2$, there is a path between q_1° and q_2° in which every state satisfies $A_e(a_e)$. This means that every path π in $\mathfrak{T}(\mathcal{P})$ has a corresponding path π° in $\mathfrak{T}(\mathcal{P}')$ s.t. for every $i \geq 0$, there exists $j \geq 0$ s.t. $\pi^\circ[j] = (\pi[i])^\circ$. Furthermore, for every consequtive values $i, j = i + 1$, we have

- $\pi^\circ[i'] = (\pi[i])^\circ$,
- $\pi^\circ[j'] = (\pi[j])^\circ$,
- $i' < j'$ and
- $\mathfrak{T}(\mathcal{P}'), \pi^\circ[k] \models A_e(a_e)$ for all k with $i' < k < j'$.

As a consequence, for every path π in $\mathfrak{T}(\mathcal{P})$ s.t. $\pi[0] = q_0$, we have $\mathfrak{T}(\mathcal{P}'), \pi^\circ \models \mathsf{G}(A_e(a_e) \vee \psi_1)$.

To show that $\mathfrak{T}(\mathcal{P}'), q_0^\circ \models \Phi^\circ$, we proceed by structural induction on Φ. For this, we have to show that for every path π in $\mathfrak{T}(\mathcal{P})$ s.t $\pi[0] = q_0$ and for every $i \geq 0$,

1. if Ψ is a state formula, then $\mathfrak{T}(\mathcal{P}), \pi[i] \models \Psi$ iff there is path π° in $\mathfrak{T}(\mathcal{P}')$ with $\pi^\circ[0] = q_0^\circ$ and an index j s.t. $(\pi[i])^\circ = \pi^\circ[j]$ and $\mathfrak{T}(\mathcal{P}'), \pi^\circ[j] \models \Psi^\circ$,
2. if Ψ is a path formula, then $\mathfrak{T}(\mathcal{P}), \pi[i..] \models \Psi$ iff there is path π° in $\mathfrak{T}(\mathcal{P}')$ with $\pi[0] = q_0^\circ$ and an index j s.t. $(\pi[i])^\circ = \pi[j]$ and $\mathfrak{T}(\mathcal{P}'), \pi^\circ[j..] \models \Psi^\circ$.

Let $\pi[i] = (\mathcal{I}, \sigma, \delta)$. We distinguish the cases based on the syntactical shape of Ψ.

1. $\Psi = C \sqsubseteq D$. Then $\mathcal{I} \models C \sqsubseteq D$ iff $\mathcal{I}^\circ \models C^\dagger \sqsubseteq D^\dagger$, and consequently also $\pi[i] \models C \sqsubseteq D$ iff $(\pi[i])^\circ \models C^\dagger \sqsubseteq D^\dagger$. The latter holds exactly iff there is a path π° in $\mathfrak{T}(\mathcal{P}')$ s.t. $\pi^\circ[0] = q_0^\circ$ and some $j > 0$ s.t. $\pi^\circ[j] = (\pi[i])^\circ$.
2. If Ψ is of one of the forms $\neg\Psi_1$ or $\Psi_1 \wedge \Psi_2$, then our claim follows directly from the inductive hypothesis.
3. If Ψ is of the form $\mathsf{X}\Psi_1$, then by our inductive hypothesis, $\mathfrak{T}(\mathcal{P}), \pi[i+1..] \models \Psi$ iff $\mathfrak{T}(\mathcal{P}'), \pi^\circ[j..] \models \Psi^\circ$ for some path π° s.t. $(\pi[i+1])^\circ = \pi^\circ[j]$. By construction, this is the case iff there is some $k < j$ s.t. $\pi^\circ[k] = (\pi[i])^\circ$ and for all l s.t. $k < l < j$, $\pi^\circ[l] \models A_e(a_e)$. This means that $\pi[i..] \models \Psi$ iff $\mathfrak{T}(\mathcal{P}'), \pi^\circ[l..] \models \mathsf{X}(A_e(a_e) \mathbin{\mathsf{U}} (\Psi')^\circ) = \Psi^\circ$.
4. Assume Ψ is of the form $\Psi_1 \mathbin{\mathsf{U}} \Psi_2$. If $\mathfrak{T}(\mathcal{P}), \pi[i..] \models \Psi$, then there exists $j > i$ s.t. $\mathfrak{T}(\mathcal{P}), \pi[j..] \models \Psi_2$ and for all k, $j \leq k < i$, $\mathfrak{T}(\mathcal{P}), \pi[k..] \models \Psi_1$. By inductive hypothesis, there is then a path π° i) there is an index j' s.t. $\mathfrak{T}(\mathcal{P}'), \pi^\circ[j'..] \models \Psi_2$ and $\pi^\circ[j'] = (\pi[j])^\circ$, and ii) for every k, $j \leq k \leq i$, there is an index k' s.t. $\mathfrak{T}(\mathcal{P}'), \pi^\circ[k'..] \models \Psi_1$ and $\pi^\circ[k'] = (\pi[k])^\circ$. For every such k, every state between $(\pi[k])^\circ$ and $(\pi[k+1])^\circ$ in π° must satisfy $A_e(a_e)$. As a consequence, we obtain $\pi^\circ[i'] \models (A_e(a_e) \vee \Psi_1^\circ) \mathbin{\mathsf{U}} (\neg A_e(a_e) \wedge \Psi_2^\circ) = \Psi^\circ$, where $\pi^\circ[i'] = (\pi[i])^\circ$. The other direction is shown correspondingly.
5. Assume Ψ is of the form $\mathsf{E}\Psi_1$. We again show only one direction of the proof, the other direction is shown accordingly. If $\mathfrak{T}(\mathcal{P}), \pi[i] \models \mathsf{E}\Psi_1$, then there exists some path π_2 in $\mathfrak{T}(\mathcal{P})$ s.t. $\pi[..i] = \pi_2[..i]$ and $\mathfrak{T}(\mathcal{P}), \pi_2[i..] \models \Psi_1$. By the inductive hypothesis, there then exists a path π_2° and an index j s.t. $\pi_2^\circ[j] = (\pi_2[j])^\circ$ and $\mathfrak{T}(\mathcal{P}'), \pi_2^\circ \models \Psi_1^\circ$. As observed above, we have $\mathfrak{T}(\mathcal{P}'), \pi_2^\circ \models \mathsf{G}(A_e(a_e) \vee \Psi_1)$, and consequently, $\mathfrak{T}(\mathcal{P}'), \pi_2^\circ[j] \models \mathsf{E}(\mathsf{G}(A_e(a_e) \vee \Psi_1) \wedge (A_e(a_e) \mathbin{\mathsf{U}} \Psi^\circ)) = \Psi^\circ$.

It follows that $\mathfrak{T}(\mathcal{P}'), q_{-1}^\circ \models \Phi'$, and that Φ' is satisfiable in $\mathcal{P}'$. □

Lemma 24. *If Φ' is satisfiable by $\mathcal{P}'$, then Φ is satisfiable by $\mathcal{P}$.*

Proof. Assume Φ' is satisfiable by $\mathcal{P}'$. There then exists some initial state $q_0 = (\mathcal{I}_0, \langle\rangle, \delta') \in I_{\mathcal{P}'}$ s.t. $\mathfrak{T}(\mathcal{P}'), q_0 \models \Phi'$.

We first show that the interpretations $\mathcal{I}$ reachable by q_0 interpret the fresh concept names $A_{\overline{D}}$ correctly, provided that $\mathcal{I} \models \psi_1$.

Claim 25. For every $q = (\mathcal{I}, \sigma, \delta) \in Q_{\mathcal{P}'}$ s.t. $q_0 \hookrightarrow_{\mathcal{P}'}{}^* q$ and $\mathfrak{T}(\mathcal{P}'), q \models \psi_1$, and for every introduced concept name $A_{\overline{D}}$, we have $A_{\overline{D}}^{\mathcal{I}} = (\neg C)^{\mathcal{I}}$.

Proof of claim. Let $q = (\mathcal{I}, \sigma, \delta)$ be as in the claim. Since δ' starts with the test ψ_2, we have $\mathcal{I}_0 \models \psi_2$, and $S_D^{\mathcal{I}_0} = \emptyset$ for every introduced concept name S_D. Because α is the first executed action, there is exactly one state q_1 s.t. $q_0 \hookrightarrow_{\mathcal{P}'} q_1$, for which we have $q_1 = (\mathcal{I}_1, \alpha, \delta^\dagger)$. Furthermore, $a_D^{\mathcal{I}_1}$ and $a_{\overline{D}}^{\mathcal{I}_1}$, are the only domain elements in $S_D^{\mathcal{I}_1}$. Since no other action in $\mathcal{P}'$ modifies the interpretation of S_D, this is also the case for $\mathcal{I}$, that is, $S_D^{\mathcal{I}} = \{a_D^{\mathcal{I}}, a_{\overline{D}}^{\mathcal{I}}\}$.

Let $d \in \Delta^{\mathcal{I}_i}$ and let $A_{\overline{D}}$ be an introduced concept name. We show that then, $d \in A_{\overline{D}}^{\mathcal{I}_i}$ iff $d \in (\neg D)^{\mathcal{I}_i}$. We first argue that, for every introduced concept name $A_{\overline{D}}$ and every domain element $d \in \Delta^{\mathcal{I}}$, d satisfies either $D^\dagger$ or $A_{\overline{D}}$. Because $\top \sqsubseteq \exists r_a.S_D \in \psi_1$ and $\mathcal{I} \models \psi_1$, d must have an r_a-successor d_2 in $\mathcal{I}$ s.t. $d_2 \in S_D^{\mathcal{I}}$. For d_2, there are exactly two possibilities.

1. $d_2 = a_{\overline{D}}^{\mathcal{I}}$. Because ψ_1 contains the axiom $\exists r_a.\{a_{\overline{D}}\} \sqsubseteq A_{\overline{D}}$, this implies that $d \in A_{\overline{D}}^{\mathcal{I}}$.
2. $d_2 = a_D^{\mathcal{I}}$. Because ψ_1 contains the axiom $\exists r_a.\{a_D\} \sqsubseteq D^\dagger$, this implies that $d \in (D^\dagger)^{\mathcal{I}}$.

Since $\mathcal{I}_i \models A_{\overline{D}} \sqcap D^\dagger \sqsubseteq \bot$, only one of the last two cases is possible. It follows that every domain element $d \in \Delta^{\mathcal{I}_i}$ is either in $(D^\dagger)^{\mathcal{I}_i}$ or in $A_{\overline{D}}^{\mathcal{I}_i}$. From here it follows by structural induction on the concept D that for every every $d \in \Delta^{\mathcal{I}_i}$, $d \in A_{\overline{D}}^{\mathcal{I}_i}$ implies $d \in (\neg D)^{\mathcal{I}_i}$, and $d \in (D^\dagger)^{\mathcal{I}_i}$ implies $d \in D^{\mathcal{I}_2}$. *End of proof of claim.*

As a consequence of the claim, we have for $q \in Q_{\mathcal{P}'}$ s.t. $q_0 \hookrightarrow_{\mathcal{P}'} q$ and $\mathfrak{T}(\mathcal{P}'), q_0 \models \psi_1$, that for every axiom $C \sqsubseteq D$ occurring in the input, $\mathfrak{T}(\mathcal{P}'), q \models C \sqsubseteq D$ iff $\mathfrak{T}(\mathcal{P}'), q \models C^\dagger \sqsubseteq D^\dagger$.

To show that Φ is satisfiable by $\mathcal{P}$, we first provide translations from paths π in $\mathfrak{T}(\mathcal{P}')$ to paths π' in $\mathfrak{T}(\mathcal{P})$. We first remove states that have no corresponding state in $\mathcal{P}$. From π, we obtain π_1 by removing the first state and every state $\pi[i]$ s.t. $\mathfrak{T}(\mathcal{P}'), \pi[i] \models A_e(a_e)$. To make sure that the states appear in $\mathfrak{T}(\mathcal{P})$, we further have to adapt the state descriptions. From π_1, we obtain π_t by replacing each state $(\mathcal{I}, \sigma, \delta_1)$ by $(\mathcal{I}, \sigma', \delta_1')$, where

- σ' is obtained from σ by removing every action introduced to an introduced name (those are the initial action α_s, as well as the actions $\alpha_{a,D}$, $\alpha_{a,\overline{D}}$ modifying the interpretation of the introduced concept names $A_{\overline{D}}$), and
- δ_1' is obtained from δ_1 by replacing α_e^+, α_e^- and $(\alpha_{C_i,d_i} \mid \alpha_{\overline{C_i},d_i})$ by $\langle\rangle$, replacing every axiom $C^\dagger \sqsubseteq D^\dagger$ by $C \sqsubseteq D$ and replacing every axiom $C^\dagger(a)$ by $C(a)$.

Note that the transformation on δ effectively undoes our transformation $\cdot^\circ$, so that for $\pi_t[i] = (\mathcal{I}, \sigma', \delta)$, we have $\pi_1[i] = (\mathcal{I}, \sigma, \delta^\circ)$.

Using our claim, it is now standard to verify that for every path π in $\mathfrak{T}(\mathcal{P}')$ s.t $\pi[0] = q_0$ and $\mathfrak{T}(\mathcal{P}'), \pi \models \mathsf{G}(A_e(a_e) \vee \psi_2)$, π_t is a path in $\mathfrak{T}(\mathcal{P})$ (we only replaced axioms in the tests by equi-satisfiable axioms, and omitted all actions that modify the introduced concept names). Note that the query Φ' only refers to paths that satisfy this property.

Now let q_1 be the first state reached from q_0, that is, $q_0 \hookrightarrow_{\mathcal{P}'} q_1$, and q_1° be the transformation of q_1 according to the above construction. Note that q_1 is the first state in any path π_t constructed from π with $\pi[0] = q_0$. Furthermore, every path π in $\mathfrak{T}(\mathcal{P})$ is such that $\pi = \pi_t'$ for some path π' in $\mathfrak{T}(\mathcal{P}')$ s.t. $\pi'[0] = q_0$ and $\mathfrak{T}(\mathcal{P}'), \pi' \models \mathsf{G}(A_e(a_e) \vee \psi_1)$.

In order to show that Φ is satifiable in $\mathcal{P}$, we show that $\mathcal{P}, q_1^\circ \models \Phi$. For this, we proceed by structural induction on Φ. Specifically, we show that for every path π_t in $\mathfrak{T}(\mathcal{P}')$ s.t. $\pi_t[0] = q_1$, every subformula Ψ of Φ, and every index $i \geq 0$, that $\mathfrak{T}(\mathcal{P}), \pi_t[i] \models \Psi$ iff $\mathfrak{T}(\mathcal{P}), \pi[j] \models \Psi^\circ$ for some $j > 0$ with $(\pi[j])^\circ = \pi_t[i]$, and $\mathfrak{T}(\mathcal{P}), \pi_t[i..] \models \Psi$ iff $\mathfrak{T}(\mathcal{P}), \pi[j..] \models \Psi^\circ$ for some $j > 0$ with $(\pi[j])^\circ = \pi_t[i]$.

1. If Ψ is of the form $C \sqsubseteq D$, then $\pi_t[i] \models C \sqsubseteq D$ iff $\pi[j] \models C^\dagger \sqsubseteq D^\dagger$ by our claim.

2. If Ψ is of the form $C(a)$, then $\pi_t[i] \models C(a)$ iff $\pi[j] \models C^\dagger(a)$ by our claim.

3. If Ψ is of one of the forms $\neg\Psi'$ and $\Psi_1 \wedge \Psi_2$, then $\pi_t[i] \models \Psi$ iff $\pi[j] \models \Psi^\circ$ by the inductive hypothesis.

4. If Ψ is of the form $\mathsf{X}\Psi'$, then $\Psi^\circ = \mathsf{X}(A_e(a_e) \,\mathsf{U}\, \Psi^\circ)$. Since π_t is obtained from π by removing all states q s.t. $q \models A_e(a_e)$, $\pi_t[i] \models \Psi$ iff $\pi[j] \models \Psi^\circ$ by the inductive hypothesis.

5. If Ψ is of the form $\Psi_1 \,\mathsf{U}\, \Psi_2$, then $\Psi^\circ = (A_e(a_e) \vee \Psi_1^\circ \,\mathsf{U}\, (\neg A_e(a_e) \wedge \Psi_2^\circ)$. Since π_t is obtained from π by removing all states q s.t. $q \models A_e(a_e)$, $\pi_t[i] \models \Psi$ iff $\pi[j] \models \Psi^\circ$ by the inductive hypothesis.

6. If Ψ is of the form $\mathsf{E}\Psi_1$, note that every path π_t in $\mathfrak{T}(\mathcal{P})$ with $\pi_t[0] = q_1^\circ$ corresponds to a path π in $\mathfrak{T}(\mathcal{P}')$ with $\mathfrak{T}(\mathcal{P}'), \pi \models \mathsf{G}(A_e(a_e) \vee \psi_1)$. Correspondingly, there exists a path π_t' with $\pi_t'[i] \models \Psi$ iff $\pi'[j] \models \Psi^\circ$ by the inductive hypothesis.

We establish that $\mathfrak{T}(\mathcal{P}'), q_0 \models \Phi'$ iff $\mathfrak{T}(\mathcal{P}), q_1^\circ \models \Phi$, and hence, since $\mathfrak{T}(\mathcal{P}'), q_0 \models \Phi'$, $\mathfrak{T}(\mathcal{P}), q_1^\circ \models \Phi$. Consequently Φ is satisfiable in $\mathcal{P}$. □

Since we can reduce satisfiability in programs over $\mathcal{ALCO}$ to satisfiability over programs in $\mathcal{ELO}_\bot$, the lower bounsd for $\mathcal{ALCO}$ directly transfer to $\mathcal{ELO}_\bot$.

Theorem 26. *Verifying $\mathcal{ELO}_\bot$-CTL formulae for $\mathcal{ELO}_\bot$-ConGolog programs is* 2ExpTime-*hard.*

We remark that our lower bound already applies to the less expressive DL $\mathcal{EL}_\bot$, which prohibits the nominal operator $\{a\}$, because this operator can be straight-forwardly simulated in our framework: for this, one simply replaces each occurrence of $\{a\}$ by a fresh concept name A_a, and uses an action to add a as single named individual to the extension of A_a. This is also the reason why all logics considered in this paper support nominals. We note that the bottom operator $\bot$ is usually harmless with respect to complexity in $\mathcal{EL}$ [2], however needed for our reduction. It is open whether we can obtain the same complexity results also for $\mathcal{EL}$, in which the bottom operator cannot be expressed.

6 Upper Bounds

We show that the lower bounds established in the last sections are indeed tight even for timed ConGolog programs and properties expressed in TCTL*, by presenting a decision procedure for the satisfiability problem.

As in the last sections, it is more convenient to focus on the problem of satisfiability rather than approaching verification directly. Given a program $\mathcal{P}$ and a model $\mathcal{I}$ of the initial KB $\mathcal{K}$, let

$$\mathfrak{T}(\mathcal{P}, \mathcal{I}) = (Q_{\mathcal{P},\mathcal{I}}, I_{\mathcal{P},\mathcal{I}} = \{\langle \mathcal{I}, \langle\rangle, \delta\rangle\}, \hookrightarrow_{\mathcal{P},\mathcal{I}}, \lambda_{\mathcal{P},\mathcal{I}})$$

be the first-order TTS induced by $\mathcal{P}$ restricted to states reachable from the initial state $\langle \mathcal{I}, \langle\rangle, \delta\rangle$. To show that a TCTL*-formula Φ is satisfiable in the timed program $\mathcal{P}$, one has to find some model $\mathcal{I}$ of the initial KB such that $\mathfrak{T}(\mathcal{P}, \mathcal{I}) \models \Phi$. A key idea of our method is to provide for a sufficient abstraction of $\mathcal{I}$ and all states in $\mathfrak{T}(\mathcal{P}, \mathcal{I})$. We call this abstraction *dynamic type*. Based on the dynamic type, we can construct a propositional abstraction of the TTS of exponential size, on which we can then evaluate a propositional abstraction of the TCTL*-formula Φ.

Let us define these dynamic types first. The intuition is that, for a fixed initial interpretation $\mathcal{I}$, all interpretations that occur in $\mathfrak{T}(\mathcal{P}, \mathcal{I})$ can be identified by the effects applied on them. The dynamic type connects these effects with the relevant axioms that should be entailed or not entailed in the interpretation obtained by applying these effects.

Let $\mathcal{P}$ be a program over the action theory $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ and Φ be a TCTL*-formula. The *context* $\mathcal{C}_\mathcal{P}$ *of* $\mathcal{P}$ contains all axioms α occurring in $\mathcal{P}$, $\mathcal{K}$, and as precondition in Eff, as well as their negation $\neg\alpha$. Note that we can assume wlog. that φ contains only axioms that also occur in $\mathcal{P}$ (if not, we simply add them in the form of tautogical tests). The *set of relevant effects* is given by $\mathsf{Lit}(\Sigma) := \{\mathsf{l} \mid \Phi \rhd \mathsf{l} \in \mathsf{Eff}(\alpha), \alpha \in \mathsf{Act}\}$. A *dynamic type for* $\mathcal{P}$ is now a set $\mathfrak{t} \subseteq \mathcal{C}_\mathcal{P} \times 2^{\mathsf{Lit}(\Sigma)}$. A dynamic type $\mathfrak{t}$ is *realisable* if there exists a model $\mathcal{I}$ of the initial KB $\mathcal{K}$ s.t. for every $(\psi, \mathsf{L}) \in \mathfrak{t}$, $\mathcal{I}^\mathsf{L} \models \psi$. We then say that $\mathcal{I}$ *realises* $\mathfrak{t}$. Note that for every model $\mathcal{I}$ of the initial KB $\mathcal{K}$, there exists a dynamic type $\mathfrak{t}$ that is realised by $\mathcal{I}$, which can be obtained from $\mathcal{I}$ by simply collecting all elements $(\psi, \mathsf{L}) \in \mathcal{C}_\mathcal{P} \times 2^{\mathsf{Lit}(\Sigma)}$ s.t. $\mathcal{I}^\mathsf{L} \models \psi$.

Realisability of dynamic types $\mathfrak{t}$ can be decided by constructing a *reduction KB* $\mathcal{K}_\mathfrak{t}$ as described in [5], which is satisfiable iff $\mathfrak{t}$ is realisable. For each name $X \in \mathsf{N_C} \cup \mathsf{N_R}$ and effect set $\mathsf{L} \subseteq \mathsf{Lit}(\Sigma)$, $\mathcal{K}_\mathfrak{t}$ uses a fresh name $X^{(\mathsf{L})}$ to represent the interpretation of X in $\mathcal{I}^\mathsf{L}$. Though [5] describe this approach only for $\mathcal{ALCO}$, careful inspection of the proofs show that the same reduction can be used to decide realisability of dynamic types for the more expressive DLs considered here. The reduction KB is exponential in the size of the program. Since satisfiability of KBs is in ExpTime for $\mathcal{L} \in \{\mathcal{ALCIO}, \mathcal{ALCQO}\}$, and in N2ExpTime for $\mathcal{L} = \mathcal{ALCQIO}$, we obtain the following lemma.

Lemma 27. *Let $\mathcal{L}$ be a DL and $\mathcal{P}$ an $\mathcal{L}$-ConGolog program. Then, realisability of dynamic types for $\mathcal{P}$ can be decided in* 2ExpTime *for $\mathcal{L} \in \{\mathcal{ALCIO}, \mathcal{ALCQO}\}$ and in* N2ExpTime *for $\mathcal{L} = \mathcal{ALCQIO}$.*

Based on a dynamic type $\mathfrak{t}$ realised by $\mathcal{I}$, we can construct a propositional TTS $\mathfrak{P}(\mathfrak{t})$ which is bisimular to the first-order TTS $\mathfrak{T}(\mathcal{P}, \mathcal{I})$. We use a mapping $\iota_\mathcal{C} : \mathcal{C}_{\mathcal{K},\Phi} \to \mathsf{AP}$ that maps each context element to a propositional symbol. Every state in $\mathfrak{T}(\mathcal{P}, \mathcal{I})$ is of the form $(\mathcal{I}^\mathsf{L}, \sigma, \delta')$ for some $\mathsf{L} \subseteq \mathsf{Lit}(\Sigma)$, and has a corresponding state in $\mathfrak{P}(\mathfrak{t})$ which is of the form (L, δ'). The labelling function λ of $\mathfrak{P}(\mathfrak{t})$ maps each state (L, δ') to the set $\{\iota_\mathcal{C}(\alpha) \mid (C \sqsubseteq D, \mathsf{L}) \in \mathfrak{t}\} \cup \{\neg\iota_\mathcal{C}(\alpha) \mid (\neg\alpha, \mathsf{L}) \in \mathfrak{t}\}$, to allow for propositional verification based on the axioms entailed in the corresponding interpretation $\mathcal{I}^\mathsf{L}$. From Φ, we obtain $\iota_\mathcal{C}(\Phi)$ by replacing every axiom α by $\iota_\mathcal{C}(\alpha)$.

Before we prove formally that this reduction works, we introduce some more terminology. To get a better handle on which actions are executed next from a program sequence, we introduce the notions of a guarded action and of the head of a program sequence.

Definition 28. A program expression over some $\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur})$ is called *guarded action* if it is of the form

$$\psi_1?; (\psi_2?; (\ldots; (\psi_n?; \alpha))),$$

where $\alpha \in \mathsf{Act}$, $n \geq 0$ and each $\psi_i?$ for $i = 1, \cdots, n$ is a test. We will often use the symbol $\mathfrak{a}$ to denote a guarded action. If $n = 0$, then the guarded action is actually an ordinary ground action, and thus $\mathfrak{a}$ may also denote a ground action. The preceding sequence of tests is called *guard.* When writing a guarded action we will often omit the parentheses.

Let $\mathcal{I}$ be an interpretation. The guarded action $\psi_1?; \cdots; \psi_n?; \alpha$ is *executable in* $\mathcal{I}$ iff

$$\mathcal{I} \models \psi_i \text{ for all } i = 1, \ldots, n.$$

▲

Next, we introduce the function $\mathsf{head}(\cdot)$ to denote the *head of a program.* Intuitively, $\mathsf{head}(\delta)$ contains those guarded actions that can be executed first when executing the program expression δ. The function $\mathsf{head}(\cdot)$ is formally defined as follows.

$$\begin{aligned}
&\mathsf{head}(\langle\rangle) := \{\boldsymbol{\epsilon}\};\\
&\mathsf{head}(\alpha) := \{\alpha\} \text{ for all } \alpha \in \mathsf{Act};\\
&\mathsf{head}(\psi?) := \{\psi?;\boldsymbol{\epsilon}\};\\
&\mathsf{head}(\delta^*) := \{\boldsymbol{\epsilon}\} \cup \mathsf{head}(\delta);\\
&\mathsf{head}(\delta_1;\delta_2) := \ \{\mathfrak{a} \mid \mathfrak{a} = \psi_1?;...;\psi_n?;\alpha \in \mathsf{head}(\delta_1) \wedge \alpha \neq \boldsymbol{\epsilon}\} \cup\\
&\qquad\qquad\qquad \{\psi_1?;\ldots;\psi_n?;\mathfrak{a} \mid \ \psi_1?;...;\psi_n?;\boldsymbol{\epsilon} \in \mathsf{head}(\delta_1) \ \wedge \mathfrak{a} \in \mathsf{head}(\delta_2)\};\\
&\mathsf{head}(\delta_1|\delta_2) := \mathsf{head}(\delta_1) \cup \mathsf{head}(\delta_2);\\
&\mathsf{head}(\delta_1\|\delta_2) := \ \{\mathfrak{a} \mid \mathfrak{a} = \psi_1?;\ldots;\psi_n?;\alpha \in \mathsf{head}(\delta_i) \wedge i \in \{1,2\} \wedge \alpha \neq \boldsymbol{\epsilon}\} \cup\\
&\qquad\qquad\qquad \{\psi_1?;\ldots;\psi_n?;\mathfrak{a} \mid \ \psi_1?;\ldots;\psi_n?;\boldsymbol{\epsilon} \in \mathsf{head}(\delta_i) \ \wedge\\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathfrak{a} \in \mathsf{head}(\delta_j) \ \wedge i,j \in \{1,2\}, i \neq j\};
\end{aligned}$$

Figure 3: Head of a program expression

Definition 29. The function $\mathsf{head}(\cdot)$ maps a program expression to a set of guarded actions. It is defined by induction on the structure of program expressions as given in Figure 3. ▲

The empty program represents the final state which means that $\boldsymbol{\epsilon}$is executed next. Since tests do not cause a separate execution step, the head of a test is given by the termination action $\boldsymbol{\epsilon}$ preceded by the test itself as a guard. Executing δ^* means executing δ zero ore more times. Hence, the head of δ^* consists of the termination action $\boldsymbol{\epsilon}$ and the heads of δ. Consider the definition of $\mathsf{head}(\delta_1;\delta_2)$. In this case, we first have to execute the program δ_1. Therefore, the first guarded action to be executed for the sequence is one of the heads of δ_1. However, if $\psi_1?;\ldots;\psi_n?;\boldsymbol{\epsilon}$ is contained in the head of δ_1, then δ_1 can terminate successfully if the tests are satisfied. But in this case the subsequent program δ_2 still needs to be executed. Therefore, we must continue with a head of δ_2. This is achieved by replacing $\boldsymbol{\epsilon}$ in $\psi_1?;\cdots;\psi_n?;\boldsymbol{\epsilon}$ with a head of δ_2. Our definition of $\mathsf{head}(\delta_1\|\delta_2)$ can be explained in a similar way. To do $\delta_1|\delta_2$ a head of δ_1 or one of δ_2 has to be done in the next step.

Next, we introduce static types, which in contrast to dynamic types only capture the entailments of a single interpretation, and the effects of applying a guarded action on them.

Definition 30 (Static types)**.** A *static type* w.r.t. $\mathcal{C}$ is a maximal consistent subset of $\mathcal{C}$. The *set of all static types w.r.t.* $\mathcal{C}$ is denoted by $\mathfrak{S}_{\mathcal{C}}$. Let $\mathcal{I}$ be an interpretation. The *static type of* $\mathcal{I}$ *w.r.t.* $\mathcal{C}$, denoted by $\mathsf{s\text{-}type}_{\mathcal{C}}(\mathcal{I})$, is given by

$$\mathsf{s\text{-}type}_{\mathcal{C}}(\mathcal{I}) := \{\psi \in \mathcal{C} \mid \mathcal{I} \models \psi\}.$$

Let $\mathcal{P} = (\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur}), \delta)$ be a program, $\mathfrak{s} \in \mathfrak{S}_{\mathcal{C}_{\mathcal{P}}}$ a static type, $\alpha \in \mathsf{Act}$ and $\mathfrak{a} = \psi_1?;\cdots;\psi_n?;\alpha$ a guarded action with $\mathfrak{a} \in \mathsf{head}(\delta')$ for some $\delta' \in \mathsf{sub}(\delta)$.

We define

$$\mathsf{Eff}(\mathfrak{s}, \alpha) := \{\mathsf{l} \mid \varphi \rhd \mathsf{l} \in \mathsf{Eff}(\alpha) \text{ and } \varphi \in \mathfrak{s}\}.$$

Furthermore, we say that $\mathfrak{a}$ *is executable in* $\mathfrak{s}$ iff $\psi_i \in \mathfrak{s}$ for all $i = 1,\ldots,n$. ▲

We have now all tools to formally define the propositional TTS $\mathfrak{P}(\mathfrak{t})$, which, based on $\mathfrak{t}$ simulates executions of $\mathcal{P}$. To be able to work fully on the propositional level, we map elements in the context to propositional symbols. Therefore, for a given context $\mathcal{C}$, we fix a function $\iota_{\mathcal{C}} : \mathcal{C} \to \mathsf{AP}$ mapping each element of $\mathcal{C}$ to some propositional symbol AP.

Definition 31 (Propositional TTS)**.** Let $\mathcal{P} = (\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur}), \delta)$ be a program, and $\mathfrak{t}$ a dynamic type for $\mathcal{P}$. The *corresponding TTS over* AP, denoted by $\mathfrak{P}(\mathfrak{t})$, is then defined as follows.

$$\mathfrak{P}(\mathfrak{t}) = (S, J, \Longrightarrow, \lambda) \text{ with}$$

- $S := \mathsf{sub}(\delta) \times 2^{\mathsf{Lit}(\Sigma)}$;
- $J := \{(\delta, \emptyset)\}$;
- it holds that $(\delta', \mathsf{L}') \overset{\rho}{\Longrightarrow} (\delta'', \mathsf{L}'')$ for some states $(\delta', \mathsf{L}'), (\delta'', \mathsf{L}'') \in S$ and interval ρ iff there exists $\mathfrak{a} = \psi_1?; \cdots; \psi_n?; \alpha \in \mathsf{head}(\delta')$ such that the following conditions are satisfied
  - $\mathfrak{a}$ is executable in the static type $\{\psi \mid (\psi, \mathsf{L}') \in \mathfrak{t}\}$;
  - $\delta'' \in \mathsf{tail}(\mathfrak{a}, \delta')$;
  - $\mathsf{L}'' = \mathsf{L}' \bowtie \mathsf{Eff}(\{\psi \mid (\psi, \mathsf{L}') \in \mathfrak{t}\}, \alpha)$ and
  - $\rho = \mathsf{Dur}(\alpha)$;
- $\lambda : (\delta', \mathsf{L}') \mapsto \{\iota_{\mathcal{C}_\mathcal{P}}(\psi) \mid (\psi, \mathsf{L}') \in \mathfrak{t}\}$ for all $(\delta', \mathsf{L}') \in S$.

▲

To be able to describe whether $\mathfrak{P}(\mathfrak{t})$ and $\mathfrak{T}(\mathcal{P}, \mathcal{I})$ behave "similar", we next define the notion of bisimulation between a first-order TTS and a propositional TTS formally. Our notion of bisimulation depends on a given context $\mathcal{C}$, which specifies which entailments in a first-order interpretations are relevant for us. These entailments have to be mapped to corresponding propositional labels in the propositional TTS, based on the mapping $\iota_\mathcal{C}$.

Definition 32 (Bisimulation)**.** Let $\mathcal{C}$ be a context such that $\iota_\mathcal{C}$ has the range AP, $\mathfrak{T} = (Q_\mathfrak{T}, I_\mathfrak{T}, \hookrightarrow_\mathfrak{T}, \lambda_\mathfrak{T})$ be a first-order TTS and $\mathfrak{P} = (Q_\mathfrak{P}, I_\mathfrak{P}, \hookrightarrow_\mathfrak{P}, \lambda_\mathfrak{P})$ a propositional TTS over AP. A binary relation $\simeq_\mathcal{C} \subseteq Q_\mathfrak{T} \times Q_\mathfrak{P}$ is called $\mathcal{C}$*-bisimulation* iff the following conditions are satisfied:

- $q_\mathfrak{T} \simeq_\mathcal{C} q_\mathfrak{P}$ implies $\lambda_\mathfrak{P}(q_\mathfrak{P}) = \{\iota_\mathcal{C}(\psi) \mid \psi \in \mathsf{s\text{-}type}_\mathcal{C}(\mathcal{I})\}$, where $\lambda_\mathfrak{T}(q_\mathfrak{T}) = \mathcal{I}$.
- If $q_\mathfrak{T} \simeq_\mathcal{C} q_\mathfrak{P}$ and there is a transition $q_\mathfrak{T} \overset{d}{\hookrightarrow}_\mathfrak{T} q'_\mathfrak{T}$, then there exists a transition $q_\mathfrak{P} \overset{d}{\hookrightarrow}_\mathfrak{P} q'_\mathfrak{P}$ such that $q'_\mathfrak{T} \simeq_\mathcal{C} q'_\mathfrak{P}$.
- If $q_\mathfrak{T} \simeq_\mathcal{C} q_\mathfrak{P}$ and there is a transition $q_\mathfrak{P} \overset{d}{\hookrightarrow}_\mathfrak{P} q'_\mathfrak{P}$, then there exists a transition $q_\mathfrak{T} \overset{d}{\hookrightarrow}_\mathfrak{T} q'_\mathfrak{T}$ such that $q'_\mathfrak{T} \simeq_\mathcal{C} q'_\mathfrak{P}$.

The relation $\simeq_\mathcal{C}$ is extended to paths as follows. Let π be a path in $\mathfrak{T}$ and $\mathfrak{p}$ a path in $\mathfrak{P}$. We write $\pi \simeq_\mathcal{C} \mathfrak{p}$ iff $\pi[i] \simeq_\mathcal{C} \mathfrak{p}[i]$ for all $i \geq 0$.

We say that $\mathfrak{T}$ and $\mathfrak{P}$ are $\mathcal{C}$*-bisimilar* iff there exists a $\mathcal{C}$-bisimulation $\simeq_\mathcal{C} \subseteq Q_\mathfrak{I} \times Q_\mathfrak{P}$ such that

- for all $q_\mathfrak{T} \in I_\mathfrak{T}$ there exists $q_\mathfrak{P} \in I_\mathfrak{P}$ such that $q_\mathfrak{T} \simeq_\mathcal{C} q_\mathfrak{P}$ and
- for all $q_\mathfrak{P} \in I_\mathfrak{P}$ there exists $q_\mathfrak{T} \in I_\mathfrak{T}$ such that $q_\mathfrak{T} \simeq_\mathcal{C} q_\mathfrak{P}$.

▲

Intuitively, if a propositional TTS $\mathfrak{P}$ a first-order TTS $\mathfrak{T}$ are bisimular, then every path in $\mathfrak{P}$ has a corresponding path in $\mathfrak{T}$ and vice versa. Using this definition, it is standard to verify the following lemma by structural induction on TCTL*-formulae.

Lemma 33. *Let $\mathcal{C}$ be a context, AP the range of $\iota_\mathcal{C}$, $\mathfrak{T}$ be a first-order TTS, $\mathfrak{P}$ a propositional TTS over AP such that there exists a $\mathcal{C}$-bisimulation $\simeq_\mathcal{C} \subseteq Q_\mathfrak{T} \times Q_\mathfrak{P}$.*

1. *For every TCTL* state formula Φ that mentions only axioms from $\mathcal{C}$, it holds that if $q_\mathfrak{T} \simeq_\mathcal{C} q_\mathfrak{P}$ for two states, then $\mathfrak{T}, q_\mathfrak{T} \models \Phi$ iff $\mathfrak{P}, q_\mathfrak{T} \models \iota_\mathcal{C}(\Phi)$.*
2. *For every TCTL* path formula Ψ that mentions only axioms from $\mathcal{C}$, it holds that if $\pi_\mathfrak{T} \simeq_\mathcal{C} \pi_\mathfrak{P}$ for two paths, then $\mathfrak{T}, \pi_\mathfrak{T} \models \Psi$ iff $\mathfrak{P}, \pi_\mathfrak{T} \models \iota_\mathcal{C}(\Psi)$.*

We have now all ingredients to show the key lemma of this section.

Lemma 34. *Φ is satisfiable by $\mathcal{P}$ iff there exists a realisable dynamic type $\mathfrak{t}$ for $\mathcal{P}$s.t. $\mathfrak{P}(\mathfrak{t}) \models \iota_{\mathcal{C}_\mathcal{P}}(\Phi)$.*

Proof. We have to show that there exists a realisable dynamic type $\mathfrak{t}$ for $\mathcal{P}$ and Φ s.t. $\mathfrak{P}(\mathfrak{t}) \models \iota_{\mathcal{C}_\mathcal{P}}(\Phi)$ iff there exists an initial state $(\mathcal{I}, \langle\rangle, \delta)$ s.t. $\mathfrak{T}(\mathcal{P}), (\mathcal{I}, \langle\rangle, \delta) \models \Phi$. Specifically, we show this for the case where $\mathcal{I}$ is the interpretation that realises $\mathfrak{t}$.

Let $\mathcal{P} = (\Sigma = (\mathcal{K}, \mathsf{Act}, \mathsf{Eff}, \mathsf{Dur}), \delta)$ be a program $\mathcal{I}$ a model of $\mathcal{K}$ and $\mathfrak{t}$ the dynamic type of $\mathcal{P}$ that is realised by $\mathcal{I}$. Further, let $\mathfrak{P}(\mathfrak{t}) = (S, J, \Longrightarrow, \lambda)$ the propositional TTS corresponding to $\mathfrak{t}$.

We define a *binary relation* $\simeq_\mathfrak{t} \subseteq Q_{\mathcal{P},\mathcal{I}} \times Q_{\mathfrak{P}(\mathfrak{t})}$ by requiring

$$\langle \mathcal{J}, \sigma, \delta' \rangle \simeq_\mathfrak{t} (\delta'', \mathsf{L}) \text{ iff } \mathsf{L} = \mathsf{Eff}(\mathcal{I}, \sigma) \text{ and } \delta'' = \delta'.$$

It is standard to define by structural induction on the program sequence δ that $\simeq_\mathfrak{t}$ is a bisimulation between $\mathfrak{T}(\mathcal{P})$ and $\mathfrak{P}(\mathfrak{t})$, and that indeed $\langle \mathcal{J}, \sigma, \delta' \rangle \simeq_\mathfrak{t} (\delta'', \mathsf{L}'')$ iff $\mathcal{J} = \mathcal{I}^\mathsf{L}$. As consequence, by Lemma 33, we obtain that $\mathfrak{T}(\mathcal{P}, \mathcal{I}) \models \Phi$ iff $\mathfrak{P}(\mathfrak{t}) \models \iota_{\mathcal{C}_\mathcal{P}}(\Phi)$.

Since for every model $\mathcal{I}$ of $\mathcal{K}$ there exists a corresponding dynamic type $\mathfrak{t}$ realised by $\mathcal{I}$, and for every realisable dynamic type $\mathfrak{t}$ there exists a model $\mathcal{I}$ of $\mathcal{K}$ that realises $\mathfrak{t}$, we obtain that Φ is satisfiable in $\mathcal{P}$ iff there exists a dynamic type $\mathfrak{t}$ for $\mathcal{P}$ s.t. $\mathfrak{P}(\mathfrak{t}) \models \iota_{\mathcal{C}_\mathcal{P}}(\Phi)$. $\square$

Using the fact that $\mathfrak{P}(\mathfrak{t})$ is exponential in the size of $\mathcal{P}$, together with Lemma 27 and the lower bounds established in the previous section, we can now establish the following theorem.

Theorem 35. *The verification problem for $\mathcal{L}$-TCTL*-formulas and timed $\mathcal{L}$-ConGolog programs is*

- 2ExpTime*-complete for $\mathcal{L} \in \{\mathcal{ELO}_\bot, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and*
- coN2ExpTime*-complete for $\mathcal{L} = \mathcal{ALCQIO}$.*

Proof. The lower bounds have been established in the last section. For the upper bounds, we notice that $\mathfrak{P}(\mathfrak{t})$ is exponential in size, and that verification of propositional TCTL*-formulae from propositional TTS is in ExpTime wrt. to the size of the TTS and in ExpSpace wrt. the size of the formula [11]. Since $\mathfrak{P}(\mathfrak{t})$ is exponential in size wrt. $\mathcal{P}$, and $\iota_\mathcal{C}(\Phi)$ is linear in size wrt. Φ, we can decide $\mathfrak{P}(\mathfrak{t}) \models \iota_\mathcal{C}(\Phi)$ in 2ExpTime wrt. the size of $\mathcal{P}$ and Φ. Thus, and by Lemma 27, deciding whether a given dynamic type $\mathfrak{t}$ is realisable and witnesses the satisfiablity of Φ can be performed in 2ExpTime for $\mathcal{L} \in \{\mathcal{ELO}_\bot, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, and in N2ExpTime for $\mathcal{L} = \mathcal{ALCQIO}$. We can thus decide satisfiability of Φ in $\mathcal{P}$ as follows. For $\mathcal{L} = \mathcal{ALCQIO}$, we guess a dynamic type $\mathfrak{t}$ and verify whether it witnesses the satisfiability of Φ. For $\mathcal{L} \in \{\mathcal{ELO}_\bot, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}\}$, we iterate over the double exponentially many dynamic types $\mathfrak{t}$ until we found one s.t. $\mathfrak{P}(\mathfrak{t}) \models \iota_{\mathcal{C}}(\Phi)$. As satisfiability is complementary to verification, we obtain the complexity results stated in the theorem. $\square$

7 Conclusion

The Golog family of programming languages is a powerful framework to model complex behvaiour of agents based on the Situation Calculus. To allow for a more realistic modelling of these actions, in timed ConGolog, actions are assigned durations of time they consume. As with durations, time intervals become more prominent, temporal specifications for programs of this form can be more accurately specified using metric temporal logics, such as the branching time logics TCTL and TCTL*.

We analysed the computational complexity of verifying TCTL formulae for timed ConGolog programs over DL actions, covering the spectrum of description logics starting from the less expressive $\mathcal{ELO}_\bot$ to expressive DLs such as $\mathcal{ALCQIO}$, with complexities ranging from 2ExpTime to coN2ExpTime. While the focus of the paper is on timed ConGolog programs and temporal properties in TCTL* and TCTL, our results also complete the picture for verifying qualitative temporal properties over non-timed Golog programs, providing for the first time tight bounds for these cases. An open question is whether complexities transfer to real life situations, or whether practical systems for the verification of discrete-timed Golog programs can be implemented. Such an implementation could for example be based on the system presented in [8], which implements verification of CTL properties over first-order formulas for (untimed) first-order ConGolog programs based on Situation Calculus theories. Since this variant of the verification problem is undecidable, the implementation cannot guarantee to terminate for any given input. It would be interesting to investigate whether one can use the special properties of DLs to obtain a system with better practical properties.

Complexity-wise, the picture of Golog programs over DL programs looks relatively complete. However, there are various extensions and restrictions that might be interesting to investigate in the future. Our results show that restricting the underlying DL to a tractable fragment has no impact on the computational complexity compared to expressive DLs such as $\mathcal{ALCIO}$ and $\mathcal{ALCQO}$. The only way to allow for verification below 2ExpTime would therefore be to go to an even less expressive DL such as DL-Lite, for which verification would almost resemble the propositional case, or to restrict the operators allowed in the Golog programs and TCTL*. Since our reductions already apply to a relatively restricted fragment of both, it is not clear whether such an investigation would result in a setting that is still useful for practical applications.

On the other hand, there are various ways in which our framework could be extended. For example, one might consider operators in temporal properties to point into the past, and allowing negative "durations" of actions. While this seems unnatural at first sight, there might be applications where this is indeed useful: for example, if we are modelling energy levels of a battery instead of the timeline, there could be actions that consume battery life, and other actions (like recharging), which increase the battery life. This setup has been considered in the context of weighted automata [15], where it was found that verifying LTL-formulae on weighted automata with negative weights is undecidable. We conjecture that this undecidability result can be transferred also to the timed $\mathcal{L}$-ConGolog-programs if we allow for negative durations of actions.

References

[1] Shqiponja Ahmetaj, Diego Calvanese, Magdalena Ortiz, and Mantas Simkus. Managing change in graph-structured data using description logics. *ACM Trans. Comput. Log.*, 18(4):27:1–27:35, 2017.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 364–369. Professional Book Center, 2005.

[3] Franz Baader, Hongkai Liu, and Anees ul Mehdi. Verifying properties of infinite sequences of description logic actions. In *Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI 2010)*, volume 215 of *Frontiers in Artifical Intelligence and Applications*, pages 53–58. IOS Press, 2010.

[4] Franz Baader, Carsten Lutz, Maja Milicic, Ulrike Sattler, and Frank Wolter. Integrating description logics and action formalisms: First results. In *Proceedings of AAAI*, pages 572–577. AAAI Press / The MIT Press, 2005.

[5] Franz Baader and Benjamin Zarrieß. Verification of golog programs over description logic actions. In *Frontiers of Combining Systems - 9th International Symposium, FroCoS 2013*, volume 8152 of *Lecture Notes in Computer Science*, pages 181–196. Springer, 2013.

[6] Jens Claßen and Gerhard Lakemeyer. A logic for non-terminating Golog programs. In *Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 589–599. AAAI Press, 2008.

[7] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[8] Jens Claßen. Symbolic verification of Golog programs with first-order BDDs. In *Proceedings of KR*, 2018. To appear.

[9] Yilan Gu and Mikhail Soutchanski. A description logic based situation calculus. *Annals of Mathematics and Artificial Intelligence*, 58(1–2):3–83, 2010.

[10] Markus Krötzsch and Sebastian Rudolph. Nominal schemas in description logics: Complexities clarified. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014*. AAAI Press, 2014.

[11] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Efficient timed model checking for discrete-time systems. *Theor. Comput. Sci.*, 353(1-3):249–271, 2006.

[12] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–83, 1997.

[13] Naiqi Li and Yongmei Liu. Automatic verification of partial correctness of Golog programs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 3113–3119. AAAI Press, 2015.

[14] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, pages 463–502. American Elsevier, New York, 1969.

[15] Karin Quaas. Model checking metric temporal logic over automata with one counter. In *Language and Automata Theory and Applications - 7th International Conference, LATA 2013*, volume 7810 of *Lecture Notes in Computer Science*, pages 468–479. Springer, 2013.

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

[17] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence.*, pages 466–471. Morgan Kaufmann, 1991.

[18] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Intell. Res.*, 12:199–217, 2000.

[19] Benjamin Zarrieß and Jens Claßen. Verifying CTL* properties of GOLOG programs over local-effect actions. In *ECAI 2014 - 21st European Conference on Artificial Intelligence*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 939–944. IOS Press, 2014.