ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-939

Verifying CTL* Properties of GOLOG Programs over Local-Effect Actions

Benjamin Zarrieß[1] and **Jens Claßen**[2]

Abstract. GOLOG is a high-level action programming language for controlling autonomous agents such as mobile robots. It is defined on top of a logic-based action theory expressed in the Situation Calculus. Before a program is deployed onto an actual robot and executed in the physical world, it is desirable, if not crucial, to verify that it meets certain requirements (typically expressed through temporal formulas) and thus indeed exhibits the desired behaviour. However, due to the high (first-order) expressiveness of the language, the corresponding verification problem is in general undecidable. In this paper, we extend earlier results to identify a large, non-trivial fragment of the formalism where verification is decidable. In particular, we consider properties expressed in a first-order variant of the branching-time temporal logic CTL*. Decidability is obtained by (1) resorting to the decidable first-order fragment C^2 as underlying base logic, (2) using a fragment of GOLOG with ground actions only, and (3) requiring the action theory to only admit local effects.

1 Introduction

GOLOG [11, 16], a family of high-level action programming languages, has proven to be a useful means for the control of autonomous agents such as mobile robots [5, 10]. It is defined on top of action theories expressed in the Situation Calculus [19, 21], a dialect of first-order logic (with some second-order features) for representing and reasoning in dynamic application domains. Before a GOLOG program is deployed onto a robot and executed in the physical world, it is often desirable, if not crucial, to verify that certain criteria are met, typical examples being safety, liveness and fairness conditions.

Verification of GOLOG programs was first considered by De Giacomo, Ternovska and Reiter [12] who presented a corresponding semantics of non-terminating processes defined by means of (second-order) Situation Calculus axioms, expressed temporal properties through (second-order) fixpoint formulas, and provided manual, meta-theoretic proofs to show the satisfaction of such properties. Since it is more preferable to do verification in an automated fashion, Claßen and Lakemeyer [6] later proposed an approach based on a new logic called $\mathcal{ESG}$, an extension of the modal Situation Calculus variant $\mathcal{ES}$ [15] by modalities for expressing temporal properties of GOLOG programs. Using regression-based reasoning and a newly introduced graph representation for programs, they provided algorithms for the verification of a fragment of the formalism resembling a first-order, but non-nested variant of the branching-time temporal logic CTL. Their procedures, which perform a fixpoint computation to do a systematic exploration of the state space, could be proven to be sound, but no general termination guarantee could be given due to the verification problem being highly undecidable.

It would of course be desirable if termination were guaranteed, which could be achieved by imposing appropriate restrictions on the input formalism such that the verification problem becomes decidable, while preferably retaining as much first-order expressiveness as possible. One corresponding approach is presented by Baader, Liu and ul Mehdi [2] who, instead of using the fully-fledged Situation Calculus and GOLOG, resort to an action language [3] based on the decidable Description Logic (DL) $\mathcal{ALC}$ [1] and approximate programs through finite Büchi automata. They could show that verification of LTL properties over $\mathcal{ALC}$ axioms thus reduces to a decidable reasoning task within the underlying DL. Baader and Zarrieß [4] later lifted these results to support a more expressive fragment of GOLOG program expressions that goes beyond what can be represented by Büchi automata, in particular regarding test conditions $\phi?$ that are needed for expressing imperative programming constructs such as while loops and conditionals, but restricts all actions to be ground.

Another approach is taken in [7] where it is shown that Claßen and Lakemeyer's original verification algorithms can be guaranteed to terminate by restricting oneself to a decidable, two-variable fragment of FOL as base logic for the Situation Calculus [13], allowing only ground action terms in GOLOG programs, and requiring actions to only have local effects [17]. Compared to the above mentioned results by Baader and Zarrieß, local-effect action theories represent an increase in expressiveness as $\mathcal{ALC}$-based action definitions only allow for basic STRIPS-style addition and deletion of literals, but the class of non-nested CTL-like properties supported by Claßen and Lakemeyer's method is less expressive than LTL.

In this paper, we turn towards unifying these earlier approaches within a single formal framework, while even increasing expressiveness. In particular, we (1) use C^2 as base logic, the two variable fragment of first-order logic with counting quantifiers, which subsumes both $\mathcal{ALC}$ and the two-variable fragment of FOL; we (2) formulate action effects through $\mathcal{ES}$-style local-effect action theories; and (3) we show that verification is decidable for GOLOG programs with only ground actions even in the case of properties expressed in the branching-time temporal logics CTL*, which is strictly more expressive than both CTL and LTL. We obtain decidability by constructing a finite abstraction of the infinite transition system induced by a program, and showing that the abstraction preserves satisfiability of CTL* properties over C^2 axioms. We also obtain a 2-NEXPTIME upper bound for the computational complexity of the problem.

The remainder of this paper is organized as follows. Section 2 presents our action formalism, namely the modal Situation Calculus variant $\mathcal{ES}$ using C^2 (the two-variable fragment of first-order logic

[1] Theoretical Computer Science, TU Dresden, Germany, email: zarriess@tcs.inf.tu-dresden.de
[2] Knowledge-Based Systems Group, RWTH Aachen University, Germany, email: classen@kbsg.rwth-aachen.de

with counting quantifiers) as decidable base logic, as well as GOLOG programs and their semantics. In Section 3 we then define the problem of verifying CTL* properties over such programs, and show (in a constructive manner) that the problem is indeed decidable. We then discuss related work and conclude. Because of space constraints, detailed proofs of our results have to be omitted. They can be found in [22].

2 Preliminaries

2.1 The Modal Situation Calculus $\mathcal{ES}$ based on C^2

In this subsection we recall the main definitions of the modal Situation Calculus variant $\mathcal{ES}$ [15]. Instead of using full first-order logic, we restrict ourselves to the *two variable fragment with equality and counting* of FOL named C^2.

Syntax: We start by fixing a set of *terms*. In our language we consider terms of two sorts, namely *object* and *action*. Terms can be built using the following symbols: an infinite supply of variables $x, y, \ldots$ of sort *object*, a single variable a of sort *action*, an infinite set N_I of *rigid object constant symbols* (i.e. 0-ary function symbols), and a finite set N_A of *rigid action function symbols*. A term is called *ground term* if it contains no variables. Note that in contrast to [13, 7] we also allow action functions with arity ≥ 2. We use the following symbols for vectors of object variables and object constants: $\vec{x}$ denotes a vector of at most two variables, $\vec{y}$ a vector of arbitrarily many variables and $\vec{c}$ a vector of arbitrarily many constants.

To build formulas, we consider a set of *fluent* predicate symbols N_F and a set of *rigid* predicate symbols N_R, each of which takes at most two arguments of sort object. Intuitively, fluents are properties that may change due to actions, while rigids do not. In addition, we use a special fluent predicate $Poss(a)$ taking an action term as argument for expressing action preconditions. Apart from equality and the usual logical connectives, formulas may contain subformulas of the form $\exists^{\leq m} x.\alpha$ (at most m individuals x satisfy α, where $m \in \mathbb{N}$), $\exists^{\geq m} x.\alpha$ (at least m individuals x satisfy α), $\Box\alpha$ (α holds after any number of actions) and $[t]\alpha$ (α holds after executing t, where t is an action term). A formula is called *fluent formula* if it neither contains $\Box$, nor $[\cdot]$, nor the predicate $Poss$, and mentions at most two non-free variables. A *fluent sentence* is a fluent formula without free variables.

Semantics: Terms and formulas are interpreted w.r.t. a semantic model called a *world*: Let $\mathcal{N}_O$ and $\mathcal{N}_A$ be the set of ground terms (also called "standard names") of sort object, and action, respectively, and $\mathcal{P}_F$ the set of all ground atoms of the form $F(n_1, \ldots, n_k)$ for predicates $F \in N_F \cup N_R$. Moreover, let $\mathcal{Z} = {\mathcal{N}_A}^*$ be the set of all finite sequences of ground action terms (including the empty sequence $\langle\rangle$). A world w then maps ground atoms and action sequences to truth values

$$w : \mathcal{P}_F \times \mathcal{Z} \to \{0, 1\}$$

respecting the *rigidity constraint*: if R is a rigid predicate, then for all $z, z' \in \mathcal{Z}$, $w[R(\vec{n}), z] = w[R(\vec{n}), z']$. Let $\mathcal{W}$ denote the set of all worlds. Given a world $w \in \mathcal{W}$ and a sentence α, we define $w \models \alpha$ as $w, \langle\rangle \models \alpha$, where for any $z \in \mathcal{Z}$:

1. $w, z \models F(t_1, \ldots, t_k)$ iff $w[F(t_1, \ldots, t_k), z] = 1$;
2. $w, z \models (t_1 = t_2)$ iff t_1 and t_2 are identical;
3. $w, z \models \alpha \wedge \beta$ iff $w, z \models \alpha$ and $w, z \models \beta$;
4. $w, z \models \neg\alpha$ iff $w, z \not\models \alpha$;
5. $w, z \models \forall x.\alpha$ iff $w, z \models \alpha^x_t$ for all $t \in \mathcal{N}_x$;
6. $w, z \models \exists^{\leq m} x.\alpha$ iff $|\{t \in \mathcal{N}_x \mid w, z \models \alpha^x_t\}| \leq m$;
7. $w, z \models \exists^{\geq m} x.\alpha$ iff $|\{t \in \mathcal{N}_x \mid w, z \models \alpha^x_t\}| \geq m$;
8. $w, z \models \Box\alpha$ iff $w, z \cdot z' \models \alpha$ for all $z' \in \mathcal{Z}$;
9. $w, z \models [t]\alpha$ iff $w, z \cdot t \models \alpha$;

Above, $\mathcal{N}_x$ refers to the set of all ground terms of the same sort as x. We moreover use α^x_t to denote the result of simultaneously replacing all free occurrences of x by t.

Basic Action Theories: To encode a dynamic application domain, we use a *basic action theory* (BAT), which is a set of formulas $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{\text{pre}} \cup \mathcal{D}_{\text{post}}$, where

1. $\mathcal{D}_0$, *the initial theory*, is a finite set of fluent sentences describing the initial state of the world.
2. $\mathcal{D}_{\text{pre}}$ is a set of *precondition axioms* such that for any action function $A \in N_A$ there is an axiom of the form $\Box Poss(A(\vec{y})) \equiv \varphi$, with φ being a fluent formula with free variables $\vec{y}$.
3. $\mathcal{D}_{\text{post}}$ is a finite set of *successor state axioms* (SSAs), one for each fluent $F \in N_F$, incorporating Reiter's [21] solution to the frame problem, and encoding the effects that actions have on the different fluents. The SSA for a fluent F has the form $\Box[a]F(\vec{x}) \equiv \gamma^+_F \vee F(\vec{x}) \wedge \neg\gamma^-_F$, where γ^+_F and γ^-_F are fluent formulas with free variables $\vec{x}$ and a.

In this paper we restrict ourselves to *local-effect* BATs, i.e. in all SSAs the positive and negative effect conditions γ^+_F and γ^-_F, respectively, are disjunctions of formulas $\exists\vec{z}[a = A(\vec{y}) \wedge \phi]$, where A is an action function, $\vec{y}$ contains $\vec{x}$, $\vec{z}$ are the remaining variables of $\vec{y}$, and ϕ is a fluent formula with free variables $\vec{y}$ and at most two non-free variables that are all of sort object. Intuitively, in a local-effect action theory executing an action $A(\vec{c})$ with arguments $\vec{c}$ means that a fluent $F(\vec{d})$ can change its truth value for $\vec{d}$ only if $\vec{d}$ is contained in $\vec{c}$.

2.2 Golog Programs

Given a BAT axiomatizing preconditions and effects of atomic actions, we now define syntax and semantics of complex actions. The *program expressions* we consider here are the ones admitted by the following grammar:

$$\delta ::= \langle\rangle \mid t \mid \alpha? \mid \delta_1; \delta_2 \mid \delta_1 | \delta_2 \mid \delta_1 \| \delta_2 \mid \delta^* \tag{1}$$

That is we consider the fragment of CONGOLOG [11] that includes the empty program $\langle\rangle$, primitive actions t (where t is a ground action term), tests $\alpha?$ (where α is a fluent sentence), sequence, nondeterministic branching, concurrency, and nondeterministic iteration. Note that thus also **if** statements and **while** loops are included:

$$\textbf{if } \alpha \textbf{ then } \delta_1 \textbf{ else } \delta_2 \textbf{ endIf} \stackrel{def}{=} [\alpha?; \delta_1] \mid [\neg\alpha?; \delta_2] \tag{2}$$

$$\textbf{while } \alpha \textbf{ do } \delta \textbf{ endWhile} \stackrel{def}{=} [\alpha?; \delta]^*; \neg\alpha? \tag{3}$$

A GOLOG *program* $\mathcal{P} = (\mathcal{D}, \delta)$ is then given by a BAT $\mathcal{D}$ and a program expression δ.

Program expressions are interpreted as follows. A *configuration* $\langle z, \delta\rangle$ consists of an action sequence $z \in \mathcal{Z}$ and a program expression δ, where intuitively z is the history of actions that have already been performed, while δ is the program that remains to be executed.

Definition 1 (Program Transition Semantics)**.** *Let $\mathcal{P} = (\mathcal{D}, \delta)$ be a Golog program. The transition relation $\xrightarrow{w}$ among configurations, given a world w with $w, \langle\rangle \models \mathcal{D}_0$, is defined by induction on the size of δ as the least set satisfying*

1. $\langle z, t\rangle \xrightarrow{w} \langle z \cdot t, \langle\rangle\rangle$, *if* $w, z \models Poss(t)$;
2. $\langle z, \delta_1; \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \gamma; \delta_2\rangle$, *if* $\langle z, \delta_1\rangle \xrightarrow{w} \langle z \cdot t, \gamma\rangle$;
3. $\langle z, \delta_1; \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$, *if* $\langle z, \delta_1\rangle \in \mathcal{F}^w$ *and* $\langle z, \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$;
4. $\langle z, \delta_1 | \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$,
   if $\langle z, \delta_1\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$ *or* $\langle z, \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$;
5. $\langle z, \delta_1 \| \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta' \| \delta_2\rangle$, *if* $\langle z, \delta_1\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$;
6. $\langle z, \delta_1 \| \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta_1 \| \delta'\rangle$, *if* $\langle z, \delta_2\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle$;
7. $\langle z, \delta^*\rangle \xrightarrow{w} \langle z \cdot t, \gamma; \delta^*\rangle$, *if* $\langle z, \delta\rangle \xrightarrow{w} \langle z \cdot t, \gamma\rangle$.

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

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

Let $\xrightarrow{w}{}^*$ denote the reflexive and transitive closure of $\xrightarrow{w}$. The *set of reachable subprograms* $\mathsf{sub}(\delta)$ is defined as follows:

$$\mathsf{sub}(\delta) := \{\delta' \mid \exists w \models \mathcal{D}, z \in \mathcal{Z} \text{ s.t. } \langle\langle\rangle, \delta\rangle \xrightarrow{w}{}^* \langle z, \delta'\rangle\}$$

Lemma 1. *Let* δ *be a program expression over ground actions. The cardinality of* $\mathsf{sub}(\delta)$ *is exponentially bounded in the size* $|\delta|$ *of* δ.

To handle non-terminating, terminating and failing runs of a program uniformly, we introduce two fresh 0-ary fluents $Term$ and $Fail$ and two 0-ary action functions ϵ and $\mathfrak{f}$ and include axioms $\Box Poss(\epsilon) \equiv true$ and $\Box Poss(\mathfrak{f}) \equiv true$ in $\mathcal{D}_{\text{pre}}$ as well as axioms $\Box[a]Term \equiv a = \epsilon \vee Term$ and $\Box[a]Fail \equiv a = \mathfrak{f} \vee Fail$ in $\mathcal{D}_{\text{post}}$. Termination and failure are thus represented through non-terminating runs by having "sinks" that continue looping ϵ and $\mathfrak{f}$ indefinitely.

Definition 2 (Transition System)**.** *Let* $\mathcal{P} = (\mathcal{D}, \delta)$ *be a Golog program. The* transition system $T_{\mathcal{P}} = (Q, \rightarrow, I)$ *induced by* $\mathcal{P}$ *consists of the set of* states

$$Q := \{(w, z, \delta') \mid w \models \mathcal{D},\ z \in \mathcal{Z},\ \delta' \in \mathsf{sub}(\delta)\},$$

a transition relation $\rightarrow \subseteq Q \times Q$*, and a set of* initial states $I \subseteq Q$*, which are defined as follows:*

- $I := \{(w, \langle\rangle, \delta) \mid w, \langle\rangle \models \mathcal{D}_0\}$
- *It holds that* $(w, z, \rho) \rightarrow (w, z \cdot t, \rho')$ *if one of the following conditions is satisfied:*
  1. $\langle z, \rho\rangle \xrightarrow{w} \langle z \cdot t, \rho'\rangle$.
  2. $\langle z, \rho\rangle \in \mathcal{F}^w$, $t = \epsilon$ *and* $\rho' = \langle\rangle$.
  3. *In case there is no* $\langle z'', \rho''\rangle$ *s.t.* $\langle z, \rho\rangle \xrightarrow{w} \langle z'', \rho''\rangle$ *and* $\langle z, \rho\rangle \notin \mathcal{F}^w$, *we have* $t = \mathfrak{f}$ *and* $\rho' = \rho$.

3 Verification

3.1 The Verification Problem

First, we define the temporal logic used to specify properties of a given program. We define the logic $\mathcal{ES}$-C^2-CTL*, whose syntax is the same as for propositional CTL* [9], but in place of propositions we allow for fluent sentences:

$$\Phi ::= \alpha \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \mathsf{E}\Psi \tag{4}$$

$$\Psi ::= \Phi \mid \neg\Psi \mid \Psi_1 \wedge \Psi_2 \mid \mathsf{X}\Psi \mid \Psi_1 \,\mathsf{U}\, \Psi_2 \tag{5}$$

Above, α can be any fluent sentence. We call formulas according to (4) *state formulas*, and formulas according to (5) *path formulas*. We use the usual abbreviations $\mathsf{A}\Psi$ for $\neg\mathsf{E}\neg\Psi$, $\mathsf{F}\Psi$ for $\top \,\mathsf{U}\, \Psi$ and $\mathsf{G}\Psi$ for $\neg\mathsf{F}\neg\Psi$. $\mathcal{ES}$-C^2-CTL* formulas are interpreted w.r.t. a *transition system* of the form $T = (Q, \rightarrow, I)$, where $\rightarrow$ is a right-total transition relation on the set of states Q, $I \subseteq Q$ and each state $s \in Q$ is associated with a world w_s and an action sequence $z_s \in \mathcal{Z}$. Clearly, the transition system $T_{\mathcal{P}}$ induced by a Golog program $\mathcal{P}$ satisfies these properties.

A *path* π in T starting in a state $s_0 \in Q$ is of the form $\pi = s_0 s_1 s_2 \cdots$ with $s_i \rightarrow s_{i+1}$ for all $i = 0, 1, 2, \ldots$. The set of all paths in T starting in $s \in Q$ is denoted by $Paths(s)$. For a given $i \in \mathbb{N}$, the suffix of a path π from position i on is denoted by $\pi[i..]$.

Definition 3 ($\mathcal{ES}$-C^2-CTL* Semantics)**.** *Let* Φ *be a* $\mathcal{ES}$-C^2-CTL* *state formula and* $T = (Q, \rightarrow, I)$ *a transition system. Truth of* Φ *in* T *in a state* $s \in Q$*, denoted by* $T, s \models \Phi$*, is defined as follows:*

- $T, s \models \alpha$ *iff* $w_s, z_s \models \alpha$;
- $T, s \models \neg\Phi$ *iff* $T, s \not\models \Phi$;
- $T, s \models \Phi_1 \wedge \Phi_2$ *iff* $T, s \models \Phi_1$ *and* $T, s \models \Phi_2$;
- $T, s \models \mathsf{E}\Psi$ *iff there exists* $\pi \in Paths(s)$ *such that* $T, \pi \models \Psi$.

Let Ψ *be a* $\mathcal{ES}$-C^2-CTL* *path formula and* T *a transition system as above. Truth of* Ψ *in* T *in a path* π *starting in some state* $s_0 \in Q$*, denoted by* $T, \pi \models \Psi$*, is defined as follows:*

- $T, \pi \models \Phi$ *iff* $T, s_0 \models \Phi$;
- $T, \pi \models \neg\Psi$ *iff* $T, \pi \not\models \Psi$;
- $T, \pi \models \Psi_1 \wedge \Psi_2$ *iff* $T, \pi \models \Psi_1$ *and* $T, \pi \models \Psi_2$;
- $T, \pi \models \mathsf{X}\Psi$ *iff* $T, \pi[1..] \models \Psi$;
- $T, \pi \models \Psi_1 \,\mathsf{U}\, \Psi_2$ *iff*
  $\exists k \geq 0 : T, \pi[k..] \models \Psi_2$ *and* $\forall j, 0 \leq j < k : T, \pi[j..] \models \Psi_1$.

We write $T \models \Phi$ *if* $T, s_0 \models \Phi$ *for all* $s_0 \in I$.

We are now ready to define the *verification problem*:

Definition 4 (Verification Problem)**.** *Let* $\mathcal{P} = (\mathcal{D}, \delta)$ *be a Golog program,* $T_{\mathcal{P}} = (Q, \rightarrow, I)$ *the corresponding transition system and* Φ *an* $\mathcal{ES}$-C^2-CTL* *state formula. The formula* Φ *is* valid *in* $\mathcal{P}$ *if* $T_{\mathcal{P}} \models \Phi$. *The formula* Φ *is* satisfiable *in* $\mathcal{P}$ *if there exists a* $s_0 \in I$ *such that* $T_{\mathcal{P}}, s_0 \models \Phi$.

Note that the logic $\mathcal{ES}$-C^2-CTL* is expressive enough to encode several variants of the verification problem. For example, one can express global domain constraints as a conjunction φ of fluent sentences. The problem of whether these constraints persist during the execution of a program $\mathcal{P}$ corresponds to validity of the formula $\mathsf{AG}\varphi$ in the program $\mathcal{P}$. Furthermore, the fluents $Term$ and $Fail$ can be used to express facts about the termination or failing of a program.

3.2 Deciding the Verification Problem

In general, the transition system induced by a GOLOG program is infinite. In order to obtain a decision procedure, we show that it is possible to construct a finite abstraction that preserves satisfiability of $\mathcal{ES}$-C^2-CTL* formulas. We do this by defining an equivalence relation over the set of semantic models (worlds) and identifying the corresponding equivalence classes (called *types*). Since it can be shown that there are only finitely many such classes for a given program and since their computation reduces to a consistency check in the underlying decidable base logic C^2, the abstract transition system can be effectively computed, and standard model checking techniques can be used to verify the given $\mathcal{ES}$-C^2-CTL* property.

3.2.1 Regression with Ground Actions

The first ingredient for our result is to use a technique presented in [18, 17] for simplifying local-effect SSAs in presence of ground actions only. The idea is that it is sufficient to consider the *ground instantiations* of SSAs obtained by substituting the action variable a by each ground action term and simplifying using unique names for constants and actions:

Lemma 2. *Let $\mathcal{D}$ be a BAT, $\Box[a]F(\vec{x}) \equiv \gamma_F^+ \vee F(\vec{x}) \wedge \neg\gamma_F^- \in \mathcal{D}_{post}$ the local-effect SSA for the fluent $F(\vec{x})$ and $t = A(\vec{c})$ a ground action term. For the ground instantiated SSA for $F(\vec{x})$ w.r.t. t, given as $\Box[t]F(\vec{x}) \equiv \gamma_{F\,t}^{+\,a} \vee F(\vec{x}) \wedge \neg\gamma_{F\,t}^{-\,a}$, it holds that both $\gamma_{F\,t}^{+\,a}$ and $\gamma_{F\,t}^{-\,a}$ are equivalent to disjunctions of the form*

$$\vec{x} = \vec{c_1} \wedge \phi_1 \vee \cdots \vee \vec{x} = \vec{c_n} \wedge \phi_n, \tag{6}$$

where the vectors of constants $\vec{c_i}$ are contained in $\vec{c}$ and the formulas ϕ_i are fluent sentences with $i = 1, ..., n$.

From now on we hence assume that in the ground instantiated SSAs, the $\gamma_{F\,t}^{+\,a}$ and $\gamma_{F\,t}^{-\,a}$ are of the form (6). We use the notation $(\vec{c}, \phi) \in \gamma_{F\,t}^{+\,a}$ and $(\vec{c}, \phi) \in \gamma_{F\,t}^{-\,a}$ to express that there is a disjunct of the form $\vec{x} = \vec{c} \wedge \phi$ in $\gamma_{F\,t}^{+\,a}$ or $\gamma_{F\,t}^{-\,a}$, respectively.

Next, we define an *effect function* mapping a world w, a finite action sequence z and a ground action t to a set of fluent literals if $Poss(t)$ is satisfied in the situation represented by w, z.

Definition 5 (Effect Function)**.** *Let $\mathcal{D}$ be a BAT over the signature Σ and Lit the set of all positive and negative ground fluent atoms, given as follows:*

$$Lit := \{F(\vec{c}), \neg F(\vec{c}) \mid \exists t \in \mathcal{N}_A : (\vec{c}, \phi) \in \gamma_{F\,t}^{+\,a} \text{ or } (\vec{c}, \phi) \in \gamma_{F\,t}^{-\,a}\}$$

The effect function $\mathcal{E} : \mathcal{W} \times \mathcal{Z} \times \mathcal{N}_A \to 2^{Lit}$ *for $\mathcal{P}$ is a partial function where*

$$\begin{aligned}\mathcal{E}(w, z, t) :=& \{F(\vec{c}) \in Lit \mid \exists(\vec{c}, \phi) \in \gamma_{F\,t}^{+\,a} \wedge w, z \models \phi\} \cup \\ & \{\neg F(\vec{c}) \in Lit \mid \exists(\vec{c}, \phi) \in \gamma_{F\,t}^{-\,a} \wedge w, z \models F(\vec{c}) \wedge \phi\}\end{aligned}$$

if $w, z \models Poss(t)$, and otherwise undefined.

Next, we show that a simplified version of Reiter's regression operator can be defined w.r.t. an effect function, given a consistent set of fluent literals and a fluent sentence. A subset $E \subseteq Lit$ is called *non-contradictory* if there is no fluent atom $F(\vec{c})$ such that $\{F(\vec{c}), \neg F(\vec{c})\} \subseteq E$.

Definition 6 (Regression Operator)**.** *Let $F(\vec{v})$ be a formula where F is a fluent and $\vec{v}$ a vector of variables or constants and let $E \subseteq Lit$ be non-contradictory. We define the* regression *of $F(\vec{v})$ through E, written as $[F(\vec{v})]^{R(E)}$, as follows:*

$$[F(\vec{v})]^{R(E)} := \left(F(\vec{v}) \wedge \bigwedge_{\neg F(\vec{c}) \in E} (\vec{v} \neq \vec{c})\right) \vee \bigvee_{F(\vec{c}) \in E} (\vec{v} = \vec{c})$$

If α is a fluent sentence, $\alpha^{R(E)}$ denotes the result of replacing any occurrence of a fluent $F(\vec{v})$ by $[F(\vec{v})]^{R(E)}$.

Clearly, the regression result $\alpha^{R(E)}$ is again a fluent sentence. Intuitively, if we want to know whether a formula α holds after executing an action with effects E, it is sufficient to test whether the regressed formula $\alpha^{R(E)}$ is satisfied in the current situation.

Lemma 3. *Let $\mathcal{D}$ be a BAT, $w \in \mathcal{W}$ with $w \models \mathcal{D}$, α a fluent sentence and $t = A(\vec{c})$ a ground action term. For all $z \in \mathcal{Z}$, it holds that for $E = \mathcal{E}(w, z, t)$, $w, z \models \alpha^{R(E)}$ iff $w, z \cdot t \models \alpha$.*

We note that an iterated application of the regression operator can be reduced to an application of the operator for a single set of fluent literals. For a set $E \subseteq Lit$ we define $\neg E := \{\neg l \mid l \in E\}$ (modulo double negation). For two non-contradictory subsets E, E' of Lit and a sentence α it holds that

$$\left[\alpha^{R(E')}\right]^{R(E)} \equiv \alpha^{R(E \setminus \neg E' \cup E')}. \tag{7}$$

3.2.2 A Finite Abstraction

Next, we identify a finite set of relevant fluent sentences (called *context*) by considering all tests occurring in δ, all formulas defined as preconditions for ground actions occurring in δ, and all formulas occurring in the ground instantiated SSAs of $\mathcal{D}$:

Definition 7 (Context for a Golog Program)**.** *Let $\mathcal{P} = (\mathcal{D}, \delta)$ be a program. A* context *$\mathcal{C}$ for $\mathcal{P}$ is a finite set of fluent sentences satisfying the following condition: Let α be a fluent sentence. If*

- *α is a test in δ,*
- *or $\alpha = \varphi_{\vec{c}}^{\vec{y}}$ and there is a ground action $A(\vec{c})$ in δ with $\Box Poss(A(\vec{y})) \equiv \varphi \in \mathcal{D}_{pre}$,*
- *or $\alpha = \phi$ and there exists a ground action t in δ and a vector of constants $\vec{c}$ such that $(\vec{c}, \phi) \in \gamma_{F\,t}^{+\,a}$ or $(\vec{c}, \phi) \in \gamma_{F\,t}^{-\,a}$,*
- *or $\alpha = F(\vec{c})$ and there exists a ground action t in δ such that $(\vec{c}, \phi) \in \gamma_{F\,t}^{-\,a}$ for some ϕ,*

then $\{\alpha, \neg\alpha\} \subseteq \mathcal{C}$. We require that $\mathcal{C}$ is closed under negation.

The following lemma shows some properties of $\mathcal{C}$. Intuitively, if we consider a situation consisting of a world and a finite sequence of ground actions, then the effects of applying a ground action in that situation depend only on the formulas in $\mathcal{C}$ that are satisfied or not satisfied in that situation. Furthermore, whether a program configuration is final in a world or has a successor configuration in this world only depends on the context as well.

Lemma 4. *Let $\mathcal{C}$ be a context for a program $\mathcal{P} = (\mathcal{D}, \delta)$. Let $w_0, w_1 \in \mathcal{W}$ satisfying $\mathcal{D}$ and $z_0, z_1 \in \mathcal{Z}$ such that $w_0, z_0 \models \alpha$ iff $w_1, z_1 \models \alpha$ for all $\alpha \in \mathcal{C}$.*

1. *Let t be a ground action that occurs in δ. It holds that $\mathcal{E}(w_0, z_0, t) = \mathcal{E}(w_1, z_1, t)$.*
2. *Let ρ be a program that contains only ground actions from δ and all tests in ρ are contained in $\mathcal{C}$.*
   (a) *$\langle z_0, \rho\rangle \in \mathcal{F}^{w_0}$ iff $\langle z_1, \rho\rangle \in \mathcal{F}^{w_1}$.*
   (b) *$\langle z_0, \rho\rangle \xrightarrow{w_0} \langle z_0 \cdot t, \rho'\rangle$ iff $\langle z_1, \rho\rangle \xrightarrow{w_1} \langle z_1 \cdot t, \rho'\rangle$.*

For the identification of equivalence classes, note that it is not sufficient to consider only contexts within single states. We rather have to make sure that equivalence is preserved when doing a transition. For this purpose, we define types of worlds such that two worlds of the same type will satisfy the same temporal properties. In the following we use the notation 2^{Lit} to denote the set of all non-contradictory subsets of Lit. First, we need the notion of a set of *type elements* for a program $\mathcal{P}$ and a context $\mathcal{C}$ for $\mathcal{P}$:

$$TE(\mathcal{P}, \mathcal{C}) := \{(\alpha, E) \mid \alpha \in \mathcal{C}, E \in 2^{Lit}\}.$$

The *type* of a world is now defined as a set of type elements.

Definition 8 (Type of a World)**.** *Let $\mathcal{P}$ be a program, $\mathcal{C}$ a context for $\mathcal{P}$ and w a world with $w \models \mathcal{D}$. The* type *of w w.r.t. $\mathcal{P}$ and $\mathcal{C}$ is:*

$$type(w) := \{(\alpha, E) \in TE(\mathcal{P}, \mathcal{C}) \mid w, \langle\rangle \models \alpha^{R(E)}\}.$$

Example 1. *Consider a single fluent $OnTable(x)$, an action $remove(x)$ and an object constant b. The initial theory is given by $\mathcal{D}_0 = \{OnTable(b)\}$, $\mathcal{D}_{post}$ contains a single SSA*

$$\Box[a]OnTable(x) \equiv OnTable(x) \wedge \neg a = remove(x).$$

As a context for the BAT $\mathcal{D}$ and program $remove(b)$ we choose the following set consisting of two formulas (abbreviated by α_b and $\alpha_\exists$, respectively) and their negations

$$\mathcal{C} = \{\alpha_b : (\neg)OnTable(b), \alpha_\exists : (\neg)\exists x.OnTable(x)\}.$$

We consider two worlds w_0, w_1 such that

$$w_0, \langle\rangle \models OnTable(b) \text{ and}$$
$$w_0, \langle\rangle \not\models OnTable(b') \text{ for all } b' \in \mathcal{N}_O \text{ with } b \neq b'$$

and in w_1 it holds that

$$w_1, \langle\rangle \models OnTable(b) \text{ and}$$
$$w_1, \langle\rangle \models OnTable(b') \text{ for some } b' \in \mathcal{N}_O \text{ with } b \neq b'.$$

We have to consider three non-contradictory sets of literals $L_0 = \emptyset$, $L^+ = \{OnTable(b)\}$ and $L^- = \{\neg OnTable(b)\}$. The different types of w_0 and w_1 are given by:

$$\begin{aligned} type(w_0) := \{&(\alpha_b, L_0), (\alpha_\exists, L_0), (\alpha_b, L^+), (\alpha_\exists, L^+),\\ &(\neg\alpha_b, L^-), \underline{(\neg\alpha_\exists, L^-)}\};\\ type(w_1) := \{&(\alpha_b, L_0), (\alpha_\exists, L_0), (\alpha_b, L^+), (\alpha_\exists, L^+),\\ &(\neg\alpha_b, L^-), \underline{(\alpha_\exists, L^-)}\}. \end{aligned}$$

In this simple example b is the only object known to be on the table initially and it is the only object that can be affected by an action. But nevertheless, since we have only incomplete information about the initial world, we also have to consider possibly unknown objects. For example, we don't know whether there is exactly one object on the table or not. As we see here, the type of w_1 is different from the type of w_0, because the formula $\exists x.OnTable(x)$ in context $\mathcal{C}$ remains true in w_1 after removing the object b.

Lemma 5. *Consider two worlds w, w' and their types w.r.t. $\mathcal{P}$ and $\mathcal{C}$. It holds that:*

1. *Let $z \in \mathcal{N}_A^*$ be a sequence of ground actions that occur in δ and t a ground action occurring in δ. If $type(w) = type(w')$, then $\mathcal{E}(w, z, t) = \mathcal{E}(w', z, t)$.*
2. *Let $\alpha \in \mathcal{C}$ and $z \in \mathcal{N}_A^*$ a sequence of ground actions that occur in δ. If $type(w) = type(w')$, then $w, z \models \alpha$ iff $w', z \models \alpha$.*

As a consequence of this lemma we can determine $\mathcal{E}(w, z, t)$ based on $type(w)$. Consider a sequence $z = t_0 t_1 \cdots t_n$ of ground actions in δ. Using the equivalence (7) we can accumulate the set of effects of each prefix of z into a single set of literals. The accumulated set of effects for z in world w is denoted by $E(w, z)$. It clearly holds that $w, z \models \alpha$ iff $w, \langle\rangle \models \alpha^{R(E(w,z))}$ for any fluent sentence $\alpha \in \mathcal{C}$. The equivalence relation on states of the transition system can thus be defined as follows:

Definition 9. *Consider $\mathcal{P}$, $\mathcal{C}$ and the transition system $T_\mathcal{P} = (Q, \to, I)$. Let $(w, z, \rho), (w', z', \rho')$ be states in Q. (w, z, ρ) and (w', z', ρ') are equivalent, written as $(w, z, \rho) \simeq (w', z', \rho')$ iff $type(w) = type(w')$ and $E(w, z) = E(w', z')$ and $\rho = \rho'$.*

Equivalent states are thus indistinguishable by temporal properties:

Lemma 6. *Let $\mathcal{C}$ be a context for the program $\mathcal{P}$ with the corresponding transition system $T_\mathcal{P} = (Q, \to, I)$. Let $s_0, s_1 \in Q$ with $s_0 \simeq s_1$.*

1. *If there exists a state s_0' with $s_0 \to s_0'$ and $z_{s_0'} = z_{s_0} \cdot t$, then there exists a state s_1' with $s_1 \to s_1'$, $z_{s_1'} = z_{s_1} \cdot t$ and $s_0' \simeq s_1'$.*
2. *If there exists a state s_1' with $s_1 \to s_1'$ and $z_{s_1'} = z_{s_1} \cdot t$, then there exists a state s_0' with $s_0 \to s_0'$, $z_{s_0'} = z_{s_0} \cdot t$ and $s_0' \simeq s_1'$.*

Basically, this means that the relation $\simeq$ on the state space Q gives us a *bisimulation* w.r.t. the formulas in $\mathcal{C}$. Therefore, the following lemma is a direct consequence of this property.

Lemma 7. *Let $\mathcal{C}$ be a context for a program $\mathcal{P}$ with the transition system $T_\mathcal{P} = (Q, \to, I)$ and Φ a $\mathcal{ES}$-C^2-CTL* formula that contains only fluent sentences from $\mathcal{C}$. Let $s, s' \in I$ and $s \simeq s'$. There exists a path π starting in s with $T_\mathcal{P}, \pi \models \Phi$ iff there exists a run π' starting in s' with $T_\mathcal{P}, \pi' \models \Phi$.*

With the above, it suffices to consider the quotient transition system $T_{\mathcal{P}/\simeq}$ of $T_\mathcal{P}$ w.r.t. $\simeq$. The equivalence classes $[w, z, \rho]_\simeq$ (corresponding to states in the quotient transition system) can be characterized by the type $type(w)$, the subset $E(w, z)$ of Lit and $\rho \in \mathsf{sub}(\delta)$. There are only finitely many world types, subsets of Lit and reachable subprograms of δ. Hence, the quotient transition system is finite.

To show how the quotient transition system can be used to decide whether $T_\mathcal{P} \models \Phi$ for a $\mathcal{ES}$-C^2-CTL* formula Φ we use the notion of *propositional abstraction*. Consider the transition system $T_\mathcal{P} = (Q, \to, I)$ of a program $\mathcal{P}$, a $\mathcal{ES}$-C^2-CTL* state formula Φ and a context $\mathcal{C}$ for $\mathcal{P}$ that also contains all fluent sentences occurring in Φ. For any $\alpha \in \mathcal{C}$ we introduce an atomic proposition $\widehat{\alpha}$. We define the relevant set of atomic proposition $\widehat{\mathcal{C}} := \{\widehat{\alpha} \mid \alpha \in \mathcal{C}\}$. Let $\widehat{\Phi}$ denote the propositional CTL* formula that is obtained from Φ by replacing any fluent sentence in Φ by the corresponding atomic proposition. Further, we introduce a labeling $\widehat{L}_\mathcal{C}$ that labels the states in $T_{\mathcal{P}/\simeq}$ with subsets of $\widehat{\mathcal{C}}$ as follows:

$$\widehat{L}_\mathcal{C}([w, z, \rho]_\simeq) := \{\widehat{\alpha} \mid (\alpha, E(w, z)) \in type(w)\}.$$

$\widehat{L}_\mathcal{C}([s]_\simeq)$ is uniquely defined since all states in the equivalence class $[s]_\simeq$ have the same context label. Together with the labeling $\widehat{L}_\mathcal{C}$, the quotient transition system $T_{\mathcal{P}/\simeq}$ can be viewed as a finite propositional transition system. To verify whether $T_{\mathcal{P}/\simeq} \models \widehat{\Phi}$, we can now apply standard model checking techniques for CTL*. It remains to be shown how the quotient transition system can be computed.

Consider a program $\mathcal{P} = (\mathcal{D}, \delta)$ and a context $\mathcal{C}$. First, we guess a set of type elements $\tau \subseteq TE(\mathcal{P}, \mathcal{C})$ such that for all $\alpha \in \mathcal{C}$ and for all non-contradictory $E \subseteq Lit$, it holds that either $(\alpha, E) \in \tau$ or $(\neg\alpha, E) \in \tau$. Using the regression operator we test whether τ is indeed a type of a world that satisfies the BAT. This is done by checking consistency of the C^2 KB, given by $\mathcal{D}_0 \cup \{\alpha^{R(E)} \mid (\alpha, E) \in \tau\}$. If this KB is consistent, then there exists a world $w, \langle\rangle \models \mathcal{D}_0$ with $type(w) = \tau$. We get that $(\tau, \emptyset, \delta)$ represents the initial state $[(w, \langle\rangle, \delta)]_\simeq$ in the quotient transition system $T_{\mathcal{P}/\simeq}$. The possible images of the effect function and the labels of all possible reachable state are encoded in the type τ. Therefore, the reachable fragment of

the quotient transition system from the initial state $(\tau, \emptyset, \delta)$ can be easily obtained from the type representation τ. For more details we refer to the technical report [22].

Having this finite abstraction of the transition system, the verification problem boils down to a propositional CTL* model checking problem, as described above. The complexity of this decision procedure is mainly determined by the complexity of the test whether a set of type elements τ represents the type of a world. To do this we have to test consistency of an exponentially large C^2 knowledge base. Knowledge base consistency in C^2 can be decided in NEXPTIME [20]. Therefore, checking whether τ is a type can be done in 2-NEXPTIME. It turns out that 2-NEXPTIME is also an upper bound for the overall complexity.

Theorem 1. *Satisfiability of an $\mathcal{ES}$-C^2-CTL* formula Φ in a Golog program $\mathcal{P} = (\mathcal{D}, \delta)$ is decidable in* 2-NEXPTIME.

4 Related Work

Apart from what was discussed in the introduction, other researchers have looked at decidable verification in the context of the Situation Calculus. One line of research is followed by De Giacomo, Lespérance and Patrizi [8] who show decidability for first-order μ-calculus properties for a class of BATs that only admit finitely many instances of fluents to hold. In contrast, our approach also allows fluents with infinite extensions. Moreover, their notion of boundedness is a semantical condition that is in general undecidable to check, whereas our approach relies on purely syntactical restrictions. In related work, Hariri et al. [14] consider the verification of μ-calculus properties in the context of light-weight DLs, where new information may be added at any time. Among other things, they show that decidability obtains if the added information is bounded. But in contrast to the action theories we consider, the frame problem is not solved in their underlying action formalism, i.e. the actions are memoryless.

5 Conclusion

We presented results on the verification of temporal properties for GOLOG programs. We have extended previous decidability results as presented in [7] and [4] to a fragment that uses the expressive, yet decidable fragment C^2 of FOL as base logic, that allows for local-effect actions with arbitrarily many arguments, and that admits properties expressed in the branching-time temporal logics CTL* over C^2 axioms. Decidability is obtained by constructing a finite abstraction of the infinite transition system induced by the GOLOG program and its action theory. Note that the restrictions we impose (decidable base logic, ground actions only, and local-effects) are all necessary in the sense that we can show [22] that dropping any one of them would instantly render the verification problem undecidable again.

There are many directions for future work. We plan to further investigate the exact computational complexity and evaluate the approach practically by means of an implementation. It would also be interesting to extend our results in terms of expressiveness, e.g. by considering SSAs that go beyond local-effect theories, or by reintroducing non-ground actions in a limited, decidable fashion.

Acknowledgements

This work was supported by the German National Science Foundation (DFG) research unit FOR 1513 on Hybrid Reasoning for Intelligent Systems (`http://www.hybrid-reasoning.org`).

REFERENCES

[1] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, Cambridge University Press, 2003.

[2] Franz Baader, Hongkai Liu, and Anees ul Mehdi, 'Verifying properties of infinite sequences of description logic actions', in *Proc. ECAI 2010*, (2010).

[3] Franz Baader, Carsten Lutz, Maja Miličić, Ulrike Sattler, and Frank Wolter, 'Integrating description logics and action formalisms: First results', in *Proc. AAAI 2005*, (2005).

[4] Franz Baader and Benjamin Zarrieß, 'Verification of Golog programs over description logic actions', in *Proc. FroCoS'13*, (2013).

[5] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun, 'Experiences with an interactive museum tour-guide robot', *Artificial Intelligence*, **114**(1–2), 3–55, (1999).

[6] Jens Claßen and Gerhard Lakemeyer, 'A logic for non-terminating Golog programs', in *Proc. KR 2008*, (2008).

[7] Jens Claßen, Martin Liebenberg, and Gerhard Lakemeyer, 'On decidable verification of non-terminating Golog programs', in *Proc. NRAC 2013*, (2013).

[8] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi, 'Bounded situation calculus action theories and decidable verification', in *Proc. KR 2012*, (2012).

[9] E. Allen Emerson and Joseph Y. Halpern, "'sometimes" and "not never" revisited: on branching versus linear time temporal logic', *J. ACM*, **33**(1), 151–178, (1986).

[10] Alexander Ferrein and Gerhard Lakemeyer, 'Logic-based robot control in highly dynamic domains', *Robotics and Autonomous Systems*, (2008).

[11] 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).

[12] Giuseppe De Giacomo, Evgenia Ternovska, and Raymond Reiter, 'Non-terminating processes in the situation calculus', in *Working Notes of "Robots, Softbots, Immobots: Theories of Action, Planning and Control", AAAI'97 Workshop*, (1997).

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

[14] Babak Bagheri Hariri, Diego Calvanese, Marco Montali, Giuseppe De Giacomo, Riccardo De Masellis, and Paolo Felli, 'Description logic knowledge and action bases', *Journal of Artificial Intelligence Research*, **46**, 651–686, (2013).

[15] Gerhard Lakemeyer and Hector J. Levesque, 'A semantic characterization of a useful fragment of the situation calculus with knowledge', *Artificial Intelligence*, **175**(1), 142–164, (2010).

[16] 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).

[17] Yongmei Liu and Gerhard Lakemeyer, 'On first-order definability and computability of progression for local-effect actions and beyond', in *Proc. IJCAI 2009*, (2009).

[18] Yongmei Liu and Hector J. Levesque, 'Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions', in *Proc. IJCAI 2005*, (2005).

[19] John McCarthy and Patrick Hayes, 'Some philosophical problems from the standpoint of artificial intelligence', in *Machine Intelligence 4*, 463–502, American Elsevier, (1969).

[20] Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera, 'Complexity results for first-order two-variable logic with counting', *SIAM Journal on Computing*, **29**(4), 1083–1117, (2000).

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

[22] Benjamin Zarrieß and Jens Claßen, 'On the decidability of verifying LTL properties of Golog programs', LTCS-Report 13–10, Chair of Automata Theory, TU Dresden, Dresden, Germany, (2013). Extended version. See http://lat.inf.tu-dresden.de/research/reports.html.