Decidable Verification of Decision-Theoretic Golog

Jens Claßen[1] and Benjamin Zarrieß[2]

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

Abstract. The GOLOG agent programming language is a powerful means to express high-level behaviours in terms of programs over actions defined in a Situation Calculus theory. Its variant DTGOLOG includes decision-theoretic aspects in the form of stochastic (probabilistic) actions and reward functions. In particular for physical systems such as robots, verifying that a program satisfies certain desired temporal properties is often crucial, but undecidable in general, the latter being due to the language's high expressiveness in terms of first-order quantification, range of action effects, and program constructs. Recent results for classical GOLOG show that by suitably restricting these aspects, the verification problem becomes decidable for a non-trivial fragment that retains a large degree of expressiveness. In this paper, we lift these results to the decision-theoretic case by providing an abstraction mechanism for reducing the infinite-state Markov Decision Process induced by the DTGOLOG program to a finite-state representation, which then can be fed into a state-of-the-art probabilistic model checker.

1 Introduction

When it comes to the design and programming of an autonomous agent, the GOLOG [12] family of action languages offers a powerful means to express high-level behaviours in terms of complex programs whose basic building blocks are the primitive actions described in a Situation Calculus [16] action theory. GOLOG's biggest advantage perhaps is the fact that a programmer can freely combine imperative control structures with non-deterministic constructs, leaving it to the system to resolve non-determinism in a suitable manner. Its extension DTGOLOG [2, 17] includes decision-theoretic aspects in the form of stochastic (probabilistic) actions and reward functions, essentially expressing a form of (infinite-state) Markov Decisions Process (MDP) [15].

In particular when GOLOG is used to control physical robots, it is often crucial to verify a program against some specification of desired behaviour, for example in order to ensure liveness and safety properties, typically expressed by means of temporal formulas. Unfortunately, the general verification problem for GOLOG is undecidable due to the language's high expressivity in terms of first-order quantification, range of action effects, and program constructs. For this reason, there have recently been endeavours to identify restricted, but non-trivial fragments of Golog where verification (and hence other reasoning tasks such as projection) becomes decidable, while a great deal of expressiveness is retained. In [20] we presented one such result for a class of action theories, called acyclic, that allows for non-local effects, i.e. where actions may affect an unbounded number of objects that are not explicitly mentioned as action parameters. Decidability of verification is achieved by restricting dependencies between fluents in successor state axioms, which allows for a wide range of applications that includes the well-known briefcase domain [14].

So far, to the best of our knowledge, the verification of temporal properties of decision-theoretic Golog programs has not received any attention, even though in most practical applications one has to deal with uncertainty, e.g. in the form of actions failing with a certain probability and not showing the desired effects. In this paper, we lift the above mentioned decidability result on acyclic theories to the decision-theoretic case by providing an abstraction mechanism for reducing the infinite-state MDP induced by a DTGolog program to a finite-state representation, which then can be fed into any state-of-the-art probabilistic model checker such as PRISM [10] and STORM [4].

2 Preliminaries

2.1 The Logic $\mathcal{ES}$

We use a fragment of the first-order action logic $\mathcal{ES}$ [11], a variant of the Situation Calculus that uses modal operators instead of situation terms to express what is true after a number of actions has occurred. Not only is the syntax of $\mathcal{ES}$ in our view more readable, but its special semantics also makes proofs for many semantic properties simpler, while retaining much of the expressive power and main benefits of the original Situation Calculus. In particular, this includes the usage of *Basic Action Theories* (BATs) [16] to encode dynamic domains.

As we aim at decidability, we further have to restrict ourselves to a decidable fragment of FOL as base logic, as otherwise reasoning about theories not involving actions, programs and temporal properties would be undecidable already. For this purpose we use C^2, the *two-variable fragment of FOL with equality and counting*, an expressive fragment that subsumes most description logics.

Syntax There are *terms* of sort *object*, *number* and *action*. Variables of sort object are denoted by symbols $x, y, \ldots$, of sort number by p, r, and of sort action by a. N_O is a countably infinite set of *object constant symbols*, N_N the countable set of rational numbers, and N_A a countably infinite set of *action function symbols* with arguments of sort object. We denote the set of all ground terms (also called *standard names*) of sort object, number and action by $\mathcal{N}_O$, $\mathcal{N}_N$, and $\mathcal{N}_A$, respectively.

Formulas are built using *fluent* predicate symbols (predicates that may vary as the result of actions) with at most two arguments of sort object, and equality, using the usual logical connectives, quantifiers, and counting quantifiers. In addition we have the two special fluents $\mathit{Prob}(a_s, a_n, p)$ (taking two actions a_s, a_n and a number p as arguments), expressing that stochastic action a_s can have outcome a_n with probability p, and $\mathit{Reward}(r)$ (taking a number r as argument), saying that the reward in the current situation is r. Furthermore, there are two modalities for referring to future situations: $\Box\phi$ says that ϕ holds after any sequence of actions, and $[t]\phi$ means that ϕ holds after executing action t.

A formula is called *fluent formula* if it contains no $\Box$, no $[\cdot]$, no *Prob* and no *Reward* (i.e. such formulas talk about the current state of the world and do not involve dynamic or decision-theoretic aspects). A C^2*-fluent formula* is a fluent formula that contains no terms of sort action and at most two variables. A *sentence* or *closed formula* is a formula without free variables.

Semantics A situation is a finite sequence (history) of actions. Let $\mathcal{Z} := \mathcal{N}_A^*$ be the set of all situations (including the empty sequence $\langle\rangle$) and $\mathcal{P}_F$ the set of all *primitive formulas* $F(n_1, ..., n_k)$, where F is a regular k-ary fluent with $0 \leq k \leq 2$ and the n_i are object standard names, together with all expressions of form $\mathit{Prob}(t_1, t_2, c_1)$ and $\mathit{Reward}(c_2)$, where $t_1, t_2 \in \mathcal{N}_A$ and $c_1, c_2 \in \mathcal{N}_N$. A *world* w then maps primitive formulas and situations to truth values:

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

The set of all worlds is denoted by $\mathcal{W}$.

Definition 1 (Truth of Formulas). *Given a world $w \in \mathcal{W}$ and a closed formula ψ, we define $w \models \psi$ as $w, \langle\rangle \models \psi$, where for any $z \in \mathcal{Z}$:*

1. $w, z \models F(n_1, \ldots, n_k)$ *iff* $w[F(n_1, \ldots, n_k), z] = 1$*;*
2. $w, z \models (n_1 = n_2)$ *iff* n_1 *and* n_2 *are identical;*
3. $w, z \models \psi_1 \wedge \psi_2$ *iff* $w, z \models \psi_1$ *and* $w, z \models \psi_2$*;*
4. $w, z \models \neg\psi$ *iff* $w, z \not\models \psi$*;*
5. $w, z \models \forall x.\phi$ *iff* $w, z \models \phi^x_n$ *for all* $n \in \mathcal{N}_x$*;*
6. $w, z \models \exists^{\leq m} x.\phi$ *iff* $|\{n \in \mathcal{N}_x \mid w, z \models \phi^x_n\}| \leq m$*;*
7. $w, z \models \exists^{\geq m} x.\phi$ *iff* $|\{n \in \mathcal{N}_x \mid w, z \models \phi^x_n\}| \geq m$*;*
8. $w, z \models \Box\psi$ *iff* $w, z \cdot z' \models \psi$ *for all* $z' \in \mathcal{Z}$*;*
9. $w, z \models [t]\psi$ *iff* $w, z \cdot t \models \psi$*.*

Above, $\mathcal{N}_x$ refers to the set of all standard names of the same sort as x. Moreover ϕ^x_n denotes the result of simultaneously replacing all free occurrences of x in ϕ by n. Note that by rule 2, the unique names assumption for constants is part of our semantics. We use the notation $\boldsymbol{x}$ and $\boldsymbol{y}$ for sequences of object variables and $\boldsymbol{v}$ for a sequence of object terms. We understand $\vee$, $\exists$, $\supset$, and $\equiv$ as the usual abbreviations.

2.2 Action Theories

Definition 2 (Basic Action Theories). *A* C^2-basic action theory *(C^2-BAT)* $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{post}$ *is a set of axioms that describes the dynamics of a specific application domain, where*

1. $\mathcal{D}_0$*,* the initial theory*, is a finite set of* C^2*-fluent sentences describing the initial state of the world;*
2. $\mathcal{D}_{post}$ *is a finite set of* successor state axioms *(SSAs), one for each fluent relevant to the application domain, incorporating Reiter's [16] solution to the frame problem to encode action effects, of the form*

$$\forall a.\forall \boldsymbol{x}.\Box\big(([a]F(\boldsymbol{x})) \equiv \gamma_F^+ \vee \big(F(\boldsymbol{x}) \wedge \neg\gamma_F^-\big)\big)$$

where the positive effect condition γ_F^+ *and* negative effect condition γ_F^- *are fluent formulas that are (possibly empty) disjunctions of formulas of the form* $\exists \boldsymbol{y}.\big(a = A(\boldsymbol{v}) \wedge \phi \wedge \phi'\big)$ *such that*

(a) $\exists \boldsymbol{y}.\big(a = A(\boldsymbol{v}) \wedge \phi \wedge \phi'\big)$ *contains the free variables* $\boldsymbol{x}$ *and* a *and no other free variables;*
(b) $A(\boldsymbol{v})$ *is an action term and* $\boldsymbol{v}$ *contains* $\boldsymbol{y}$*;*
(c) ϕ *is a fluent formula with no terms of sort action and the number of variable symbols in it not among* $\boldsymbol{v}$ *or bound in* ϕ *is less or equal two;*
(d) ϕ' *is a fluent formula with free variables among* $\boldsymbol{v}$*, no action terms, and at most two bound variables.*

ϕ *is called* effect descriptor *and* ϕ' context condition.

The restrictions 2a and 2b on SSAs are without loss of generality and describe the usual syntactic form of SSAs. Intuitively, the effect descriptor ϕ defines a set of (pairs of) objects that are added to or deleted from the relational fluent F when $A(\boldsymbol{v})$ is executed. If free occurrences of variables in ϕ that appear as arguments of $A(\boldsymbol{v})$ are instantiated, condition 2c ensures definability of the (instantiated) effect descriptor in our base logic C^2. In contrast to the effect descriptor, the context condition ϕ' only tells us *whether* $A(\boldsymbol{v})$ has an effect on F, but not *which* objects are affected. Condition 2d again ensures that after instantiation of the action, the context condition is a sentence in C^2. The variables $\boldsymbol{x}$ mentioned in 2a may hence have free occurrences in ϕ but not in ϕ'.

Note that for simplicity we do not include precondition axioms, again without loss of generality: To ensure that action t only gets executed when precondition ϕ_t holds, simply precede every occurrence of t in the program expression (cf. Section 2.3) by a test for ϕ_t.

For representing the decision-theoretic aspects, we assume that action function symbols are subdivided into two disjoint subsets, *deterministic* actions and *stochastic* actions. We then associate every stochastic action with a probability distribution over a finite number of possible outcomes in the form of deterministic actions. Moreover, (state-based) rewards are represented by assigning numeric values to situations:

Definition 3 (Decision-Theoretic BATs). *A* C^2-decision-theoretic action theory *(C^2-DTBAT)* $\mathcal{DDT} = \mathcal{D} \cup \mathcal{D}_{prob} \cup \mathcal{D}_{reward}$ *extends a BAT* $\mathcal{D}$ *over deterministic actions by*

1. $\mathcal{D}_{prob}$*, an axiom of the form* $\Box Prob(a_s, a_n, p) \equiv \phi$*, where* a_s *and* a_n *are action variables,* p *is a number variable, and* ϕ *is a disjunction of formulas of the form*

$$\exists \boldsymbol{x}.\ a_s = A(\boldsymbol{x}) \wedge \bigvee_i a_n = A_i(\boldsymbol{x}_i) \wedge p = c_i,$$

where A *is a stochastic action, the* A_i *are deterministic actions defined in* $\mathcal{D}$*, the* $\boldsymbol{x}_i$ *are contained in* $\boldsymbol{x}$*, and the* c_i *are rational constants with* $0 < c_i \leq 1$ *and* $\sum_i c_i = 1$*. Furthermore, we assume that Prob is defined to be functional in the sense that for any ground action terms* t_s *and* t_n*, there is at most one* c *such that* $Prob(t_s, t_n, c)$*.*

2. $\mathcal{D}_{reward}$*, an axiom of the form* $\Box Reward(r) \equiv \psi$*, where* ψ *is a fluent formula with free variable* r*, no terms of sort action and at most two bound variables. Reward is assumed to be partially functional, i.e. in any situation there is at most one* r *such that* $Reward(r)$ *holds.*

Example 1. Consider a warehouse domain with *shelves* holding *boxes* containing *items*. The fluent $Broken(x)$ denotes that a box or item x is currently broken, $On(x, y)$ says that box or item x is currently on shelf y, and $Contains(x, y)$ is true for a box x containing an item y.

The agent is a robot that can move a box v from shelf s to shelf s' using the action $Move(v, s, s')$. We also have actions with undesired effects: $Drop(v, s)$ stands for dropping a box v from shelf s to the ground, causing all fragile objects in it to break if there is no bubble wrap in it. Finally, $Repair(s)$ is an action by means of which the robot can repair a box or an item that is not fragile.

Figure 2 exemplarily shows the effect conditions for $Broken(x)$ and $On(x, y)$. Effect descriptors are underlined with a solid line, context conditions with a dashed line. If for example the agent were to drop the *box* in an initial situation incompletely described by the axioms in Figure 1, everything in it will break if the box contains no bubble wrap, i.e. the BAT entails

$$\begin{aligned}&\neg\exists x\big(Contains(box, x) \wedge BubbleWrap(x)\big)\\&\quad\supset [Drop(box)]\big(\forall y. Contains(box, y) \supset Broken(y)\big).\end{aligned}$$

$MoveS(v, s, s')$ is a stochastic action that has the desired effect in 90% of the cases, but there is a 10% chance to drop v from shelf s; having the unbroken

$$
\begin{aligned}
&\mathit{On}(\mathit{box}, s_1),\\
&\forall x \exists^{\leq 1} y.\, \mathit{On}(x, y),\\
&\forall x.(\mathit{BubbleWrap}(x) \supset \neg \mathit{Fragile}(x)),\\
&\mathit{Contains}(\mathit{box}, \mathit{vase}),\\
&\forall x.(\mathit{Contains}(\mathit{box}, x) \supset \mathit{Fragile}(x))\\
&\forall y \exists^{\leq 1} x.\, \mathit{Contains}(x, y),
\end{aligned}
$$

Fig. 1. Example initial theory

$$
\begin{aligned}
\gamma^{+}_{\mathit{Broken}} &:= \exists v, s.\big(\, a = \mathit{Drop}(v, s) \land \underline{\mathit{On}(v, s) \land \mathit{Contains}(v, x) \land \mathit{Fragile}(x)} \land \\
&\qquad \underline{\neg \exists y.\, \mathit{Contains}(v, y) \land \mathit{BubbleWrap}(y)}\,\big);\\
\gamma^{-}_{\mathit{Broken}} &:= \exists s.\big(a = \mathit{Repair}(s) \land \underline{s = x \land \neg \mathit{Fragile}(x)}\big);\\
\gamma^{+}_{\mathit{On}} &:= \exists v, s, s'.\big(\, a = \mathit{Move}(v, s, s') \land \underline{y = s' \;\land \big(\mathit{Contains}(v, x) \lor x = v\big)}\big);\\
\gamma^{-}_{\mathit{On}} &:= \exists v, s, s'.\big(\, a = \mathit{Move}(v, s, s') \land \underline{y = s \;\land \big(\mathit{Contains}(v, x) \lor x = v\big)}\big) \lor \\
&\qquad \exists v, s.\big(\, a = \mathit{Drop}(v, s) \land \underline{y = s \land \big(v = x \lor \mathit{Contains}(v, x)\big)}\big)
\end{aligned}
$$

Fig. 2. Example effect conditions

vase on shelf s_1 gives a reward of 5, while on s_2 it gives a reward of 10:

$$
\begin{aligned}
&\Box \mathit{Prob}(a_s, a_n, p) \equiv \exists v, s, s'.\, a_s = \mathit{MoveS}(v, s, s') \land \\
&\quad \big(\, a_n = \mathit{Move}(v, s, s') \land p = 0.9 \lor \\
&\quad\;\; a_n = \mathit{Drop}(v, s) \land p = 0.1 \,\big)\\
&\Box \mathit{Reward}(r) \equiv \\
&\quad \big(\, \mathit{On}(\mathit{vase}, s_1) \land \neg \mathit{Broken}(\mathit{vase}) \land r = 5 \lor \\
&\quad\;\; \mathit{On}(\mathit{vase}, s_2) \land \neg \mathit{Broken}(\mathit{vase}) \land r = 10 \,\big)
\end{aligned}
$$

2.3 DTGolog and the Verification Problem

In a Golog program over ground actions we combine actions, whose effects are defined in a C^2-BAT, and tests, using a set of programming constructs to define a complex action.

Definition 4 (Programs). *A* program expression δ *is built according to the following grammar:*

$$\delta ::= t \mid \psi? \mid \delta;\delta \mid \delta|\delta \mid \delta^{*}$$

A program expression can thus be a (deterministic or stochastic) ground action term t, a test $\psi?$ *where ψ is a C^2-fluent sentence, or constructed from subprograms by means of* sequence $\delta;\delta$, non-deterministic choice $\delta|\delta$, *and* non-deterministic iteration δ^*. *Furthermore, **if** statements and **while** loops can be defined as abbreviations in terms of these constructs:*

$$\textbf{\textit{if}}\ \phi\ \textbf{\textit{then}}\ \delta_1\ \textbf{\textit{else}}\ \delta_2\ \textbf{\textit{endIf}} \overset{def}{=} [\phi?;\delta_1] \mid [\neg\phi?;\delta_2]$$

$$\textbf{\textit{while}}\ \phi\ \textbf{\textit{do}}\ \delta\ \textbf{\textit{endWhile}} \overset{def}{=} [\phi?;\delta]^*;\neg\phi?$$

A Golog program $\mathcal{G} = (\mathcal{D}, \delta)$ *consists of a C^2-BAT $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{post}$ and a program expression δ where all fluents occurring in $\mathcal{D}$ and δ have an SSA in $\mathcal{D}_{post}$.*

To handle termination and failure of a program we use two 0-ary fluents $Final$ and $Fail$ and two 0-ary action functions ϵ and $\mathfrak{f}$ and include the SSAs $\square[a]Final \equiv a = \epsilon \vee Final$ and $\square[a]Fail \equiv a = \mathfrak{f} \vee Fail$ in $\mathcal{D}_{post}$. Furthermore, we require that $\neg Final \in \mathcal{D}_0$ and $\neg Fail \in \mathcal{D}_0$, and that the fluents $Final$, $Fail$ and actions ϵ and $\mathfrak{f}$ do not occur in δ.

Following [3] we define the transition semantics of programs meta-theoretically. First, consider program expressions that only contain deterministic actions. A *configuration* $\langle z, \rho\rangle$ consists of a situation $z \in \mathcal{Z}$ and a program expression ρ, where z represents the actions that have already been performed, while ρ is the program that remains to be executed. Execution of a program in a world $w \in \mathcal{W}$ yields a *transition relation* $\xrightarrow{w}$ *among configurations* defined inductively over program expressions, given by the smallest set that satisfies:

1. $\langle z, t\rangle \xrightarrow{w} \langle z \cdot t, \langle\rangle\rangle$;
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 $\langle z, \delta_1\rangle \in \mathsf{Fin}(w)$ 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. $\langle z, \delta^*\rangle \xrightarrow{w} \langle z \cdot t, \gamma;\delta^*\rangle$, if $\langle z, \delta\rangle \xrightarrow{w} \langle z \cdot t, \gamma\rangle$.

The set of final configurations $\mathsf{Fin}(w)$ w.r.t. a world w is defined similarly as the smallest set such that:

1. $\langle z, \psi?\rangle \in \mathsf{Fin}(w)$ if $w, z \models \psi$;
2. $\langle z, \delta_1;\delta_2\rangle \in \mathsf{Fin}(w)$ if $\langle z, \delta_1\rangle \in \mathsf{Fin}(w)$ and $\langle z, \delta_2\rangle \in \mathsf{Fin}(w)$;
3. $\langle z, \delta_1|\delta_2\rangle \in \mathsf{Fin}(w)$ if $\langle z, \delta_1\rangle \in \mathsf{Fin}(w)$ or $\langle z, \delta_2\rangle \in \mathsf{Fin}(w)$;
4. $\langle z, \delta^*\rangle \in \mathsf{Fin}(w)$.

The set of *failing configurations* w.r.t. a world w is given by

$$\mathsf{Fail}(w) := \{\langle z, \delta\rangle \mid \langle z, \delta\rangle \notin \mathsf{Fin}(w),\ \text{there is no}\ \langle z \cdot t, \delta'\rangle\ \text{s.t.}\ \langle z, \delta\rangle \xrightarrow{w} \langle z \cdot t, \delta'\rangle\}.$$

We now turn to the decision-theoretic case. A DTGolog *program* $\mathcal{G} = (\mathcal{DDT}, \delta)$ consists of a C^2-DTBAT $\mathcal{DDT} = \mathcal{D} \cup \mathcal{D}_{\text{prob}} \cup \mathcal{D}_{\text{reward}}$ and a program expression

δ that only contains stochastic actions,[3] and where all fluents occurring in $\mathcal{DDT}$ and δ have an SSA in $\mathcal{D}_{\text{post}}$. Given a world $w \in \mathcal{W}$ with $w \models \mathcal{DDT}$, execution of δ in w induces an *infinite-state MDP w.r.t. w* given by $\mathsf{M}_\delta^w = \langle \mathsf{S}, \mathsf{s}^0, \mathsf{A}, \mathsf{P}, \mathsf{R} \rangle$, where

- the (infinite) set of states S is given by $\mathsf{Reach}(w, \delta_{\mathsf{det}})$, which denotes the set of configurations reachable from $\langle \langle\rangle, \delta_{\mathsf{det}} \rangle$ via $\xrightarrow{w}$, where δ_{det} is the program obtained by replacing every stochastic action $A(\boldsymbol{v})$ in δ by the expression $(A_1(\boldsymbol{v}_1)|\cdots|A_k(\boldsymbol{v}_k))$ such that the $A_i(\boldsymbol{v}_i)$ are all deterministic actions for which
$$w, z \models \mathit{Prob}(A(\boldsymbol{x}), A_i(\boldsymbol{x}_i), p)^{\boldsymbol{x}}_{\boldsymbol{v}};$$
- the initial state is $\mathsf{s}^0 = \langle \langle\rangle, \delta_{\mathsf{det}} \rangle$;
- the (finite) set of actions A are all (stochastic) ground action terms occurring in δ;
- the transition function $\mathsf{P} : \mathsf{S} \times \mathsf{A} \times \mathsf{S} \to \mathbb{R}$ is such that
$$\mathsf{P}(\langle z, \rho \rangle, t, \langle z \cdot t', \rho' \rangle) = \begin{cases} p, & w, z \models \mathit{Prob}(t, t', p) \\ & \quad \text{and } \langle z, \rho \rangle \xrightarrow{w} \langle z \cdot t', \rho' \rangle \\ 1, & \langle z, \rho \rangle \in \mathsf{Fin}(w),\ t = t' = \rho' = \epsilon \\ 1, & \langle z, \rho \rangle \in \mathsf{Fail}(w),\ t = t' = \rho' = \mathfrak{f} \\ 0, & \text{otherwise} \end{cases}$$
- the reward function $\mathsf{R} : \mathsf{S} \to \mathbb{R}$ is given by
$$\mathsf{R}(\langle z, \rho \rangle) = \begin{cases} r, & w, z \models \mathit{Reward}(r) \\ 0, & \text{otherwise} \end{cases}$$

In addition, final and failing configurations are absorbing states, i.e. if s is reached by ϵ, then $\mathsf{P}(\mathsf{s}, \epsilon, \mathsf{s}) = 1$, and if s is reached by $\mathfrak{f}$, then $\mathsf{P}(\mathsf{s}, \mathfrak{f}, \mathsf{s}) = 1$.

The non-determinism on the agent's side is resolved by means of a *policy* σ, which is a mapping $\sigma : \mathsf{S} \to \mathsf{A}$ such that $\mathsf{P}(\mathsf{s}, \sigma(\mathsf{s}), \mathsf{s}') > 0$ for some $\mathsf{s}' \in \mathsf{S}$. An infinite path $\pi = \mathsf{s}_0 \xrightarrow{\mathsf{a}_1} \mathsf{s}_1 \xrightarrow{\mathsf{a}_2} \cdots$ is called a *σ-path* if $\sigma(\mathsf{s}_j) = \mathsf{a}_{j+1}$ for all $j \geq 0$. The j-th state s_j of any such path is denoted by $\pi[j]$. The *set of all σ-paths* starting in s is $\mathsf{Paths}^\sigma(\mathsf{s}, \mathsf{M}_\delta^w)$.

Every policy σ induces a probability space $\mathit{Pr}_{\mathsf{s}}^\sigma$ on the sets of infinite paths starting in s, using the cylinder set construction [8]: For any finite path prefix $\pi_{\mathsf{fin}} = \mathsf{s}_0 \xrightarrow{\mathsf{a}_1} \mathsf{s}_1 \xrightarrow{\mathsf{a}_2} \cdots \mathsf{s}_n$, we define the probability measure
$$\mathit{Pr}_{\mathsf{s}_0,\mathsf{fin}}^\sigma = \mathsf{P}(\mathsf{s}_0, \mathsf{a}_1, \mathsf{s}_1) \cdot \mathsf{P}(\mathsf{s}_1, \mathsf{a}_2, \mathsf{s}_2) \cdot \ldots \cdot \mathsf{P}(\mathsf{s}_{n-1}, \mathsf{a}_n, \mathsf{s}_n).$$

This extends to a unique measure $\mathit{Pr}_{\mathsf{s}}^\sigma$.

[3] Note that we can always simulate a deterministic action by a stochastic one that has only one outcome.

Definition 5 (Temporal Properties of Programs). *To express temporal properties of probabilistic systems represented by* DTGolog *programs, we use a probabilistic variant of* CTL *called* PRCTL *[1], which extends* PCTL *[7] with rewards. However, in place of atomic propositions, we allow for C^2-fluent sentences ψ:*

$$\Phi ::= \psi \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathbf{P}_I[\Psi] \mid \mathbf{R}_J[\Phi] \tag{1}$$

$$\Psi ::= \mathsf{X}\Phi \mid (\Phi\ \mathsf{U}\,\Phi) \mid (\Phi\ \mathsf{U}^{\leq k}\ \Phi) \tag{2}$$

Above, $I \subseteq [0,1]$ and J are intervals with rational bounds. We call formulas according to (1) state formulas*, and formulas according to (2)* path formulas. *Intuitively, $\mathbf{P}_I[\Psi]$ expresses that the probability of the set of paths satisfying Ψ lies in the interval I, while $\mathbf{R}_J[\Phi]$ says that the expected reward cumulated before reaching a state that satisfies Φ is in J. Rather than providing intervals explicitly, we often use abbreviations such as $\mathbf{P}_{\geq 0.9}[\Psi]$ to denote $\mathbf{P}_{[0.9,1]}[\Psi]$, $\mathbf{P}_{=1}[\Psi]$ for $\mathbf{P}_{[1,1]}[\Psi]$, or $\mathbf{P}_{>0}[\Psi]$ for $\mathbf{P}_{]0,1]}[\Psi]$.*

$(\Phi_1\ \mathsf{U}^{\leq k}\,\Phi_2)$ *is the* step-bounded *version of the until operator, expressing that Φ_2 will hold within at most k steps, where Φ_1 holds in all states before. We use the usual abbreviations $\mathsf{F}\Phi$ (*eventually Φ*) for $(\mathit{true}\ \mathsf{U}\,\Phi)$ and $\mathsf{G}\Phi$ (*globally Φ*) for $\neg\mathsf{F}\neg\Phi$, as well as their corresponding step-bounded variants.*

Let Φ be a temporal state formula, M^w_δ the infinite-state MDP of a program $\mathcal{G} = (\mathcal{D}, \delta)$ w.r.t. a world w with $w \models \mathcal{DDT}$, and $\mathsf{s} = \langle z, \rho\rangle \in \mathsf{S}$. Truth of Φ in $\mathsf{M}^w_\delta, \mathsf{s}$, denoted by $\mathsf{M}^w_\delta, \mathsf{s} \models \Phi$ is defined as follows:

- $\mathsf{M}^w_\delta, \mathsf{s} \models \psi$ *iff* $w, z \models \psi$;
- $\mathsf{M}^w_\delta, \mathsf{s} \models \neg\Phi$ *iff* $\mathsf{M}^w_\delta, \mathsf{s} \not\models \Phi$;
- $\mathsf{M}^w_\delta, \mathsf{s} \models \Phi_1 \wedge \Phi_2$ *iff* $\mathsf{M}^w_\delta, \mathsf{s} \models \Phi_1$ *and* $\mathsf{M}^w_\delta, \mathsf{s} \models \Phi_1$;
- $\mathsf{M}^w_\delta, \mathsf{s} \models \mathbf{P}_I[\Psi]$ *iff for all policies σ, $\mathit{Pr}^\sigma_\mathsf{s}(\Psi) \in I$;*
- $\mathsf{M}^w_\delta, \mathsf{s} \models \mathbf{R}_J[\Phi]$ *iff for all policies σ, $\mathit{ExpRew}^\sigma_\mathsf{s}(\Phi) \in J$,*

where

$$\mathit{Pr}^\sigma_\mathsf{s}(\Psi) = \mathit{Pr}^\sigma_\mathsf{s}(\{\pi \in \mathsf{Paths}^\sigma(\mathsf{s}, \mathsf{M}^w_\delta) \mid \mathsf{M}^w_\delta, \pi \models \Psi\})$$

and $\mathit{ExpRew}^\sigma_\mathsf{s}(\Phi)$ is the expectation (wrt. measure $\mathit{Pr}^\sigma_\mathsf{s}$) of the random variable $X_\Phi(\pi) : \mathsf{Paths}^\sigma(\mathsf{s}, \mathsf{M}^w_\delta) \to \mathbb{R}_{\geq 0}$ such that for any path $\pi = \mathsf{s}_0 \xrightarrow{\mathsf{a}_1} \mathsf{s}_1 \xrightarrow{\mathsf{a}_2} \cdots$,

$$X_\Phi(\pi) = \begin{cases} 0, & \mathsf{M}^w_\delta, \mathsf{s}_0 \models \Phi \\ \infty, & \mathsf{M}^w_\delta, \mathsf{s}_i \not\models \Phi\ \forall\ i \in \mathbb{N} \\ \sum_{i=0}^{\min\{j \mid \mathsf{M}^w_\delta, \mathsf{s}_j \models \Phi\}-1} \mathsf{R}(\mathsf{s}_i), & \mathit{otherwise} \end{cases}$$

Let Ψ be a temporal path formula, M^w_δ and $\mathsf{s} = \langle z, \rho\rangle$ as above, and $\pi \in \mathsf{Paths}^\sigma(\mathsf{s}, \mathsf{M}^w_\delta)$ for some σ. Truth of Ψ in M^w_δ, π, denoted by $\mathsf{M}^w_\delta, \pi \models \Psi$, is defined as follows:

- $\mathsf{M}^w_\delta, \pi \models \mathsf{X}\Phi$ *iff* $\mathsf{M}^w_\delta, \pi[1] \models \Phi$;
- $\mathsf{M}^w_\delta, \pi \models (\Phi_1\ \mathsf{U}\,\Phi_2)$ *iff* $\exists i \geq 0 : \mathsf{M}^w_\delta, \pi[i] \models \Phi_2$
  and $\forall j, 0 \leq j < i : \mathsf{M}^w_\delta, \pi[j] \models \Phi_1$;

– $\mathsf{M}_\delta^w, \pi \models (\Phi_1 \; \mathsf{U}^{\leq k} \; \Phi_2)$ *iff* $\exists i, k \geq i \geq 0 : \mathsf{M}_\delta^w, \pi[i] \models \Phi_2$ *and* $\forall j, 0 \leq j < i : \mathsf{M}_\delta^w, \pi[j] \models \Phi_1$.

Definition 6 (Verification Problem). *A temporal state formula Φ is* valid *in a program $\mathcal{G} = (\mathcal{DDT}, \delta)$ iff for all worlds $w \in \mathcal{W}$ with $w \models \mathcal{DDT}$ it holds that $\mathsf{M}_\delta^w, \mathsf{s}^0 \models \Phi$.*

Example 2. Assume that due to the fact that the action may fail, the agent decides to simply execute the $\mathit{MoveS}(\mathit{box}, s_1, s_2)$ action repeatedly until the desired situation is reached where the unbroken vase is on shelf s_2:

$$\delta = \textbf{while } \neg\big(\mathit{On}(\mathit{vase}, s_2) \wedge \neg \mathit{Broken}(\mathit{vase})\big) \textbf{ do } \mathit{MoveS}(\mathit{box}, s_1, s_2) \textbf{ endWhile}$$

Temporal properties one might want to verify for this program expression could be whether it is very likely that this can be achieved within exactly one, at least k, or an arbitrary number of steps:

$$\mathbf{P}_{\geq 0.95}[\mathsf{X}\big(\mathit{On}(\mathit{vase}, s_2) \wedge \neg \mathit{Broken}(\mathit{vase})\big)] \tag{3}$$

$$\mathbf{P}_{\geq 0.95}[\mathsf{F}^{\leq k}\big(\mathit{On}(\mathit{vase}, s_2) \wedge \neg \mathit{Broken}(\mathit{vase})\big)] \tag{4}$$

$$\mathbf{P}_{\geq 0.95}[\mathsf{F}\big(\mathit{On}(\mathit{vase}, s_2) \wedge \neg \mathit{Broken}(\mathit{vase})\big)] \tag{5}$$

3 Decidability of Verification

We first note that in general:

Theorem 1. *The verification problem for* DTGolog *is undecidable.*

Proof (sketch). In [20] it is shown that given a two-counter machine M, a Golog program and BAT can be constructed where $\mathsf{EF}\mathit{Halt}$ is valid iff M halts, which is undecidable. Since regular Golog programs are a subset of DTGolog, and since the corresponding temporal property can be expressed as $\mathbf{P}_{]0,1]}[\mathsf{F}\mathit{Halt}]$ in PRCTL, we also get undecidability in the decision-theoretic case.

3.1 Fluent Dependencies and Acyclic Theories

One source of undecidability lies in cyclic dependencies between fluents in the effect descriptors of SSAs.

Definition 7 (Fluent Dependencies). *The* fluent dependency graph $G_\mathcal{D}$ *for a C^2-BAT $\mathcal{D}$ consists of a set of nodes, one for each fluent in $\mathcal{D}$. There is a directed edge (F, F') from fluent F to fluent F' iff there is a disjunct $\exists \boldsymbol{y}.\big(a = A(\boldsymbol{v}) \wedge \phi \wedge \phi'\big)$ in γ_F^+ or γ_F^- such that F' occurs in the effect descriptor ϕ. We call $\mathcal{D}$* acyclic *iff $G_\mathcal{D}$ is acyclic. The* fluent depth *of an acyclic action theory $\mathcal{D}$, denoted by $\mathsf{fd}(\mathcal{D})$, is the length of the longest path in $G_\mathcal{D}$. The* fluent depth *of F* w.r.t. *$\mathcal{D}$, $\mathsf{fd}_\mathcal{D}(F)$, is the length of the longest path in $G_\mathcal{D}$ starting in F.*

While the BAT used in the construction for the undecidability proof has a cyclic dependency graph, the one for Example 1 is acyclic (with fluent depth 2), as shown in Figure 3. Note that only effect descriptors are relevant. Important special cases of acyclic action theories are the *local-effect* ones [18] (corresponding to fluent depth 0) and the *context-free* [13] (fluent depth 1).

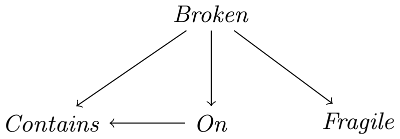


Fig. 3. Example fluent dependencies

3.2 Decidable Verification with Acyclic Theories

Let us now restrict our attention to programs over ground actions with an acyclic C^2-DTBAT $\mathcal{DDT}$. Let $\mathcal{A}$ denote the finite set of ground deterministic actions (including ϵ and $\mathfrak{f}$) occurring in δ_{det}. The goal is to construct a finite propositional abstraction of the infinite-state MDP M_δ^w with $w \models \mathcal{DDT}$. Following the construction for Golog programs presented in [20] and elaborated in [19], the essential part is a compact representation of effects from executing a *sequence* of such ground actions in a given world satisfying the BAT.

First we simplify SSAs as follows. If $F(\boldsymbol{x})$ is a fluent and $t \in \mathcal{A}$, the *grounding* of the SSA of F w.r.t. t is of the form

$$\square[t]F(\boldsymbol{x}) \equiv \left(\gamma_F^+\right)_t^a \vee F(\boldsymbol{x}) \wedge \neg\left(\gamma_F^-\right)_t^a.$$

The instantiated positive and negative effect conditions $\left(\gamma_F^+\right)_t^a$ and $\left(\gamma_F^-\right)_t^a$ then are each equivalent to a disjunction

$$\phi_1^{\mathsf{eff}} \wedge \phi_1^{\mathsf{con}} \vee \cdots \vee \phi_n^{\mathsf{eff}} \wedge \phi_n^{\mathsf{con}}$$

for some $n \geq 0$, where the ϕ_i^{eff} (effect descriptors) are C^2-fluent formulas with $\boldsymbol{x}$ as their only free variables, and the ϕ_i^{con} (context conditions) are C^2-fluent sentences. We often view $\left(\gamma_F^+\right)_t^a$ and $\left(\gamma_F^-\right)_t^a$ as sets and write $(\phi_i^{\mathsf{eff}}, \phi_i^{\mathsf{con}}) \in \left(\gamma_F^+\right)_t^a$ to express that the corresponding disjunct is present. An *effect function* then represents the effects of a ground action:

Definition 8 (Effects). *Let $F(\boldsymbol{x})$ be a fluent and ϕ a C^2-fluent formula with free variables $\boldsymbol{x}$, where $\boldsymbol{x}$ is empty or $\boldsymbol{x} = x$ or $\boldsymbol{x} = (x, y)$. We call the expression $\langle F^+, \phi\rangle$ a* positive effect on F*, and the expression $\langle F^-, \phi\rangle$ a* negative effect on F*. We use the notation $\langle F^\pm, \phi\rangle$ for an effect if we do not explicitly distinguish between a positive or a negative effect on F. Let $\mathcal{D}$ be a C^2-BAT, w a world with $w \models \mathcal{D}$, $z \in \mathcal{Z}$ and $t \in \mathcal{A}$. The* effects of executing t in (w, z) *are defined as:*

$$\begin{aligned}
&\mathcal{E}_{\mathcal{D}}(w, z, t) := \\
&\{\langle F^+, \phi^{\mathsf{eff}}\rangle \mid \exists(\phi^{\mathsf{eff}}, \phi^{\mathsf{con}}) \in \left(\gamma_F^+\right)_t^a \textit{ s. t. } w, z \models \phi^{\mathsf{con}}\} \cup \\
&\{\langle F^-, \phi^{\mathsf{eff}}\rangle \mid \exists(\phi^{\mathsf{eff}}, \phi^{\mathsf{con}}) \in \left(\gamma_F^-\right)_t^a \textit{ s. t. } w, z \models \phi^{\mathsf{con}}\}.
\end{aligned}$$

Intuitively, if $\langle F^+, \phi\rangle \in \mathcal{E}_{\mathcal{D}}(w, z, t)$ and $\boldsymbol{c}$ is an instance of ϕ *before* executing t in w, z, then $F(\boldsymbol{c})$ will be true *after* the execution (similar for negative effects).

To accumulate effects of consecutively executed actions, we define a regression operator applied to a C^2-fluent formula given a set of effects. Without loss of generality we assume that only variable symbols x and y occur.

Definition 9 (Regression). *Let E be a set of effects and φ a C^2-fluent formula. The* regression of φ through E*, denoted by $\mathcal{R}[\mathsf{E}, \varphi]$, is a C^2-fluent formula obtained from φ by replacing each occurrence of a fluent $F(\boldsymbol{v})$ in φ by the formula*

$$F(\boldsymbol{v}) \wedge \bigwedge_{\langle F^-,\phi\rangle\in\mathsf{E}} \neg\phi^{\boldsymbol{x}}_{\boldsymbol{v}} \vee \bigvee_{\langle F^+,\phi\rangle\in\mathsf{E}} \phi^{\boldsymbol{x}}_{\boldsymbol{v}}.$$

By appropriately renaming variables in the effect descriptors ϕ it can be ensured that $\mathcal{R}[\mathsf{E}, \varphi]$ is again a C^2-fluent sentence.

The result of first executing effects E_0 and afterwards E_1 is a new set of effects $\mathsf{E}_0 \rhd \mathsf{E}_1$ given by:

$$\begin{aligned}&\{\langle F^{\pm}, \mathcal{R}[\mathsf{E}_0, \varphi]\rangle \mid \langle F^{\pm}, \varphi\rangle \in \mathsf{E}_1\} \cup \\ &\{\langle F^{+}, \big(\varphi \wedge \bigwedge_{\langle F^-,\varphi'\rangle\in\mathsf{E}_1} \neg\mathcal{R}[\mathsf{E}_0, \varphi']\big)\rangle \mid \langle F^{+}, \varphi\rangle \in \mathsf{E}_0\} \ \cup \ \{\langle F^{-}, \varphi\rangle \in \mathsf{E}_0\}.\end{aligned}$$

It can be shown that for any C^2-fluent sentence ϕ,

$$\mathcal{R}[\mathsf{E}_0, \mathcal{R}[\mathsf{E}_1, \phi]] \equiv \mathcal{R}[\mathsf{E}_0 \rhd \mathsf{E}_1, \phi].$$

Let w be a world with $w \models \mathcal{D}$. To accumulate the effects of a sequence $z = t_1 t_2 \cdots t_n \in \mathcal{A}^*$ of deterministic actions into a single set, let $z[i]$ denote the subsequence of the first $i \leq n$ elements of z. Then we set

$$\begin{aligned}&\mathsf{E}_1 := \mathcal{E}_{\mathcal{D}}(w, \langle\rangle, t_1)\\ &\mathsf{E}_i := \mathsf{E}_{i-1} \rhd \mathcal{E}_{\mathcal{D}}(w, z[i-1], t_i) \text{ for } i = 2, \ldots, n\end{aligned}$$

and say that E_n *is generated by executing $t_1 t_2 \cdots t_n$ in w*. Then, for the effects E_z generated by z in w and a C^2-fluent sentence ψ, it holds that

$$w, z \models \psi \text{ iff } w, \langle\rangle \models \mathcal{R}[\mathsf{E}_z, \psi].$$

For a given DTGolog program $\mathcal{G} = (\mathcal{DDT}, \delta)$ with an acyclic BAT $\mathcal{D}$ and finitely many deterministic ground actions $\mathcal{A}$ occurring in δ_{det} we show that there are only finitely many possible effects that can be generated by action sequences from $\mathcal{A}$. We observe that for an effect $\langle F^{\pm}, \varphi\rangle$ on fluent F with depth $\mathsf{fd}_{\mathcal{D}}(F) = i$ all fluents occurring in φ have a depth that is strictly smaller than i. Thus, for regressing the effect descriptor φ only effects on fluents with depth strictly smaller than i are relevant. Using this argument we can define the set of all relevant effects as follows: For a fluent F the set of all positive effect descriptors for F are given by

$$\mathsf{eff}^{+}_{\mathcal{A}}(F) := \{\phi^{\mathsf{eff}} \mid (\phi^{\mathsf{eff}}, \phi^{\mathsf{con}}) \in \big(\gamma^{+}_{F}\big)^{a}_{t} \text{ for some } t \in \mathcal{A}\},$$

and analogous for the negative effect descriptors $\mathsf{eff}^{-}_{\mathcal{A}}(F)$. For an acyclic BAT $\mathcal{D}$ and finite set of ground actions $\mathcal{A}$ the *set of all relevant effects* on all fluents with depth $\leq j$ with $j = 0, \ldots, \mathsf{fd}(\mathcal{D})$ is denoted by $\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{j}$ and is given in Figure 4. We define $\mathfrak{E}^{\mathcal{D},\mathcal{A}} := \mathfrak{E}^{\mathcal{D},\mathcal{A}}_{n}$ with $\mathsf{fd}(\mathcal{D}) = n$. For a given fluent F with $\mathsf{fd}_{\mathcal{D}}(F) = 0$ it holds that the effects on F can be described without referring to any other fluent. Consequently, all effects on F generated by a ground action sequence from $\mathcal{A}$ must be contained in $\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{0}$. For fluents F with $\mathsf{fd}_{\mathcal{D}}(F) = i$ and $i > 0$ the fluents in the effect descriptors may also be subject to changes but have a depth strictly smaller than i. To obtain all relevant effects on F it is therefore sufficient to consider the effects in $\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{i-1}$.

$$
\begin{aligned}
\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{0} :=\, & \{\langle F^{-}, \varphi\rangle \mid \mathsf{fd}_{\mathcal{D}}(F) = 0, \varphi \in \mathsf{eff}^{-}_{\mathcal{A}}(F)\} \cup \\
& \{\langle F^{+}, \varphi \wedge \bigwedge_{\varphi' \in X} \neg\varphi'\rangle \mid \mathsf{fd}_{\mathcal{D}}(F) = 0, \varphi \in \mathsf{eff}^{+}_{\mathcal{A}}(F),\, X \subseteq \mathsf{eff}^{-}_{\mathcal{A}}(F)\}; \\
\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{i} :=\, & \mathfrak{E}^{\mathcal{D},\mathcal{A}}_{i-1} \cup \{\langle F^{-}, \mathcal{R}[\mathsf{E}, \varphi]\rangle \mid \mathsf{fd}_{\mathcal{D}}(F) = i, \varphi \in \mathsf{eff}^{-}_{\mathcal{A}}(F),\, \mathsf{E} \in 2^{\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{i-1}}\} \cup \\
& \{\langle F^{+}, \Xi\rangle \mid \mathsf{fd}_{\mathcal{D}}(F) = i, \phi \in \mathsf{eff}^{+}_{\mathcal{A}}(F), \mathsf{E} \in 2^{\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{i-1}},\, X \subseteq \mathsf{eff}^{-}_{\mathcal{A}}(F) \times 2^{\mathfrak{E}^{\mathcal{D},\mathcal{A}}_{i-1}}\} \\
& \text{with } \Xi := \big(\mathcal{R}[\mathsf{E}, \phi] \wedge \bigwedge_{(\varphi, \mathsf{E}') \in X} \neg\mathcal{R}[\mathsf{E}', \varphi]\big)
\end{aligned}
$$

Fig. 4. Sets of all relevant effects with $1 \leq i \leq \mathsf{fd}(\mathcal{D})$

Lemma 1. *Let $\mathcal{D}$ and $\mathcal{A}$ be as above, $z \in \mathcal{A}^{*}$, $w \models \mathcal{D}$ and E_{z} the effects generated by executing z in w. For each $\langle F^{\pm}, \varphi\rangle \in \mathsf{E}_{z}$ there exists $\langle F^{\pm}, \varphi'\rangle \in \mathfrak{E}^{\mathcal{D},\mathcal{A}}$ with $\varphi \equiv \varphi'$.*

Using the finite representation of action effects we can construct a finite abstraction of the infinite-state MDP induced by a program with a C^2-DTBAT and an acyclic $\mathcal{D}$. First, we identify a finite set of *relevant C^2-fluent sentences* called *context of a program*, denoted by $\mathcal{C}(\mathcal{G})$. It consists of

- all sentences in the initial theory,
- all context conditions in the instantiated SSAs,
- all instantiations ψ^{r}_{c} of the right-hand side of axiom $\mathcal{D}_{\text{reward}}$ for all occurring numeric constants c,
- all C^2-fluent subformulas in the temporal property, and
- all tests in the program.

Furthermore, the context is closed under negation.

Central for the abstraction is the notion of a *type of a world*, representing an equivalence class over $\mathcal{W}$. Intuitively, a type says which of the context axioms are satisfied initially and in all relevant future situations of that world.

Definition 10 (Types). *Let $\mathcal{G} = (\mathcal{DDT}, \delta)$ be a* DTGOLOG *program with an acyclic BAT $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_{post}$ w.r.t. a finite set of ground actions $\mathcal{A}$ (including ϵ and $\mathfrak{f}$). Furthermore, let $\mathcal{C}(\mathcal{G})$ be the context of $\mathcal{G}$ and $\mathfrak{E}^{\mathcal{D},\mathcal{A}}$ the set of all relevant effects. The* set of all type elements *is given by*

$$\mathsf{TE}(\mathcal{G}) := \{(\psi, \mathsf{E}) \mid \psi \in \mathcal{C}(\mathcal{G}), \mathsf{E} \subseteq \mathfrak{E}^{\mathcal{D},\mathcal{A}}\}.$$

A type w.r.t. $\mathcal{G}$ *is a set $\tau \subseteq \mathsf{TE}(\mathcal{G})$ that satisfies:*

1. *For all $\psi \in \mathcal{C}(\mathcal{G})$ and all $\mathsf{E} \subseteq \mathfrak{E}^{\mathcal{D},\mathcal{A}}$ either $(\psi, \mathsf{E}) \in \tau$ or $(\neg\psi, \mathsf{E}) \in \tau$.*
2. *There exists a world $w \in \mathcal{W}$ such that*

$$w \models \mathcal{D}_0 \cup \{\mathcal{R}[\mathsf{E}, \psi] \mid (\psi, \mathsf{E}) \in \tau\}.$$

The set of all types w.r.t. $\mathcal{G}$ *is denoted by* $\mathsf{Types}(\mathcal{G})$*. The* type of a world $w \in \mathcal{W}$ w.r.t. $\mathcal{G}$ *is given by*

$$\mathsf{type}(w) := \{(\psi, \mathsf{E}) \in \mathsf{TE}(\mathcal{G}) \mid w \models \mathcal{R}[\mathsf{E}, \psi]\}.$$

The abstraction of a world state consisting of a world $w \in \mathcal{W}$ with $w \models \mathcal{DDT}$ and an action sequence $z \in \mathcal{A}^*$ is then given by $\mathsf{type}(w)$ and the set of effects $\mathsf{E}_z \subseteq \mathfrak{E}^{\mathcal{D},\mathcal{A}}$ generated by executing z in w. Furthermore, the program only admits finitely many control states. Here we use a representation similar to the *characteristic program graphs* from [3] where nodes are the reachable subprograms $\mathit{Sub}(\delta)$, each of which is associated with a termination condition $\mathsf{Fin}(\delta')$, and where an edge $\delta_1 \xrightarrow{t/\psi} \delta_2$ represents a transition from δ_1 to δ_2 via action t if test condition ψ holds. Moreover, failure conditions are given by

$$\mathsf{Fail}(\delta') := \neg\big(\mathsf{Fin}(\delta') \vee \bigvee_{\delta' \xrightarrow{t/\psi} \delta''} \psi\big).$$

The abstract, finite MDP for a type τ can then be constructed using the Cartesian product of effect sets and subprograms as states, the same actions as in the original MDP, and the context formulas as labels. Formally, $\mathsf{M}^{\tau}_{\delta\,\mathsf{fin}} = \langle \mathsf{S}_{\mathsf{fin}}, \mathsf{s}^0_{\mathsf{fin}}, \mathsf{A}_{\mathsf{fin}}, \mathsf{P}_{\mathsf{fin}}, \mathsf{R}_{\mathsf{fin}}, \mathsf{L}_{\mathsf{fin}} \rangle$ consists of

- the set of states $\mathsf{S}_{\mathsf{fin}} = 2^{\mathfrak{E}^{\mathcal{D},\mathcal{A}}} \times \mathit{Sub}(\delta_{\mathsf{det}})$;
- the initial state $\mathsf{s}^0_{\mathsf{fin}} = \langle \emptyset, \delta_{\mathsf{det}} \rangle$;
- the set of actions $\mathsf{A}_{\mathsf{fin}} = \mathsf{A}$;
- the transition function $\mathsf{P}_{\mathsf{fin}}$ such that

$$\mathsf{P}_{\mathsf{fin}}(\langle \mathsf{E}_1, \delta_1 \rangle, t, \langle \mathsf{E}_2, \delta_2 \rangle) = \begin{cases} c, & \mathcal{D}_{\text{prob}} \models \mathit{Prob}(t, t', c), \\ & \delta_1 \xrightarrow{t'/\psi} \delta_2,\ (\psi, \mathsf{E}_1) \in \tau, \\ & \mathsf{E}_2 = \mathsf{E}_1 \rhd \mathcal{E}_{\mathcal{D}}(\tau, \mathsf{E}_1, t') \\ 1, & (\mathsf{Fin}(\delta_1), E_1) \in \tau,\ t = t' = \delta_2 = \epsilon \\ 1, & (\mathsf{Fail}(\delta_1), E_1) \in \tau,\ t = t' = \delta_2 = \mathfrak{f} \\ 0, & \text{otherwise} \end{cases}$$

and all $\langle E, \epsilon \rangle$ as well as all $\langle E, \mathfrak{f} \rangle$ are absorbing states;

	$\varPhi_1$	$\varPhi_{\leq 1}$	$\varPhi_{\leq 2}$	$\varPhi_{\leq 3}$	$\varPhi_{\infty}$
with bubble wrap	false	false	true	true	true
without bubble wrap	false	false	false	false	false

Table 1. Verification Results for Example Properties

- the reward function $\mathsf{R}_{\mathsf{fin}}$ such that $\mathsf{R}_{\mathsf{fin}}(\langle \mathsf{E}_1, \delta_1 \rangle) = c$ iff $(\psi_c^r, \mathsf{E}_1) \in \tau$;
- and the labeling function $\mathsf{L}_{\mathsf{fin}}(\langle \mathsf{E}_1, \delta_1 \rangle) = \{\psi \in \mathcal{C}(\mathcal{G}) \mid (\psi, \mathsf{E}_1) \in \tau\}$.

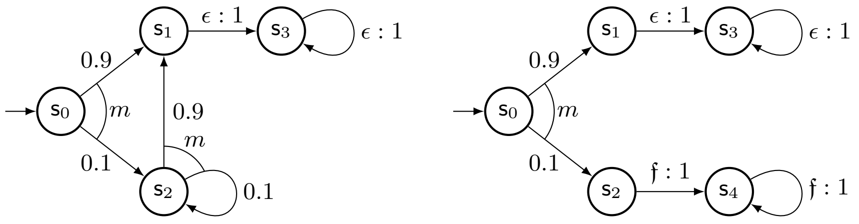
We can thus regard the finitely many context formulas as atomic propositions, and hence apply propositional probabilistic model checking. The finitely many world types can be computed using a decidable consistency check in C^2, so this yields a decision procedure for the verification problem:

Theorem 2. *Let $\mathcal{G} = (\mathcal{DDT}, \delta)$ be a* DTGolog *program with an acyclic C^2-BAT and $\varPhi$ a temporal state formula. It is decidable to verify whether $\varPhi$ is valid in $\mathcal{G}$.*

Example 3. In our running example we obtain two types, one for the case that the box contains bubble wrap and one where it does not. This is due to the fact that our initial theory (Fig. 1) does not say anything about the truth of the context condition $\neg\exists y.\mathit{Contains}(\mathit{box}, y) \wedge \mathit{BubbleWrap}(y)$ for the *Drop* action in $\gamma^{+}_{\mathit{Broken}}$ (Fig. 2).

The corresponding abstract MDPs are depicted in Figures 5(a) and 5(b), respectively, where m stands for the ground action $\mathit{MoveS}(\mathit{box}, s_1, s_2)$. That is to say when there is bubble wrap, a successful attempt of moving the box leads to state s_1, from where only successful termination of the program is possible, represented by entering absorbing state s_3. Should the box be dropped, state s_2 is entered, and m may be retried indefinitely until it succeeds. On the other hand, if the box does not contain any bubble wrap, the agent only has one attempt. Should it fail, absorbing state s_4 is reached, representing program failure.

We can now feed these finite MDPs into a probabilistic model checker such as STORM [4] in order to verify (the propositionalized versions of) the example properties. Table 1 shows the corresponding results, where $\varPhi_1$ stands for formula (3), $\varPhi_{\leq k}$ for (4) with $k \in \{1, 2, 3\}$, and $\varPhi_{\infty}$ for (5). None of the properties holds in both types, i.e. none is valid. We can see that in order to obtain a 95% certainty that the unbroken vase ends up on shelf s_2, we need to allow for at least two move attempts (hence bubble wrap is required). Intuitively, this is because the first one only has a 90% chance to succeed, but with two attempts we already get $0.9 + 0.1 \cdot 0.9 = 99\%$ success probability, 99.9% with three, and so on. The desired situation is thus reached eventually "almost surely", meaning with a 100% probability.



(a) with bubble wrap

(b) without bubble wrap

Fig. 5. Example Abstract MDPs

4 Conclusion

In this paper we lifted recent results on the decidability of verification of temporal properties of classical Golog programs to the decision-theoretic case. The class of acyclic theories is very expressive in the sense that it subsumes many of the popular classes, including the context-free and local-effect ones. Our result not only enables us to employ recent advances in probabilistic model checking [6, 9, 4] for the verification of DTGolog agents, variants of which have been used e.g. for controlling soccer robots [5]. Our abstraction, which can be performed as a preprocessing step, also opens the application range of methods normally working on finite MDPs to a large class of infinite-state problems.

Acknowledgments. This work was supported by the German Research Foundation (DFG) research unit FOR 1513 on Hybrid Reasoning for Intelligent Systems, project A1.

References

1. Andova, S., Hermanns, H., Katoen, J.: Discrete-time rewards model-checked. In: Larsen, K.G., Niebert, P. (eds.) Proceedings of the First International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003). Lecture Notes in Computer Science, vol. 2791, pp. 88–104. Springer (2003)
2. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-theoretic, high-level agent programming in the situation calculus. In: Kautz, H., Porter, B. (eds.) Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000). pp. 355–362. AAAI Press (2000)
3. Claßen, J., Lakemeyer, G.: A logic for non-terminating Golog programs. In: Brewka, G., Lang, J. (eds.) Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008). pp. 589–599. AAAI Press (2008)
4. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A Storm is coming: A modern probabilistic model checker. In: Kuncak, V., Majumdar, R. (eds.) Proceedings of the Twentyninth International Conference on Computer Aided Verification (CAV 2017). Springer (2017)

5. Ferrein, A., Lakemeyer, G.: Logic-based robot control in highly dynamic domains. Robotics and Autonomous Systems (2008)
6. Forejt, V., Kwiatkowska, M.Z., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Bernardo, M., Issarny, V. (eds.) Formal Methods for Eternal Networked Software Systems - Eleventh International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM 2011). Lecture Notes in Computer Science, vol. 6659, pp. 53–113. Springer (2011)
7. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
8. Kemeny, J.G., Snell, J.L., Knapp, A.W.: Denumerable Markov Chains, Graduate Texts in Mathematics, vol. 40. Springer (1976)
9. Kwiatkowska, M., Parker, D.: Advances in probabilistic model checking. In: Nipkow, T., Grumberg, O., Hauptmann, B. (eds.) Software Safety and Security - Tools for Analysis and Verification, NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 33, pp. 126–151. IOS Press (2012)
10. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proceedings of the Twentythird International Conference on Computer Aided Verification (CAV 2011). pp. 585–591. Springer (2011)
11. Lakemeyer, G., Levesque, H.J.: A semantic characterization of a useful fragment of the situation calculus with knowledge. Artificial Intelligence 175(1), 142–164 (2010)
12. Levesque, H.J., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.B.: GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming 31(1–3), 59–83 (1997)
13. Lin, F., Reiter, R.: How to progress a database. Artificial Intelligence 92(1–2), 131–167 (1997)
14. Pednault, E.P.D.: Synthesizing plans that contain actions with context-dependent effects. Computational Intelligence 4, 356–372 (1988)
15. Puterman, M.L.: Markov decision processes: Discrete stochastic dynamic programming. John Wiley & Sons (1994)
16. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press (2001)
17. Soutchanski, M.: An on-line decision-theoretic Golog interpreter. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001). pp. 19–26. Morgan Kaufmann Publishers Inc. (2001)
18. Vassos, S., Lakemeyer, G., Levesque, H.J.: First-order strong progression for local-effect basic action theories. In: Brewka, G., Lang, J. (eds.) Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008). pp. 662–672. AAAI Press (2008)
19. Zarrieß, B., Claßen, J.: Decidable verification of Golog programs over non-local effect actions. LTCS-Report 15-19, Chair of Automata Theory, TU Dresden, Dresden, Germany (2015)
20. Zarrieß, B., Claßen, J.: Decidable verification of Golog programs over non-local effect actions. In: Schuurmans, D., Wellman, M. (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016). pp. 1109–1115. AAAI Press (2016)