Artificial Intelligence 174 (2010) 865–888

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Property persistence in the situation calculus

Ryan F. Kelly, Adrian R. Pearce *

Department of Computer Science and Software Engineering, The University of Melbourne, Victoria 3010, Australia

ARTICLE INFO

Article history:
Received 25 May 2009
Received in revised form 6 May 2010
Accepted 6 May 2010
Available online 8 May 2010

Keywords:
Situation calculus
Automated reasoning
Property persistence

ABSTRACT

We develop a new automated reasoning technique for the situation calculus that can handle a class of queries containing universal quantification over situation terms. Although such queries arise naturally in many important reasoning tasks, they are difficult to automate in the situation calculus due to the presence of a second-order induction axiom. We show how to reduce queries about property persistence, a common type of universally-quantified query, to an equivalent form that does not quantify over situations and so is amenable to existing reasoning techniques. Our algorithm replaces induction with a meta-level fixpoint calculation; crucially, this calculation uses only first-order reasoning with a limited set of axioms. The result is a powerful new tool for verifying sophisticated domain properties in the situation calculus.

1. Introduction

The situation calculus is one of the most popular and influential AI formalisms for reasoning about action and change, having found application in a wide variety of both theoretical and practical works [5,6,9,26,27,30]. A major contributor to the success of the formalism is that it combines a powerful modelling language built on first-order logic with easily implementable techniques for effective automated reasoning.

A key challenge when working with the situation calculus is managing this balance between expressivity and effectiveness. An induction axiom is used to define the structure of situation terms, so answering arbitrary queries requires reasoning in second-order logic. While certain special cases are known to be decidable [31], such reasoning is prohibitively expensive in general [24].

If queries are restricted to certain syntactic forms, it is possible to obtain much more effective reasoning procedures – for example, queries restricted to existential quantification over situations can be answered using only first-order logic [23], while queries containing only ground situation terms permit special-purpose techniques such as regression [26].

However, there are many important reasoning tasks that require *universal* quantification over situations, for which the situation calculus currently offers no effective reasoning tools. One simple example is the problem of goal impossibility – establishing that all possible situations fail to satisfy a goal. In this paper we study a subset of universally-quantified queries which we refer to as *property persistence queries*: under a particular situation calculus theory $\mathcal{D}$, and given some formula ϕ and situation σ, determine whether ϕ will hold in all situations in the future of σ:

$$\mathcal{D} \models \forall s\colon \sigma \sqsubseteq s \rightarrow \phi[s]$$

The need for second-order logic has traditionally limited automated reasoning about such queries. We introduce a new approach to property persistence that is similar in spirit to the standard regression operator, by defining a meta-level

* Corresponding author. Tel.: +613 8344 1399; fax: +613 9348 1184.
E-mail addresses: rfk@csse.unimelb.edu.au (R.F. Kelly), adrianrp@unimelb.edu.au (A.R. Pearce).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.05.003

operator $\mathcal{P}_{\mathcal{D}}$ such that ϕ persists at σ if and only if $\mathcal{P}_{\mathcal{D}}(\phi)$ holds at σ. We term the resulting formula the *persistence condition* of ϕ and show how to calculate it as a fixpoint of applications of an operator based on regression; crucially, this calculation requires only first-order logic and a limited set of axioms. The persistence condition is also guaranteed to be in a form amenable to existing automated reasoning techniques.

Importantly, our results do not require restrictions on the domain theory $\mathcal{D}$ – they are generally applicable to the full first-order situation calculus, and are based purely on standard first-order reasoning techniques.

The result is a powerful new technique for exploring sophisticated domain properties in the situation calculus. It allows some second-order aspects of the theory to be "factored out" and handled using a special-purpose algorithm. The technique is always sound, and we show that it is complete for important standard variants of the situation calculus. Perhaps most importantly, it builds upon and integrates well with standard techniques for effective automated reasoning, so our technique is directly applicable to existing theories and systems based on the situation calculus.

A preliminary version of this paper has previously appeared as [15]; this revised edition includes extended and additional proofs, a more comprehensive discussion of the termination properties of our algorithm, and a detailed example of how the persistence condition can be used to reason about goal impossibility – a deceptively simple task which is nonetheless beyond the reach of existing reasoning techniques.

The paper now proceeds with a brief review of the situation calculus, before formally defining the persistence condition and establishing its effectiveness as a reasoning tool. Readers familiar with the situation calculus are encouraged to review the background material in Sections 2 and 4, as we make several small modifications to the standard notation that greatly simplify the development of our approach: the unique names axioms $\mathcal{D}_{una}$ are incorporated into a general background theory $\mathcal{D}_{bg}$; the *Poss* fluent is subsumed by a general class of *action description predicates* defined in $\mathcal{D}_{ad}$; we parameterise the "future situations" predicate $s \sqsubset s'$ to assert that all intermediate actions satisfy a given predicate using $s <_{\alpha} s'$; and we use the single-step variant of the regression operator, with corresponding definitions of regressable formulae.

2. The situation calculus

The situation calculus is a powerful formalism for describing and reasoning about dynamic worlds. It was first introduced by McCarthy and Hayes [22] and has since been significantly expanded and formalised [23,26]. We use the particular variant due to Reiter et al. at the University of Toronto, sometimes called the "Toronto school" or "situations-as-histories" version. The formalisation below is based on the standard definitions from [16,23,25], with some simple modifications.

The language $\mathcal{L}_{sitcalc}$ of the situation calculus is a many-sorted language of second-order logic with equality, containing the following disjoint sorts:

- ACTION terms denote individual instantaneous events that can cause the state of the world to change;
- SITUATION terms are histories of the actions that have occurred in the world, with the initial situation represented by S_0 and successive situations built using the function $do : Action \times Situation \rightarrow Situation$;
- OBJECT terms represent any other object in the domain.

Fluents are predicates representing properties of the world that may change between situations, and so take a situation term as their final argument. Predicates and functions that do not take a situation term are called *rigid*. We use the term *primitive fluent* to describe fluents that are directly affected by actions, rather than being defined in terms of other fluents. No functions other than S_0 and *do* produce values of sort SITUATION. For the sake of clarity we will not consider functional fluents in this paper; this is a common simplifying assumption in the situation calculus literature and does not result in a loss of generality.

$\mathcal{L}_{sitcalc}$ contains the standard alphabet of logical connectives, constants $\top$ and $\bot$, countably infinitely many variables of each sort, countably infinitely many predicates of each arity, etc.; for a complete definition, consult the foundational paper by Pirri and Reiter [23]. We follow standard naming conventions for the situation calculus: upper-case roman names indicate constants; lower-case roman names indicate variables; Greek characters indicate meta-variables or formula templates. All axioms universally close over their free variables at outermost scope. The notation $\bar{t}$ indicates a vector of terms of context-appropriate arity and type. The connectives $\wedge$, $\neg$, $\exists$ are taken as primitive, with $\vee$, $\rightarrow$, $\equiv$, $\forall$ defined in the usual manner.

Complex properties of the state of the world are represented using *uniform formulae*. These are basically logical combinations of fluents referring to a common situation term.

Definition 1 *(Uniform formulae).* Let σ be a fixed situation term, R_i an arbitrary rigid predicate, F_i an arbitrary primitive fluent predicate, τ_i an arbitrary term that is not of sort SITUATION, and x_i an arbitrary variable that is not of sort SITUATION. Then the formulae uniform in σ are the smallest set of syntactically-valid formulae satisfying:

$$\phi ::= F_i(\bar{\tau_i}, \sigma) \mid R_i(\bar{\tau_i}) \mid \tau_i = \tau_j \mid \phi_i \wedge \phi_j \mid \neg\phi \mid \exists x_i.\ \phi$$

We will call a formula *uniform* if it is uniform in some situation. The important aspect of this definition is that the formula refers to no situation other than σ, which appears as the final argument of all fluents in the formula. In particular, uniform formulae cannot quantify over situations or compare situation terms, and cannot contain non-primitive fluents.

The meta-variable ϕ is used throughout to refer to an arbitrary uniform formula. The notation $\phi[s']$ represents a uniform formula with the particular situation s' inserted into all its fluents, replacing whatever situation term was previously there. Note that this is simply a syntactic shorthand designed to keep the presentation clean and readable – it is *not* an operation from the logic itself.

The dynamics of a particular domain are captured by a set of sentences called a *basic action theory*. Queries about the behaviour of the world are posed as logical entailment queries relative to this theory.

Definition 2 *(Basic action theory).* A basic action theory, denoted $\mathcal{D}$, is a set of situation calculus sentences (of the specific syntactic form outlined below) describing a particular dynamic world. It consists of the following disjoint sets: the foundational axioms of the situation calculus (Σ); action description axioms defining various aspects of action performance, such as preconditions ($\mathcal{D}_{ad}$); successor state axioms describing how primitive fluents change between situations ($\mathcal{D}_{ssa}$); axioms describing the value of primitive fluents in the initial situation ($\mathcal{D}_{S_0}$); and axioms describing the static background facts of the domain ($\mathcal{D}_{bg}$):

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ad} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{bg}$$

These axioms must satisfy some simple consistency criteria in order to constitute a valid domain description [23]. This is a straightforward modification of the standard definition of a basic action theory, intended to simplify the details of our forthcoming development.

The axiom set $\mathcal{D}_{S_0}$ is a collection of sentences uniform in S_0 that describe the initial state of the world, while the set $\mathcal{D}_{bg}$ contains all the situation-independent facts about the domain. Standard notation includes situation-independent facts in $\mathcal{D}_{S_0}$, but our upcoming definitions require that they be separate. $\mathcal{D}_{bg}$ includes the standard unique names axioms for actions [23].

The axiom set $\mathcal{D}_{ssa}$ contains one *successor state axiom* for each primitive fluent in the domain, providing a monotonic solution to the frame problem for that fluent. They have the following form, where Φ is uniform in s:

$$F\big(\bar{x}, do(a, s)\big) \equiv \Phi_F(\bar{x}, a, s)$$

The axiom set $\mathcal{D}_{ad}$ defines fluents that describe various aspects of the performance of an action, which we call *action description predicates*. For each such predicate $ADP(\bar{x}, a, s)$ the set $\mathcal{D}_{ad}$ contains a single axiom of the following form, where Π_{ADP} is uniform in s:

$$ADP(\bar{x}, a, s) \equiv \Pi_{ADP}(\bar{x}, a, s)$$

The canonical example of an action description predicate is the precondition predicate $Poss(a, s)$, which indicates whether it is possible to perform an action in a given situation. In principle there can be any number of predicates or functions defined in a similar way – a common example is the function SR used to axiomatise sensing actions in [29]. We will henceforth use the meta-predicate α to denote an arbitrary action description predicate.

Note that this is a departure from the standard notion, where a separate axiom specifies the preconditions for each function of sort Action [23]. The single-axiom approach used here embodies a domain closure assumption on the Action sort, and is necessary when reasoning about formulae that universally quantify over actions [28,34].

We will sometimes write the definition of an action description predicate in terms of other previously-defined action description predicates. This is purely a notational convenience; the definitions in $\mathcal{D}_{ad}$ must use primitive fluents only.

The foundational axioms Σ ensure that situations form a branching-time account of the world state. There is a distinguished situation term S_0 called the *initial situation*, and situations in general form a tree structure with S_0 at the root and $do(a, s)$ constructing the situation that results from performing action a in situation s. We abbreviate the performance of several successive actions by writing:

$$do\big([a_1 \ldots a_n], s\big) \stackrel{\text{def}}{=} do\big(a_n, do\big(\ldots, do(a_1, s)\big)\big)$$

The relation $s \sqsubset s'$ indicates that s' is in the future of s:

$$\neg(s \sqsubset S_0)$$
$$s \sqsubset do\big(a, s'\big) \equiv s \sqsubseteq s'$$

Here $s \sqsubseteq s'$ is the standard abbreviation for $s \sqsubset s' \vee s = s'$. This notation for "in the future of" can be extended to consider only those futures in which all actions satisfy a particular action description predicate. We include a relation $<_\alpha$ for each action description predicate α, with the following definitions:

$$\neg(s <_\alpha S_0)$$
$$s <_\alpha do\big(a, s'\big) \equiv s \leqslant_\alpha s' \wedge \alpha\big(a, s'\big)$$

For example, by stating that $s <_{Poss} s'$ we assert that not only is s' in the future of s, but that all actions performed between s and s' were actually possible; this is equivalent to the $<$ operator of Pirri and Reiter [23]. Note that we suppress the action and situation arguments of the action description predicate in order to simplify the presentation.

Finally, a second-order induction axiom is used to assert that all situations must be constructed by performing a finite sequence of actions:

$$\forall P\colon \left[P(S_0) \wedge \forall s, a\colon \left(P(s) \rightarrow P\left(do(a, s)\right)\right)\right] \rightarrow \forall s\colon P(s)$$

This axiom is the only second-order sentence in a basic action theory, and is vital to the proper semantics of statements that universally quantify over situation terms [24].

3. Property persistence queries

With this notation in hand, let us now formally define the kinds of query that are of interest in this paper. Given some uniform formula ϕ and situation σ, a *property persistence query* asks whether ϕ will hold in all situations in the future of σ:

$$\mathcal{D} \models \forall s\colon \sigma \sqsubseteq s \rightarrow \phi[s]$$

More generally, one may wish to limit the futures under consideration to those brought about by actions satisfying a certain predicate α, which is easily accomplished using the $\leqslant_\alpha$ relation. We thus have the following definition of a property persistence query:

Definition 3 *(Property persistence query).* Let ϕ be a uniform formula, α an action description predicate, and σ a situation term. Then a property persistence query is a query of the form:

$$\mathcal{D} \models \forall s\colon \sigma \leqslant_\alpha s \rightarrow \phi[s]$$

If the query contains free variables, we treat them as universally quantified at outermost scope.

In words, a persistence query states that "ϕ holds in σ, and assuming all subsequent actions satisfy α, ϕ will continue to hold". For succinctness we will henceforth describe this as "ϕ persists under α". Such queries are involved in many useful reasoning tasks; the following are a small selection:

Goal impossibility. Given a goal G, establish that there is no legal situation in which that goal is achieved:

$$\mathcal{D} \models \forall s\colon S_0 \leqslant_{Poss} s \rightarrow \neg G(s)$$

Goal futility. Given a goal G and situation σ, establish that the goal cannot be achieved in any legal future of σ:

$$\mathcal{D} \models \forall s\colon \sigma \leqslant_{Poss} s \rightarrow \neg G(s)$$

Note how this differs from goal impossibility: while the goal may have initially been achievable, subsequent actions have rendered the goal unachievable. Detecting and avoiding such situations could be a very important task.

Checking state constraints. Given a state constraint SC, show that the constraint holds in every legal situation:

$$\mathcal{D} \models \forall s\colon S_0 \leqslant_{Poss} s \rightarrow SC(s)$$

This can be seen as a variant of goal impossibility, by showing that the constraint can never be violated.

Need for cooperation. Given an agent agt, goal G and situation σ, establish that no sequence of actions performed by that agent only can achieve the goal. Suppose we define *MyAction* to identify the agent's own actions:

$$MyAction(a, s) \stackrel{\text{def}}{=} actor(a) = agt$$

Then the appropriate query is:

$$\mathcal{D} \models \forall s\colon \sigma \leqslant_{MyAction} s \rightarrow \neg G(s)$$

If this is the case, the agent will need to seek cooperation from another agent in order to achieve its goal.

Knowledge with hidden actions. Consider an agent reasoning about its own knowledge in a multi-agent domain, where the other agents can perform "hidden" actions that it is unable to observe. To reason correctly in such asynchronous domains, the agent must take into account arbitrarily-long sequences of hidden actions [14,32].

For example, suppose an agent can only observe actions if they occur in the same room as it and the lights are on:

$$Hidden(a, s) \stackrel{\text{def}}{=} InSameRoom\big(actor(a), agt, s\big) \wedge LightsOn(s)$$

Intuitively, to establish that it knows ϕ the agent must establish that ϕ cannot become false through a sequence of hidden actions:

$$\mathcal{D} \models \forall s\colon\ \sigma \leqslant_{Hidden} s \rightarrow \phi[s]$$

Here $\sigma \leqslant_{Hidden} s$ denotes that a sequence of possible, but hidden, actions was performed by other agents between σ and s.

The "gold thief" domain. As a more detailed example, which we will revisit in Section 7, consider a domain in which a thief may try to steal some gold from a safe. There is a light in the room, and a security camera that will detect the thief's actions as long as the light is on. The safe can be open or closed, but the gold can only be stolen if the safe is open. It is possible for the thief to crack the safe and force it open, but only if the light is on.

The actions in this domain are *takeGold*, *crackSafe* and *toggleLight*, the primitive fluents are *SafeOpen*, *LightOn* and *Stolen*, and the action description predicates include the standard $Poss(a, s)$ and a custom predicate $Undet(a, s)$ indicating that action a would not be detected by the security camera. The complete axioms can be found in Appendix B.

As the owners of the gold, we would like to ensure that the thief cannot steal it. Unfortunately this is not possible, as nothing prevents him from simply cracking the safe and taking the gold. We can, however, ensure that the thief cannot steal the gold *undetected*. Formally, we want to establish that "no sequence of undetected actions results in the gold being stolen":

$$\mathcal{D} \models \forall s\colon\ S_0 \leqslant_{Undet} s \rightarrow \neg Stolen(s) \tag{1}$$

Intuitively, this will be the case as long as the gold is not already stolen, and either the light is on (so the thief's actions will be detected) or the safe is closed (so the thief must switch on the light to crack it):

$$\mathcal{D}_{bg} \cup \mathcal{D}_{S_0} \models \neg Stolen(S_0) \wedge \big[\neg SafeOpen(S_0) \vee LightOn(S_0)\big] \tag{2}$$

A manual proof that (1) iff (2) is straightforward, but it is beyond the reach of the standard automated reasoning tools of the situation calculus. The difficulty, as we shall explore in the next section, stems from the use of a second-order induction axiom to define the set of all situations.

4. Effective reasoning

An important feature of the situation calculus is the existence of effective reasoning procedures for certain types of query. In the general case, answering a query about a basic action theory $\mathcal{D}$ is a theorem-proving task in second-order logic (denoted SOL) due to the induction axiom:

$$\mathcal{D} \models_{SOL} \psi$$

This is clearly problematic for effective automated reasoning. Fortunately, restricting the syntactic form of queries can allow us to discard some axioms from $\mathcal{D}$ and make automated reasoning easier.

A core result of Pirri and Reiter [23] is that if a query performs only *existential* quantification over situations, then it can be answered without the induction axiom (denoted I) and thus using only first-order logic (FOL):

$$\mathcal{D} \models_{SOL} \exists s\colon\ \psi(s)$$

iff

$$\mathcal{D} - \{I\} \models_{FOL} \exists s\colon\ \psi(s)$$

While this is a substantial improvement over requiring a second-order theorem prover, it is still far from an effective technique. Effective reasoning requires that the set of axioms be reduced as much as possible.

In their work on state constraints, Lin and Reiter [19] show how to reduce the task of verifying a state constraint to a reasoning task we call *static domain reasoning*, where only the background axioms need to be considered:

$$\mathcal{D}_{bg} \models_{FOL} \forall s\colon\ \phi[s]$$

Since the axioms in $\mathcal{D}_{bg}$ do not mention situation terms, the leading quantification in such queries has no effect – ϕ will hold for all s if and only if it holds for some s. This is an important result because it is usually not valid to drop the

induction axiom for queries that universally quantify over situations. Their work has shown that this can be circumvented in some cases.

Simpler still are queries uniform in the initial situation, which can be answered using only first-order logic and a limited set of axioms:

$$\mathcal{D} \models_{SOL} \phi[S_0]$$
$$\text{iff}$$
$$\mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models_{FOL} \phi[S_0]$$

We call such reasoning *initial situation reasoning*. Since the axioms $\mathcal{D}_{S_0} \cup \mathcal{D}_{bg}$ often satisfy the closed-world assumption, provers such as Prolog can be employed to handle this type of query quite effectively.

While few useful queries happen to precisely match these restricted forms, it is possible to answer quite broad classes of query by *transforming* them into such a form. This insight is at the heart of the principle tool for effective reasoning in the situation calculus: regression.

4.1. Regression

The regression meta-operator $\mathcal{R}_{\mathcal{D}}$ is a syntactic manipulation that encodes the preconditions and effects of actions into the query itself, meaning fewer axioms are needed for the final reasoning task [23,26]. The idea is to reduce a query about some future situation to a query about the initial situation only, which is much easier to answer.

There are two styles of regression operator commonly defined in the literature: the single-pass operator as defined in [23] which reduces to S_0 in a single application, and the single-step operator as defined in [29] which operates one action at a time. We use the single-step variant as it is the more powerful of the two; the single-pass operator can only be applied to situations rooted at S_0, while the single-step operator can handle formulae containing situation variables.

Regression is only defined for formulae that are *regressable:*

Definition 4 *(Regressable formulae).* Let σ_i be an arbitrary situation term, x_i an arbitrary variable not of sort Situation, τ_i an arbitrary term not of sort Situation, a_i an arbitrary term of sort Action, R_i an arbitrary rigid predicate, F_i an arbitrary primitive fluent predicate, and α_i an arbitrary action description predicate. Then the regressable formulae are the smallest set of syntactically-valid formulae satisfying:

$$\varphi ::= F_i(\bar{\tau}_i, \sigma_i) \mid \alpha_i(\bar{\tau}_i, a_i, \sigma_i) \mid R_i(\bar{\tau}_i) \mid \tau_i = \tau_j \mid \neg\varphi \mid \varphi_i \wedge \varphi_j \mid \exists x_i{:}\ \varphi$$

Regressable formulae are more general than uniform formulae. In particular, they can contain action description predicates and may mention different situation terms. They cannot, however, quantify over situation terms or compare situations using the $\sqsubset$ predicate. Note also that our definition is more general than that of [23], where the single-pass regression operator is used.

The regression operator is defined using a series of *regression rules* such as those shown below, mirroring the structure of regressable formulae:

Definition 5 *(Regression operator).* Let R_i be an arbitrary rigid predicate, α_i be an arbitrary action description predicate with axiom $\alpha_i(\bar{\nu}, a, s) \equiv \Pi_\alpha(\bar{\nu}, a, s)$ in $\mathcal{D}_{ad}$, F_i be an arbitrary primitive fluent with axiom $F_i(\bar{x}, do(a, s)) \equiv \Phi_{F_i}(\bar{x}, a, s)$ in $\mathcal{D}_{ssa}$, τ_i be an arbitrary term not of sort Situation, s_i be an arbitrary variable of sort Situation, and a_i be an arbitrary term of sort Action. Then the regression of ϕ, denoted $\mathcal{R}_{\mathcal{D}}(\phi)$, is defined according to the following structural rules:

$$\mathcal{R}_{\mathcal{D}}(\varphi_i \wedge \varphi_j) \stackrel{\text{def}}{=} \mathcal{R}_{\mathcal{D}}(\varphi_i) \wedge \mathcal{R}_{\mathcal{D}}(\varphi_j)$$
$$\mathcal{R}_{\mathcal{D}}(\exists x_i{:}\ \varphi) \stackrel{\text{def}}{=} \exists x_i{:}\ \mathcal{R}_{\mathcal{D}}(\varphi)$$
$$\mathcal{R}_{\mathcal{D}}(\neg\varphi_i) \stackrel{\text{def}}{=} \neg\mathcal{R}_{\mathcal{D}}(\varphi_i)$$
$$\mathcal{R}_{\mathcal{D}}\big(\alpha_i(\bar{\tau}_i, a_i, \sigma_i)\big) \stackrel{\text{def}}{=} \mathcal{R}_{\mathcal{D}}\big(\Pi_\alpha(\bar{\tau}_i, a_i, \sigma_i)\big)$$
$$\mathcal{R}_{\mathcal{D}}\big(F_i\big(\bar{\tau}_i, do(a_i, \sigma_i)\big)\big) \stackrel{\text{def}}{=} \Phi_{F_i}(\bar{\tau}_i, a_i, \sigma_i)$$
$$\mathcal{R}_{\mathcal{D}}\big(F_i(\bar{\tau}_i, s_i)\big) \stackrel{\text{def}}{=} F_i(\bar{\tau}_i, s_i)$$
$$\mathcal{R}_{\mathcal{D}}\big(F_i(\bar{\tau}_i, S_0)\big) \stackrel{\text{def}}{=} F_i(\bar{\tau}_i, S_0)$$
$$\mathcal{R}_{\mathcal{D}}(\tau_i = \tau_j) \stackrel{\text{def}}{=} \tau_i = \tau_j$$
$$\mathcal{R}_{\mathcal{D}}\big(R_i(\bar{\tau}_i)\big) \stackrel{\text{def}}{=} R_i(\bar{\tau}_i)$$

The key point here is that each application of $\mathcal{R}_\mathcal{D}$ replaces action description predicates with the RHS of their definitions from $\mathcal{D}_{ad}$ and primitive fluents with the RHS of their successor state axioms from $\mathcal{D}_{ssa}$, "unwinding" a single action from each $do(a, \sigma)$ situation term in the query. If the situation term is not constructed using do, it is left unchanged.

Let us briefly state some important properties of the regression operator. First, and most importantly, it preserves equivalence of formulae:

Proposition 1. *For φ a regressable formula, $\mathcal{D} \models \varphi \equiv \mathcal{R}_\mathcal{D}(\varphi)$.*

Proof. By Pirri and Reiter [23, Theorem 2]. □

Any formula uniform in $do(a, s)$ is regressable, and the resulting formula will always be uniform in s:

Proposition 2. *For ϕ uniform in $do(a, s)$, $\mathcal{R}_\mathcal{D}(\phi)$ is uniform in s.*

Proof. By induction on the structure of regressable formulae. □

Let $\mathcal{R}^*_\mathcal{D}$ denote repeated applications of $\mathcal{R}_\mathcal{D}$ until the formula remains unchanged. Such applications can transform a query about some future situation into a query about the initial situation only:

Proposition 3. *For ϕ uniform in $do([a_1 \ldots a_n], S_0)$, $\mathcal{R}^*_\mathcal{D}(\phi)$ is uniform in S_0.*

Proof. By Pirri and Reiter [23, Theorem 3, part 1]. □

This last property is key to effective reasoning in the situation calculus, as it allows one to answer the *projection problem*. To determine whether ϕ holds in a given future situation, it suffices to determine whether $\mathcal{R}^*_\mathcal{D}(\phi)$ holds in the initial situation:

Proposition 4. *For ϕ uniform in $do([a_1 \ldots a_n], S_0)$:*

$$\mathcal{D} \models \phi \quad \textit{iff} \quad \mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models \mathcal{R}^*_\mathcal{D}(\phi).$$

Proof. By Pirri and Reiter [23, Theorem 3, part 2]. □

The regressed form is usually easier to answer, as it requires only the initial state axioms and background theory. The axioms $\mathcal{D}_{ad}$ and $\mathcal{D}_{ssa}$ are essentially "compiled into the query" by the $\mathcal{R}^*_\mathcal{D}$ operator. While an efficiency gain is not guaranteed, regression has proven a very effective technique in practice [17,23], particularly when combined with techniques to limit the resulting increase in query size [33].

Unfortunately, this powerful technique cannot be applied to formulae that are not regressable. Its operation depends crucially on knowing how many actions there are in each situation term, so that each action can be "unwound" from the query in turn. Queries that universally quantify over situations, such as the property persistence queries of interest in this paper, fall squarely outside the reach of standard regression techniques.

4.2. Inductive reasoning

While there is a rich and diverse literature base for the situation calculus, there appears to have been little work on queries that universally quantify over situation terms. Reiter [24] has shown why the induction axiom cannot in general be eliminated when proving such statements, demonstrating the use of the axiom in manual proofs but offering no automated procedure.

Other work on universally-quantified queries focuses on highly specialised applications, such as verifying state constraints [1,19] or studying properties of ConGolog programs [2,4,12]. While these works have produced useful results, they are typically intended as stand-alone techniques rather than general reasoning tools for the situation calculus.

Pirri and Reiter [19] have shown that the induction axiom can be "compiled away" when verifying a state constraint, by means of the following equivalence[1]:

$$\mathcal{D} - \mathcal{D}_{S_0} \models \phi[S_0] \rightarrow \big(\forall s\colon\ S_0 \leqslant_{Poss} s \rightarrow \phi[s]\big)$$

iff

$$\mathcal{D}_{bg} \models \forall s, a\colon\ \phi[s] \wedge \mathcal{R}_\mathcal{D}\big(Poss(a, s)\big) \rightarrow \mathcal{R}_\mathcal{D}\big(\phi\big[do(a, s)\big]\big)$$

[1] We have modified the original equations for consistency with our modified notation.

Verification of a state constraint can thus be reduced to the comparatively straightforward task of *static domain reasoning*. Verification of state constraints was also approached by Bertossi et al. [1], who develop an automatic constraint verification system using an induction theorem prover.

However, there are many issues that are not addressed by these highly specialised works. What if we are interested in the future of some arbitrary situation σ, rather than only S_0? What if we restrict future actions according to an arbitrary action description predicate? Can we integrate a method for handling universally-quantified queries with existing regression techniques? Our treatment of property persistence can provide a concrete basis for these considerations, and is hence significantly more general than this existing work.

Formulating various safety, liveness and starvation properties of ConGolog programs also requires universal quantification over situations. De Giacomo et al. [4] show how to re-cast these properties as fixpoint queries in second-order logic, and a preliminary model-checker capable of verifying them is described in [12]. Claßen and Lakemeyer [2] have developed a logic of ConGolog programs based on an iterative fixpoint computation similar to the one we propose in this paper. Their technique is based on a modal variant of the situation calculus known as $\mathcal{ES}$ and is designed for quite specific applications; by contrast, our approach aims to be a general-purpose reasoning tool for the classical situation calculus.

If one is willing to restrict attention to propositional domains, it is possible to use techniques from the propositional mu-calculus to answer a broad range of inductive queries [11]. In a similar vein, Ternovska [31] has proven decidability for a variant of the situation calculus with monadic fluents. Our work differs by focusing on a narrower class of queries, by constructing fixpoints at the meta-level rather than in the language, and by its applicability to the full first-order situation calculus.

Finally, let us introduce an important property of situations first formally identified by Savelli [28]: that universal quantification over situation terms is equivalent to a kind of infinite conjunction over the *levels* of the tree of situations:

$$\mathcal{D} \models \forall s\colon\ \psi(s)$$

iff

$$\mathcal{D} \models \bigcup_{n\in\mathbb{N}} \{\forall a_1 \ldots a_n\colon\ \psi\big(do\big([a_1 \ldots a_n], S_0\big)\big)\}$$

This is a direct consequence of the induction axiom for situations, which restricts situations to be constructed by performing some countable number of actions in the initial situation.

5. The persistence condition

To enable the use of persistence queries in practical systems, we clearly need a more effective reasoning technique than open-ended second-order theorem proving. Our approach is inspired by the success of Reiter's regression technique: use a meta-level operator to transform the query into one that is easier to answer. Specifically, our technique transforms a property persistence query at σ into the evaluation of a uniform formula at σ – a much simpler query which can be approached using existing techniques.

Formally, we seek a method for transforming a uniform formula ϕ and action description predicate α into a uniform formula $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ that is true at precisely the situations in which ϕ persists under α. We call such a formula the *persistence condition* of ϕ under α.

Definition 6 *(Persistence condition).* The persistence condition of ϕ under α, denoted $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$, is a uniform formula such that:

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon\ \big(\mathcal{P}_{\mathcal{D}}(\phi, \alpha)[s] \equiv \forall s'\colon\ s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]\big)$$

In other words, $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ holds at s iff ϕ persists under α at s.

Defining $\mathcal{P}_{\mathcal{D}}$ to be independent of the initial world state allows it to be calculated regardless of what (if anything) is known about the actual state of the world – after all, a situated agent may not know all the details of $\mathcal{D}_{S_0}$, and we still want it to be able to use this technique.

Provided that such a formula $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ is given, we can use standard regression to reduce reasoning about situation-invariant properties to a first-order reasoning task as follows:

$$\mathcal{D} \models \forall s\colon\ \sigma \leqslant_\alpha s \rightarrow \phi[s]$$

iff

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}(\phi, \alpha)[\sigma]$$

iff

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models \mathcal{R}^*_{\mathcal{D}}\big(\mathcal{P}_{\mathcal{D}}(\phi, \alpha)[\sigma]\big)$$

Note that this generalises the work of Lin and Reiter [19] on state constraints, where queries are restricted to the form $\forall s\colon\ S_0 \leqslant_{Poss} s \rightarrow \phi[s]$.

Definition 6 alone clearly does not make the task of answering a persistence query any easier – it gives no indication of how the persistence condition might be calculated in practice, or even whether such a formula actually exists for a given ϕ and α. In order to establish these results, we first need to define the weaker notion of a formula *persisting to depth n* in a situation.

Definition 7 *(Persistence to depth 1).* A uniform formula ϕ persists to depth 1 under α in situation s when the formula $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)[s]$ holds, as defined by:

$$\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)[s] \stackrel{\text{def}}{=} \phi[s] \wedge \forall a\colon\ \mathcal{R}_{\mathcal{D}}\big(\alpha[a,s]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a,s)\big]\big)$$

Since α is an action description predicate and ϕ is a uniform formula, the expressions $\mathcal{R}_{\mathcal{D}}(\alpha[a,s])$ and $\mathcal{R}_{\mathcal{D}}(\phi[do(a,s)])$ are always defined and the resulting formula is always uniform in s. Note that $\mathcal{P}^1_{\mathcal{D}}$ is a literal encoding of the requirement that "ϕ holds in s and in all its direct successors".

Successive applications of $\mathcal{P}^1_{\mathcal{D}}$ can then assert persistence to greater depths:

Definition 8 *(Persistence to depth n).* For any $n \geqslant 0$, a uniform formula ϕ persists to depth n under α in situation s when the formula $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s]$ holds, as defined by:

$$\mathcal{P}^0_{\mathcal{D}}(\phi,\alpha) \stackrel{\text{def}}{=} \phi$$
$$\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha) \stackrel{\text{def}}{=} \mathcal{P}^1_{\mathcal{D}}\big(\mathcal{P}^{n-1}_{\mathcal{D}}(\phi,\alpha),\alpha\big)$$

The following theorem confirms that $\mathcal{P}^n_{\mathcal{D}}$ operates according to this intuition – for any sequence of actions of length $i \leqslant n$, if each action satisfies α when it is executed, then ϕ will hold after performing those actions.

Theorem 1. *For any $n \in \mathbb{N}$, $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ holds in σ iff ϕ holds in σ and in all successors of σ reached by performing at most n actions satisfying α:*

$$\mathcal{D} \models \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[\sigma] \quad \equiv \bigwedge_{i \leqslant n} \forall a_1 \ldots a_i\colon \left(\bigwedge_{j \leqslant i} \alpha\big[a_j, do\big([a_1 \ldots a_{j-1}],\sigma\big)\big] \rightarrow \phi\big[do\big([a_1 \ldots a_i],\sigma\big)\big] \right)$$

Proof sketch. By induction on the natural numbers. For $n = 0$ we have $\phi[\sigma] \equiv \phi[\sigma]$ by definition. For the inductive case, we expand the definition of $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[\sigma]$ to get the following for the LHS:

$$\mathcal{P}^{n-1}_{\mathcal{D}}(\phi,\alpha)[\sigma] \wedge \forall a\colon\ \mathcal{R}_{\mathcal{D}}\big(\alpha[a,\sigma]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\mathcal{P}^{n-1}_{\mathcal{D}}(\phi,\alpha)\big[do(a,\sigma)\big]\big)$$

Substituting for $\mathcal{P}^{n-1}_{\mathcal{D}}$ using the inductive hypothesis gives us a conjunction ranging over $i \leqslant n-1$, with universally quantified variables $a_1 \ldots a_i$, and we must establish the $i = n$ case. Pushing this conjunction inside the scope of the $\forall a$ quantifier, we can rename $a \Rightarrow a_1$, $a_1 \Rightarrow a_2$, etc. to get the required expression. For a detailed proof see Appendix A. □

The $\mathcal{P}^n_{\mathcal{D}}$ operator thus allows us to express the persistence of a formula to any given depth using a simple syntactic translation based on regression. Intuitively, one would expect $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ to be some sort of fixpoint of $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)$, since $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ must imply persistence up to any depth. Such a fixpoint could then be calculated by iterative application of $\mathcal{P}^1_{\mathcal{D}}$. The remainder of this section is devoted to verifying this intuition.

We begin with two straightforward generalisations of results from the situation calculus literature, adapting them to our $\leqslant_\alpha$ notation:

Proposition 5. *For any action description predicate α, the foundational axioms of the situation calculus entail the following induction principle*:

$$\forall W, s\colon\ W(s) \wedge \big[\forall a, s'\colon\ \alpha\big[a,s'\big] \wedge s \leqslant_\alpha s' \wedge W\big(s'\big) \rightarrow W\big(do\big(a,s'\big)\big)\big] \rightarrow \forall s'\colon\ s \leqslant_\alpha s' \rightarrow W\big(s'\big)$$

Proof. A trivial adaptation of Theorem 3.2 in [24]. □

Proposition 6. *For any basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α:*

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon \phi[s] \rightarrow \big(\forall s'\colon s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]\big)$$

iff

$$\mathcal{D}_{bg} \models \forall s, a\colon \phi[s] \wedge \mathcal{R}_{\mathcal{D}}\big(\alpha[a, s]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a, s)\big]\big)$$

Proof. A straightforward generalisation of the model-construction proof of Lemma 5.3 in [19], utilising Proposition 5. The details of this proof are reproduced in Appendix A. □

Proposition 6 will be key in our algorithm for calculating the persistence condition. It allows one to establish the result "if ϕ holds in s, then ϕ persists in s" by using static domain reasoning, a comparatively straightforward reasoning task.

We next formalise some basic relationships between our hypothetical $\mathcal{P}_{\mathcal{D}}$ operator and $\mathcal{P}^n_{\mathcal{D}}$:

Lemma 1. *Given a basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α, then for any n:*

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon \big(\forall s'\colon s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]\big) \equiv \big(\forall s'\colon s \leqslant_\alpha s' \rightarrow \mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)\big[s'\big]\big)$$

That is, ϕ persists under α iff $\mathcal{P}^n_{\mathcal{D}}[\phi, \alpha]$ persists under α.

Proof. Since $\mathcal{P}^n_{\mathcal{D}}[\phi, \alpha]$ implies ϕ by definition, the *if* direction is trivial. For the *only-if* direction we proceed by induction on n.

For the base case of $\mathcal{P}^1_{\mathcal{D}}$, let $\mathcal{M}$ be a model of $\mathcal{D}$ and μ an assignment to the free variables in ϕ. Suppose that ϕ persists at s but $\mathcal{P}^1_{\mathcal{D}}(\phi, \alpha)$ does not:

$$\mathcal{M}, \mu \models \forall s'\colon s \leqslant_\alpha s' \rightarrow \phi\big[s'\big] \tag{3}$$

$$\mathcal{M}, \mu \not\models \forall s'\colon s \leqslant_\alpha s' \rightarrow \mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)\big[s'\big] \tag{4}$$

For (4) to hold, there must be some situation element from $\mathcal{M}$ that is in the future of $\mu(s)$ but at which $\mathcal{P}^1_{\mathcal{D}}(\phi, \alpha)$ is false. Let μ assign this situation element to the fresh variable $\dot{s}$, so that:

$$\mathcal{M}, \mu \models s \leqslant_\alpha \dot{s} \wedge \neg \mathcal{P}^1_{\mathcal{D}}(\phi, \alpha)[\dot{s}]$$

Expanding the definition of $\mathcal{P}^1_{\mathcal{D}}$ we have:

$$\mathcal{M}, \mu \models s \leqslant_\alpha \dot{s} \wedge \neg\big(\phi[\dot{s}] \wedge \forall a\colon \alpha[a, \dot{s}] \rightarrow \phi\big[do(a, \dot{s})\big]\big) \tag{5}$$

But by our assumption that (3) holds, we must have:

$$\mathcal{M}, \mu \models \phi[\dot{s}]$$

$$\mathcal{M}, \mu \models \forall a\colon \alpha[a, \dot{s}] \rightarrow \phi\big[do(a, \dot{s})\big]$$

It is thus impossible for (5) to hold, and we have a contradiction. Since our choice of $\mathcal{M}$ and μ was arbitrary, the result will hold for any s and we have the lemma as required.

For the inductive case, assume that $\mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha)$ persists but $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ does not. By definition we have $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha) = \mathcal{P}^1_{\mathcal{D}}(\mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha), \phi)$, and we repeat the base case proof using $\phi' = \mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha)$ in place of ϕ to obtain a contradiction. □

Lemma 2. *Given a basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α, then for any n:*

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon \big(\mathcal{P}_{\mathcal{D}}(\phi, \alpha)[s] \rightarrow \mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)[s]\big)$$

Proof. $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ implies the persistence of ϕ by definition. If ϕ persists at s, then by Lemma 1 we have that $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ persists at s. Since the persistence of $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ at s implies that $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ holds at s by definition, we have the lemma as desired. □

We are now equipped to prove the major theorem of this paper: that if $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ implies $\mathcal{P}^{n+1}_{\mathcal{D}}(\phi, \alpha)$, then $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ is equivalent to the persistence condition for ϕ under α.

Theorem 2. *Given a basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α, then for any n:*

$$\mathcal{D}_{bg} \models \forall s\colon\ \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \rightarrow \mathcal{P}^{n+1}_{\mathcal{D}}(\phi,\alpha)[s] \tag{6}$$

iff

$$\mathcal{D} - \mathcal{D}_{s_0} \models \forall s\colon\ \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \equiv \mathcal{P}_{\mathcal{D}}(\phi,\alpha)[s] \tag{7}$$

In other words, if we can calculate a fixpoint of applications of $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)[s]$ then that fixpoint is the persistence condition for ϕ under α.

Proof. For the *if* direction, we begin by expanding Eq. (6) using the definition of $\mathcal{P}^1_{\mathcal{D}}$ to get the equivalent form:

$$\mathcal{D}_{bg} \models \forall s\colon\ \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \rightarrow \mathcal{P}^1_{\mathcal{D}}\big(\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha),\alpha\big)[s]$$

$$\mathcal{D}_{bg} \models \forall s\colon\ \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \rightarrow \big(\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \wedge \forall a\colon\ \mathcal{R}_{\mathcal{D}}\big(\alpha[a,s]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)\big[do(a,s)\big]\big)\big)$$

$$\mathcal{D}_{bg} \models \forall s,a\colon\ \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \wedge \forall a\colon\ \mathcal{R}_{\mathcal{D}}\big(\alpha[a,s]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\mathcal{P}^1_{\mathcal{D}}(\phi\alpha)\big[do(a,s)\big]\big)$$

By Proposition 6, Eq. (6) thus lets us conclude that $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ persists under α. By Lemma 1 this is equivalent to the persistence of ϕ under α, which is equivalent to $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ by definition, giving:

$$\mathcal{D} - \mathcal{D}_{s_0} \models \forall s\colon\ \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \rightarrow \mathcal{P}_{\mathcal{D}}(\phi,\alpha)[s]$$

By Lemma 2 this is an equivalence, yielding Eq. (7) as required.

The *only-if* direction is a straightforward reversal of this reasoning process: $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ implies the persistence of ϕ, which implies the persistence of $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$, which yields Eq. (6) by Proposition 6. □

Since $\mathcal{D}_{bg} \models \mathcal{P}^{n+1}_{\mathcal{D}}(\phi,\alpha) \rightarrow \mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ by definition, Eq. (6) identifies $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ as a fixpoint of the $\mathcal{P}^1_{\mathcal{D}}$ operator, as our initial intuition suggested. We can therefore apply some standard results from fixpoint theory to the calculation of $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$, which we will do in the next section.

To conclude this section, we establish what is essentially the "dual" of the theorem above – that is there is *any* uniform formula satisfying the definition of $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ then it is the fixpoint of applications of $\mathcal{P}^1_{\mathcal{D}}$.[2]

Theorem 3. *Given a basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α, suppose that ψ is a regressable formula with s the only zero-arity term of sort situation and*:

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon\ \psi(s) \equiv \forall s'\colon\ s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]$$

$\psi(s)$ thus identifies precisely those situations in which ϕ persists under α. Then for any situation term σ:

$$\mathcal{D} \models \mathcal{R}^*_{\mathcal{D}}(\psi)[\sigma]$$

iff

$$\mathcal{D} \models \bigcup_{n\in\mathbb{N}} \big\{\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[\sigma]\big\}$$

In other words, $\mathcal{R}^*_{\mathcal{D}}(\psi)$ is a uniform formula representing the fixpoint of applications of $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)[s]$.

Proof sketch. The restricted form of ψ means that it is "about" only the situation s and its successors, so we can apply regression to transform it into a formula uniform in s. The theorem is then a straightforward adaptation of Lemma 6 in [28], using Savelli's technique of splitting the quantification over situation terms into an infinite conjunction. For a detailed proof see Appendix A. □

6. Calculating $\mathcal{P}_{\mathcal{D}}$

Since we can easily calculate $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ for any n, we have a straightforward algorithm for determining $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$: search for an n such that

$$\mathcal{D}_{bg} \models \forall s\colon\ \big(\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s] \rightarrow \mathcal{P}^{n+1}_{\mathcal{D}}(\phi,\alpha)[s]\big)$$

Since we expect $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ to be simpler than $\mathcal{P}^{n+1}_{\mathcal{D}}(\phi,\alpha)$, we should look for the smallest such n. Algorithm 1 presents an iterative procedure for doing just that. Note that the calculation of $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)$ is a straightforward syntactic transformation, so we do not present an algorithm for it.

[2] Previous versions of this work [13,14] incorrectly claimed that such a formula would always exist. Our thanks to an anonymous reviewer for helping to clarify this point.

Algorithm 1 Calculate $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$

```
pn ⇐ φ
pn1 ⇐ P^1_D(pn, α)
while D_bg ⊭ ∀s : pn[s] → pn1[s] do
    pn ⇐ pn1
    pn1 ⇐ P^1_D[pn, α]
end while
return pn
```

6.1. Soundness

If Algorithm 1 terminates, it terminates returning a value of *pn* for which Eq. (6) holds. By Theorem 2 this value of *pn* is equivalent to the persistence condition for ϕ under α. The algorithm therefore correctly calculates the persistence condition.

In particular, note that Eq. (6) holds when $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ is unsatisfiable for any situation, as it appears in the antecedent of an implication. The algorithm thus correctly returns an unsatisfiable condition (equivalent to $\bot$) when ϕ can never persist under α.

6.2. Completeness

There are two aspects to the completeness of our technique: whether the necessary fixpoint exists at all, and whether the algorithm for calculating it will terminate in a finite number of iterations.

6.2.1. Existence of the persistence condition

By Theorem 3 we know that the persistence condition is always the fixpoint of applications of $\mathcal{P}^1_{\mathcal{D}}$. In other words, if $\mathcal{P}^1_{\mathcal{D}}$ has no finitely-expressible fixpoint for a given ϕ and α, then a uniform formula satisfying the definition of $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ does not exist.

Unfortunately, it is relatively easy to construct a fluent for which a first-order persistence condition does not exist. Consider a domain with a single object sort modelled after the standard first-order axioms for the natural numbers, a single fluent $F(x, s)$ and a single action A that makes $F(x, s)$ false whenever $F(suc(x), s)$ is false:

$$F\big(x, do(A, s)\big) \equiv F\big(suc(x), s\big)$$

Let us attempt to calculate the persistence condition of $F(0, s)$ with the action description predicate α set to true. The sequence of iterations produced by Algorithm 1 would be:

$$\begin{aligned}
&\mathcal{P}^1_{\mathcal{D}}\big(F(0,s)\big) \equiv F(0,s) \wedge F\big(suc(0), s\big)\\
&\mathcal{P}^2_{\mathcal{D}}\big(F(0,s)\big) \equiv F(0,s) \wedge F\big(suc(0), s\big) \wedge F\big(suc\big(suc(0)\big), s\big)\\
&\quad\vdots\\
&\mathcal{P}^n_{\mathcal{D}}\big(F(0,s)\big) \equiv \bigwedge_{i=0}^{i=n} F\big(suc^i(0), s\big)
\end{aligned}$$

It is straightforward to manually demonstrate that $F(0, s)$ will persist if and only if F holds for all objects built by successive applications of *suc* to 0. Such a condition can only be expressed using a transitive closure, and hence there is no first-order formula that is equivalent to the persistence condition of $F(0, s)$.

Of course, if $\mathcal{D}_{bg}$ provides an axiomatisation of the standard "greater than" predicate $x < y$ then we can finitely identify these objects, and the persistence condition can be expressed as:

$$\mathcal{P}_{\mathcal{D}}\big(F(0,s)\big) \equiv F(0,s) \wedge \forall x\colon\ 0 < x \rightarrow F(x,s)$$

However, calculating such a condition would still be beyond the reach of Algorithm 1; since it cannot be constructed by finitely many applications of $\mathcal{P}^1_{\mathcal{D}}$, the algorithm would fail to terminate.

Nevertheless, Theorem 3 does provide an important completeness result – it demonstrates that if there exists *any* technique to replace reasoning about a persistence query at σ with a regression-based reasoning scheme at σ, then it is equivalent (up to termination of the algorithm) to our approach.

In fact, we suspect that a stronger result than Theorem 3 holds: that if a persistence query at σ can be replaced by any first-order formula that does not universally quantify over situation terms, then that query has a well-defined persistence condition.

Conjecture 1. *Given a basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α, suppose that $\psi(s)$ is a sentence of $\mathcal{L}_{sitcalc}$ whose prenex normal form contains only existential quantifiers over situations, with s the only free variable of sort situation, such that*:

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon\ \psi(s) \equiv \forall s'\colon\ s \leqslant_\alpha s' \rightarrow \phi\left[s'\right]$$

Thus $\psi(s)$ identifies precisely those situations in which ϕ persists under α. Then there is a uniform formula $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ such that:

$$\mathcal{D} \models \forall s\colon\ \psi(s) \equiv \mathcal{P}_\mathcal{D}(\phi, \alpha)[s]$$

In other words: if the domain $\mathcal{D}$ permits *some* technique for transforming a persistence query into a query that doesn't universally quantify over situations, then $\mathcal{D}$ is also amenable to the technique presented in this paper.

The proof from Theorem 3 cannot be applied to this conjecture because we have no way to reduce this more general $\psi(s)$ to a uniform formula. As part of our ongoing research, we aim to either confirm this intuition or to find a counter-example – either result would shed valuable light on the study of universally-quantified queries in the situation calculus.

6.2.2. Termination

As shown in the previous section, even if the persistence condition $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ is known to exist there is no guarantee that Algorithm 1 can calculate it in a finite amount of time. The algorithm may in fact fail to terminate for two distinct reasons: the loop condition may never be satisfied, or the first-order logical inference in the loop condition may be undecidable.

The later case indicates that the background theory $\mathcal{D}_{bg}$ is undecidable. While this is a concern, it affects more than just our algorithm – any system implemented around such an action theory will be incomplete. With respect to this source of incompleteness, our algorithm is no more incomplete than any larger reasoning system it would form a part of. We will concern ourselves only with the former case.

To ensure the completeness of our approach, we must restrict the domain theory and/or the form of queries being posed so that $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ is not only guaranteed to exist, but is guaranteed to be calculable within a finite number of iterations. Some standard results from fixpoint theory can be applied towards this task:

Definition 9. Let $L(\mathcal{D}_{bg})$ be the Lindenbaum–Tarski algebra of the background theory for formulae uniform in s. It is thus a boolean lattice where:

- elements are sets of uniform formulae grouped by equivalence w.r.t. $\mathcal{D}_{bg}$,
- meet and join are $\wedge$ and $\vee$ respectively,
- logical implication w.r.t. $\mathcal{D}_{bg}$ forms a partial ordering relation, and
- top and bottom are the equivalence classes of $\top$ and $\bot$ respectively.

Theorem 4. *For any fixed value of α, $\mathcal{P}^1_\mathcal{D}(\phi, \alpha)[s]$ is a monotone decreasing function over $L(\mathcal{D}_{bg})$.*

Proof. The domain and range of $\mathcal{P}^1_\mathcal{D}$ are uniform formulae and so correspond to the elements of $L(\mathcal{D}_{bg})$. By the definition of $\mathcal{P}^1_\mathcal{D}$ it is always the case that $\mathcal{D}_{bg} \models \mathcal{P}^1_\mathcal{D}(\phi, \alpha) \rightarrow \phi$. Since $\rightarrow$ is the partial ordering relation of $L(\mathcal{D}_{bg})$, $\mathcal{P}^1_\mathcal{D}$ is a monotone decreasing function as required. □

Corollary 1. *If $L(\mathcal{D}_{bg})$ is a complete lattice, then given a uniform formula ϕ and action description predicate α, the persistence condition $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ always exists and is unique up to equivalence under $\mathcal{D}_{bg}$.*

Proof. A standard result from fixpoint theory. Since $\mathcal{P}^1_\mathcal{D}$ is a monotone decreasing function over a complete lattice, the constructive proof of Tarski's fixpoint theorem [3] means it has a unique greatest fixpoint less than the equivalence class of ϕ. This fixpoint can be found by transfinite iteration of applications of $\mathcal{P}^1_\mathcal{D}$ and is equivalent to $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ by Theorem 2. □

Corollary 2. *If $L(\mathcal{D}_{bg})$ is a well-founded lattice, then given a uniform formula ϕ and action description predicate α, the persistence condition $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ always exists and can be calculated by finitely many applications of $\mathcal{P}^1_\mathcal{D}$.*

Proof. A standard result from fixpoint theory. A well-founded lattice $L(\mathcal{D}_{bg})$ as no infinite descending chains of elements, so the transfinite iteration used by [3] to find the fixpoint of $\mathcal{P}^1_\mathcal{D}$ in Corollary 1 must terminate after finitely many iterations. □

6.3. Guaranteeing completeness

The most obvious restriction that can be applied to guarantee completeness is to ensure that the action and object sorts are finite. In such theories the lattice $L(\mathcal{D}_{bg})$ is finite, and any finite lattice is both complete and well-founded. These theories also have the advantage that the static domain reasoning performed by Algorithm 1 can be done using propositional logic, meaning it is decidable and so providing a strong termination guarantee.

There are notable advantages to maintaining the expressive power of first-order logic even when the domain can be propositionalised – for example, the use of quantifiers can produce exponentially-shorter formulae and hence lead to shorter proofs.

An alternate approach is to restrict the form of the successor state axioms and/or the queries being posed so that applications of $\mathcal{P}^1_{\mathcal{D}}$ operate on a *subset* of $L(\mathcal{D}_{bg})$ which meets the requirements set out in Corollaries 1 and 2.

For example, suppose we restrict successor state axioms and action description predicates to the following forms, where $\bar{y}_i \subseteq \bar{x}$ and $\Phi^{+/-}_{F,i}$ and $\Pi_{\alpha,i}$ mention no terms other than $\bar{x}$ and s:

$$F\big(\bar{x}, do(a,s)\big) \equiv \bigvee_{i=1}^{n} a = a_i(\bar{y}_i) \wedge \Phi^{+}_{F,i}(\bar{x},s) \vee F(\bar{x},s) \wedge \neg \bigvee_{i=1}^{n} a = a_i(\bar{y}_i) \wedge \Phi^{-}_{F,i}$$

$$\alpha(\bar{x},a,s) \equiv \bigvee_{i=1}^{n} a = a_i(\bar{y}_i) \wedge \Pi_{\alpha,i}(\bar{x},s)$$

Such domains have a finite number of actions, each of which is only capable of changing fluents about objects it is given as direct arguments. For example, the action $a_i(A)$ can change the fluent $F(A,s)$ but not $F(B,s)$.

Now, suppose we have a *quantifier free* uniform formula ϕ. When calculating $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)$ it is possible to simplify away the action terms introduced by the successor state axioms, so that $\mathcal{P}^1_{\mathcal{D}}(\phi,\alpha)$ is also a quantifier-free uniform formula mentioning only the terms found in ϕ. The range of $\mathcal{P}^1_{\mathcal{D}}$ applied to ϕ is thus a finite subset of $L(\mathcal{D}_{bg})$, which ensures a terminating calculation of $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$. A detailed proof appears in Appendix A (Theorem 5).

This restriction is similar to the local-effect theories of [21,35], with the additional requirement that the right-hand side of the axiom contain no quantifiers. Whether local-effect theories can guarantee termination without additional restrictions would be an interesting avenue for further research.

As an example, the classic "holding" predicate can be expressed in a form that meets this restriction:

$$Holding\big(obj, do(a,s)\big) \equiv a = pickup(obj) \vee Holding(obj,s) \wedge a \neq drop(obj)$$

$$Poss\big(pickup(obj), s\big) \equiv \neg Holding(obj,s)$$

$$Poss\big(drop(obj), s\big) \equiv Holding(obj,s)$$

However, if there are multiple agents acting in the domain then the possibility predicate for *pickup* would no longer meet this restriction, as it contains a quantifier on the right-hand side:

$$Poss\big(pickup(agt,obj), s\big) \equiv \neg\exists agt' .\ Holding\big(agt', obj, s\big)$$

Clearly this is a strong restriction on the structure of the theory, as the successor state axioms are not able to contain any quantifiers. It does demonstrate, however, that certain syntactic restrictions on $\mathcal{D}$ are able to guarantee terminating calculation of $\mathcal{P}_{\mathcal{D}}$. It seems there should be a more general "syntactic well-foundedness" restriction that can be applied to these axioms, but we have not successfully formulated one at this stage.

In a similar vein, suppose that the theory of action is *context free* [20]. In such theories successor state axioms have the following form:

$$F\big(\bar{x}, do(a,s)\big) \equiv \Phi^{+}_{F}(\bar{x},a) \vee \big(F(\bar{x},s) \wedge \neg\Phi^{-}_{F}(\bar{x},a)\big)$$

The effects of an action are thus independent of the situation in which it is performed. Lin and Levesque [18, Lemma 6.2] have show that context-free theories with a finite number of parameterless actions have a finite state space, which is sufficient to ensure termination of our algorithm.

Intuitively, if there are at most N distinct states in the domain then any situation more than N actions into the future has the same state as some situation less than N actions into the future. Algorithm 1 will therefore terminate after at most N iterations. A detailed proof is available in Appendix A (Theorem 6).

From a slightly different perspective, suppose that ϕ can never persist under α, so that $\mathcal{P}_{\mathcal{D}}(\phi,\alpha) \equiv \bot$. Further suppose that $\mathcal{D}$ has the *compactness* property as in standard first-order logic. Then the "quantum levels" of Savelli [28] guarantee that there is a fixed, finite number of actions within which $\neg\phi$ can always be achieved. In this case Algorithm 1 will determine $\mathcal{P}_{\mathcal{D}}(\phi,\alpha) \equiv \bot$ within finitely many iterations.

It would also be interesting to determine whether known variants of the situation calculus in which the projection problem is decidable (such as [10]) are able to guarantee termination of the fixpoint construction, or whether more sophisticated fixpoint algorithms can be applied instead of simple iterative approximation. Investigating such algorithms would be a promising avenue for future research.

The important point here is not that we can guarantee termination in general, but that we have precisely characterised the inductive reasoning necessary to answer property persistence queries, and shown how it can be replaced by the evaluation of a uniform formula at the situation in question.

6.4. Effectiveness

Our algorithm replaces a single reasoning task based on the full action theory $\mathcal{D}$ with a series of reasoning tasks based on the static background theory $\mathcal{D}_{bg}$. Is this a worthwhile trade-off in practice? The following points weigh strongly in favour of our approach.

First and foremost, we avoid the need for the second-order induction axiom. All the reasoning tasks can be performed using standard first-order reasoning, for which there are high-quality automated provers. Second, the calculation of $\mathcal{P}_{\mathcal{D}}$ performs only *static reasoning*, which as discussed in Section 2 is a comparatively straightforward task which can be made decidable under certain conditions. Third, $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)[\sigma]$ is in a form amenable to regression, a standard tool for effective reasoning in the situation calculus. Fourth, the persistence condition for a given ϕ and α can be cached and re-used for a series of related queries about different situations, a significant gain in amortised efficiency. Finally, in realistic domains we expect many properties to fail to persist beyond a few situations into the future, meaning that our algorithm will require few iterations in a large number of cases.

Of course, we also inherit the potential disadvantage of the regression operator: the length of $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ may be much larger than the length of ϕ. Since there is no bound on the number of iterations required for Algorithm 1 to terminate, the length of $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ is actually *unbounded* in the general case. As with regression, our experience has been that this is rarely a problem in practice, and is more than compensated for by the reduced complexity of the resulting reasoning task.

6.5. Related algorithms

As noted in Section 2, our use of fixpoints in this paper has much in common with the study of properties of ConGolog programs [2,4]. In particular, our algorithm has deep similarities to the CheckEU and CheckEG algorithms used by Claßen and Lakemeyer [2]. Like our work, they seek a fixpoint using iterative application of a meta-level function based on regression. Indeed, Algorithm 1 could be re-cast as a special case of their approach, but without the need to maintain a complex graph structure.

Claßen and Lakemeyer also note that their approach is not able to guarantee completeness in the general case, and they identify as future work the discovery of general classes of theory for which their technique is complete. Given the underlying similarities, we are confident that such advances in reasoning about ConGolog programs will advance our ability to answer persistence queries, and vice-versa.

We also note that work on state invariants in other planning formalisms (e.g. [7,8]) uses ideas broadly similar to our approach – an iterative algorithm that explores longer and longer sequences of actions until a stable state is reached. In these formalisms, the domain typically has a finiteness restriction that guarantees eventual termination of the algorithm.

7. Examples

To demonstrate the applicability of our technique, consider again the example persistence queries given in Section 3. The persistence condition is readily applicable to each example, and the transformed queries can then be answered using standard regression.

Goal impossibility. Given a goal G, establish that there is no legal situation in which that goal is satisfied:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}(\neg G, \mathit{Poss})[S_0]$$

The persistence condition of $\neg G$ with respect to action legality allows goal impossibility to be checked easily.

Goal futility. Given a goal G and situation σ, establish that the goal cannot be satisfied in any legal future situation from σ:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}(\neg G, \mathit{Poss})[\sigma]$$

Precisely the same formula is required for checking goal impossibility and goal futility. This highlights the advantage of re-using the persistence condition at multiple situations. Our approach makes it feasible for an agent to check for goal futility each time it considers performing an action, and avoid situations that would make its goals unachievable.

Checking state constraints. Given a state constraint SC, show that the constraint holds in every legal situation:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}(SC, \mathit{Poss})[S_0]$$

However, since we want a state constraint to *always* persist, it must satisfy the following equivalence:

$$\mathcal{D}_{bg} \models \phi \equiv \mathcal{P}_{\mathcal{D}}(\phi, \mathit{Poss})$$

If this equivalence does not hold then $\mathcal{P}_{\mathcal{D}}(\phi, \mathit{Poss})$ indicates the additional conditions that are necessary to ensure that ϕ persists, which may be useful for adjusting the action theory to enforce the constraint. This particular application has strong parallels to the approach to state constraints developed by Lin and Reiter [19].

Need for cooperation. Given an agent agt, goal G and situation σ, establish that no sequence of actions performed by that agent only can achieve the goal:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}(\neg G, \mathit{MyAction})[\sigma]$$

Knowledge with hidden actions. In recent work we have developed a regression rule for knowledge that uses the persistence condition to account for arbitrarily-long sequences of hidden actions [14]. While the details of this formulation are outside the scope of the current paper, the general form of the regression rule is:

$$\mathcal{R}_{\mathcal{D}}\big(\mathbf{Knows}\big(\phi, do(a,s)\big)\big) \stackrel{\text{def}}{=} \mathbf{Knows}\big(\mathcal{R}_{\mathcal{D}}\big(\mathcal{P}_{\mathcal{D}}(\phi, \mathit{Hidden})\big[do(a,s)\big]\big), s\big)$$

The key point of this definition is that the agent can only know ϕ if it knows that ϕ will persist after any sequence of hidden actions.

This highlights an important benefit of our approach – it integrates well with existing reasoning techniques. If one is willing to assume that $\mathcal{P}_{\mathcal{D}}(\phi, \alpha)$ will always exist, it can be used to "factor out" the inductive reasoning and produce a regression rule for formulae that universally quantify over situations.

The "gold thief" domain. As a detailed example of our technique in action, consider again the "gold thief" domain as described in Section 3 and axiomatised in Appendix B. We want to establish that:

$$\mathcal{D} \models \forall s\colon\ S_0 \leqslant_{\mathit{Undet}} s \rightarrow \neg \mathit{Stolen}(s)$$

By the definition of the persistence condition, this is equivalent to:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}(\neg \mathit{Stolen}, \mathit{Undet})[\sigma]$$

To answer this query we will employ Algorithm 1, calculating $\mathcal{P}^n_{\mathcal{D}}$ for successively larger values of n until the series converges to a fixpoint. Full details of this calculation can be found in Appendix C; we present only the major results below. The case of $n = 0$ is trivial:

$$\mathcal{P}^0_{\mathcal{D}}(\neg \mathit{Stolen}, \mathit{Undet})[s] = \neg \mathit{Stolen}(s)$$

The $n = 1$ case is given by Definition 7 as:

$$\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg \mathit{Stolen}(s) \wedge \forall a\colon\ \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(a,s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg \mathit{Stolen}\big(do(a,s)\big)\big)$$

Expanding the $\forall a$ quantifier over each of the three actions in this domain we obtain:

$$\begin{aligned}\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = {} & \neg \mathit{Stolen}(s) \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{takeGold}, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg \mathit{Stolen}\big(do(\mathit{takeGold}, s)\big)\big) \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{crackSafe}, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg \mathit{Stolen}\big(do(\mathit{crackSafe}, s)\big)\big) \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{toggleLight}, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg \mathit{Stolen}\big(do(\mathit{toggleLight}, s)\big)\big)\end{aligned}$$

Applying regression and simplifying produces the final result:

$$\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg \mathit{Stolen}(s) \wedge \big[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\big]$$

Intuitively, this indicates that the gold cannot be stolen by a single undetected action if the safe is not open (since it would first have to be cracked) or the light is on (since the action would be detected). Since this is clearly not entailed by the $\mathcal{P}^0_{\mathcal{D}}$ case, we must continue to the $n = 2$ case. Again applying Definition 7 and expanding out each individual action, we get:

$$\begin{aligned}\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = {} & \neg \mathit{Stolen}(s) \wedge \big[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\big] \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{takeGold}, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg \mathit{Stolen}\big(do(\mathit{takeGold}, s)\big) \\ & \wedge \big[\neg \mathit{SafeOpen}\big(do(\mathit{takeGold}, s)\big) \vee \mathit{LightOn}\big(do(\mathit{takeGold}, s)\big)\big]\big) \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{crackSafe}, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg \mathit{Stolen}\big(do(\mathit{crackSafe}, s)\big) \\ & \wedge \big[\neg \mathit{SafeOpen}\big(do(\mathit{crackSafe}, s)\big) \vee \mathit{LightOn}\big(do(\mathit{crackSafe}, s)\big)\big]\big) \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{toggleLight}, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}(\neg \mathit{Stolen}\big(do(\mathit{toggleLight}, s)\big) \\ & \wedge \big[\neg \mathit{SafeOpen}\big(do\big(\mathit{toggleLight}(s)\big) \vee \mathit{LightOn}\big(do(\mathit{toggleLight}, s)\big)\big]\big)\end{aligned}$$

Performing the regression and simplifying down, we eventually obtain:

$$\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = \neg \mathit{Stolen}(s) \wedge \big[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\big]$$

We have clearly satisfied the termination condition of Algorithm 1, since $\mathcal{P}^1_{\mathcal{D}} \rightarrow \mathcal{P}^2_{\mathcal{D}}$, and have thus successfully calculated the persistence condition:

$$\mathcal{P}_{\mathcal{D}}(\neg \mathit{Stolen}, \mathit{Undet}) = \neg \mathit{Stolen} \wedge [\neg \mathit{SafeOpen} \vee \mathit{LightOn}]$$

Checking whether the gold is safe is now a simple matter of reasoning about the initial situation:

$$\mathcal{D} \models \neg \mathit{Stolen}(S_0) \wedge \big[\neg \mathit{SafeOpen}(S_0) \vee \mathit{LightOn}(S_0)\big]$$

With $\mathcal{P}_{\mathcal{D}}(\neg \mathit{Stolen}, \mathit{Undet})$ in hand, we can also perform more sophisticated reasoning about the safety of the gold – for example, we can check whether a proposed action would jeopardise the safety of the gold and refuse to perform the action if so.

8. Conclusions

In this paper we have developed an algorithm that transforms property persistence queries, a quite general and useful class of situation calculus query, into a form that is amenable to standard techniques for effective reasoning in the situation calculus. The algorithm replaces a second-order induction axiom with a meta-level fixpoint calculation based on iterative application of the standard regression operator. It is shown to be sound, and complete in several interesting cases.

Our approach generalises previous work on universally-quantified queries in several important ways. It can consider sequences of actions satisfying a range of conditions, not just the standard ordering over action possibility, enabling us to treat problems such as need for cooperation and knowledge with hidden actions. It can establish that properties persist in the future of an arbitrary situation, not necessarily the initial situation, enabling us to answer the question of goal futility. The results of calculating the persistence condition can be cached, allowing for example the goal futility question to be efficiently posed on a large number of situations once the persistence condition has been calculated.

Perhaps most importantly for the wider situation calculus community, we have *factored out* the inductive reasoning required to answer these queries. Work on increasing the effectiveness of this inductive reasoning, and on guaranteeing a terminating calculation in stronger classes of action theory, can proceed independently from the development of formalisms that utilise persistence queries. This opens the possibility of wider application of the property persistence approach – subsequent applications can use the $\mathcal{P}_{\mathcal{D}}$ operator as a kind of "black box" for dealing with persistence queries, for example to formulate regression rules as in our own work on knowledge [14].

This paper has thus significantly increased the scope of queries that can be posed within automated reasoning systems built upon the situation calculus.

Appendix A. Detailed proofs

Theorem 1. *For any $n \in \mathbb{N}$, $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)$ holds in σ iff ϕ holds in σ and in all successors of σ reached by performing at most n actions satisfying α:*

$$\mathcal{D} \models \mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)[\sigma] \equiv$$
$$\bigwedge_{i \leqslant n} \forall a_1 \ldots a_i \colon \Big(\bigwedge_{j \leqslant i} \alpha\big[a_j, do\big([a_1 \ldots a_{j-1}], \sigma\big)\big] \rightarrow \phi\big[do\big([a_1 \ldots a_i], \sigma\big)\big] \Big)$$

Proof. By induction on the natural numbers. For $n = 0$ we have $\phi[\sigma] \equiv \phi[\sigma]$ by definition. For the inductive case, we expand the definition of $\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)[\sigma]$ to get the following for the LHS:

$$\mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha)[\sigma] \wedge \forall a \colon \mathcal{R}_{\mathcal{D}}\big(\alpha[a, \sigma]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha)\big[do(a, \sigma)\big]\big)$$

By the inductive hypothesis we can equate $\mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha)[\sigma]$ in this LHS with all but the $i = n$ clause from the RHS conjunction, and we suppress them on both sides. If we also drop the regression operators we are left with the following to establish:

$$\mathcal{D} \models \forall a \colon \alpha[a, \sigma] \rightarrow \mathcal{P}^{n-1}_{\mathcal{D}}(\phi, \alpha)\big[do(a, \sigma)\big] \equiv$$
$$\forall a_1 \ldots a_n \colon \Big(\bigwedge_{j \leqslant n} \alpha\big[a_j, do\big([a_1 \ldots a_{j-1}], \sigma\big)\big] \rightarrow \phi\big[do\big([a_1 \ldots a_n], \sigma\big)\big] \Big)$$

We can again use the inductive hypothesis on $\mathcal{P}_{\mathcal{D}}^{n-1}$ in the LHS of this equivalence. If we then distribute the $\alpha[a,\sigma]$ implication over the outermost conjunction and collect quantifiers, we obtain the following for the LHS:

$$\bigwedge_{i\leqslant n-1} \forall a, a_1 \ldots a_i\colon \left(\alpha[a,\sigma] \wedge \bigwedge_{j\leqslant i} \alpha\big[a_j, do\big([a,a_1\ldots a_{j-1}],\sigma\big)\big] \rightarrow \phi\big[do\big([a,a_1\ldots a_i],\sigma\big)\big]\right)$$

Renaming $a \Rightarrow a_1$, $a_1 \Rightarrow a_2, \ldots, a_i \Rightarrow a_{i+1}$, we see that each of the $i < n-1$ clauses on the LHS is equivalent to one of the $i < n$ clauses that have been suppressed on the RHS. The remaining $i = n-1$ clause is equivalent to the required RHS:

$$\forall a_1 \ldots a_n\colon \left(\bigwedge_{j\leqslant n} \alpha\big[a_j, do\big([a_1\ldots a_{j-1}],\sigma\big)\big] \rightarrow \phi\big[do\big([a_1\ldots a_n],\sigma\big)\big]\right)$$

We therefore have the desired equivalence. □

Lemma 3. *For any basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α:*

if

$$\Sigma \cup \mathcal{D}_{bg} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{ad} \cup \big\{\phi[S_0]\big\} \models \forall s\colon\ S_0 \leqslant_\alpha s \rightarrow \phi[s]$$

then

$$\mathcal{D}_{bg} \models \forall s, a\colon\ \phi[s] \wedge \mathcal{R}_{\mathcal{D}}\big(\alpha[a,s]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a,s)\big]\big)$$

Proof. This is a straightforward generalisation of the proof of Lemma 5.3 in [19] using our modified notation. We repeat the details of the proof below for convenience. Begin by assuming the antecedent:

$$\Sigma \cup \mathcal{D}_{bg} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{ad} \cup \big\{\phi[S_0]\big\} \models \forall s\colon\ S_0 \leqslant_\alpha s \rightarrow \phi[s] \tag{A.1}$$

Let $\mathcal{M}$ be a model of $\mathcal{D}_{bg}$. Suppose μ_s and μ_a are variable assignments for situation and action variables respectively, such that:

$$\mathcal{M}, \mu_s, \mu_a \models \phi[s] \wedge \mathcal{R}_{\mathcal{D}}\big(\alpha[a,s]\big)$$

We will show that $\mathcal{M}, \mu_s, \mu_a \models \mathcal{R}_{\mathcal{D}}(\phi[do(a,s)])$. Following the construction in [23], we construct a model $\mathcal{M}'$ with the following properties:

1. $\mathcal{M}'$ and $\mathcal{M}$ share the same domains for sorts Action and Object.
2. $\mathcal{M}'$ interprets every rigid predicate and function the same as M.
3. $\mathcal{M}' \models \Sigma \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{ad}$.
4. For every variable assignment for object variables μ_o, and for every fluent $F(\vec{x}, s)$, $\mathcal{M}', \mu_o \models F(\vec{x}, S_0)$ iff $\mathcal{M}, \mu_s, \mu_o \models F(\vec{x}, s)$.

Since $\mathcal{M}, \mu_s, \mu_a \models \mathcal{R}_{\mathcal{D}}(\alpha[a,s])$ and $\mathcal{M}' \models \mathcal{D}_{ad}$ we have:

$$\mathcal{M}', \mu_a \models \alpha[a, S_0]$$

Since $\mathcal{M}, \mu_s, \mu_a \models \phi[s]$, property (4) of $\mathcal{M}'$ ensures that:

$$\mathcal{M}', \mu_a \models \phi[S_0]$$

By our assumption (A.1), $\mathcal{M}', \mu_a \models \phi[do(a, S_0)]$. Since $\mathcal{M}'$ satisfies $\mathcal{D}_{ssa}$ the regression operator can be employed to yield:

$$\mathcal{M}', \mu_a \models \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a, S_0)\big]\big)$$

Finally, property (4) of $\mathcal{M}'$ gives us:

$$\mathcal{M}', \mu_a \models \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a, S_0)\big]\big) \quad \text{iff} \quad \mathcal{M}, \mu_s, \mu_a \models \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a, s)\big]\big)$$

Since we have the LHS of this equivalence by construction, we can conclude the RHS. This suffices to establish the lemma. □

Proposition 6. *For any basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α:*

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon \phi[s] \rightarrow \big(\forall s'\colon s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]\big)$$
iff
$$\mathcal{D}_{bg} \models \forall s, a\colon \phi[s] \wedge \mathcal{R}_{\mathcal{D}}\big(\alpha[a, s]\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\phi\big[do(a, s)\big]\big)$$

Proof. The *if* direction is straightforward using the induction axiom from Proposition 5. For the *only-if* direction, we can take the S_0 case of the LHS to obtain:

$$\mathcal{D} - \mathcal{D}_{S_0} \models \phi[S_0] \rightarrow \big(\forall s\colon S_0 \leqslant_\alpha s \rightarrow \phi[s]\big)$$

Lifting $\phi[S_0]$ into the axioms, this precisely matches the form of Lemma 3 and we have the theorem as desired. □

Theorem 3. *Given a basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α, suppose that ψ is a regressable formula with s the only zero-arity term of sort situation and*:

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s\colon \psi(s) \equiv \forall s'\colon s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]$$

$\psi(s)$ thus identifies precisely those situations in which ϕ persists under α. Then for any situation term σ:

$$\mathcal{D} \models \mathcal{R}^*_{\mathcal{D}}(\psi)[\sigma]$$
iff
$$\mathcal{D} \models \bigcup_{n\in\mathbb{N}} \big\{\mathcal{P}^n_{\mathcal{D}}(\phi, \alpha)[\sigma]\big\}$$

In other words, $\mathcal{R}^*_{\mathcal{D}}(\psi)$ is a uniform formula representing the fixpoint of applications of $\mathcal{P}^1_{\mathcal{D}}(\phi, \alpha)[s]$.

Proof. The restrictions on ψ mean it is a regressable formula containing only situations of the form s and $do([a_1 \ldots a_i], s)$. $\mathcal{R}^*_{\mathcal{D}}(\psi)$ can therefore unwind these action terms and produce an equivalent formula uniform in s.

Let $\mathcal{M}$ be a model of $\mathcal{D} - \mathcal{D}_{S_0}$ and μ an assignment to the free variables in ϕ. Adopting the technique used by Savelli in [28, Lemma 6], we can split the quantification over s' into a kind of infinite conjunction over the levels of the tree of situations:

$$\mathcal{M}, \mu \models \forall s'\colon s \leqslant_\alpha s' \rightarrow \phi\big[s'\big]$$
iff
$$\mathcal{M}, \mu \models \bigcup_{n\in\mathbb{N}} \left\{\forall a_1 \ldots a_n\colon \bigwedge_{i=1}^{n} \alpha\big(a_i, do\big([a_1 \ldots a_{i-1}], s\big)\big) \rightarrow \phi\big[do\big([a_1 \ldots a_n], s\big)\big]\right\} \tag{A.2}$$

Note that each element of the set (A.2) is a statement about all situations n actions into the future of s. Moreover, each element can be regressed to give a formula uniform in s.

Now take the finite subset of (A.2) for n up to any natural number m. Clearly such a subset can be written as a finite conjunction:

$$\mathcal{M}, \mu \models \bigcup_{n\leqslant m} \left\{\forall a_1 \ldots a_n\colon \bigwedge_{i=1}^{n} \alpha\big(a_i, do\big([a_1 \ldots a_{i-1}], s\big)\big) \rightarrow \phi\big[do\big([a_1 \ldots a_n], s\big)\big]\right\}$$
iff
$$\mathcal{M}, \mu \models \bigwedge_{n\leqslant m} \forall a_1 \ldots a_n\colon \bigwedge_{i=1}^{n} \alpha\big(a_i, do\big([a_1 \ldots a_{i-1}], s\big)\big) \rightarrow \phi\big[do\big([a_1 \ldots a_n], s\big)\big] \tag{A.3}$$

Eq. (A.3) precisely matches the form of the RHS of the equivalence in Theorem 1, so we can substitute to LHS to give:

$$\mathcal{M}, \mu \models \bigcup_{n\leqslant m} \left\{\forall a_1 \ldots a_n\colon \bigwedge_{i=1}^{n} \alpha\big(a_i, do\big([a_1 \ldots a_{i-1}], s\big)\big) \rightarrow \phi\big[do\big([a_1 \ldots a_n], s\big)\big]\right\}$$
iff
$$\mathcal{M}, \mu \models \mathcal{P}^m_{\mathcal{D}}(\phi, \alpha)[s]$$

By definition $\mathcal{P}^m_{\mathcal{D}}(\phi,\alpha)$ implies $\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)$ for any $n<m$, so we have:

$$\mathcal{M},\mu \models \bigcup_{n\leqslant m}\left\{\forall a_1\ldots a_n\colon \bigwedge_{i=1}^{n}\alpha\big(a_i, do\big([a_1\ldots a_{i-1}],s\big)\big)\rightarrow \phi\big[do\big([a_1\ldots a_n],s\big)\big]\right\}$$

iff

$$\mathcal{M},\mu \models \bigcup_{n\leqslant m}\big\{\mathcal{P}^n_{\mathcal{D}}(\phi,\alpha)[s]\big\}$$

Noting that this construction works for any $\mathcal{M}$, μ and $m\in\mathbb{N}$ gives the theorem as required. □

Theorem 5. *Let $\mathcal{D}$ be such that all successor state axioms and action description predicates take the following restricted forms, where $\bar{y}_i\subseteq\bar{x}$ and $\Phi^{+/-}_{F,i}$ and $\Pi_{\alpha,i}$ mention no terms other than $\bar{x}$ and s:*

$$F\big(\bar{x},do(a,s)\big)\equiv\bigvee_{i=1}^{n}a=a_i(\bar{y}_i)\wedge\Phi^{+}_{F,i}(\bar{x},s)\vee F(\bar{x},s)\wedge\neg\bigvee_{i=1}^{n}a=a_i(\bar{y}_i)\wedge\Phi^{-}_{F,i}$$

$$\alpha(\bar{x},a,s)\equiv\bigvee_{i=1}^{n}a=a_i(\bar{y}_i)\wedge\Pi_{\alpha,i}\bar{x},s)$$

Then for a quantifier-free uniform formula ϕ and any α, the persistence condition $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ exists and can be calculated in finitely many iterations.

Proof. We will treat only the case where $\phi=F(\bar{x},s)$; the general case follows by induction on the structure of ϕ. Calculating $\mathcal{P}^1_{\mathcal{D}}(F(\bar{x},s),\alpha)$ gives:

$$\mathcal{P}^1_{\mathcal{D}}\big(F(\bar{x},s),\alpha\big)=F(\bar{x},s)\wedge\forall a\colon\left(\bigvee_{i=1}^{n}a=a_i(\bar{y}_i)\wedge\Pi_{\alpha,i}(\bar{x},s)\right)\rightarrow$$
$$\left(\bigvee_{j=1}^{n}a=a_j(\bar{y}_j)\wedge\Phi^{+}_{F,i}(\bar{x},s)\vee F(\bar{x},s)\wedge\neg\bigvee_{j=1}^{n}a=a_j(\bar{y}_j)\wedge\Phi^{-}_{F,i}\right)$$

If we take the i-disjunction outside the implication, it becomes a conjunction over action types:

$$\mathcal{P}^1_{\mathcal{D}}\big(F(\bar{x},s),\alpha\big)=F(\bar{x},s)\wedge\forall a\colon\bigwedge_{i=1}^{n}\left(a=a_i(\bar{y}_i)\wedge\Pi_{\alpha,i}(\bar{x},s)\rightarrow\right.$$
$$\left.\bigvee_{i=1}^{n}a=a_i(\bar{y}_i)\wedge\Phi^{+}_{F,i}(\bar{x},s)\vee F(\bar{x},s)\wedge\neg\bigvee_{i=1}^{n}a=a_i(\bar{y}_i)\wedge\Phi^{-}_{F,i}\right)$$

This allows us to remove the quantification over a and eliminate the explicit references to a_i to produce:

$$\mathcal{P}^1_{\mathcal{D}}\big(F(\bar{x},s)\alpha\big)=F(\bar{x},s)\wedge\bigwedge_{i=1}^{n}\big(\Pi_{\alpha,i}(\bar{x},s)\rightarrow\Phi^{+}_{F,i}(\bar{x},s)\vee F(\bar{x},s)\wedge\neg\Phi^{-}_{F,i}\big)$$

Let $\mathbf{terms}(\phi)$ denote the set of all terms mentioned in ϕ:

$$\mathbf{terms}\big(F(\bar{\tau},\sigma)\big)\stackrel{\text{def}}{=}\bar{\tau}\cup\{\sigma\}$$
$$\mathbf{terms}(\tau_i=\tau_j)\stackrel{\text{def}}{=}(\tau_i,\tau_j)$$
$$\mathbf{terms}(\phi_i\wedge\phi_j)\stackrel{\text{def}}{=}\mathbf{terms}(\phi_i)\cup\mathbf{terms}(\phi_j)$$
$$\mathbf{terms}(\neg\phi)\stackrel{\text{def}}{=}\mathbf{terms}(\phi)$$

Recall that $\Phi^{+/-}_{F,i}$ and $\Pi_{\alpha,i}$ mention no terms other than $\bar{x}$ and s, so:

$$\mathbf{terms}\big(F(\bar{x},s)\big)=\mathbf{terms}\big(\mathcal{P}^1_{\mathcal{D}}\big(F(\bar{x},s),\alpha\big)\big)$$

And by induction, we have that for any n:

$$\mathbf{terms}\big(F(\bar{x},s)\big) = \mathbf{terms}\big(\mathcal{P}^n_{\mathcal{D}}\big(F(\bar{x},s),\alpha\big)\big)$$

Since $\mathbf{terms}(F(\bar{x},s))$ is finite, it follows that applications of $\mathcal{P}^1_{\mathcal{D}}$ to $F(\bar{x},s)$ can only generate finitely many non-equivalent formulae. $\mathcal{P}^1_{\mathcal{D}}$ thus operates over a finite subset of $L(\mathcal{D}_{bg})$, which suffices to ensure the terminating calculation of $\mathcal{P}_{\mathcal{D}}(F(\bar{x},s),\alpha)$. □

Theorem 6. *Let $\mathcal{D}$ be a context-free domain with finitely many parameterless actions, then for any ϕ and α the persistence condition $\mathcal{P}_{\mathcal{D}}(\phi,\alpha)$ exists and can be calculated in finitely many iterations.*

Proof. Following the notation used in [18], say that two situations have the same *state* if they satisfy precisely the same fluents:

$$\mathit{SameState}\big(s,s'\big) \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n}\big(\forall \bar{x}_i : F_i(\bar{x}_i,s) \equiv F_i\big(\bar{x}_i,s'\big)\big)$$

Observe that such situations also satisfy the same uniform formulae:

$$\mathcal{D} \models \mathit{SameState}\big(s,s'\big) \rightarrow \big(\phi[s] \equiv \phi\big[s'\big]\big)$$

Let $\|do(\xi,S_0)\|$ be the set of all situations that have the same state as $do(\xi,S_0)$ for a given sequence of actions ξ. By [18, Lemma 6.1] there are at most N distinct sets $\|do(\xi,S_0)\|$ for some natural number N.

We show that Algorithm 1 terminates after at most N iterations. Suppose otherwise, i.e. suppose that there is a model $\mathcal{M}$ of $\mathcal{D}$ and a variable assignment μ such that:

$$\mathcal{M},\mu \models \mathcal{P}^N_{\mathcal{D}}(\phi,\alpha)[s]$$
$$\mathcal{M},\mu \models \neg\mathcal{P}^{N+1}_{\mathcal{D}}(\phi,\alpha)[s] \tag{A.4}$$

Then there must exist a sequence of actions $[A_1 \ldots A_{N+1}]$ such that:

$$\mathcal{M},\mu \models \neg\phi\big[do\big([A_1 \ldots A_{N+1}],s\big)\big] \tag{A.5}$$

We will show that such a sequence cannot exist. First, assume that $\|do([A_1 \ldots A_i],s)\|$ is distinct for each $i \leqslant N$. Since there are at most N such distinct states in the domain, $\|do([A_1 \ldots A_{N+1})\|$ must be the same as $\|do([A_1 \ldots A_i],s)\|$ for some i. By (A.4) we have that $\phi[do([A_1 \ldots A_i],s)]$ for any $i \leqslant N$, and so (A.5) cannot hold.

Alternately, assume that there is some $i < j \leqslant N$ such that:

$$\mathcal{M},\mu \models \mathit{SameState}\big(do\big([A_1 \ldots A_i],s\big), do\big([A_1 \ldots A_j],s\big)\big)$$

Then we can remove the redundant actions between i and j to get:

$$\mathcal{M},\mu \models \mathit{SameState}\big(do\big([A_1 \ldots A_i, A_{j+1} \ldots A_{N+1}],s\big), do\big([A_1 \ldots A_{N+1}],s\big)\big)$$

By (A.4) we know that ϕ holds at $do([A_1 \ldots A_i, A_{j+1} \ldots A_{N+1}])$, so (A.5) cannot hold and we have the desired contradiction.

There can therefore be no such model $\mathcal{M}$, and persistence to depth N suffices to establish persistence to any depth. □

Appendix B. Axioms for the "gold thief" domain

This section gives the axioms used for the "gold thief" example domain in Sections 3 and 7. In this domain a thief may try to steal some gold from a safe. There is a light in the room, and a security camera that will detect the thief's actions as long as the light is on. The safe can be open or closed, but the gold can only be stolen if the safe is open. It is possible for the thief to crack the safe and force it open, but only if the light is on.

The actions in this domain are *takeGold*, *crackSafe* and *toggleLight*, the primitive fluents are *SafeOpen*, *LightOn* and *Stolen*, and the action description predicates include the standard $\mathit{Poss}(a,s)$ and a custom predicate $\mathit{Undet}(a,s)$ indicating that action a would not be detected by the security camera.

The successor state axioms $\mathcal{D}_{ssa}$ consist of the following sentences:

$$\mathit{Stolen}\big(do(a,s)\big) \equiv a = \mathit{takeGold} \vee \mathit{Stolen}(s)$$
$$\mathit{SafeOpen}\big(do(a,s)\big) \equiv a = \mathit{crackSafe} \vee \mathit{SafeOpen}(s)$$
$$\mathit{LightOn}\big(do(a,s)\big) \equiv a = \mathit{toggleLight} \wedge \neg\mathit{LightOn}(s) \vee \mathit{LightOn}(s) \wedge a \neq \mathit{toggleLight}$$

The action description predicates in $\mathcal{D}_{ad}$ are defined by the following:

$$Poss(a, s) \equiv a = toggleLight \vee a = takeGold \wedge SafeOpen(s) \vee a = crackSafe \wedge LightOn(s)$$

$$Undet(a, s) \equiv Poss(a, s) \wedge a \neq toggleLight \wedge \neg LightOn(s)$$

The background theory $\mathcal{D}_{bg}$ contains only the standard unique names assumptions for actions, and the initial situation axioms $\mathcal{D}_{S_0}$ are not specified since they are not used in the discussion.

Appendix C. Complete calculation for the "gold thief" domain

This section presents complete details of the calculation of $\mathcal{P}_{\mathcal{D}}$ for the "gold thief" example as outlined in Section 7. We need to calculate:

$$\mathcal{P}_{\mathcal{D}}(\neg Stolen, Undet)[\sigma]$$

To calculate this persistence condition we follow Algorithm 1, calculating $\mathcal{P}^n_{\mathcal{D}}$ for successively larger values of n until the series converges to a fixpoint. The case of $n = 0$ is trivial:

$$\mathcal{P}^0_{\mathcal{D}}(\neg Stolen, Undet)[s] = \neg Stolen(s)$$

The $n = 1$ case is given by Definition 7 as:

$$\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg Stolen(s) \wedge \forall a\colon \mathcal{R}_{\mathcal{D}}\big(Undet(a, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg Stolen\big(do(a, s)\big)\big)$$

Some straightforward simplifications can be applied at this stage. First, since there are finitely many actions in this domain, the $\forall a$ quantification can be replaced with a finite conjunction:

$$\begin{aligned}\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg Stolen(s)\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(takeGold, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg Stolen\big(do(takeGold, s)\big)\big)\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(crackSafe, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg Stolen\big(do(crackSafe, s)\big)\big)\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(toggleLight, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg Stolen\big(do(toggleLight, s)\big)\big)\end{aligned}$$

From the domain axioms we know that *crackSafe* and *toggleLight* do not affect whether the gold has been stolen, while *takeGold* will always make *Stolen* true. Performing the regression of $\neg Stolen$ we thus obtain the following result:

$$\begin{aligned}\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg Stolen(s) &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(takeGold, s)\big) \rightarrow \neg\top\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(crackSafe, s)\big) \rightarrow \neg Stolen(s)\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(toggleCam, s)\big) \rightarrow \neg Stolen(s)\end{aligned}$$

Since the conjunction already contains $\neg Stolen(s)$ unconditionally, we can simplify away the final three cases. We obtain:

$$\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg Stolen(s) \wedge \neg\mathcal{R}_{\mathcal{D}}\big(Undet(takeGold, s)\big)$$

Performing the remaining regression, we obtain the final result for $\mathcal{P}^1_{\mathcal{D}}$:

$$\mathcal{P}^1_{\mathcal{D}}(\ldots)[s] = \neg Stolen(s) \wedge \big[\neg SafeOpen(s) \vee LightOn(s)\big]$$

Since this is clearly not entailed by the $\mathcal{P}^0_{\mathcal{D}}$ case, we must continue to the $n = 2$ case. Again applying Definition 7 and expanding out each individual action, we get:

$$\begin{aligned}\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = \neg Stolen(s) &\wedge \big[\neg SafeOpen(s) \vee LightOn(s)\big]\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(takeGold, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg Stolen\big(do(takeGold, s)\big)\\ &\wedge \big[\neg SafeOpen\big(do(takeGold, s)\big) \vee LightOn\big(do(takeGold, s)\big)\big]\big)\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(crackSafe, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}\big(\neg Stolen\big(do(crackSafe, s)\big)\\ &\wedge \big[\neg SafeOpen\big(do(crackSafe, s)\big) \vee LightOn\big(do(crackSafe, s)\big)\big]\big)\\ &\wedge \mathcal{R}_{\mathcal{D}}\big(Undet(toggleLight, s)\big) \rightarrow \mathcal{R}_{\mathcal{D}}(\neg Stolen\big(do(toggleLight, s)\big)\\ &\wedge \big[\neg SafeOpen\big(do\big(toggleLight(s)\big) \vee LightOn\big(do(toggleLight, s)\big)\big]\big)\end{aligned}$$

Applying similar simplifications to the $\mathcal{P}^1_{\mathcal{D}}$ case, we can obtain:

$$\begin{aligned}\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = {} & \neg \mathit{Stolen}(s) \wedge \left[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\right] \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{takeGold}, s)\big) \rightarrow \neg\top \wedge \left[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\right] \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{crackSafe}, s)\big) \rightarrow \neg \mathit{Stolen}(s) \wedge \left[\neg\top \vee \mathit{LightOn}(s)\right] \\ & \wedge \mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{toggleLight}, s)\big) \rightarrow \neg \mathit{Stolen}(s) \wedge \left[\neg \mathit{SafeOpen}(s) \vee \neg \mathit{LightOn}(s)\right]\end{aligned}$$

Further simplifying:

$$\begin{aligned}\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = {} & \neg \mathit{Stolen}(s) \wedge \left[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\right] \wedge \neg\mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{takeGold}, s)\big) \\ & \wedge \left[\neg\mathcal{R}_{\mathcal{D}}\big(\mathit{Undet}(\mathit{toggleLight}, s)\big) \vee \neg \mathit{SafeOpen}(s) \vee \neg \mathit{LightOn}(s)\right]\end{aligned}$$

Now performing the second regression:

$$\begin{aligned}\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = {} & \neg \mathit{Stolen}(s) \wedge \left[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\right] \wedge \left[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\right] \\ & \wedge \left[\neg\bot \vee \neg \mathit{SafeOpen}(s) \vee \neg \mathit{LightOn}(s)\right]\end{aligned}$$

Further simplification gives:

$$\mathcal{P}^2_{\mathcal{D}}(\ldots)[s] = \neg \mathit{Stolen}(s) \wedge \left[\neg \mathit{SafeOpen}(s) \vee \mathit{LightOn}(s)\right]$$

We have clearly satisfied the termination condition of Algorithm 1, since $\mathcal{P}^1_{\mathcal{D}} \rightarrow \mathcal{P}^2_{\mathcal{D}}$. We have thus successfully calculated the persistence condition:

$$\mathcal{P}_{\mathcal{D}}(\neg \mathit{Stolen}, \mathit{Undet}) = \neg \mathit{Stolen} \wedge [\neg \mathit{SafeOpen} \vee \mathit{LightOn}]$$

References

[1] Leopoldo E. Bertossi, Javier Pinto, Pablo Saez, Deepak Kapur, Mahadevan Subramaniam, Automating proofs of integrity constraints in situation calculus, in: Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems, in: Lecture Notes in Artificial Intelligence, Springer, 1996, pp. 212–222.

[2] Jens Claßen, Gerhard Lakemeyer, A logic for non-terminating Golog programs, in: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08), 2008, pp. 589–599.

[3] Patrick Cousot, Radhia Cousot, Constructive versions of Tarski's fixed point theorems, Pacific Journal of Mathematics 82 (1) (1979) 43–57.

[4] Giuseppe De Giacomo, Evgenia Ternovska, Ray Reiter, Non-terminating processes in the situation calculus, in: Proceedings of the AAAI'97 Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control, 1997.

[5] Alessandro Farinelli, Alberto Finzi, Thomas Lukasiewicz, Team programming in Golog under partial observability, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), 2007, pp. 2097–2012.

[6] A. Ferrein, Ch. Fritz, G. Lakemeyer, Using Golog for deliberation and team coordination in robotic soccer, Kunstliche Intelligenz I (2005) 24–43.

[7] Maria Fox, Derek Long, The automatic inference of state invariants in TIM, Journal of Artificial Intelligence Research 9 (1998) 367–421.

[8] Alfonso Gerevini, Lenhart K. Schubert, Discovering state constraints in DISCOPLAN: Some new results, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI), 2000, pp. 761–767.

[9] Hojjat Ghaderi, Hector Levesque, Yves Lespérance, A logical theory of coordination and joint ability, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07), 2007, pp. 421–426.

[10] Yilian Gu, Mikhail Soutchanski, Decidable reasoning in a modified situation calculus, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), 2007, pp. 1891–1897.

[11] Mayu Ishida, Reasoning about actions: A model-theoretic approach, Masters' thesis in computer science, School of Computing Science, Simon Fraser University, 2002.

[12] L. Kalantari, E. Ternovska, A model checker for verifying ConGolog programs, in: Proceedings of the 18th AAAI Conference on Artificial Intelligence (AAAI'02), 2002.

[13] Ryan Kelly, Asynchronous multi-agent reasoning in the situation calculus, PhD thesis, Department of Computer Science and Software Engineering, The University of Melbourne, Melbourne, Victoria, Australia, 2008.

[14] Ryan F. Kelly, Adrian R. Pearce, Knowledge and observations in the situation calculus, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07), 2007, pp. 841–843.

[15] Ryan F. Kelly, Adrian R. Pearce, Property persistence in the situation calculus, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), 2007, pp. 1948–1953.

[16] H. Levesque, F. Pirri, R. Reiter, Foundations for the situation calculus, Electronic Transactions on Artificial Intelligence 2 (3–4) (1998) 159–178.

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

[18] F. Lin, H. Levesque, What robots can do: Robot programs and effective achievability, Artificial Intelligence 101 (1998) 201–226.

[19] Lin Fangzhen, Ray Reiter, State constraints revisited, Journal of Logic and Computation 4 (5) (1994) 655–678.

[20] Lin Fangzhen, Ray Reiter, How to progress a database, Artificial Intelligence 92 (1997) 131–167.

[21] Yongmei Liu, Gerhard Lakemeyer, On first-order definability and computability of progression for local-effect actions and beyond, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09), 2009, pp. 860–866.

[22] John McCarthy, Patrick J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence, vol. 4, Edinburgh University Press, 1969, pp. 463–502.

[23] Fiora Pirri, Ray Reiter, Some contributions to the metatheory of the situation calculus, Journal of the ACM 46 (3) (1999) 325–361.

[24] Ray Reiter, Proving properties of states in the situation calculus, Artificial Intelligence 64 (1993) 337–351.

[25] Ray Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, The MIT Press, 2001.
[26] Ray Reiter, The frame problem in situation the calculus: A simple solution (sometimes) and a completeness result for goal regression, in: Vladimir Lifschitz (Ed.), Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, Academic Press Professional, Inc., 1991, pp. 359–380.
[27] Sebastian Sardina, Giuseppe De Giacomo, Yves Lespénce, Hector Levesque, On the semantics of deliberation in IndiGolog – from theory to implementation, Annals of Mathematics and Artificial Intelligence 41 (2–4) (August 2004) 259–299.
[28] Francesco Savelli, Existential assertions and quantum levels on the tree of the situation calculus, Artificial Intelligence 170 (6) (2006) 643–652.
[29] Richard Scherl, Hector Levesque, Knowledge, action, and the frame problem, Artificial Intelligence 144 (2003) 1–39.
[30] Steven Shapiro, Yves Lespérance, Hector J. Levesque, The cognitive agents specification language and verification environment for multiagent systems, in: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02), 2002, pp. 19–26.
[31] E. Ternovska, Automata theory for reasoning about actions, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99), 1999, pp. 153–158.
[32] Johan van Benthem, Eric Pacuit, The tree of knowledge in action: Towards a common perspective, Advances in Modal Logic 6 (2006) 87–106.
[33] Hans van Ditmarsch, Andreas Herzig, Tiago de Lima, Optimal regression for reasoning about knowledge and actions, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007, pp. 1070–1075.
[34] Stavros Vassos, Hector Levesque, On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture, in: Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08), 2008, pp. 1004–1009.
[35] Stavros Vassos, Gerhard Lakemeyer, Hector J. Levesque, First-order strong progression for local-effect basic action theories, in: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08), 2008, pp. 662–671.