Property Persistence in the Situation Calculus

Ryan F. Kelly and Adrian R. Pearce
NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne
Victoria, 3010, Australia
{rfk,adrian}@csse.unimelb.edu.au

Abstract

We develop an algorithm for reducing universally quantified situation calculus queries to a form more amenable to automated reasoning. Universal quantification in the situation calculus requires a second-order induction axiom, making automated reasoning difficult for such queries. We show how to reduce queries about property persistence, a common family of universally-quantified query, to an equivalent form that does not quantify over situations. The algorithm for doing so utilizes only first-order reasoning. We give several examples of important reasoning tasks that are facilitated by our approach, including checking for goal impossibility and reasoning about knowledge with partial observability of actions.

1 Introduction

The situation calculus [Pirri and Reiter, 1999] has long been one of the most popular formalisms for reasoning about dynamic worlds. It has more recently become a popular choice for *implementing* systems situated in a dynamic world, as it offers, among other advantages: a formalism based in first-order logic; an elegant monotonic solution to the frame problem; and an effective reasoning procedure for the projection problem [Reiter, 1991]. Coupled with logic programming languages such as Prolog, it has facilitated the implementation of a wide range of systems.

The foundational axioms of the situation calculus include a second-order induction axiom defining the set of all situations [Reiter, 1993]. Sentences that contain only existential quantification over situation terms can be proven without the induction axiom and so are more amenable to automated reasoning [Pirri and Reiter, 1999]. It is therefore important for systems built on a situation calculus theory of action to limit queries to existential form. Much of the work on implementing systems with the situation calculus has been on reducing the number of axioms required to answer a given query, to allow more efficient reasoning.

Unfortunately there are many reasoning tasks that require universal quantification over situations. For example, it is often desirable to show that no situation can satisfy a particular goal. In this paper we are interested in a subset of such queries which we refer to as *persistence queries*: under a particular situation calculus theory $\mathcal{D}$, and given some property ϕ and situation s, determine whether ϕ will hold in all situations in the future of s:

$$\mathcal{D} \models \forall s' \,.\, s \sqsubseteq s' \;\rightarrow\; \phi[s']$$

The need for second-order logic has traditionally limited automated reasoning about such queries. We introduce a new approach that is similar in spirit to the regression operator of [Reiter, 1991]: define an operator that transforms a formula ϕ into a formula $\mathcal{P}_{\mathcal{D}}[\phi]$, such that ϕ persists in s if and only if $\mathcal{P}_{\mathcal{D}}[\phi]$ holds in s. We term this the *persistence condition* of ϕ, and show how to calculate it in a form suitable for effective automated reasoning.

To determine $\mathcal{P}_{\mathcal{D}}[\phi]$, we first define an operator $\mathcal{P}^1_{\mathcal{D}}[\phi]$ that holds in s whenever ϕ holds in s and all its immediate successors. The persistence condition is shown to be a fixed-point of this operator, which can be sought using a straightforward iterative algorithm. Since this requires only first-order reasoning with a limited number of axioms, the result is an effective procedure with which to answer a family of queries that universally quantify over situations.

The paper is organized as follows: Section 2 gives a brief introduction to the situation calculus, formally defines the persistence problem, discusses related work and gives several examples of important persistence queries; Section 3 formally defines the persistence condition, presents an algorithm that calculates it, shows that the algorithm is correct and discusses the conditions necessary for completeness; and Section 4 concludes with a summary of our results.

2 Background

The situation calculus is a many-sorted language of first-order logic augmented with a second-order induction axiom. It has the following sorts: *Action* terms are functions denoting 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 up using the function $do : Action \times Situation \rightarrow Situation$; *Object* terms represent any other object in the domain. It further distinguishes *Fluents* as predicates or functions representing properties of the world that may change from one situation to

another, and so take a situation term as their final argument. For a detailed description consult [Pirri and Reiter, 1999].

A *basic action theory* is a set $\mathcal{D}$ of situation calculus sentences (with a specific syntactic form as specified in [Pirri and Reiter, 1999]) that describes a particular dynamic world. Queries about the behavior or evolution of the world are posed as logical entailment queries relative to this theory. It consists of the following disjoint sets: the foundational axioms of the situation calculus (Σ); successor state axioms describing how fluents change between situations ($\mathcal{D}_{ss}$); precondition axioms indicating when actions can be performed ($\mathcal{D}_{ap}$); unique names axioms ensuring that action terms are distinct ($\mathcal{D}_{una}$); and axioms describing the value of fluents in the initial situation ($\mathcal{D}_{S_0}$):

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$$

The foundational axiom of most import for this paper is the induction axiom, which defines the set of all situations as the least set containing S_0 and closed under application of the do function, as follows:

$$\forall P\,.\,[P(S_0) \wedge \forall s, a\,.\,(P(s) \rightarrow P(do(a,s))) \rightarrow \forall s\,.\,P(s)]$$

The fundamental importance of this axiom is described in [Reiter, 1993]. Much research has focused on identifying sentences of the situation calculus that can be proven without this axiom, and thus require only first-order logic. Our work continues this tradition.

There is a distinguished predicate $Poss(a, s)$ that indicates when it is possible to perform an action in a given situation. The set of action precondition axioms $\mathcal{D}_{ap}$ contains one axiom for each type of action A, of the general form:

$$\forall s, \overrightarrow{x}\,.\,[Poss(A(\overrightarrow{x}), s) \equiv \Pi_A(\overrightarrow{x}, s)]$$

Here Π_A represents a uniform formula (see below) defining action possibility on a per-action basis.

It is often useful to introduce analogous predicates to describe different aspects of action performance. For example, a predicate $CantObs(agt, a, s)$ could indicate when an agent will be unable to observe the occurrence of an action. Such predicates are easily incorporated into a basic action theory by axiomatizing them in $\mathcal{D}_{ap}$ in the same manner as $Poss$. We refer to these predicates in general as *action description predicate*s, and use the meta-variable α to represent an arbitrary such predicate.

The *uniform formulae* as defined in [Pirri and Reiter, 1999] can be thought of as formulae that describe a *property* of the state of the world. They are basically logical combinations of fluents referring to a common situation term, which cannot mention action description predicates nor compare situation terms. We will use the meta-variable ϕ to refer to an arbitrary uniform formula.

It is often useful to determine the truth of a property at a given situation. The formula $\phi[s]$ represents the uniform formula ϕ with all occurrences of its unique situation term replaced by the situation s.

2.1 Ordering over Situations

Situations form a tree with S_0 at the root and do constructing child situations from parents. There is a basic ordering relation $\sqsubset$ defined by the following foundational axioms:

$$\forall s\,.\,\neg\,(s \sqsubset S_0)$$

$$\forall s, s', a\,.\,s \sqsubset do(a, s') \equiv s \sqsubseteq s'$$

Where $s \sqsubseteq s'$ is the standard abbreviation of $s \sqsubset s' \vee s = s'$. It is frequently useful to restrict the situations under consideration to those possible in the real world, by defining an ordering relation $<$ over only these "executable" situations:

$$\forall s\,.\,\neg\,(s < S_0)$$

$$\forall s, s', a\,.\,s < do(a, s') \equiv s \leq s' \wedge Poss(a, s')$$

Many results of the situation calculus are derived relative to the executable situations - for example, [Reiter, 1993] develops an induction principle for all situations where $S_0 \leq s$. This notation is so useful that we propose a parametrization of it: let α be an action description predicate specifying the actions of interest, then define the "α-ordering" over situations as follows:

$$\forall s\,.\,\neg\,(s <_\alpha S_0)$$

$$\forall s, s', a\,.\,s <_\alpha do(a, s') \equiv s \leq_\alpha s' \wedge \alpha(a, s')$$

The standard ordering over executable situations is clearly $<_{Poss}$. If s is thought of as the "current situation", then $s <_\alpha s'$ means that s' is one potential future, and all actions that will occur in that future satisfy α.

2.2 Effective Reasoning

Answering an arbitrary situation calculus query ψ involves, in general, an entailment problem in second-order logic (SOL):

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

This can be problematic for efficient automated reasoning. Fortunately, there exist particular syntactic forms for which some of the axioms in $\mathcal{D}$ are not required. In [Pirri and Reiter, 1999] the "$\exists$s sentences" are defined as those in which every situation variable is in the scope of a positive existential quantifier. They show that such sentences can always be proven without the induction axiom (I) and hence are answerable by first-order logical entailment (FOL):

$$\mathcal{D} \models_{SOL} \exists s\,\psi \;\; \textit{iff} \;\; \mathcal{D} - \{I\} \models_{FOL} \exists s\,\psi$$

Such queries can be approached with standard first-order reasoning systems. To increase the efficiency of reasoning it is desirable to eliminate further axioms from $\mathcal{D}$, which [Pirri and Reiter, 1999] show possible for several syntactic forms.

Axiom reduction is also the key idea behind the regression operator, the principal tool for effective reasoning in the situation calculus [Reiter, 1991]. The regression operator $\mathcal{R}_\mathcal{D}$ is a syntactic manipulation whose behavior can be summarized[1] for our purposes as follows: given a formula ϕ uniform in $do(a, s)$, regression transforms it into a formula $\mathcal{R}_\mathcal{D}[\phi]$ that

[1] The full behavior of $\mathcal{R}_\mathcal{D}$ is beyond the scope of this paper. Experienced readers please note that we use the single-step version of the regression operator, as in [Lin and Reiter, 1994]. Since any formula uniform in $do(a, s)$ is by definition single-step regressable, the development below omits the traditional qualification that formulae must be regressable.

is uniform in s and is equivalent to ϕ under the theory of action $\mathcal{D}$:

$$\mathcal{D} \models \phi \equiv \mathcal{R}_{\mathcal{D}}[\phi]$$

Regression also replaces instances of the $Poss$ predicate with appropriate instantiations of the corresponding uniform formula from the axioms in $\mathcal{D}_{ap}$, and other action description predicates can easily be treated in the same way. In order to regress such a predicate over an action variable one must assume that there are only a finite number of action types, so that it can be replaced with the disjunctive closure of its definitional axioms. We assume this restriction throughout.

If ϕ refers to a situation that is rooted at S_0, repeated applications of the regression operator (denoted by $\mathcal{R}^*_{\mathcal{D}}$) can transform it into an equivalent formula uniform in the initial situation. The successor state and action precondition axioms are "compiled into" the regressed formula, and so are not required for the reasoning task:

$$\mathcal{D} \models \phi \ \ \textit{iff} \ \ \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \mathcal{R}^*_{\mathcal{D}}[\phi][S_0]$$

The trade-off is that the length of $\mathcal{R}^*_{\mathcal{D}}[\phi]$ may be exponential in the length of ϕ. In practice this is typically more than compensated for by the reduction in the number of axioms required, and regression has proven a very effective technique.

2.3 Property Persistence

One important form of situation calculus query is to ask whether a given property ϕ will hold in *all future situations* of a given situation s, as well as at s itself:

$$\mathcal{D} \models \forall s' \,.\, s \sqsubseteq s' \rightarrow \phi(s')$$

More generally, one may wish to limit the futures considered to those brought about by a certain class of actions α:

$$\mathcal{D} \models \forall s' \,.\, s \leq_{\alpha} s' \rightarrow \phi[s']$$

In words this states "given that all subsequent actions satisfy α, ϕ will remain true" or, more succinctly, "ϕ persists under α". We term queries of this form *property persistence queries*, and they are involved in many useful reasoning tasks. The following are a small selection:

Goal Impossibility: Given a goal G, establish that there is no possible situation in which that goal is satisfied:

$$\mathcal{D} \models \forall s \,.\, S_0 \leq s \rightarrow \neg G(s)$$

Goal Futility: Given a goal G and situation s, establish that the goal cannot be satisfied in any possible situation in the future of s:

$$\mathcal{D} \models \forall s' \,.\, s \leq s' \rightarrow \neg G(s')$$

Note how this is different from goal impossibility: while the agent may have initially been able to achieve its goal, the actions that have subsequently been performed have rendered the goal unachievable. Agents would be well advised to avoid such situations.

Checking State Constraints: This is a variant of goal impossibility - show that the constraint can never be violated.

Need for Cooperation: To establish that it is absolutely necessary to cooperate with another agent, an agent must determine persistence with respect to actions performed by itself:

$$\mathcal{D} \models \forall s' \,.\, s \leq_{OwnAction} s' \rightarrow \neg G[s']$$

Assuming that the action description predicate $OwnAction$ has been defined to identify actions performed by the agent in question, this states that no situation in the future of s in which all actions were performed by that agent can satisfy G. If this is the case, it will need to cooperate with another agent in order to achieve its goal.

Knowledge under Partial Observability: In recent unpublished work we develop a new account of knowledge in the situation calculus when not all actions are observable by all agents. To facilitate reasoning in this formalism, agents must be able to reason about what cannot be changed by actions that they cannot observe, a form of persistence query.

Due to the universal quantification over situations, the techniques for effective automated reasoning in Section 2.2 cannot be applied to persistence queries. This paper is devoted to developing of a complementary technique for handling such queries.

2.4 Related Work

While there is a rich and diverse literature base for the situation calculus, there appears to have been little work dealing with universally quantified queries. The work of [Reiter, 1993] shows how to handle such queries by hand using an appropriate instantiation of the induction axiom, but makes no mention of automating this reasoning.

Other work considering persistence focuses exclusively on verifying state constraints. These are uniform formulae that must hold in every possible situation, a highly specialized form of persistence query. The work of [Lin and Reiter, 1994] "compiles away" the induction axiom when verifying a state constraint, by means of the following equivalence:

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \models \phi[S_0] \rightarrow (\forall s \,.\, S_0 \leq s \rightarrow \phi[s])$$
$$\textit{iff}$$
$$\mathcal{D}_{una} \models \forall s, a \,.\, \phi[s] \wedge \mathcal{R}_{\mathcal{D}}[Poss(a, s)] \rightarrow \mathcal{R}_{\mathcal{D}}[\phi[do(a, s)]]$$

Verification of a state constraint can thus be reduced to reasoning about a universally quantified uniform formula using only the unique names axioms, a comparatively straightforward reasoning task. This problem is also approached by [Bertossi *et al.*, 1996], who develop a system for automatically verifying state constraints based on an induction theorem prover.

However, there are many issues related to persistence that are not addressed by such work, including: persistence at situations other than S_0; action description predicates other than $Poss$; how to combine notions of persistence and regression; and determining what additional conditions may be necessary to guarantee the persistence of ϕ. As our treatment of persistence can provide a concrete basis for these considerations, it is significantly more general than existing work.

3 The Persistence Condition

For implementing persistence queries in practical systems, we clearly need to transform the query into a form suitable for effective reasoning. Specifically, we will transform a persistence query based at s into the evaluation of a uniform formula at s, which can be done effectively using the regression operator. We need some transformation of a property

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, given a particular action theory $\mathcal{D}$. We call $\mathcal{P}_\mathcal{D}$ the *persistence condition* of ϕ under α.

Definition 1. *The persistence condition $\mathcal{P}_\mathcal{D}[\phi, \alpha]$ is a uniform formula that is the weakest precondition for the persistence of ϕ under α, given a basic action theory $\mathcal{D}$ without the initial situation axioms. That is:*

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

Defining $\mathcal{P}_\mathcal{D}$ to be independent of the initial world state allows an agent to calculate it regardless of what (if anything) is known about the actual state of the world. To see how $\mathcal{P}_\mathcal{D}$ may be calculated, consider the weaker notion of a formula *persisting to depth* n in a situation:

Definition 2. *A uniform formula ϕ persists to depth 1 under α in situation s when the formula $\mathcal{P}^1_\mathcal{D}[\phi, \alpha]$ holds in s, as defined by:*

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

More generally, for any $n \geq 0$, a uniform formula ϕ persists to depth n under α in situation s when the formula $\mathcal{P}^n_\mathcal{D}[\phi, \alpha]$ holds in s, as defined by:

$$\mathcal{P}^0_\mathcal{D}[\phi, \alpha] \stackrel{def}{=} \phi$$

$$\mathcal{P}^n_\mathcal{D}[\phi, \alpha] \stackrel{def}{=} \mathcal{P}^1_\mathcal{D}[\mathcal{P}^{n-1}_D[\phi, \alpha], \alpha]$$

Note that $\mathcal{P}^1_\mathcal{D}$ is a literal encoding of the requirement "ϕ holds in s and in all its direct successors". Since α is an action description predicate and ϕ is a uniform formula, the expression $\alpha(a, s) \rightarrow \phi[do(a, s)]$ can always be regressed and the result will always be uniform in s. Successive applications of $\mathcal{P}^1_\mathcal{D}$ assert persistence to greater depths.

Intuitively, one would expect $\mathcal{P}_\mathcal{D}[\phi, \alpha]$ to be a fixed-point of $\mathcal{P}^1_\mathcal{D}[\phi, \alpha]$, since $\mathcal{P}_\mathcal{D}[\phi, \alpha]$ implies persistence up to any depth. Such a fixed-point can then be calculated using standard iterative approximation techniques. To show that this is in fact the case we require a number of theoretical results, presented below.

3.1 Formal Development

We begin by adapting two existing results involving induction to operate with our generalized $\leq_\alpha$ notation, and be based at situations other than S_0:

Proposition 1. *For any action description predicate α, the foundational axioms of the situation calculus entail the following induction principle:*

$$\forall W, s \,.\, W(s) \wedge [\forall a, s' \,.\, \alpha(a, s') \wedge s \leq_\alpha s' \wedge W(s') \rightarrow W(do(a, s'))] \rightarrow \forall s' \,.\, s \leq_\alpha s' \rightarrow W(s')$$

Proof. A trivial adaptation of theorem 1 in [Reiter, 1993]. □

Proposition 2. *For any basic action theory $\mathcal{D}$, uniform formula ϕ and action description predicate α:*

$$\mathcal{D} - \mathcal{D}_{S_0} \models \forall s \,.\, \phi[s] \rightarrow (\forall s' \,.\, s \leq_\alpha s' \rightarrow \phi[s'])$$

iff

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

Proof. A straightforward generalization of Lemma 5 in [Lin and Reiter, 1994], utilizing Proposition 1. □

Proposition 2 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 checking entailment of a uniform formula by the unique names axioms, a straightforward first-order reasoning task.

Next we must formalize some basic relationships between $\mathcal{P}_\mathcal{D}$ and $\mathcal{P}^n_\mathcal{D}$, as follows:

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

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

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

Proof. By the definition of $\mathcal{P}^n_\mathcal{D}$, any s' that falsifies the right-hand side of this equivalence will also falsify the left-hand side, and vice-versa. □

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

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

Proof. $\mathcal{P}_\mathcal{D}[\phi, \alpha]$ implies the persistence of ϕ by definition, which implies $\mathcal{P}^n_\mathcal{D}[\phi, \alpha]$ by Lemma 1. □

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

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

iff

$$\mathcal{D} - \mathcal{D}_{s_0} \models \forall s \,.\, \mathcal{P}^n_\mathcal{D}[\phi, \alpha][s] \equiv \mathcal{P}_\mathcal{D}[\phi, \alpha][s] \quad (2)$$

Proof. For the *if* direction, we first see by the definition of $\mathcal{P}^1_\mathcal{D}$ that equation (1) is equivalent to:

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

which by Proposition 2 implies the persistence of $\mathcal{P}^n_\mathcal{D}[\phi, \alpha]$ under α. By Lemma 1 this implies the persistence of ϕ under α, which in turn implies $\mathcal{P}_\mathcal{D}[\phi, \alpha]$, giving:

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

By Lemma 2 this implication is in fact an equivalence, yielding equation (2) as required.

The *only if* direction is a straightforward reversal of this reasoning: $\mathcal{P}_\mathcal{D}[\phi, \alpha]$ implies the persistence of ϕ, which implies the persistence of $\mathcal{P}^n_\mathcal{D}[\phi, \alpha]$, which implies (1) by Proposition 2. □

Since $\mathcal{P}^{n+1}_\mathcal{D}[\phi, \alpha] \rightarrow \mathcal{P}^n_\mathcal{D}[\phi, \alpha]$ by definition, equation (1) identifies $\mathcal{P}^n_\mathcal{D}[\phi, \alpha]$ as a fixed-point of the $\mathcal{P}^1_\mathcal{D}$ operator, as our initial intuition suggested.

3.2 Algorithm for the Persistence Condition

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

$$\mathcal{D}_{una} \models \forall s \,.\, \left(\mathcal{P}^n_{\mathcal{D}}[\phi, \alpha][s] \rightarrow \mathcal{P}^{n+1}_{\mathcal{D}}[\phi, \alpha][s]\right)$$

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.

Algorithm 1 Calculate $\mathcal{P}_{\mathcal{D}}[\phi, \alpha]$

```
pn ⇐ φ
pn1 ⇐ P^1_D[pn, α]
while D_una ⊭ ∀s . pn[s] → pn1[s] do
   pn ⇐ pn1
   pn1 ⇐ P^1_D[pn, α]
end while
return pn
```

Note that the calculation of $\mathcal{P}^1_{\mathcal{D}}[\phi, \alpha]$ is a straightforward syntactic transformation and so requires no further treatment.

Correctness

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

Note that equation (1) holds when $\mathcal{P}^n_{\mathcal{D}}[\phi, \alpha]$ is unsatisfiable for any situation, as it appears in the antecedent of the implication. The algorithm thus correctly returns an unsatisfiable condition (equivalent to $false$) when ϕ can never persist under α.

Completeness

As Theorem 1 is an equivalence, the only source of incompleteness will be failure to terminate. Algorithm 1 may fail to terminate for two reasons: the loop condition may never be satisfied, or the first-order logical inference in the loop condition may be undecidable and fail to terminate.

The later indicates that the basic action theory $\mathcal{D}$ 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. Thus, with respect to this source of incompleteness, our algorithm is no more incomplete than any larger system it would form a part of.

The former is of more direct consequence to our work, and raises two questions: is the persistence condition guaranteed to exist, and is it guaranteed to be reachable in a finite number of iterations? Since $\mathcal{P}^1_{\mathcal{D}}$ is clearly monotone, the constructive proof of Tarski's fixed-point theorem [Cousot and Cousot, 1979] guarantees the existence of a fixed-point that can be calculated via transfinite iteration, as performed by our algorithm. Moreover, this will be the least fixed-point greater than ϕ, a satisfying confirmation of correctness since the persistence condition must be the weakest precondition for the persistence of ϕ.

Unfortunately, there is no guarantee that this fixed-point can be reached via *finite* iteration, which is required for termination. Indeed, it is straightforward to construct a fluent for which the algorithm never terminates: consider a fluent $Fp(x, s)$ taking integers x, that is affected by a single action that makes it false whenever $Fp(x + 1, s)$ is false. Letting α be vacuously true, the sequence of iterations produced by our algorithm would be:

$$\mathcal{P}^1_{\mathcal{D}}[Fp(x, s)] \equiv Fp(x, s) \wedge Fp(x + 1, s)$$
$$\mathcal{P}^2_{\mathcal{D}}[Fp(x, s)] \equiv Fp(x, s) \wedge Fp(x + 1, s) \wedge Fp(x + 2, s)$$
$$\vdots$$
$$\mathcal{P}^n_{\mathcal{D}}[Fp(x, s)] \equiv \bigwedge_{i=0}^{i=n} Fp(x + i, s)$$

The persistence condition in this case is $\mathcal{P}_{\mathcal{D}}[Fp(x, s)] \equiv \forall y \,.\, x \leq y \rightarrow Fp(y, s)$, and while this is the limit of the iteration it is clearly unachievable in any finite number of steps.

$\mathcal{P}^1_{\mathcal{D}}$ operates over the set of equivalence classes of formulae uniform in s, and the theory of fixed-points requires that this set be a *well-founded partial order* to guarantee termination of an iterative approximation algorithm. There are certain classes of basic action theory for which this well-foundedness can be guaranteed. The most obvious is theories with finite action and object domains, in which the set of equivalence classes is finite. Another is the case where successor-state axioms refer only to the direct arguments of their fluents, in which case repeated applications of $\mathcal{P}^1_{\mathcal{D}}$ will pick out a subset of the equivalence classes that refers to a finite number of objects, giving the same guarantee.

In general, termination requires that repeated applications of the successor-state axioms (via the regression operator) do not construct infinite chains. We are currently investigating syntactic restrictions on successor state axioms that can enforce this requirement, and have found it to be easily met in practice. We are also investigating more advanced fixed-point algorithms that may give better efficiency and termination guarantees.

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 unique names axioms $\mathcal{D}_{una}$. Is this a worthwhile trade-off in practice? The following points weigh strongly in favor 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 many high-quality automated provers. Second, the calculation of $\mathcal{P}_{\mathcal{D}}$ only reasons based on the unique names axioms, which as discussed is a comparatively straightforward task. Third, $\mathcal{P}_{\mathcal{D}}[\phi, \alpha][s]$ 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 amortized 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 exponential in the length of ϕ. As with regression, our experience thus far has been that this is rarely a problem, and is more than compensated for by the reduced number of axioms required for reasoning.

3.3 Applications

The persistence condition is readily applicable to the persistence query problems given in Section 2.3. Since $\mathcal{P}_{\mathcal{D}}$ yields a uniform formula, the techniques outlined in Section 2.2 can be directly applied to the following transformed queries.

Goal Impossibility: Given a goal G, establish that there is no possible situation in which that goal is satisfied:

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

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

Goal Futility: Given a goal G and situation s, establish that the goal cannot be satisfied in any possible situation in the future of s:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}[\neg G, Poss][s]$$

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: This can be handled as per goal impossibility above. A state constraint ϕ should always satisfy:

$$\mathcal{D}_{una} \models \phi \equiv \mathcal{P}_{\mathcal{D}}[\phi, Poss]$$

If this is not the case then $\mathcal{P}_{\mathcal{D}}[\phi, Poss]$ indicates the additional conditions that are necessary to ensure that ϕ persists, which may be useful in adjusting the action theory to accommodate the constraint.

Need for Cooperation: To establish that it is absolutely necessary to cooperate with another agent, an agent must determine persistence with respect to actions performed by itself:

$$\mathcal{D} \models \mathcal{P}_{\mathcal{D}}[\neg G, OwnAction][s]$$

Knowledge under Partial Observability: In recent unpublished work, we have used the persistence condition to formulate a regression procedure for knowledge formulae when actions may be unobservable. To know ϕ, the agent must know that ϕ will persist under any actions that it would be unable to observe, resulting in a regression rule like the following:

$$\mathcal{R}_{\mathcal{D}}[\mathbf{Knows}(\phi, do(a, s))] = \mathbf{Knows}(\mathcal{R}_{\mathcal{D}}[\mathcal{P}_{\mathcal{D}}[\phi, CantObs][do(a, s)]], s)$$

4 Conclusions

We have developed an algorithm that transforms persistence queries, a very general and useful class of situation calculus query, to a form that is amenable to standard techniques for effective reasoning in the situation calculus. The algorithm is based on iterative application of the standard regression operator, and uses only first-order reasoning with a small number of axioms. Our algorithm is shown to be correct, and complete given some basic restrictions on the theory of action.

Our approach generalizes previous work on persistence in several important ways. It can consider sequences of actions satisfying a range of conditions, not just the standard ordering over possibility, enabling us to treat problems such as need for cooperation and knowledge under partial observability. 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. Finally, the fact that $\mathcal{P}_{\mathcal{D}}$ operates similarly to the standard regression operator $\mathcal{R}_{\mathcal{D}}$ allows them to interact in non-trivial ways, such as in the regression of knowledge formulae under partial observability of actions.

As a result, we have significantly increased the scope of queries that can be effectively posed in practical systems built on the situation calculus.

References

[Bertossi *et al.*, 1996] Leopoldo E. Bertossi, Javier Pinto, Pablo Saez, Deepak Kapur, and Mahadevan Subramaniam. Automating proofs of integrity constraints in situation calculus. In *International Syposium on Methodologies for Intelligent Systems*, pages 212–222, 1996.

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

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

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

[Reiter, 1991] Ray Reiter. The frame problem in situation the calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, pages 359–380, San Diego, CA, USA, 1991. Academic Press Professional, Inc.

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