University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Kelly, RF;Pearce, AR

Title:
Asynchronous knowledge with hidden actions in the situation calculus

Date:
2015-01-01

Citation:
Kelly, R. F. & Pearce, A. R. (2015). Asynchronous knowledge with hidden actions in the situation calculus. Artificial Intelligence, 221, pp.1-35. https://doi.org/10.1016/j.artint.2014.12.005.

Persistent Link:
https://hdl.handle.net/11343/54840

Asynchronous Knowledge with Hidden Actions in the Situation Calculus

Ryan F. Kelly, Adrian R. Pearce[1,*]

Department of Computing and Information Systems

The University of Melbourne

Victoria, 3010, Australia

Abstract

We present a powerful new account of multi-agent knowledge in the situation calculus and an automated reasoning procedure for knowledge queries. Existing accounts of epistemic reasoning in the situation calculus require that whenever an action occurs, all agents *know* that an action has occurred. This demands a level of synchronicity that is unreasonable in many multi-agent domains. In asynchronous domains, each agent's knowledge must instead account for arbitrarily-long sequences of *hidden actions*. By using a persistence condition meta-operator to augment traditional regression techniques, we show how agents can reason about their own knowledge using only their internal history of observations, rather than requiring a full history of the world. The result is a more robust and flexible account of knowledge in the situation calculus suitable for asynchronous, partially-observable multi-agent domains.

Keywords: Situation calculus, Knowledge, Epistemic reasoning

1. Introduction

In their landmark paper *Knowledge, Action, and the Frame Problem,* Scherl and Levesque [45] incorporate knowledge-producing actions into the

*Corresponding Author

Email addresses: ryan@rfk.id.au (Ryan F. Kelly), adrianrp@unimelb.edu.au (Adrian R. Pearce)

[1]Tel: +61 3 8344 1399, Fax: +61 3 9348 1184

situation calculus [37], inheriting Reiter's solution to the frame problem [38] and so enabling use of the regression operator to reason about the changing knowledge of an agent. Extensions to multiple agents [48] and concurrent actions [44] have produced an expressive logic of knowledge, action and change in which regression provides an automated reasoning procedure

While powerful, this formalism has a restriction that can make it unsuitable for modelling complex multi-agent domains. It requires that whenever an action occurs, all agents *know* that an action has occurred, demanding a level of synchronicity that is unreasonable in many multi-agent domains. If this restriction is lifted then each agent's knowledge must account for arbitrarily-long sequences of hidden actions [21], and proofs about knowledge must use a second-order induction axiom for quantifying over all future situations. This precludes the use of regression for automated reasoning. It also makes it difficult for agents to reason about their own knowledge, as they may not have enough information to formulate an appropriate query.

We overcome this limitation by combining two elements – an explicit representation of an agent's local perspective and a persistence condition meta-operator – to formulate an account of knowledge in the situation calculus that can faithfully represent the hidden actions inherent in asynchronous domains while maintaining regression as a key tool for automated reasoning.

To decouple knowledge from the specific actions that have occurred, we explicitly reify the local *observations* made by each agent, so that every situation corresponds to an agent-local *view*. An agent is said to know proposition ϕ if and only if ϕ is true in all situations matching its current view. Our work thus has strong parallels with the classic view-based account of knowledge of Halpern and Moses [15], but grounded in the situation calculus and with an emphasis on regression for automated reasoning. This work appeared in preliminary form in [17]; the current paper presents an expanded treatment including proofs, properties and application domains.

The main challenge we overcome is developing regression rules that can handle arbitrarily long sequences of hidden actions. To ensure its knowledge is valid, an agent must reason about all future situations that are compatible with its observations. Such universal quantification over situation terms requires a second-order induction axiom, which the standard regression operator cannot handle. In previous work we have developed the *persistence condition* operator to handle this induction as a meta-level fixpoint calculation [19]. Using this operator to augment the regression techniques developed by Scherl and Levesque [45], we maintain their elegant solution to the frame

problem while handling arbitrarily long sequences of hidden actions.

The formulation is shown to respect basic intuitions about how knowledge should behave, and to preserve important properties of the agent's epistemic state through the occurrence of actions. Moreover, it is *elaboration tolerant*, automatically preserving these properties in the face of more complex information-producing actions, such as the guarded sensing actions of [32], that can be tricky to axiomatize correctly in existing formalisms.

Decoupling knowledge from action in this manner makes it easy to express varying degrees of observability, from actions that are public through to ones that are completely hidden. To illustrate, we present a running example based on the well-known "Hunt the Wumpus" domain [41] which has been previously used to demonstrate the interleaved sensing and action typical of realistic domains in the situation calculus [42]. Our variant is complicated by the presence of multiple agents and partial observability of actions:

> Ann and Bob are hunting a Wumpus in a dungeon with many interconnecting rooms. They can fully observe each other's actions if they are in the same room, can hear each other's actions from adjacent rooms, and have no other means of synchronisation.
>
> Like any Wumpus, this one does not move, causes a stench in all adjacent rooms, and if shot will emit a piercing scream that can be heard anywhere in the dungeon.
>
> Can Ann and Bob coordinate their knowledge and actions in order to find and shoot the Wumpus?

The possibility of hidden actions makes this domain difficult to represent, let alone reason about, in standard theories of knowledge in the situation calculus. Our approach offers a straightforward formulation and an automated reasoning procedure.

Further demonstrating the utility of our approach, we show how the new regression rules can be applied using an agent's individual view, rather than requiring a full situation term. Agents can thus use our techniques to reason about their own knowledge using only their local information, making the formalism suitable both for reasoning *about*, and for reasoning *in*, rich multi-agent domains.

The end result is a significantly more general and robust theory of knowledge in the situation calculus that still permits an automated reasoning procedure. There is a large body of work that could benefit from our formalism,

including: specification and verification of multi-agent systems [52]; theories of coordination [13] and ability [22]; reasoning about the epistemic feasibility of plans [23]; analysing multi-player games [1]; and our own work on the computation of complex epistemic modalities [18] and the cooperative execution of Golog programs [16].

The paper now proceeds with a review of the standard account of multi-agent knowledge in the situation calculus, before treating the individual knowledge of each agent in the face of hidden actions. We develop the axioms for our new observation-based account of knowledge in Section 3 and develop a regression rule for our formalism using the persistence condition operator in Section 4. Potential applications are shown in Section 5, where we show how our approach to axiomatising observations overcomes several difficulties encountered in previous formulations. An illustrative example of its use for reasoning about a partially-observable domain is provided in Section 6.

2. Background

Our work utilises the situation calculus [28, 37] with multiple agents [48] and concurrent actions [40], and we begin from the standard account of knowledge due to Scherl and Levesque [45]. Several conservative extensions to the standard situation calculus meta-theory are also employed: the *Poss* predicate is subsumed by a general class of *action description predicates*; the unique names axioms for actions are subsumed by a general *background theory*; reasoning is performed using the *single-step regression operator* along with a new reasoning tool called the *persistence condition* operator [19].

There are of course a wide range of related formalisms for reasoning about knowledge, action and change, which we do not directly consider in this paper. We find the notation and meta-theory of the situation calculus particularly suitable for expressing our main ideas. Moreover, the strong underlying similarities between the major action formalisms should allow these ideas to transcend the specifics of the situation calculus [46, 53, 54].

2.1. The Situation Calculus for Multiple Agents

The situation calculus is a many-sorted language of first-order logic augmented with a second-order induction axiom. Its has the following sorts: AGENT terms represent the agents operating in the world; ACTION terms are functions denoting individual instantaneous events that can cause the state of the world to change, with the initiating agent indicated by their first

argument; CONCURRENT terms are non-empty, finite sets of actions that occur simultaneously; 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 $do : Concurrent \times Situation \rightarrow Situation$; OBJECT terms represent any other object in the domain. *Fluents* are predicates or functions that represent properties of the world that may change between situations; they take a situation term as their final argument.

We follow standard naming conventions for the situation calculus: lowercase Roman names indicate variables; upper-case Roman names indicate constants; Greek letters indicate meta-variables and meta-operators; macros are presented in bold face; all axioms universally close over their free variables. The connectives $\wedge$, $\neg$, $\exists$ are taken as primitive, with $\vee$, $\rightarrow$, $\equiv$, $\forall$ defined in the usual manner. The name s is used for variables of sort SITUATION, a for sort ACTION, c for sort CONCURRENT, agt for sort AGENT and x for sort OBJECT. The notation $\bar{x}$ indicates a vector of terms of context-appropriate sort and arity.

Some example statements from the "Hunt the Wumpus" domain are "Initially everyone is in the first room of the dungeon, $R1$" and "Bob is not in room $R1$ after moving to room $R2$". Written formally:

$$\forall agt : In(agt, R1, S_0)$$
$$\neg In(Bob, R1, do(\{move(Bob, R2)\}, S_0))$$

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 1 (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 to constitute a valid domain description; see [37] for details. This is a conservative extension of the definition used by Pirri and Reiter, designed to accommodate our forthcoming extensions.

The *uniform formulae* can be thought of as properties of the state of the world, and are basically logical combinations of fluents referring to a common situation term. The meta-variable ϕ is used to refer to an arbitrary uniform formula. For the moment we restrict ourselves to *objective* uniform formulae; the complete definition includes statements about knowledge and will be introduced in the next section.

Definition 2 (Uniform Terms)**.** *Let σ be a fixed situation term, r an arbitrary rigid*[2] *function symbol, f an arbitrary fluent function symbol, and x a variable that is not of sort* SITUATION*. Then the terms uniform in σ are the smallest set of syntactically-valid terms satisfying:*

$$\tau ::= x \mid r(\bar{\tau}) \mid f(\bar{\tau}, \sigma)$$

Definition 3 (Objective Uniform Formulae)**.** *Let σ be a fixed situation term, R an arbitrary rigid predicate, F an arbitrary primitive fluent predicate, τ an arbitrary term uniform in σ, and x an arbitrary variable that is not of sort* SITUATION*. Then the objective formulae uniform in σ are the smallest set of syntactically-valid formulae satisfying:*

$$\phi ::= F(\bar{\tau}, \sigma) \mid R(\bar{\tau}) \mid \tau_1 = \tau_2 \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x : \phi$$

Since they represent properties of the world, it is often useful to evaluate uniform formulae at several different situation terms, and to suppress the situation terms in order to simplify the presentation. The notation $\phi[s']$ represents a uniform formula in which all fluents have their situation argument replaced with the particular situation term s', while ϕ^{-1} represents a uniform formula with the situation argument removed from all its fluents.

The axiom set $\mathcal{D}_{bg}$ contains all the static background facts about the domain. It must include: the definitions for sets and set membership; unique names axioms to ensure that action terms with different names or different arguments are in fact different; and the function $actor(a)$ giving the agent

[2]Fluents are predicates or functions that change as a result of actions, and hence take a situation as their final argument; rigids do not change and take no situation argument.

performing an action, which is always the action's first argument. It may also contain domain-specific background facts, such as situation-independent formulae or simple state invariants.

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. These axioms have the following general form:

$$F(\bar{x}, do(c, s)) \equiv \Phi_F^+(\bar{x}, c, s) \ \vee \ F(\bar{x}, s) \wedge \neg\Phi_F^-(\bar{x}, c, s)$$

Here Φ_F^+ and Φ_F^- are formulae uniform in s. This may be read as "F is true after performing c if c made it true, or it was previously true and c did not make it false".

The axiom set $\mathcal{D}_{ad}$ defines fluents that describe various aspects of the performance of an action, which we call *action description predicates*. The precondition predicate $Poss(c, s)$ is the canonical example, indicating whether it is possible to perform a set of actions c in a given situation s. In principle there can be any number of predicates or functions defined in a similar way – forthcoming examples include the sensing-result function SR and the observability predicate $CanObs$. The meta-predicate α will denote an arbitrary action description predicate. For each such predicate the set $\mathcal{D}_{ad}$ contains a single axiom of the following form, where Π_α is uniform in s:

$$\alpha(\bar{x}, c, s) \ \equiv \ \Pi_\alpha(\bar{x}, c, s)$$

Note that this is a departure from the standard approach of Pirri and Reiter [37] where a separate axiom specifies the preconditions for each individual action type. The single-axiom approach used in this paper embodies a domain-closure assumption on the ACTION sort. It is necessary when reasoning about formulae that universally quantify over actions [43, 58], and is assumed to appropriately axiomatise any interactions that may occur between primitive actions when they are performed concurrently [36].

For convenience we will allow action description predicates to be defined in terms of other action description predicates; as long as these definitions are well-founded they can be expanded out to a definition that uses only uniform formulae.

The foundational axioms Σ capture axiomatically the intuition that situations are finite sequences of actions. There are *initial situations* identified by the predicate $Init(s)$, with a distinguished initial situation S_0 called the *actual* initial situation. Other initial situations are used to represent the

agents' epistemic uncertainty, and will be discussed in the following section. Situations in general form a tree structure with an initial situation at the root and $do(c, s)$ constructing the successor situation resulting when the actions c are performed in s; all situations thus produced are distinct:

$$do(c_1, s_1) = do(c_2, s_2) \rightarrow c_1 = c_2 \wedge s_1 = s_2$$

We abbreviate the performance of several successive actions by writing:

$$do([c_1, \ldots, c_n], s) \stackrel{\text{def}}{=} do(c_n, do(\ldots, do(c_1, s)))$$

There is also a second-order induction axiom asserting that all situations must be constructed in this way, which is needed to prove statements that universally quantify over situations [39]:

$$(\forall P) : [\forall\, c, s : (Init(s) \rightarrow P(s)) \wedge (P(s) \rightarrow P(do(c, s)))] \rightarrow \forall s : P(s)$$

The *root* of a situation is the initial situation from which it was constructed:

$$Init(s) \rightarrow root(s) = s$$
$$root(do(c, s)) = root(s)$$

The relation $s \sqsubset s'$ indicates that s' is in the future of s and is defined as follows:

$$Init(s) \rightarrow \neg(s' \sqsubset s)$$
$$s \sqsubset do(c, s') \equiv s \sqsubseteq s'$$

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 $<_{\alpha(\bar{x})}$ for each action description predicate $\alpha(\bar{x}, c, s)$, with the following definitions:

$$Init(s) \rightarrow \neg\left(s' <_{\alpha(\bar{x})} s\right)$$
$$s <_{\alpha} do(c, s') \equiv s \leq_{\alpha(\bar{x})} s' \wedge \alpha(\bar{x}, c, s')$$

Note that we suppress the action and situation arguments to the action description predicate in order to simplify the presentation. For notational

simplicity we will henceforth leave any additional arguments $\bar{x}$ implicit, writing just $<_\alpha$ for a generic relation of this kind.

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 [37] in the sense that the *legal situations* are those in which every action performed was actually possible:

$$Legal(s) \equiv root(s) \leq_{Poss} s$$

Finally, the axiom set $\mathcal{D}_{S_0}$ describes the state of the world before any actions are performed. It is a collection of sentences uniform in S_0 stating what holds in the actual initial situation. The initial knowledge of each agent is also captured in these axioms, as described in the next section.

2.2. Knowledge and Sensing

Epistemic reasoning was first introduced to the situation calculus by Moore [29], and formalised extensively by Scherl and Levesque [45] whose paper is now the canonical reference for these techniques. Their work has been extended to include concurrent actions [44] and multiple agents [48].

The semantics of knowledge are based on a reification of the "possible worlds" semantics of modal logic, using situation terms rather than abstract worlds. A special fluent $K(agt, s', s)$ is used to indicate that "in situation s, the agent agt considers the alternate situation s' to be possible". The macro **Knows** is then defined as a shorthand for the standard possible-worlds definition of knowledge, stating that an agent knows ϕ when ϕ is true in all situations considered possible:

$$\mathbf{Knows}(agt, \phi, s) \stackrel{\text{def}}{=} \forall s' : K(agt, s', s) \rightarrow \phi[s'] \tag{1}$$

The foundational axioms Σ define a special fluent $K_0(agt, s', s)$ that is used to model the initial epistemic uncertainty of the agents, with $\mathcal{D}_{S_0}$ containing sentences of the form $\mathbf{Knows_0}(agt, \phi, S_0)$ to specify what is initially known[3]:

$$K_0(agt, s', s) \rightarrow Init(s) \wedge Init(s')$$
$$\mathbf{Knows_0}(agt, \phi, s) \stackrel{\text{def}}{=} \forall s' : K_0(agt, s', s) \rightarrow \phi[s']$$

[3]The standard account does not require a separate K_0 fluent, as evidenced by Equation (2). It will be required when we incorporate hidden actions, so we introduce it now to maintain consistency.

The action description function $SR(a, s)$ specifies the sensing result returned by a when executed in situation s. For non-sensing actions the value of SR is an arbitrary constant. The dynamics of knowledge are then specified by an additional set of axioms.

Definition 4. *We will denote by $\mathcal{D}_K^{std}$ the axioms of the standard account of knowledge due to Scherl and Levesque [45], as detailed in Equations (2,3) below:*

$$Init(s) \rightarrow (K(agt, s', s) \equiv K_0(agt, s', s)) \tag{2}$$

$$\begin{aligned} K(agt, s'', do(c, s)) \equiv & \exists s' : s'' = do(c, s') \wedge K(agt, s', s) \wedge Poss(c, s') \\ & \wedge \forall a \in c : (actor(a) = agt \rightarrow SR(a, s) = SR(a, s')) \end{aligned} \tag{3}$$

Equation (2) ensures that the agents begin with their knowledge as specified by $\mathcal{D}_{S_0}$. Equation (3) takes the form of a standard successor state axiom for the K fluent. It ensures that s'' is considered a possible alternative to $do(c, s)$ when s'' is the result of doing the same actions c in a situation s' that is considered a possible alternative to s. It must furthermore have been possible to perform those actions in s', and the sensing results must match for each action that was carried out by the agent in question. Thus an agent's knowledge after an action occurs is completely determined by its knowledge before the action, and the sensing results from the action.

Scherl and Levesque [45] show how the **Knows** macro can be treated syntactically as if it were a primitive fluent, allowing it to appear as part of a uniform formula for the purposes of reasoning. We can now present the complete definition of a uniform formula, which may contain statements about knowledge:

Definition 5 (Uniform Formulae)**.** *Let σ be a fixed situation term, let τ, τ_1 and τ_2 be arbitrary terms uniform in σ and x an arbitrary variable that is not of sort* SITUATION*. Then the formulae uniform in σ are the smallest set of syntactically-valid formulae satisfying:*

$$\begin{aligned} \phi ::= & F(\bar{\tau}, \sigma) \mid R(\bar{\tau}) \mid \tau_1 = \tau_2 \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x : \phi \\ & \mid \mathbf{Knows}(agt, \phi, \sigma) \mid \mathbf{Knows_0}(agt, \phi, \sigma) \end{aligned}$$

While powerful, this knowledge-representation formalism has an important limitation: it is fundamentally *synchronous.* Each agent is assumed to have full knowledge of all actions that have occurred - in other words, all actions are public. While suitable for some domains, there are clearly many multi-agent domains where achieving total awareness of actions would be infeasible. A major contribution of this paper is a more flexible formalism for knowledge that can be applied to a much wider range of domains.

2.3. Reasoning and Regression

One of the attractions of the situation calculus is the existence of automated reasoning procedures for certain types of query. These are generally based on syntactic manipulation of a query into a form that is more amenable to reasoning – for example, because it can be proven without using some of the axioms from $\mathcal{D}$.

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 included in the foundational axioms. This is clearly problematic for automated reasoning, but fortunately there exist particular syntactic forms for which some of the axioms in $\mathcal{D}$ are not required [37]. In particular, queries about the initial situation can be answered using only first-order logic (FOL) and a limited set of axioms:

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

By making a closed-world assumption over $\mathcal{D}_{S_0} \cup \mathcal{D}_{bg}$, logic programming environments such as Prolog can be used to handle this type of query quite effectively [38]. Automated reasoning depends on transforming queries into more easily-handled forms such as this.

The principle tool for automated reasoning in the situation calculus is the regression meta-operator $\mathcal{R}_{\mathcal{D}}$, 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 [37]. The idea is to reduce a query about some future situation to a query about the current situation only.

Regression is only defined for a certain class of formulae, the *regressable formulae.*

Definition 6 (Regressable Terms)**.** *Let σ be an arbitrary situation term, x an arbitrary variable not of sort situation, r an arbitrary rigid function and*

f an arbitrary fluent function. Then the regressable terms are the smallest set of syntactically-valid terms satisfying:

$$\nu ::= \sigma \mid x \mid f(\bar{\nu}, \sigma) \mid r(\bar{\nu})$$

Definition 7 (Regressable Formulae)**.** *Let σ be an arbitrary situation term, x an arbitrary variable not of sort situation, ν an arbitrary regressable term, R an arbitrary rigid predicate, F an arbitrary primitive fluent predicate, α an arbitrary action description predicate and c a set of actions. Then the regressable formulae are the smallest set of syntactically-valid formulae satisfying:*

$$\varphi ::= F(\bar{\nu}, \sigma) \mid \alpha(\bar{\nu}, c, \sigma) \mid R(\bar{\nu}) \mid \nu_1 = \nu_2 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x : \varphi$$

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 a $\leq_\alpha$ relation.

The regression operator is then defined using a series of *regression rules* such as those shown below, which mirror the structural definition of regressable formulae.

Definition 8 (Regression Operator)**.** *Let R be a rigid predicate, α be an action description predicate with axiom $\alpha(\bar{\nu}, c, s) \equiv \Pi_\alpha(c, s)$ in $\mathcal{D}_{ad}$, and F be a primitive fluent with axiom $F(\bar{x}, s) \equiv \Phi_F(\bar{x}, s)$ in $\mathcal{D}_{ssa}$. Then the regression of ϕ, denoted $\mathcal{R}_\mathcal{D}(\phi)$, is defined according to the following structural rules:*

$$\begin{aligned}
\mathcal{R}_\mathcal{D}(\varphi_1 \wedge \varphi_2) &= \mathcal{R}_\mathcal{D}(\varphi_1) \wedge \mathcal{R}_\mathcal{D}(\varphi_2) \\
\mathcal{R}_\mathcal{D}(\exists x : \varphi) &= \exists x : \mathcal{R}_\mathcal{D}(\varphi) \\
\mathcal{R}_\mathcal{D}(\neg\varphi) &= \neg\mathcal{R}_\mathcal{D}(\varphi) \\
\mathcal{R}_\mathcal{D}(\alpha(\bar{\nu}, c, \sigma)) &= \mathcal{R}_\mathcal{D}(\Pi_\alpha(\bar{\nu}, c, \sigma)) \\
\mathcal{R}_\mathcal{D}(F(\bar{\nu}, do(c, \sigma))) &= \Phi_F(\bar{\nu}, c, \sigma) \\
\mathcal{R}_\mathcal{D}(F(\bar{\nu}, s)) &= F(\bar{\nu}, s) \\
\mathcal{R}_\mathcal{D}(F(\bar{\nu}, S_0)) &= F(\bar{\nu}, S_0)
\end{aligned}$$

We have omitted some technical details here, such as the handling of functional fluents; consult [37] for the details. The key point is that each application of the regression operator replaces action description predicates

with their definitions from $\mathcal{D}_{ad}$ and primitive fluents with their successor state axioms from $\mathcal{D}_{ssa}$, "unwinding" a single action from each $do(c, \sigma)$ term in the query. Situation terms not constructed using *do* are left unchanged.

Since $\mathcal{D}$ is fixed, we will henceforth drop the subscript and simply write $\mathcal{R}$ for the regression operator. When dealing with situation-suppressed uniform formulae, we will use a two-argument operator $\mathcal{R}(\phi, c)$ to indicate the regression of ϕ over the action c. It should be read as a shorthand for $\mathcal{R}(\phi[do(c, s)])^{-1}$ using the situation-suppression operator from Section 2.1.

Repeated applications of this operator, denoted by $\mathcal{R}^*$, can transform a query about some future situation into a query about the initial situation only. This is typically easier to answer as it requires fewer axioms from the theory, as shown in [37]. The axioms $\mathcal{D}_{ad}$ and $\mathcal{D}_{ssa}$ are essentially "compiled into" the query. The trade-off is that the length of the regressed query may be exponential in the length of ϕ. While an efficiency gain is not guaranteed, regression has proven a very effective technique in practice [24] and there are techniques to avoid exponential growth of the query in some cases [57].

A key contribution of Scherl and Levesque [45] was showing how to apply the regression operator to formulae containing the **Knows** macro, allowing it to be treated syntactically as if it were a primitive fluent. This means that epistemic queries can be approached using standard reasoning techniques of the situation calculus. Although we have changed the notation somewhat to foreshadow the techniques we will develop in Section 3, their definition operates as follows.

First, we introduce a notational convenience for pairing an action with its corresponding sensing result:

Definition 9 (ACTION#RESULT PAIR)**.** *An* ACTION#RESULT PAIR*, denoted $a\#r$, represents an* ACTION *term a and a corresponding* RESULT *term r as might be returned from the sensing-result function $r = SR(a, s)$. We require unique-names axioms for the pairing operator:*

$$a\#r = a'\#r' \equiv a = a' \wedge r = r'$$

We will use the special RESULT *term Nil as the sensing result for non-information-producing actions.*

We then define the *results* of a concurrent action to be the set of *action#result* pairs for all primitive actions performed by the agent in question.

$$\mathbf{res}(agt, c, s) \stackrel{\text{def}}{=} \{a\#SR(a, s) \mid a \in c \wedge actor(a) = agt\}$$

This definition can be used to formulate a regression rule (denoted $\mathcal{R}_{(SL)}$ to distinguish it from our subsequent rule) as follows:

$$\mathcal{R}_{(SL)}(\mathbf{Knows}(agt, \phi, do(c, s)) = \quad \exists y : y = \mathbf{res}(agt, c, s) \\ \wedge \mathbf{Knows}(agt, [Poss(c) \wedge \mathbf{res}(agt, c) = y] \rightarrow \mathcal{R}_{(SL)}(\phi[do(c, s)]), s) \quad (4)$$

This works by collecting the sensing results from each action performed by the agent into the set y, then ensuring matching sensing results in every situation considered possible. It expresses the knowledge of the agent after a concurrent action in terms of what it knew before the action, along with the information returned by the action. This technique relies heavily on the fact that all actions are public, since it requires every agent's knowledge to be updated in response to every action.

Repeated applications of $\mathcal{R}$ can thus transform a knowledge query into one that is uniform in the initial situation. While it would be valid to then expand the **Knows** macros and handle the query using first-order logic, in practice the reasoning procedure would leave **Knows** intact and use a specialised prover based on modal logic.

It is possible to formulate an alternate successor state axiom for knowledge that does not assume all actions are public, such as that of Lespérance et al. [21]. Such formulations invariably require universal quantification over situation terms, to account for arbitrarily-long sequences of hidden actions. This is incompatible with regression rules like the above, and these formulations offer no reasoning procedure other than general second-order theorem proving. By utilising a new reasoning technique called the *persistence condition* to factor out this universal quantification, our work is the first to provide an account of knowledge with hidden actions while maintaining regression in the style presented above as an effective reasoning tool.

2.4. Property Persistence and the Persistence Condition

Queries that universally quantify over situation terms are often in a simple syntactic form called *property persistence* queries [19]. Such queries assert that a uniform formula ϕ holds in a given situation, and will continue to hold as long as all future actions satisfy some action description predicate α:

$$\forall s' : s \leq_\alpha s' \rightarrow \phi[s']$$

In [19] we developed the *persistence condition* meta-operator $\mathcal{P}_\mathcal{D}(\phi, \alpha)$ to handle such queries more effectively. It is a meta-level fixpoint calculation

producing a uniform formula such that:

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

As with regression, we will use the simpler notation $\mathcal{P}(\phi, \alpha)$ and leave $\mathcal{D}$ implicit. We briefly describe its operation below since we will need some details for a worked example; for a full treatment see our previous work [19].

First, the one-step-persistence operator $\mathcal{P}^1$ is defined to assert that ϕ holds in s and all its immediate successors:

$$\mathcal{P}^1(\phi, \alpha)[s] = \phi[s] \wedge \forall c : \alpha(c, s) \rightarrow \mathcal{R}(\phi, c)[s]$$

Repeated application asserts persistence after greater numbers of actions:

$$\mathcal{P}^n(\phi, \alpha) = \mathcal{P}^1(\mathcal{P}^{n-1}(\phi, \alpha), \alpha)$$

Intuitively $\mathcal{P}(\phi, \alpha)$ should be a fixpoint of $\mathcal{P}^1(\phi, \alpha)$, and indeed it can be shown that:

$$\mathcal{D}_{bg} \models \mathcal{P}^n(\phi, \alpha) \rightarrow \mathcal{P}^{n+1}(\phi, \alpha) \quad \text{iff} \quad \mathcal{D} - \mathcal{D}_{S_0} \models \mathcal{P}^n(\phi, \alpha) \equiv \mathcal{P}(\phi, \alpha)$$

The calculation of $\mathcal{P}(\phi, \alpha)$ thus replaces a second-order reasoning task with a fixpoint search, using iterated first-order reasoning against the background theory. This is in contrast to the regression operator, which is a purely syntactic transformation, and so it comes with two important caveats: the fixpoint $\mathcal{P}(\phi, \alpha)$ may not exist at all, or it may be impossible to calculate within finitely many iterations.

Nevertheless, our experience has shown this technique to be useful in practice, thanks to the relative simplicity of the axioms in $\mathcal{D}_{bg}$. Claßen and Lakemeyer [4] have also had success with a similar approach in a modal variant of the situation calculus know as $\mathcal{ES}$, where they study fixpoint properties of non-terminating Golog programs.

Our paper [19] discusses several restricted domains in which a terminating calculation of $\mathcal{P}(\phi, \alpha)$ can be guaranteed, including finite domains (which can be reduced to propositional logic) and certain kinds of context-free domains (which can be shown to have a finite state-space). We shall have more to say on the effectiveness of this technique in Section 4.

The important point is that $\mathcal{P}(\phi, \alpha)$ allows one to factor out certain kinds of universally-quantified queries, dealing with them via special-purpose meta-level reasoning machinery, and reducing them to a uniform formula that can then be approached via standard regression techniques.

3. Observations, Views and Knowledge

Existing accounts of knowledge in the situation calculus directly express the dynamics of knowledge update in terms of the actions that have occurred, as seen in Section 2.2. This works well when agents can be assumed to have full knowledge of the actions that have been performed, but quickly becomes cumbersome when trying to allow for hidden actions. In this Section we first develop a principled axiomatisation of the *observability* of actions, then build a powerful yet succinct axiomatisation of knowledge upon it.

The basic idea is as follows: each occurrence of an action results in an agent making a set of *observations*. Every situation then corresponds to a local *view* for that agent: the sequence of all its observations, excluding cases where the set of observations was empty. An agent knows something if it is true in all situations matching its current view. Decoupling knowledge in this manner makes it easy to express various degrees of observability of actions, from public actions through to actions that are completely hidden.

The direct coupling between knowledge and action also has undesirable implications for situated agents reasoning about their own knowledge. As a consequence of using regression to handle knowledge queries, one can only reason about knowledge if one has a situation term rooted in S_0, as the required query is:

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models \mathcal{R}^*(\mathbf{Knows}(agt, \phi, do([c_1, \ldots, c_n], S_0)))$$

In asynchronous domains with hidden actions, where agents are not necessarily aware how many actions have been performed, the agents cannot use this formulation to reason about their own knowledge. Since knowledge is directly coupled to actions, and hence to situation terms, they cannot construct the appropriate query.

Our approach will allow this kind of query to be posed in terms of an agent's local view, rather than requiring a full situation term, allowing them to reason about their own knowledge using only their local information

3.1. Observations and Views

To remove the direct coupling between knowledge and actions, we introduce an explicit notion of *observations*. These are internal notifications that agents receive when an action has occurred. By expressing knowledge in terms of observations rather than actions, we ensure that agents can always reason about their own knowledge based on their own internal history of observations.

Definition 10 (Observations)**.** *An observation is a notification event received by an agent, making it aware of some change in the state of the world. When an agent receives such a notification, we may say that it "observed", "made" or "perceived" that observation.*

Since the state of the world may only change in response to some action, observations can only be made as a result of some action. For simplicity we assume that agents perceive observations instantaneously, i.e. in the same instant as the actions that cause them.

Since "observation" is quite a loaded term, it is worth re-iterating this point: observations are instantaneous *events* generated internally by each agent in response to actions occurring in the world. We make no commitment as to how these events are generated, preferring a clean delineation between the task of observing change and the dynamics of knowledge update based on those observations. As with the work of Scherl and Levesque [45], we consider change-awareness to be the responsibility of a lower-level component of the agent's control software.

To make this idea concrete, let us introduce an additional sort OBSERVATIONS to the language of the situation calculus, for the moment without any particular commitment towards what this sort will contain. We then add to $\mathcal{D}_{ad}$ an axiom for the action description function Obs:

$$Obs(agt, c, s) = o$$

This function returns a finite set of observations, and should be read as "when actions c are performed in situation s, agent agt will perceive o". Using a set of observations allows an agent to perceive any number of observations in response to an action occurrence - perhaps several observations, perhaps none. When $Obs(agt, c, s)$ is the empty set (denoted $\emptyset$) then the agent makes no observations and the actions c are completely hidden.

The concept of a *view* follows naturally - it is the sequence of all the observations made by an agent as the world has evolved.

Definition 11 (Views)**.** *An agent's view in a given situation* s *is the corresponding sequence of observations made by the agent as a result of each action in* s*, excluding those actions for which no observations were made.*

We introduce another sort VIEWS consisting of sequences of finite sets of observations. Using ϵ to represent the empty sequence and $\cdot$ to prepend a

new element to a sequence[4] , the functional fluent $View$ is defined as follows to give the history of observations associated with a particular situation:

$$\begin{aligned} Init(s) &\rightarrow View(agt, s) = \epsilon \\ Obs(agt, c, s) = \emptyset &\rightarrow View(agt, do(c, s)) = View(agt, s) \\ Obs(agt, c, s) \neq \emptyset &\rightarrow View(agt, do(c, s)) = Obs(agt, c, s) \cdot View(agt, s) \end{aligned} \quad (5)$$

Observations and views can be seen as localised analogues of actions and situations respectively. An action is a global event that causes the state of the world to change, while an observation is an internal event that causes an agent's knowledge of the state of the world to change. Similarly, a situation represents a complete, global history of all the actions that have occurred in the world, while a view is an agent's local history of all the observations it has made. The situation is an omniscient perspective on the world, the view a local perspective. This distinction is fundamental to developing a truly general multi-agent semantics for knowledge, as we shall see.

The axiomatisation of Obs will depend on the particulars of the domain being modelled, and we make no commitment beyond requiring it to be formulated as an action description predicate. Under the common assumption that all actions are public, it may be as simple as defining:

$$a \in Obs(agt, c, s) \equiv a \in c$$

With sensing actions included, Obs may be extended to include Action#Result pairs so that:

$$\begin{aligned} a\#r \in Obs(agt, c, s) &\equiv a \in c \\ \wedge\, (actor(a) = agt &\rightarrow r = SR(a, s)) \\ \wedge\, (actor(a) \neq agt &\rightarrow r = Nil) \end{aligned}$$

For an asynchronous domain where all actions are private, Obs may instead be defined to include only the actions of the agent in question:

$$a\#r \in Obs(agt, c, s) \equiv a \in c \wedge actor(a) = agt \wedge SR(a, s) = r$$

We will discuss some more detailed axiomatisations for Obs in Section 5. The main point, as we shall see, is that the particular axiomatisation of Obs does not affect the general axioms and behaviour of knowledge.

[4]We rely on the standard semantics of sequences for function symbol '$\cdot$'; specifically we assume it excludes non-standard infinite sequences that do not grow from ϵ.

3.2. Knowledge

It is a basic tenet of epistemic reasoning that an agent's knowledge at any particular time must depend solely on its local history: the knowledge that it started out with combined with the observations it has made since then [15]. Given an explicit account of the observations made by each agent, the required semantics of the K relation are clear: $K(agt, s', s)$ must hold whenever s' is legal, both s and s' would result in the same view for the agent, and s and s' are rooted at K-related initial situations:

$$K(agt, s', s) \equiv \\ Legal(s') \wedge View(agt, s') = View(agt, s) \wedge K_0(agt, root(s'), root(s)) \quad (6)$$

In essence, this is a direct encoding into the situation calculus of the definition of knowledge in classical epistemic reasoning [30, 15, 10].

While a wonderfully succinct definition of how knowledge should behave, this formulation cannot be used directly in a basic action theory. The dynamics of fluent change must be specified by a successor state axiom, expressing K at $do(c, s)$ in terms of K at situation s. We must formulate a successor state axiom for the K fluent which enforces the equivalence in Equation 6.

For notational convenience, let us first introduce an action description predicate **PbU** (for "possible but unobservable") indicating that the actions c are possible in s, but no observations will be made by agt if they occur:

$$\mathbf{PbU}(agt, c, s) \stackrel{\text{def}}{=} Poss(c, s) \wedge Obs(agt, c, s) = \emptyset \quad (7)$$

By stating that $s \leq_{\mathbf{PbU}(agt)} s'$, we assert that agt would make no observations were the world to move from situation s to s'. This means that the agent's view in both situations would be identical, so if it considers s possible then it must also consider s' possible. Following this intuition, we propose the following successor state axiom to capture the desired dynamics of the knowledge fluent:

$$K(agt, s'', do(c, s)) \equiv [\, Obs(agt, c, s) = \emptyset \rightarrow K(agt, s'', s)\,] \\ \wedge [\, Obs(agt, c, s) \neq \emptyset \rightarrow \exists c', s' :\ Obs(agt, c', s') = Obs(agt, c, s) \\ \wedge Poss(c', s') \wedge K(agt, s', s) \wedge do(c', s') \leq_{\mathbf{PbU}(agt)} s''\,] \quad (8)$$

This axiom considers two independent cases, depending on whether the actions c produced any observations. The cases are illustrated in Figure 1 and Figure 2 respectively.

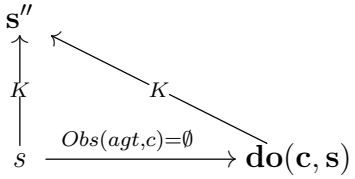


Figure 1: Illustration of the successor state axiom for knowledge fluent $\mathbf{K(agt, s'', do(c, s))}$ from Equation 8, for the case where $Obs(agt, c, s) = \emptyset$.

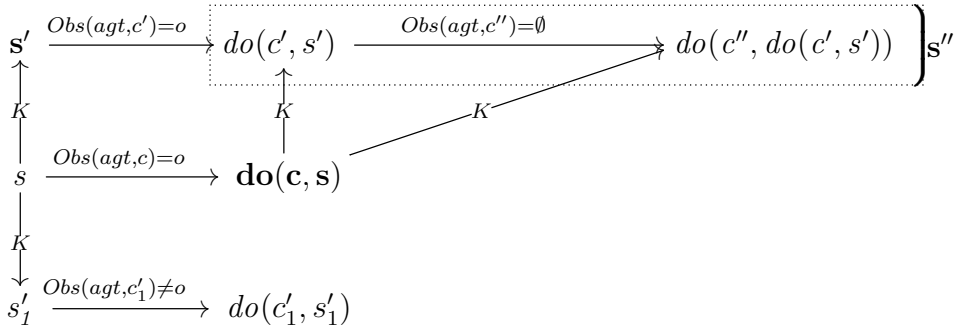


Figure 2: Illustration of the successor state axiom for knowledge fluent $\mathbf{K(agt, s'', do(c, s))}$ from Equation 8, for the case where $Obs(agt, c, s) \neq \emptyset$.

Figure 1 shows the case where $Obs(agt, c, s) = \emptyset$ and the actions c are hence totally unobservable. In this case the agent's state of knowledge does not change – the situations considered possible in $do(c, s)$ are precisely those considered possible in s.

Figure 2 shows the case where $Obs(agt, c, s) \neq \emptyset$ and hence the actions c produce some observations o. In this case the agent considers possible any legal successor to a possible alternate situation s' that can be brought about by actions c' yielding identical observations. Actions that would not produce matching observations do not produce K-related situations. Crucially, the agent also considers possible any future of $do(c', s')$ that could be reached through a sequence of unobservable actions.

To reiterate: unlike the standard successor state axiom from Equation (3), our approach requires agents to consider any possible future situation in which they would make no further observations. This requirement is fundamental to the correct specification of knowledge in asynchronous domains.

It remains to specify K in the initial situation. The relation K_0 defines

knowledge before *any* actions have occurred, but the agents must consider the possibility that some hidden actions have occurred. In other words, we must include situations where $root(s) \leq_{\mathbf{PbU}(agt)} s$ in the K-relation for initial situations. We therefore propose the following axiom:

$$Init(s) \rightarrow \left[K(agt, s'', s) \equiv \exists s' : K_0(agt, s', s) \wedge s' \leq_{\mathbf{PbU}(agt)} s''\right] \quad (9)$$

Definition 12. *We will denote by $\mathcal{D}_K^{obs}$ the axioms for our new observation-based semantics for knowledge, as detailed in Equations (8,9) above.*

These axioms suffice to ensure that knowledge behaves as we require: two situations will be related by $K(agt, s', s)$ if and only if they result in identical views for that agent, s' is legal, and their root situations were initially related:

Theorem 1. *For any agent agt and situations s and s'':*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models K(agt, s'', s) \equiv$$
$$Legal(s'') \wedge View(agt, s'') = View(agt, s) \wedge K_0(agt, root(s''), root(s))$$

Proof sketch. For the *if* direction we show how to construct an s' satisfying the $\exists s'$ parts of Equations (8,9). For the *only-if* direction we establish each of the three conjuncts individually. The *root* case is trivial since Equation (8) always expresses $K(s'', do(c, s))$ in terms of $K(s', s)$, while Equation (9) relates K for initial situations back to K_0. The *Legal* case relies on the fact that **PbU** implies $Poss$, while the $View$ case relies on the fact that $s \leq_{\mathbf{PbU}} s' \rightarrow View(s) = View(s')$. For the complete proof, see Appendix C. □

Using this new formulation, an agent's knowledge is completely decoupled from the global notion of actions, instead depending only on the local information that it has observed. Of course, this must be combined with a specific axiomatisation of how the *Obs* function behaves. We detail a variety of axiomatisations in Section 5, and our account of knowledge is directly applicable to all of them.

3.3. Properties of Knowledge

As a demonstration of the correctness of their axioms, Scherl and Levesque [45] prove five properties of their formalism: that knowledge-producing actions have only knowledge-producing effects; that unknown fluents remain

unknown by default; that knowledge incorporates the results of sensing actions; that known fluents remain known by default; and that agents have knowledge of the effects of their actions.

However, the intuition behind these properties depends heavily on the assumption of public actions and on the separation of actions into two classes: knowledge-producing actions that only return sensing information, and ordinary actions that only affect the state of the world. In asynchronous multi-agent domains, these restrictions cannot be meaningfully applied.

For example, it is entirely possible that a knowledge-producing action and an ordinary action are performed concurrently by two different agents, so the results of a sensing action might immediately be made invalid. Moreover, suppose that an agent performs an action to make a formula ϕ true, but there is a series of hidden actions that could subsequently make ϕ false. The agent cannot meaningfully claim to know ϕ, since it could become false without updating the local view of that agent.

The proofs used in [45] all hinge on showing that the situations K-related to $do(a, s)$ are precisely the "correct" ones, where correctness is formulated in terms of the preconditions and effects of a. We claim that in our formulation, the "correct" situations to be related to $do(c, s)$ are precisely those that are legal and have the same view, and the validity of Theorem 1 provides sufficient justification for the correctness of our knowledge axioms.

Indeed, it can be shown that our formulation is strictly equivalent to the standard account of Scherl and Levesque [45] when all actions are public. Let each agent observe all actions, as well as the sensing results of actions that they themselves perform:

$$\begin{aligned} a\#r \in Obs(agt, c, s) \;&\equiv\; a \in c \\ \wedge\, actor(a) = agt \;&\rightarrow\; r = SR(a, s) \\ \wedge\, actor(a) \neq agt \;&\rightarrow\; r = Nil \end{aligned} \tag{10}$$

Our new account of knowledge will then behave identically to the standard account:

Theorem 2. *Suppose $\mathcal{D}_{ad}$ contains Equation (10) as its definition of the Obs function, then for any legal situation terms σ and σ':*

$$\mathcal{D} \cup \mathcal{D}_K^{std} \models K(agt, \sigma', \sigma) \;\text{ iff }\; \mathcal{D} \cup \mathcal{D}_K^{obs} \models K(agt, \sigma', \sigma)$$

Proof. Equation (10) means that $Obs(agt, c, s)$ cannot be empty for $c \neq \emptyset$, so $s \leq_{\mathbf{PbU}(agt)} s'$ iff $s = s'$. Since we restrict our attention to legal situations,

we can substitute $\bot$ for $Obs(agt, c, s) = \emptyset$ and $\top$ for $Obs(agt, c, s) \neq \emptyset$ into Equations (8,9) to obtain the following:

$$\begin{aligned} K(agt, s'', do(c, s)) \equiv\ & [\bot \rightarrow K(agt, s'', s)\,] \\ & \wedge\, [\,\top \rightarrow \exists c', s' :\ Obs(agt, c', s') = Obs(agt, c, s) \\ & \qquad \wedge\, Poss(c', s') \wedge K(agt, s', s) \wedge do(c', s') = s''\,] \end{aligned}$$

$$Init(s) \rightarrow [K(agt, s'', s) \equiv \exists s' :\ K_0(agt, s', s) \wedge s' = s'']$$

Which further simplifies to:

$$\begin{aligned} K(agt, s'', do(c, s)) \equiv\ & \exists c', s' :\ Obs(agt, c', s') = Obs(agt, c, s) \\ & \wedge\ Poss(c', s') \wedge K(agt, s', s) \wedge do(c', s') = s'' \end{aligned} \quad (11)$$

$$Init(s) \rightarrow [K(agt, s'', s) \equiv\ K_0(agt, s'', s)] \quad (12)$$

Using Equation (10), it is straightforward to show that:

$$\begin{aligned} & Obs(agt, c', s') = Obs(agt, c, s) \equiv \\ & \qquad c = c' \wedge \forall a \in c :\ actor(a) = agt \rightarrow [SR(a, s) = SR(a, s')] \end{aligned}$$

Equations (11,12) are therefore equivalent to Equations (2,3) from $\mathcal{D}_K^{std}$, meaning that K behaves the same under both theories. □

Having established that our account subsumes the standard "public actions" account of knowledge, we can also show that it maintains many of its desirable properties in the general case. One of the fundamental results in [45] is that if the initial knowledge relation K_0 is reflexive, symmetric, transitive or Euclidean, then the K relation has these properties for any situation. In our formalism, such preservation of accessibility properties follows immediately from Theorem 1 and the reflexive, symmetric, transitive and Euclidean nature of the equality operator.

Theorem 3. *If the K_0 relation is restricted to be reflexive, transitive, symmetric or Euclidean, then the K relation defined by $\mathcal{D}_K^{obs}$ will satisfy the same restrictions at every legal situation.*

Proof. Each follows directly from Theorem 1 and the properties of equality. We will take the transitive case as an example; the rest are virtually identical.

Suppose that K_0 is transitive, and we have legal situations s_1, s_2, s_3 such that $K(agt, s_2, s_1)$ and $K(agt, s_3, s_2)$. Then by Theorem 1 we have the following:

$$\begin{aligned} K_0(agt, root(s_2), root(s_1)) \\ K_0(agt, root(s_3), root(s_2)) \\ View(agt, s_1) = View(agt, s_2) \\ View(agt, s_2) = View(agt, s_3) \end{aligned}$$

From the transitivity of K_0 we can conclude that $K_0(agt, root(s_3), root(s_1))$ and from the transitivity of equality we can conclude that $View(agt, s_1) = View(agt, s_3)$. Since s_3 is restricted to be legal, we have enough to satisfy the RHS of the equivalence in Theorem 1, so $K(agt, s_3, s_1)$ and K is therefore transitive. □

That these properties hold regardless of the axiomatisation of *Obs* is a compelling argument in favour of our approach. Certain kinds of sensing actions can easily invalidate these properties if not axiomatised carefully under the standard account, such as the *guarded sensing actions* of [32]. We will explore this case in detail in Section 5.

Our formalism is not only a proper generalisation of the standard account of knowledge in the situation calculus, it is an *elaboration tolerant* generalisation. It maintains important general properties of knowledge as more complex models of sensing and observability are introduced.

One problem remains: the successor state axiom for K in Equation 8 is incompatible with standard regression techniques. The right-hand side is not a regressable formula, as the $\leq_{\mathbf{PbU}(agt)}$ component universally quantifies over situation terms. A similar problem prevented the development of a regression rule in [21]. In the following section we will use the persistence condition operator to formulate a regression rule that can handle these arbitrarily-long sequences of hidden actions.

4. Reasoning about Knowledge

For our new account of knowledge to be useful in practice, we must extend the standard techniques for automated reasoning in the situation calculus to handle the modified formalism – that is to say, we must provide a regression rule for **Knows**.

4.1. Regressing Knowledge Queries

The appearance of $\leq_{\mathbf{PbU}(agt)}$ in Equation (8) means that our new successor state axiom universally quantifies over situations, so the regression technique developed in [45] cannot be used directly. We must appeal to the persistence condition meta-operator introduced in Section 2.4 to handle the inductive component of this reasoning, by transforming the quantification into a uniform formula so that standard regression techniques can be applied.

We propose the following as the regression rule for **Knows** under our formalism:

$$\begin{aligned}\mathcal{R}(\mathbf{Knows}(agt, \phi, do(c, s))) = \exists o :\ & Obs(agt, c, s) = o \\ & \wedge [o = \emptyset \rightarrow \mathbf{Knows}(agt, \phi, s)] \\ & \wedge [o \neq \emptyset \rightarrow \mathbf{Knows}(agt, \forall c' : Obs(agt, c') = o \\ & \quad \wedge Poss(c') \rightarrow \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt)), c'), s)]\end{aligned} \tag{13}$$

Note the similarity to the standard regression rule for knowledge from Equation (4). New in our version are: the replacement of the **res** macro with an explicit definition of what the agent has observed; explicit handling of the case when the agent makes no observations; and use of the persistence condition to account for arbitrarily-long sequences of hidden actions.

As required for a regression rule, Equation (13) reduces a knowledge query at $do(c, s)$ to a knowledge query at s. It is also intuitively appealing: to know that ϕ holds, the agent must know that in all situations that agree with its observations, ϕ cannot become false without it making some further observation. This is the meaning of $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ in the above, to state that "if ϕ were to become false, I would notice".

We must also specify a regression rule for **Knows** in the initial situation, as Equation (9) also uses the $\leq_{\mathbf{PbU}(agt)}$ ordering. This clause produces an expression in $\mathbf{Knows_0}$ at S_0, meaning that it can be handled by epistemic reasoning about the initial situation only:

$$\mathcal{R}(\mathbf{Knows}(agt, \phi, S_0)) \ = \ \mathbf{Knows_0}(agt, \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt))[S_0])^{-1}, S_0) \tag{14}$$

The use of $\mathcal{P}$ here is similar to its use in the previous regression rule. The use of $\mathcal{R}(\ldots[S_0])^{-1}$ ensures that the transform is applied recursively to any nested knowledge formulae – when the enclosed formula ϕ does not contain

nested knowledge macros, regressing it at S_0 and suppressing the situation term leaves it unchanged.

Assuming that $\mathcal{P}$ can be calculated as described in Section 2.4, these regression rules provide a sound and complete procedure for reducing queries about knowledge to queries about the initial knowledge of the agents:

Theorem 4. *Given a basic action theory $\mathcal{D}$ and a uniform formula ϕ for which $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ exists:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(agt, \phi, s) \equiv \mathcal{R}(\mathbf{Knows}(agt, \phi, s))$$

Proof sketch. By induction on situation terms. In the $do(c, s)$ case, we proceed by expanding the definition for **Knows** using our new successor state axiom for K, collecting sub-formulae that match the form of the **Knows** macro, and using regression and the persistence condition to render the resulting expressions uniform in s. In the base case, we apply the persistence condition to an expansion of **Knows** at S_0 to produce the desired result. For the complete proof, see Appendix C. □

These regression rules thus enable us to handle knowledge queries in our formalism using a standard regression-based approach, with the persistence condition operator allowing us to "factor out" the inductive component of the reasoning.

4.2. Regression over Views

While this regression rule is suitable for modelling and simulation purposes, it would be unreasonable for a situated agent to ask "do I know ϕ in the current situation?" using the situation calculus query $\mathcal{D} \models \mathbf{Knows}(agt, \phi, \sigma)$. As discussed in Section 3, an agent cannot be expected to have the full current situation σ. It will however have its current view v and can construct a query about its own knowledge as follows:

$$\mathcal{D} \models \forall s : \mathit{View}(agt, s) = v \wedge \mathit{root}(s) = S_0 \wedge \mathit{Legal}(s) \rightarrow \mathbf{Knows}(agt, \phi, s)$$

Such a query universally quantifies over situations and so cannot be handled using regression. It is also not in a form amenable to the persistence condition operator.

However, we should expect from Theorem 1 that the quantification over situations is unnecessary in this case – after all, any situation with the same

view for that agent should result in it having the same knowledge. Let us explicitly define knowledge with respect to a view as follows:

$$\mathbf{Knows}(agt, \phi, v) \stackrel{\text{def}}{=} \\ \forall s : \mathit{View}(agt, s) = v \wedge \mathit{root}(s) = S_0 \wedge \mathit{Legal}(s) \rightarrow \mathbf{Knows}(agt, \phi, s)$$

We can then modify the regression rules in Equations (13,14) to work directly on formulae of this form. The resulting rules are actually simpler than for regression over situations, as there are no empty observations in a view. The result is:

$$\mathcal{R}(\mathbf{Knows}(agt, \phi, o \cdot v)) = \mathbf{Knows}(agt, \forall c : \mathit{Obs}(agt, c) = o \\ \wedge \mathit{Poss}(c) \rightarrow \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt)), c), v) \quad (15)$$

$$\mathcal{R}(\mathbf{Knows}(agt, \phi, \epsilon)) = \mathbf{Knows_0}(agt, \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt))[S_0])^{-1}, S_0) \quad (16)$$

Using regression in this way, an agent can reduce the query $\mathbf{Knows}(agt, \phi, v)$ to an equivalent query about its knowledge in the initial situation.

Theorem 5. *Given a basic action theory $\mathcal{D}$ and a uniform formula ϕ for which $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ exists:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(agt, \phi, v) \equiv \mathcal{R}(\mathbf{Knows}(agt, \phi, v))$$

Proof sketch. The proof hinges on a simple corollary of Theorem 1: that situations with the same root and same view entail the same knowledge:

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \forall s, s', s'' : \mathit{root}(s) = \mathit{root}(s') \wedge \mathit{View}(s) = \mathit{View}(s') \\ \wedge K(agt, s'', s) \rightarrow K(agt, s'', s')$$

We can then proceed by induction over views. For the ϵ and $o \cdot v$ cases we split on whether there exists a situation having that view. If no such situation exists, we show that the regression rules (15, 16) generate a formula that is vacuously true, as an invalid view causes anything to be known. If such a situation does exist, we select an arbitrary witness and demonstrate that rules (15, 16) generate an equivalent formula to rules (13,14) using that witness. By the above corollary, this is enough to establish equivalence for any such situation. For the complete proof, see Appendix C. □

Our formalism thus allows agents to reason about their own knowledge using only their local information, even in asynchronous domains where they do not know how many actions have been performed.

4.3. Reasoning in Practice

It is worth re-iterating that our regression rules are no longer straightforward syntactic transformations – rather, they involve a fixpoint calculation to generate $\mathcal{P}(\phi, \mathbf{PbU}(agt))$. Our previous work on the persistence condition meta-operator [19] discussed the advantages of this approach in detail, but its primary advantage is to make automated reasoning approachable at all – the alternative is second-order theorem proving against the full theory $\mathcal{D}$.

Section 6 demonstrates our technique in action, answering multi-agent knowledge queries in a purely mechanical fashion. We have also completed a preliminary implementation of our technique in Prolog, using a modal variant of the LeanT^{A}P theorem prover [3, 11] as its core reasoning engine, and have verified its operation on several simple domains, including some of the Hunt the Wumpus examples presented in Section 6. More details of the domains and examples are available along with the implementation[5].

At a high level, we replace a single second-order theorem-proving task:

$$\mathcal{D} \models_{SOL} \mathbf{Knows}(agt, \phi, \sigma)$$

With iterated first-order reasoning to calculate the persistence condition:

$$\mathcal{D}_{bg} \models_{FOL} \mathcal{P}^n(\phi, \mathbf{PbU}(agt)) \rightarrow \mathcal{P}^{n+1}(\phi, \mathbf{PbU}(agt))$$

Followed by first-order reasoning about the initial situation only:

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models_{FOL} \mathcal{R}^*(\mathbf{Knows}(agt, \phi, \sigma))$$

We thus "factor out" the components of the reasoning process, allowing each to be treated in isolation. The inductive aspects are handed by property persistence, and the dynamic aspects are handled by regression, leaving just static epistemic reasoning in the target domain. Each may be more or less challenging depending on the target domain, and each may be studied and improved in isolation.

The problem of performing possible-worlds reasoning in the initial situation has received considerable attention in the literature [32], as has the general problem of full epistemic reasoning about unrestricted forms of nested

[5]To obtain the code please visit http://people.eng.unimelb.edu.au/adrianrp/ or http://agentlab.cis.unimelb.edu.au/

knowledge in K_n [14]. Since our approach has these challenges in common with the standard account from [45], we will not discuss them further here.

Our insistence on considering "all possible future actions" introduces an additional challenge not faced by the standard account – the calculation of $\mathcal{P}$ may be very complex, and is not guaranteed to terminate in the general case. Our previous work identified several classes of basic action theory in which the persistence condition can be computed [19], and we will review some specific examples below.

Since decidability results for the most general reasoning tasks in the situation calculus are known to be rather rare, recent approaches frequently seek to restrict action theories in an appropriate way to achieve termination, depending upon the problem domain. Importantly, our approach is compatible with a range of existing techniques, facilitated by our first-order representation used for answering property persistence queries.

We consider below three classes of action theory in which $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ is guaranteed to be calculable.

4.3.1. Synchronous action theories

An important sanity-check for our approach is that is does not introduce additional complexity for purely synchronous domains.

Definition 13 (Synchronous Action Theory)**.** *A basic action theory $\mathcal{D}$ is synchronous if every agent observes something whenever an action occurs:*

$$\mathcal{D} \models \forall agt, c, s : \mathit{Poss}(c, s) \rightarrow \mathit{Obs}(agt, c, s) \neq \emptyset$$

It is straightforward to show that $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ in synchronous domains is always equivalent to ϕ. The regression rules in Equations (13,14) then reduce to purely syntactic manipulations:

Theorem 6. *Let $\mathcal{D}_{sync}$ be a synchronous basic action theory, then:*

$$\mathcal{D}_{sync} \models \forall s, agt : \phi[s] \equiv \mathcal{P}(\phi, \mathbf{PbU}(agt))[s]$$

Proof. By definition, we have:

$$\mathcal{D}_{sync} \models \forall agt, c, s : \mathit{Poss}(c, s) \rightarrow \mathit{Obs}(agt, c, s) \neq \emptyset$$

Recall from Equation (7) that:

$$\mathbf{PbU}(agt, c, s) \equiv \mathit{Poss}(c, s) \wedge \mathit{Obs}(agt, c, s) = \emptyset$$

So clearly:

$$\mathcal{D}_{sync} \models \forall agt, c, s : \mathbf{PbU}(agt, c, s) \equiv \bot$$

The definition of $\mathcal{P}^1(\phi, \mathbf{PbU}(agt))$ will then produce:

$$\mathcal{P}^1(\phi, \mathbf{PbU}(agt)) \equiv \phi \wedge (\forall c : \bot \rightarrow \mathcal{R}(\phi, c)) \equiv \phi$$

The calculation of $\mathcal{P}$ thus terminates immediately at the first iteration, giving $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ equal to $\mathcal{P}^1(\phi, \mathbf{PbU}(agt))$, which is equivalent to ϕ as desired. □

We thus do not introduce unnecessary complications for domains in which effective reasoning procedures already exist, while extending the reach of our formalism into richer domains where some inductive reasoning is required.

4.3.2. Propositional action theories

For domains in which all non-situation sorts are finite, reasoning about uniform formulae can be reduced to propositional logic and is therefore strongly decidable. Calculating the persistence condition is also decidable in this case – since the domain has a finite state-space, the fixpoint calculation operates over a complete well-founded lattice and is therefore guaranteed to terminate [19].

The Scherl and Levesque [45] account of knowledge has been shown to admit to an optimal decision procedure and regression algorithm by integrating observation actions and modal knowledge operators into a dynamic epistemic logic encoding [57].

There are advantages to maintaining the expressive power of first-order logic, even when the domain can be propositionalized, as the use of quantifiers and free variables can produce exponentially-shorter formulae and hence lead to shorter proofs.

4.3.3. Context-free action theories

Suppose that the theory of action is *context free* [27]. In such theories successor state axioms have the following form:

$$F(\bar{x}, do(a, s)) \equiv \Phi_F^+(\bar{x}, a) \vee \left(F(\bar{x}, s) \wedge \neg\Phi_F^-(\bar{x}, a)\right)$$

The effects of an action are thus independent of the situation in which it is performed. Quantification is still permitted, provided the quantifiers only range over the situation-independent formulae.

Lin and Levesque [25, lemma 6.2] show that context-free theories with a finite number of parameterless actions have a finite state space. This is therefore sufficient to ensure a terminating computation of persistence condition, $\mathcal{P}$, in the same manner as propositional domains [19].

Context free theories capture the expressivity of the usual formation of classical planning problems encoded in STRIPS, with add and delete lists, in which every action is considered to be context-free, as identified in [25].

4.4. Approximate Reasoning

Rather than restricting the domain to guarantee terminating calculation of $\mathcal{P}(\phi, \mathbf{PbU}(agt))$, it is also possible to restrict the form of queries so that it can be pre-calculated offline. This kind of *approximate* reasoning about knowledge is key to the formalism of Demolombe and Pozos Parra [9], in which knowledge is limited to be about fluent literals only.

Their basic idea is to introduce, for each fluent F in the domain, two additional fluents $K^+_{agt}F$ and $K^-_{agt}F$ to explicitly represent "agt knows F" and "agt knows $\neg F$". By formulating ordinary successor state axioms for these fluents, literal-level knowledge can be reasoned about using standard regression and does not require an explicit possible-worlds K-relation. The trade-off is that this approach cannot represent indeterminate disjunctive knowledge such as "agt knows F or G".

The Demolombe and Pozos Parra approach has been formally related to the standard Scherl and Levesque approach by Petrick and Levesque [31]. They show there is an equivalence between the two approaches when an agent's knowledge is restricted to be *disjunctive*, so that the following holds:

$$\mathbf{Knows}(agt, \phi_1 \vee \phi_2, s) \rightarrow \mathbf{Knows}(agt, \phi_1, s) \vee \mathbf{Knows}(agt, \phi_{2,}s)$$

In [32] this equivalence is extended to cover existential quantification by restricting knowledge to also satisfy the following:

$$\mathbf{Knows}(agt, \exists x : \phi(x), s) \rightarrow \exists x : \mathbf{Knows}(agt, \phi(x), s)$$

These disjunctive properties of knowledge are *not* entailed by a general possible-worlds style theory in the tradition of [45], although there are restrictions that can be placed on the theory in order to enforce them [33, 32].

Even when these disjunctive knowledge properties are not enforced by the domain, they permit a sound *approximation* of knowledge that can be

reasoned about more tractably than the standard possible-worlds account. Following the style of [31] we can provide the following definitions:

$$\mathbf{Knows}_A(agt, \phi_1 \wedge \phi_2, s) = \mathbf{Knows}_A(agt, \phi_1, s) \wedge \mathbf{Knows}_A(agt, \phi_2, s)$$
$$\mathbf{Knows}_A(agt, \neg(\phi_1 \wedge \phi_2), s) = \mathbf{Knows}_A(agt, \neg\phi_1, s) \vee \mathbf{Knows}_A(agt, \neg\phi_2, s)$$
$$\mathbf{Knows}_A(agt, \forall x : \phi(x), s) = \forall x : \mathbf{Knows}_A(agt, \phi(x), s)$$
$$\mathbf{Knows}_A(agt, \neg\forall x : \phi(x), s) = \exists x : \mathbf{Knows}_A(agt, \neg\phi(x), s)$$
$$\mathbf{Knows}_A(agt, \neg\neg\phi, s) = \mathbf{Knows}_A(agt, \phi, s)$$
$$\mathbf{Knows}_A(agt, F, s) = \mathbf{Knows}(agt, F, s)$$
$$\mathbf{Knows}_A(agt, \neg F, s) = \mathbf{Knows}(agt, \neg F, s)$$

A knowledge query is split across the logical operators until we are left with only knowledge of fluent literals, which is then handled using our formalism. If we assume a finite number of fluents, then we can use our regression rule for knowledge to *pre-calculate* an explicit successor state axiom for $\mathbf{Knows}_A(agt, F, s)$ and $\mathbf{Knows}_A(agt, \neg F, s)$, allowing them to be treated as primitive fluents and reasoned about at run-time using purely syntactic transformations.

Unlike the approach of Demolombe and Pozos Parra [9] in which knowledge fluents must be axiomatised separately from the concrete fluents they describe, the approach suggested here would allow a successor state axiom for literal-level knowledge to be derived from the dynamics of the domain. The calculation of $\mathcal{P}(F, \mathbf{PbU}(agt))$ for each fluent could be performed once, offline, and then used directly for approximate reasoning about knowledge.

5. Axiomatising Observations

We now show how observations and views can be used to model a variety of common domain dynamics from the situation calculus literature. We argue that these axiomatisations intuitively capture the "correct" information in each case, and show how our explicit representation of the agent's local perspective can overcome difficulties encountered in previous formulations.

5.1. Public Actions

In Section 3 we showed how public actions can be easily captured using the following axiom for *Obs* (reproduced from Equation 10):

$$\begin{aligned} a\#r \in Obs(agt, c, s) \equiv\ & a \in c \\ \wedge\ actor(a) = agt \rightarrow\ & r = SR(a, s) \\ \wedge\ actor(a) \neq agt \rightarrow\ & r = Nil \end{aligned}$$

It should be clear that these definitions capture the intuition behind the most common model of action observability, and indeed Section 3 formally proved an equivalence to the standard knowledge axioms of [45].

This approach clearly leads to synchronous domains, since an agent's set of observations can only be empty if the set of actions is also empty, and the empty action set is never legal to perform.

5.2. Private Actions

Private actions can be easily modelled by the following definition of *Obs*:

$$a\#r \in Obs(agt, c, s) \equiv a \in c \wedge actor(a) = agt \wedge SR(a, s) = r$$

As noted by Lespérance et al. [21], private actions mean that agents need to consider arbitrarily-long sequences of hidden actions which may or may not have occurred, and standard regression techniques are insufficient to handle this case. By explicitly formalising this situation, we are able to provide an approach to effective automated reasoning in such asynchronous domains.

Private actions also serve to demonstrate the elaboration tolerance of our approach. Consider the successor state axiom for knowledge with hidden actions that is formulated in [21]:

$$\begin{aligned} K(agt, s'', do(a, s)) \equiv\ & \exists s' :\ K(agt, s', s) \\ & \wedge\ (actor(a) \neq agt\ \rightarrow\ s' \leq_{actor(a) \neq agt}\ s'')) \\ & \wedge\ (actor(a) = agt\ \rightarrow\ \exists s^* :\ \big[s' \leq_{actor(a) \neq agt}\ s^* \wedge \\ & \quad s'' = do(a, s^*) \wedge Poss(a, s^*) \wedge SR(a, s) = SR(a, s^*))\big] \end{aligned}$$

While the axiom seems intuitively plausible, it has a subtle problem: an agent's knowledge can change in response to actions performed by others. Suppose that *agt* has just performed action a_1, so the world is in situation $do(a_1, s)$. Another agent then performs the action a_2, leaving the world in

situation $do(a_2, do(a_1, s))$. Since it is not aware of the occurrence of a_2, the knowledge of agt should be unchanged between these two situations. This is not the case under the formulation of [21], which introduces arbitrarily-long sequences of hidden actions into the *past* of the possible situation s''. Based on our explicit formalisation of the agent's local view, our axiom includes hidden actions in the *future* of s'' and avoids this unintuitive behaviour.

By explicitly formalising the notion of a view, Theorem 1 guarantees that our formulation will avoid subtle problems such as this.

5.3. Guarded Sensing Actions

Petrick [32] extends the approach of [45] to include *guarded* sensing actions. These actions cause the agent to learn that some formula ϕ holds, but only if an additional guard formula ψ also holds in the world. They are introduced by modifying the successor state axiom for K as follows:

$$K(agt, s'', do(a, s)) \equiv \exists s' : s'' = do(a, s) \wedge K(agt, s', s) \wedge \ldots$$
$$a = sense_{\phi,\psi} \rightarrow [\psi(s) \rightarrow \phi(s) \equiv \phi(s')]$$

Petrick [32] demonstrates that this modified successor state axiom is no longer guaranteed to preserve the symmetry of the K fluent, invalidating one of the fundamental properties of knowledge. The problem is that although the agent will learn ϕ if the guard ψ is true, it cannot conclude that the guard was false by virtue of *not* learning ϕ. Since the agent's local perspective is only modelled implicitly, it has no way of detecting that the action failed to produce its sensing result.

To ensure that symmetry is preserved through action, it is necessary to axiomatise these sensing actions in such a way that the status of the guard formula itself also becomes known. This is achieved in [32] through special syntactic restrictions, but our approach of explicitly representing the observations made by each agent avoids this problem automatically.

To model guarded sensing actions in our framework, one would leave K unchanged and instead modify the definition of Obs to add guard conditions Ψ as follows:

$$a\#r \in Obs(agt, c, s) \equiv a \in c \wedge actor(a) = agt \wedge SR(a, s) = r \wedge \Psi(a, s)$$

For example, an action $sense_{\phi,\psi}$ that senses the truth of some formula ϕ when the guard condition ψ is true would require something like the following

to be entailed by the definition of Obs:

$$\begin{aligned}
sense_{\phi,\psi}\#T \in Obs(agt, c, s) \equiv\ & sense_{\phi,\psi} \in c \\
& \wedge\ actor(sense_\phi) = agt \wedge \psi(s) \wedge \phi(s) \\
sense_{\phi,\psi}\#F \in Obs(agt, c, s) \equiv\ & sense_{\phi,\psi} \in c \\
& \wedge\ actor(sense_\phi) = agt \wedge \psi(s) \wedge \neg\phi(s) \\
sense_{\phi,\psi}\#Nil \in Obs(agt, c, s) \equiv\ & sense_{\phi,\psi} \in c \\
& \wedge\ actor(sense_\phi) = agt \wedge \neg\psi(s)
\end{aligned}$$

An agent using our formalism can therefore explicitly conclude, by virtue of not receiving a sensing result from $sense_{\phi,\psi}$, that the guard condition must not hold. This is sufficient to maintain symmetry of the knowledge accessibility relation as guaranteed by Theorem 3.

5.4. Speech Acts

Communication in the situation calculus is traditionally modelled using explicit communicative actions or "speech acts" [50, 49]. These actions are axiomatised as per standard actions, but special-case handling is introduced in the axioms for knowledge in order to model their communicative effects.

Instantaneous communication is modelled using actions such as $inform$, where $inform(agt_s, agt_r, \phi)$ means the sender agt_s informs the receiver agt_r of the truth of some formula ϕ. If we assert that only truthful speech acts are allowed, and all actions are publicly observable, then this requires no further axiomatisation:

$$Poss(inform(agt_s, agt_r, \phi), s) \equiv \mathbf{Knows}(agt_s, \phi[s], s)$$

The *inform* action will be included in each agent's observations whenever it occurs, from which the agent can conclude that it was possible and thus that the contained formula holds in the world.

However, this simple approach can lead to third-party agents being aware of what was communicated, which is often undesirable. In [50] encrypted speech acts are introduced to overcome this limitation, ensuring that only the intended recipient of a message is able to access its contents by performing a special *decrypt* action. While it would be straightforward to copy this approach in our formalism, the problem it was introduced to solve no longer exists – we can directly limit the accessibility of the message contents adding

another type of observation, $inform(agt_s, agt_r)$, that indicates the occurrence of the action but not its contents:

$$inform(agt_s, agt_r) \in Obs(agt, c, s) \equiv \exists m : inform(agt_s, agt_r, m) \in c$$

$$inform(agt_s, agt_r, m) \in Obs(agt, c, s) \equiv$$
$$inform(agt_s, agt_r, m) \in c \wedge (agt = r \vee agt = s)$$

Here all agents will observe that the communication occurred, but only the sender and recipient can observe the contents of the message.

Non-instantaneous communication can be modelled using a message queue for each agent with separate *send* and *check* actions [21]. The *send* action adds a message to the queue, while the *check* action returns the details of pending messages as its sensing result. Since this approach uses the standard sensing-result machinery, it requires no special axiomatisation here.

5.5. Explicit Observability Axioms

Our approach offers a straightforward way to explore the middle ground between the two extremes of "public actions" and "private actions" discussed above. To axiomatise general *partial observability* of actions, we introduce a new action description predicate $CanObs(agt, a, s)$ that defines the conditions under which agent agt would observe action a being performed in situation s. If $CanObs(agt, a, s)$ is false, then that action will be hidden. We can then define Obs as follows:

$$a \in Obs(agt, c, s) \equiv a \in c \wedge CanObs(agt, a, s)$$

This permits a great deal of flexibility in the axiomatisation. In our Hunt the Wumpus domain, agents are aware of all the actions performed in the same room as themselves:

$$CanObs(agt, a, s) \equiv \exists r : \wedge In(agt, r) \wedge In(actor(a), r)$$

It is also possible to allow partial observability of sensing results using an analogous predicate $CanSense(agt, a, s)$ and the following definition of Obs:

$$a \in Obs(agt, c, s) \equiv a \in c \wedge CanObs(agt, a, s) \wedge \neg CanSense(agt, a, s)$$
$$a\#r \in Obs(agt, c, s) \equiv a \in c \wedge SR(a, s) = r$$
$$\wedge CanObs(agt, a, s) \wedge CanSense(agt, a, s)$$

For example, consider an agent waiting for a train who activates a speaker to determine when it will arrive. The results of this sensing action would provide information to any other agent within earshot:

$$CanSense(agt, activateSpeaker(agt_2), s) \equiv CloseToSpeaker(agt)$$

We feel that this formulation provides a good balance between simplicity and expressiveness; it allows the observability of actions to vary according to the state of the world, but provides agents with a complete description of each action that they are capable of observing.

5.6. Observability Interaction

Reasoning about observability of concurrent actions raises the potential for *observability interaction*, in which some actions produce different observations when they are performed concurrently with another action. Like the precondition interaction problem for $Poss$, which has undergone extensive research and is discussed in [36, 34, 35], we assume that the axiom defining Obs contains the appropriate logic to handle such interaction. A simple axiomatisation might have actions being "masked" by the co-occurrence of another action, and would appear like so:

$$a \in Obs(agt, c, s) \equiv a \in c \wedge CanObs(agt, a, s) \wedge \neg\exists a' \in c : Masks(a', a, s)$$

The important point is that, given an explicit account of the local perspective of each agent, such interaction can be axiomatised independently of the rest of the action theory.

5.7. Observing the Effects of Actions

In many domains it would be infeasible for an agent to observe all of the details of a particular action when it occurs, but it may observe some of the effects of that action. For example, suppose that an agent monitors the state of a light in its environment, such that it notices it changing from dark to light. While it knows that *some* action must have occurred to produce that effect, it may not be sure precisely what action took place (e.g. precisely *who* turned on the light). This can be modelled by further extending the OBSERVATION sort to contain a special "effect observation" term $lightCameOn$, and axiomatising like so:

$$lightCameOn \in Obs(agt,c, s) \equiv \neg lightIsOn(s) \wedge \exists agt' : turnLightOn(agt') \in c$$

When the light is switched on, each agent's observation set will contain the term $lightCameOn$, and they will be able to deduce that this change has occurred without necessarily knowing the specific action responsible for the change. This is similar to the "fluent change" actions proposed by De Giacomo et al. [6], but embedded in the theory itself rather than generated by the agent when it discovers that it must update its beliefs.

5.8. Delayed Communication

Delayed communication can be modelled using separate *send* and *recv* actions. However, unlike the use of explicit communication channels discussed previously, we do not want the receiving agent to have to poll the message queue. Rather, the *recv* action should occur automatically some time after the *send* action.

This is easily modelled using *natural actions* as defined by Reiter [40], actions that occur automatically at a particular time. By making *recv* a natural action, *send*/*recv* pair can be axiomatised mirroring the standard account of long-running tasks in the situation calculus. A fluent $PendMsg(s, r, m, t)$. indicates that some message is pending and will be delivered at time t. We have:

$$natural(recv(agt_s, agt_r, m, t))$$

$$send(agt_s, agt_r, m, t) \in Obs(agt, c, s) \equiv send(agt_s, agt_r, m, t) \in c \wedge agt = agt_s$$

$$recv(agt_s, agt_r, m, t) \in Obs(agt, c, s) \equiv recv(agt_s, agt_r, m, t) \in c \wedge agt = agt_r$$

$$Poss(recv(agt_s, agt_r, m, t), s) \equiv PendMsg(agt_s, agt_r, m, t, s)$$

$$\begin{aligned} PendMsg(s, r, m, t_m, do(c, s)) \equiv \\ send(s, r, m, t) \in c \wedge t_m = t + delay(s, r, m, s) \\ \vee\ PendMsg(s, s, m, t_m, s) \wedge recv(s, r, m, t_m) \notin c \end{aligned}$$

A *send* action thus causes the message to become pending, with its delivery time determined by the functional fluent *delay*. Once the delay time has elapsed, the natural action *recv* will be triggered and the message delivered. The *send* and *recv* actions are observed only by the sender and receiver respectively.

If the agents have incomplete information about the *delay* function, this could easily model domains in which the message delay is unpredictable or even unbounded, giving asynchronous communication in the style of [15].

6. An Illustrative Example

We now give a brief demonstration of our formalism in action, using it to model the "Hunt The Wumpus" domain outlined in Section 1. Our variant is complicated by the presence of multiple hunters and partial observability, but we also simplify it by omitting common features such as pits and breezes. The full axiomatisation is available in Appendix A.

We take as a static background fact the predicate $Adjacent(r, r')$ defining the layout of the dungeon, as shown in Figure 3. Since it is part of $\mathcal{D}_{bg}$, the layout is known to all agents.

The actions of interest in the domain are $move(agt, r)$ by which agents may move between adjacent rooms, $shoot(agt, r)$ by which an agent may shoot into an adjacent room, and $alert(agt)$ by which agents may alert each other to the presence of a stench in their current room.

The fluents of interest are the function $r = Loc(agt, s)$ giving the current room each agent is in, $r = Wumpus(s)$ giving the room containing the Wumpus, $Stench(r, s)$ indicating whether a room contains a stench, and $Killed(s)$ indicating whether the Wumpus has been killed.

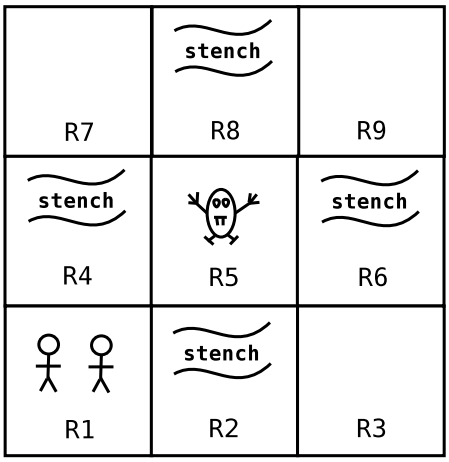


Figure 3: Hunt-the-Wumpus Dungeon Layout.

We have the following precondition axioms:

$$\begin{aligned} Poss(move(agt, r), s) &\equiv Adjacent(r, Loc(agt, s)) \\ Poss(shoot(agt, r), s) &\equiv Adjacent(r, Loc(agt, s)) \\ Poss(alert(agt), s) &\equiv Stench(Loc(agt, s), s) \end{aligned}$$

Along with the following successor state axioms:

$$\begin{aligned} r = Loc(agt, do(c, s)) &\equiv move(agt, r) \in c \\ &\quad \vee (r = Loc(agt, s) \wedge \neg\exists r' : move(agt, r') \in c) \\ Killed(do(c, s)) &\equiv Killed(s) \\ &\quad \vee \exists agt, r : shoot(agt, r) \in c \wedge r = Wumpus(s) \\ r = Wumpus(do(c, s)) &\equiv r = Wumpus(s) \\ Stench(r, do(c, s)) &\equiv Stench(r, s) \end{aligned}$$

Note that *Wumpus* and *Stench* do not vary, but they must be fluents to allow for agents having partial knowledge of these facts. They are related by the following background theory axiom, stating that only rooms adjacent to the Wumpus have a stench:

$$Stench(r, s) \equiv Adjacent(r, Wumpus(s))$$

This is a simple state invariant that, if true initially, is implied for all situations by the successor state axioms above. Typical approaches would discard such invariants after encoding their effects into the successor state axioms, as in [26]. We keep it as part of the background theory since it is useful when calculating the persistence condition; see Appendix B for a detailed treatment.

For OBSERVATION terms we include all actions along with the following: *footsteps* indicating movement of an agent in an adjacent room, *alert* indicating an announcement by an agent in an adjacent room, *stench* as a sensing result when moving into a stench-filled room, and *scream* to indicate

the death of the Wumpus. They are axiomatized as follows:

$$
\begin{aligned}
move(agt1, r1) \in Obs(agt, c, s) &\equiv move(agt1, r1) \in c \\
&\quad \wedge [Loc(agt, s) = Loc(agt1, s) \vee Loc(agt, s) = r1] \\
shoot(agt1, r1) \in Obs(agt, c, s) &\equiv shoot(agt1, r1) \in c \\
&\quad \wedge Loc(agt, s) = Loc(agt1, s) \\
alert(agt1) \in Obs(agt, c, s) &\equiv alert(agt1) \in c \\
&\quad \wedge Loc(agt, s) = Loc(agt1, s) \\
footsteps \in Obs(agt, c, s) &\equiv \exists agt1, r1 : move(agt1, r1) \in c \\
&\quad \wedge [Adjacent(Loc(agt, s), r1) \vee \\
&\qquad Adjacent(Loc(agt, s), Loc(agt1, s))] \\
alert \in Obs(agt, c, s) &\equiv \exists agt1 : alert(agt1) \in c \\
&\quad \wedge Adjacent(Loc(agt, s), Loc(agt1, s)) \\
stench \in Obs(agt, c, s) &\equiv \exists r1 : move(agt, r1) \in c \\
&\quad \wedge Stench(r1, s) \\
scream \in Obs(agt, c, s) &\equiv \exists agt1, r1 : shoot(agt1, r1) \in c \\
&\quad \wedge r1 = Wumpus(s) \wedge \neg Killed(s)
\end{aligned}
$$

The hunters know their initial state, being located in room $R1$ where there is no Wumpus and no stench:

$$
\begin{aligned}
Init(s) &\rightarrow \forall agt : Loc(agt, s) = R1 \\
Init(s) &\rightarrow Wumpus(s) \neq R1 \wedge \neg Stench(R1, s)
\end{aligned}
$$

The Wumpus is in room $R5$, but this is not known to the agents. It is a fact about the *actual* initial situation rather than all possible initial situations:

$$Wumpus(S_0) = R5$$

The agents have no additional knowledge about the state of the dungeon.

The following are examples of knowledge queries that can be posed in our formalism, a brief explanation of their outcome, and a demonstration of how they can be answered using our new regression rules.

Example 1. *Initially, Ann does not know where the wumpus is:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \neg\exists r : \mathbf{Knows}(A, \mathit{Wumpus}() = r, S_0)$$

Intuitively: it is given that $\neg\exists r : \mathbf{Knows_0}(A, \mathit{Wumpus}() = r, S_0)$, and there is no sequence of hidden actions that could result in her learning the location of the wumpus.

Formally: we answer the query by regressing it to produce a query about her initial knowledge $\mathbf{Knows_0}$:

$$\begin{aligned}&\mathcal{R}(\neg\exists r : \mathbf{Knows}(A, \mathit{Wumpus}() = r, S_0)) \Rightarrow \\ &\neg\exists r : \mathbf{Knows_0}(A, \mathcal{R}((\mathcal{P}(\mathit{Wumpus}() = r, \mathbf{PbU}(A))[S_0])^{-1}, S_0)\end{aligned}$$

The calculation of this persistence condition is trivial since the location of the wumpus cannot change, so the fixpoint search terminates after a single iteration:

$$\mathcal{P}(\mathit{Wumpus}() = r, \mathbf{PbU}(A)) \;\Rightarrow\; \mathit{Wumpus}() = r$$

Which gives:

$$\begin{aligned}&\mathcal{R}(\neg\exists r : \mathbf{Knows}(A, \mathit{Wumpus}() = r, S_0)) \Rightarrow \\ &\quad \neg\exists r : \mathbf{Knows_0}(A, \mathcal{R}((\mathit{Wumpus}() = r)[S_0])^{-1}, S_0)\end{aligned}$$

The nested regression of $\mathit{Wumpus}(r)$ at S_0 leaves the formula unchanged. The query thus regresses to the following, which is entailed by the domain:

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models \neg\exists r : \mathbf{Knows_0}(A, \mathit{Wumpus}() = r, S_0)$$

Example 2. *Bob knows that the wumpus is not in an adjacent room, since he knows there is no stench in room* $R1$.

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(B, \mathit{Wumpus}() \neq R2 \wedge \mathit{Wumpus}() \neq R4, S_0)$$

It is therefore safe for him to move to an adjacent room.

Regressing as in the previous example gives the equivalent query:

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models \mathbf{Knows_0}(B, \mathit{Wumpus}() \neq R2 \wedge \mathit{Wumpus}() \neq R4, S_0)$$

All initial situations have $\neg\mathit{Stench}(R1)$ and the background theory has $\mathit{Adjacent}(R2, R1)$, $\mathit{Adjacent}(R4, R1)$, and the axiom relating a stench to an adjacent wumpus. All initial situations thus have no Wumpus in rooms adjacent to $R1$, so the regressed query is entailed by the domain.

Example 3. *After Bob moves into room R2, he knows that it has a stench.*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(B, Stench(R2), do(\{move(B, R2)\}, S_0))$$

The *move* action causes Bob to receive a *stench* observation, and since a room's stenchiness cannot change, he can be sure the stench persists after any hidden actions.

Formally, we regress over the *move* action like so:

$$\begin{gathered}\mathcal{R}(\mathbf{Knows}(B, Stench(R2), do(\{move(B, R2)\}, S_0)) \Rightarrow \\ \exists o: \ Obs(B, \{move(B, R2)\}, S_0) = o \\ \wedge [o = \emptyset \rightarrow \mathbf{Knows}(B, Stench(R2), s)] \\ \wedge [o \neq \emptyset \rightarrow \mathbf{Knows}(B, \forall c' : Obs(B, c') = o \\ \wedge Poss(c') \rightarrow \mathcal{R}(\mathcal{P}(Stench(R2), \mathbf{PbU}(B)), c'), s)]\end{gathered}$$

Since this domain has a finite number of possible observations, we can expand the "$\exists o$" clause into a finite disjunction. Indeed there are only two possible values for $Obs(B, \{move(B, R2)\}, s)$, corresponding to the room having or not having a stench:

- $Stench(R2, s) \equiv Obs(B, \{move(B, R2)\}, s) = \{move(B, R2), stench\}$
- $\neg Stench(R2, s) \equiv Obs(B, \{move(B, R2)\}, s) = \{move(B, R2)\}$

Expanding and replacing each observation with its preconditions yields:

$$\begin{gathered}\mathcal{R}(\mathbf{Knows}(B, Stench(R2), do(\{move(B, R2)\}, S_0)) \Rightarrow \\ (Stench(R2, S_0) \wedge [\mathbf{Knows}(B, \dots)]) \\ \vee (\neg Stench(R2, S_0) \wedge [\mathbf{Knows}(B, \dots)])\end{gathered}$$

The domain entails $Stench(R2, S_0)$ so we can simplify the other option away, leaving:

$$\begin{gathered}\mathcal{R}(\mathbf{Knows}(B, Stench(R2), do(\{move(B, R2)\}, S_0))) \ \Rightarrow \\ \mathbf{Knows}(B, \forall c' : Poss(c') \wedge Obs(B, c') = \{move(B, R2), stench\} \rightarrow \\ \mathcal{R}(\mathcal{P}(Stench(R2), \mathbf{PbU}(B)), c'), S_0)\end{gathered}$$

The persistence condition calculation again terminates after one iteration:

$$\begin{gathered}\mathcal{P}(Stench(R2), \mathbf{PbU}(B)) \Rightarrow Stench(R2) \\ \mathcal{R}(\mathcal{P}(Stench(R2), \mathbf{PbU}(B)), c') \Rightarrow Stench(R2)\end{gathered}$$

So the query further simplifies to:

$$\mathbf{Knows}(B, \forall c' : Poss(c') \wedge Obs(B, c') = \{move(B, R2), stench\} \rightarrow Stench(R2), S_0)$$

Since this domain has a finite number of possible actions, we can expand the "$\forall c'$" clause into a finite conjunction – indeed, the only value of c' that can produce those observations is $\{move(B, R2)\}$. Substituting it and its action description predicates $Poss$ and Obs gives:

$$\mathbf{Knows}(B, Adjacent(R2, Loc(B)) \wedge Stench(R2) \rightarrow Stench(R2), S_0)$$

This tautology is clearly entailed by the domain.

Example 4. *Ann learns that Bob is in room R2 by observing Bob's move.*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, Loc(B) = R2, do(\{move(B, R2)\}, S_0))$$

Ann observes $move(B, R2)$ and learns that Bob is in room $R2$. If Bob leaves room $R2$, Ann will observe *footsteps* and know that he has left, provided she remains in room $R1$.

Regressing and enumerating the possible observations as before, we know that Ann's observations in this case will be $\{move(B, R2)\}$ and we have:

$$\mathcal{R}(\mathbf{Knows}(A, Loc(B) = R2, do(\{move(B, R2)\}, S_0))) \Rightarrow$$
$$\mathbf{Knows}(A, \forall c' : Poss(c') \wedge Obs(A, c') = \{move(B, R2)\} \rightarrow$$
$$\mathcal{R}(\mathcal{P}(Loc(B) = R2, \mathbf{PbU}(A)), c'), S_0)$$

As before, there is only a single possible value of c' that would match these observations. Setting $c' = \{move(B, R2\}$ and its definitions for $Poss$ and Obs gives:

$$\mathcal{R}(\mathbf{Knows}(A, Loc(B) = R2, do(\{move(B, R2)\}, S_0))) \Rightarrow$$
$$\mathbf{Knows}(A, Adjacent(Loc(B), R2) \wedge (Loc(A) = Loc(B) \vee Loc(A) = R2) \rightarrow$$
$$\mathcal{R}(\mathcal{P}(Loc(B) = R2, \mathbf{PbU}(A)), \{move(B, R2)\}), S_0)$$

Unlike previous examples, it *is* possible for $Loc(B) = R2$ to become false by some sequence of hidden actions. Calculation of $\mathcal{P}$ terminates after two iterations, yielding:

$$\mathcal{P}(Loc(B) = R2, \mathbf{PbU}(A)) \Rightarrow$$
$$Loc(B) = R2 \wedge (Loc(A) = R2 \vee Adjacent(R2, Loc(A)))$$

Ann will observe any actions that falsify $Loc(B) = R2$, as long as she is either in the same room (so she will see Bob move) or in an Adjacent room (so she will hear Bob's footsteps).

Since $move(B, R2)$ makes $Loc(B) = R2$ true, regressing over it gives the following:

$$\mathcal{R}(\mathcal{P}(Loc(B) = R2, \mathbf{PbU}(A)), \{move(B, R2)\}) \Rightarrow \\ Loc(A) = R2 \vee Adjacent(R2, Loc(A))$$

Substituting into the overall expression results in:

$$\mathbf{Knows}(A, Adjacent(Loc(B), R2) \wedge (Loc(A) = Loc(B) \vee Loc(A) = R2) \rightarrow \\ Loc(A) = R2 \vee Adjacent(R2, Loc(A)), S_0)$$

At this point we must apply $\mathcal{R}$ again to convert it into a formula about $\mathbf{Knows_0}$, the details of which proceed identically to the previous examples. It suffices to note that $\mathcal{D} \models \mathbf{Knows}(A, Adjacent(R2, Loc(A)), S_0)$ which is enough to satisfy the implication in the above formula.

Example 5. *After Bob alerts that there is a stench, Ann knows there is a stench in room R2 , since Ann knows that Bob is in room R2.*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, Stench(R2), do([\{move(B, R2)\}, \{alert(B)\}], S_0))$$

Ann observes *alert* which informs her that there is a stench in an adjacent room, and since she knows that Bob is in room $R2$, she infers $Stench(R2)$.

We begin by regressing over the most recent action $\{alert(B)\}$ to get a formula uniform in $\sigma \stackrel{\text{def}}{=} do(\{move(B, R2)\}, S_0)$:

$$\mathcal{R}(\mathbf{Knows}(A, Stench(R2), do(\{alert(B)\}, \sigma)) \Rightarrow \\ \exists o : o = Obs(A, \{alert(B)\}, \sigma) \\ \wedge \mathbf{Knows}(A, \forall c' : Poss(c') \wedge Obs(A, c') = o \rightarrow \\ \mathcal{R}(\mathcal{P}(Stench(R2), \mathbf{PbU}(A)), c'), \sigma)$$

The only possible case for the "$\exists o$" clause is $o = \{alert\}$, and the only possible case for the "$\forall c'$" clause is $c' = \{alert(B)\}$. From a previous example we have $\mathcal{R}(\mathcal{P}(Stench(R2), \mathbf{PbU}(B)), c') \Rightarrow Stench(R2)$. We can thus simplify this to:

$$\mathbf{Knows}(A, Poss(\{alert(B)\}) \wedge Obs(A, \{alert(B)\}) = \{alert\} \rightarrow \\ Stench(R2), do(\{move(B, R2)\}, S_0))$$

Inserting definitions of *Poss* and *Obs* gives:

$$\mathbf{Knows}(A, Stench(Loc(B)) \wedge Adjacent(Loc(A), Loc(B)) \rightarrow \\ Stench(R2), do(\{move(B, R2)\}, S_0))$$

From a previous example, we have that Ann knows $Loc(B, do(\{move(B, R2)\}, S_0)) = R2$, so this will reduce the following tautology:

$$\mathbf{Knows}(A, Stench(R2) \wedge Adjacent(Loc(A), R2) \rightarrow \\ Stench(R2), do(\{move(B, R2)\}, S_0))$$

Which is clearly entailed by the domain.

Example 6. *After moving to room R4 Ann observes a stench, knows that there is a stench in both R2 and R4:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, Stench(R2) \wedge Stench(R4), \\ do([\{move(B, R2)\}, \{alert(B)\}, \{move(A, R4)\}], S_0))$$

And hence knows that the wumpus is in room R5:

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, Wumpus() = R5, \\ do([\{move(B, R2)\}, \{alert(B)\}, \{move(A, R4)\}], S_0))$$

The mechanics of Ann learning $Stench(R4)$ are identical to Example 3, so we will not repeat them here. The interesting point is that, since agents know all logical consequences of their knowledge, Ann can deduce the location of the Wumpus based on:

$$\mathcal{D} \models Stench(R2, s) \wedge Stench(R4, s) \rightarrow Wumpus(s) = R5$$

Example 7. *Ann doesn't know where Bob is, since she can't observe his footsteps from room R4.*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \neg\exists r : \mathbf{Knows}(A, Loc(B) = r, \\ do([\{move(B, R2)\}, \{alert(B)\}, \{move(A, R4)\}], S_0))$$

Ann isn't adjacent to Bob, so she is no longer assured of hearing his footsteps if he moves out of room $R2$.

Formally, using $\sigma = do([\{move(B, R2)\}, \{alert(B)\}], S_0)$ we have:

$$\begin{aligned}\mathcal{R}(\neg\exists r : \mathbf{Knows}(A, Loc(B) = r, do(\{move(A, R4)\}, \sigma))) \Rightarrow \\ \exists o : \; o = Obs(A, \{move(A, R4)\}, \sigma) \\ \wedge \neg\exists r : \mathbf{Knows}(A, \forall c' : Poss(c') \wedge Obs(A, c') = o \rightarrow \\ \mathcal{R}(\mathcal{P}(Loc(B) = r), \mathbf{PbU}(A), c'), \sigma)\end{aligned}$$

Expanding "$\exists o$" and "$\forall c'$" as before, we find that the only relevant value of c' is $\{move(A, R4)\}$. Recall from Example 4 that:

$$\begin{aligned}\mathcal{P}(Loc(B) = r, \mathbf{PbU}(A)) \Rightarrow \\ Loc(B) = r \wedge (Loc(A) = r \vee Adjacent(r, Loc(A)))\end{aligned}$$

Regressing this over $move(A, R4)$ fixes Ann's location as $R4$, producing:

$$\begin{aligned}\mathcal{R}(\mathcal{P}(Loc(B) = r, \mathbf{PbU}(A)), \{move(A, R4\}) \Rightarrow \\ Loc(B) = r \wedge (R4 = r \vee Adjacent(r, R4))\end{aligned}$$

In Example 4 we also established that:

$$\mathbf{Knows}(A, Loc(B) = R2, \sigma)$$

So there can be no value of r satisfying $(R4 = r \vee Adjacent(r, R4))$, which is required in this expression, and therefore no r satisfying the initial query.

Example 8. *Ann shoots the wumpus, observes the scream and knows the wumpus is dead.*

$$\begin{aligned}\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, Killed, \\ do([\{move(B, R2)\}, \{alert(B)\}, \{move(A, R4)\}, \{shoot(A, R5)\}], S_0))\end{aligned}$$

Abbreviating all but the last action as σ, and using $Obs(A, \{shoot(A, R5)\}, \sigma) = \{shoot(A, R5), scream\}$, we have:

$$\begin{aligned}\mathcal{R}(\mathbf{Knows}(A, Killed, do(\{shoot(A, R5)\}, \sigma))) \Rightarrow \\ \mathbf{Knows}(A, Poss(\{shoot(A, R5)\}) \\ \wedge Obs(A, \{shoot(A, R5)\}) = \{shoot(A, R5), scream\} \\ \rightarrow \mathcal{R}(\mathcal{P}(Killed, \mathbf{PbU}(A), \{shoot(A, R5)\}), \sigma)\end{aligned}$$

No actions can cause *Killed* to become false, so:

$$\mathcal{P}(Killed, \mathbf{PbU}(A)) \Rightarrow Killed$$

Regressing over action $\{shoot(A, B5)\}$ results in:

$$\mathcal{R}(\mathcal{P}(Killed, \mathbf{PbU}(A)), \{shoot(A, R5)\}) \Rightarrow Killed \vee Wumpus() = R5$$

Substituting definitions of *Poss* and *Obs* we get:

$$\mathbf{Knows}(A, Adjacent(Loc(A), R5) \wedge \neg Killed \wedge Wumpus() = R5 \rightarrow Killed \vee Wumpus() = R5, \sigma)$$

This tautology is clearly entailed by the domain.

Example 9. *Ann knows that Bob knows the wumpus is dead, as he will have heard the scream regardless of his location.*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, \mathbf{Knows}(B, Killed), do([\{move(B, R2)\}, \{alert(B)\}, \{move(A, R4)\}, \{shoot(A, R5)\}], S_0))$$

Abbreviating all but the last action as σ, and using $Obs(A, \{shoot(A, R5)\}, \sigma) = \{shoot(A, R5), scream\}$, as in Example 8, we have:

$$\begin{aligned}\mathcal{R}(\mathbf{Knows}(A,&\mathbf{Knows}(B, Killed), do(\{shoot(A, R5)\}, \sigma))) \Rightarrow \\ &\mathbf{Knows}(A, Poss(\{shoot(A, R5)\}) \\ &\quad \wedge Obs(A, \{shoot(A, R5)\}) = \{shoot(A, R5), scream\} \\ &\quad \rightarrow \mathcal{R}(\mathcal{P}(\mathbf{Knows}(B, Killed), \mathbf{PbU}(A), \{shoot(A, R5)\}), \sigma)\end{aligned}$$

Since no future actions can cause *Killed* to be false:

$$\mathcal{P}(\mathbf{Knows}(B, Killed), \mathbf{PbU}(A)) \Rightarrow \mathbf{Knows}(B, Killed)$$

We then need to regress this inner knowledge expression over the action $\{shoot(A, R5)\}$, which can proceed in much the same way as previous examples:

$$\begin{aligned}&\mathcal{R}(\mathbf{Knows}(B, Killed), \{shoot(A, R5)\}) \Rightarrow \\ &\quad \exists o : o = Obs(B, \{shoot(A, R5)\}) \\ &\quad \wedge [o = \emptyset \rightarrow \mathbf{Knows}(B, Killed)] \\ &\quad \wedge [o \neq \emptyset \rightarrow \mathbf{Knows}(B, \forall c' : Poss(c') \wedge Obs(B, c') = o \rightarrow \mathcal{R}(\mathcal{P}(Killed), c'))]\end{aligned}$$

Note, however, that we cannot expand the "$\exists o$" into a disjunction based on the value of $Obs(B, \{shoot(A, R5)\}, \sigma)$. We must instead enumerate all possible observations that Ann thinks he might make. There are three cases to consider:

- $Obs(B, \{shoot(A, R5)\}, s) = \{shoot(A, R5), scream\}$
- $Obs(B, \{shoot(A, R5)\}, s) = \{shoot(A, R5)\}$
- $Obs(B, \{shoot(A, R5)\}, s) = \{scream\}$

From previous examples, we have that $\mathbf{Knows}(A, Wumpus() = R5, \sigma)$ and $\mathbf{Knows}(A, Loc(B) \neq R4, \sigma)$. This is enough to limit Bob's potential observations to a single option:

$$\mathbf{Knows}(A, Obs(B, \{shoot(A, R5)\}) = \{scream\}, \sigma)$$

So regression of the inner expression can proceed in an identical manner to Example 8.

In fact, at this point the hunters will have *common knowledge* that the Wumpus is dead. The current formalism is not rich enough to reason directly about common knowledge, but we have preliminary work on this topic in [18] that could be integrated with the approach taken here.

7. Related and Future Work

There has been a great deal of work on multi-agent systems in the situation calculus, including: specification of multi-agent systems Shapiro et al. [52]; theories of coordination [13] and ability [22]; reasoning about the epistemic feasibility of plans [23]; analysing multi-player games [1]; and our own work on the cooperative execution of Golog programs [16]. Many of these explicitly assume that all actions are public, in order to use regression to reason about knowledge. Our richer theory of knowledge could immediately be used to extend and generalise formalisms such as these to more complex multi-agent domains.

We have developed the first principled axiomatisation of the observability of actions, using the notion of observations and views as analogues of actions

and situations that are localised to an individual agent. This terminology has been deliberately chosen to match similar concepts in other formalisations of knowledge, such as the well-known treatise of Halpern and Moses [15]. By reifying these concepts as terms in the logic, we are able to give a succinct definition of the dynamics of the knowledge fluent and prove that its behaviour matches our intuitive expectations.

A further advantage of our explicit axiomatisation of observations is in establishing properties of the knowledge relation. A major theorem of Scherl and Levesque [45] states that if the K-relation is reflexive, symmetric or transitive at the initial situation, then it has that property at every situation. In our formulation these are all simple corollaries of Theorem 1, based on the reflexive, symmetric and transitive nature of the equality symbol.

As we discussed in Sections 3 and 5, different kinds of information-generating actions have previously been modelled by directly modifying the successor state axiom for K [48, 21, 50, 32, 49]. Unfortunately, these extensions do not in general maintain the important theorems of the standard account of knowledge [45]; there is no guarantee that a modified axiom will preserve accessibility properties of K, or will permit a regression rule for **Knows**. This approach is not *elaboration tolerant*.

By contrast, the proof of Theorem 3 and the examples in Section 5 have shown that our approach is naturally elaboration tolerant in the face of private actions, guarded sensing actions, speech acts, and other extensions that have required special treatment in previous work. By basing our approach on an explicit, separately-axiomatised account of the local perspective of each agent, our formulation robustly maintains both accessibility property preservation and a regression rule regardless of the specific semantics of information-producing actions.

Our approach depends crucially on the ability to do epistemic reasoning about unrestricted forms of nested knowledge in K_n. Several promising tractable approaches have been developed for such reasoning, such as restricting theories to proper epistemic knowledge bases (PEKBs) and transforming queries into a certain normal form, which is complete and tractable for important classes of applications [20]. Lakemeyer and Lespérance [20] point out that reasoning about other agents' actions is really also required in such systems, and highlight the importance of future work on the first-order case. Our approach could form a component of such work.

While our formalism has explicitly focused on modelling knowledge, there has also been substantial work on modelling *belief* in the situation calculus,

where agents may be mistaken about the state of the world [51]. In such systems there are alternatives to reasoning about arbitrarily-long sequences of hidden actions, such as assuming that no hidden actions occur until there is evidence to the contrary. Shapiro and Pagnucco [47] show how agents may hypothesise the occurrence of hidden exogenous actions when they discover that their beliefs are wrong. We note that these belief-based systems are also based on the original formulation of epistemic modalities by Scherl and Levesque [45], therefore aspects of our approach could be used to enrich such systems.

Our explicit notions of observations and views could also provide benefits for such formalisms, as well as for the formulation of other modalities (e.g goals as in [49]) in the situation calculus. Our approach could facilitate the monitoring of goal achievement during execution, where ongoing commitment to the planned execution of goals is subject to what an agent believes. We could potentially answer the relativized persistent goal queries of Cohen and Levesque [5], where agents must consider dropping persistent goals should they believe that they have become unachievable.

The knowledge formalism we have developed here is *permissive*, in that it assumes the world could potentially evolve via any legal sequence of actions. In the wider field of epistemic reasoning, it is common to constrain the world to evolve according to a given *protocol* [15, 10, 55]. One then speaks of an agent's knowledge under a particular protocol. The situation calculus has shown promise for verifying protocols, such as cryptographic protocols [8].

As discussed in [55], permissive formulations of knowledge can easily be extended to support *local* protocols, where the allowable next actions can be determined based on the current state of the world. Our use of $Poss$ in the axioms for knowledge could easily be replaced with predicates axiomatising actions that are, for example, $Permissable$ or $Motivated$. But recent work by Fritz et al. [12] also presents an intriguing possibility to extend our approach to more general protocols.

The most natural language for expressing a protocol in the situation calculus is Golog, so one may wish to reason about an agent's knowledge assuming the world evolves as specified by the Golog program δ:

$$K_p(agt, \delta, s', s) \stackrel{\text{def}}{=} K(agt, s', s) \wedge \exists s'', \delta' : \mathit{Init}(s'') \wedge \mathit{Trans}^*(\delta, s'', \delta', s')$$

Such knowledge would be queried like so:

$$\mathcal{D} \cup \mathcal{D}_{golog} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}_p(agt, \delta, \phi, \sigma)$$

It is this kind of knowledge that would be needed to integrate epistemic reasoning into our MIndiGolog execution planner [16], as the agents in that case know their teammates will act according to the shared control program.

Fritz et al. [12] have demonstrated that the details of a given ConGolog program δ can be *compiled into* a theory of action $\mathcal{D}$, producing a new theory $\mathcal{D}^\delta$ in which the only legal situations are those that form part of a legal execution of δ:

$$\mathcal{D} \cup \mathcal{D}_{golog} \models \exists \delta' : Trans^*(\delta, S_0, \delta', \sigma)$$
$$\text{iff}$$
$$\mathcal{D}^\delta \models Legal(\sigma)$$

So to investigate the knowledge of an agent under a protocol, we could use the compilation technique of [12] to get a query which can be handled using our formalism:

$$\mathcal{D} \cup \mathcal{D}_{golog} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}_p(agt, \delta, \phi, \sigma)$$
$$\text{iff}$$
$$\mathcal{D}^\delta \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(agt, \phi, \sigma)$$

The details are not quite so straightforward, as the compilation procedure introduces some auxiliary actions and fluents that should be hidden from the agent's knowledge. However, it does offer a possibility for future work.

Another promising avenue of future research is to bound the number of fluents maintained at any point in time, based on the implicit assumption that agents will eventually forget facts after a certain period of time. In a model checking setting this leads to an abstract transition system with a finite number of states that can be decidably checked, and can still represent infinite executions [7]. This bounds the number of fluents in the active domain and could have merit for computing the property persistence fixpoint of $\mathcal{P}$.

Finally, we note that a multi-agent logic of knowledge and action naturally raises the question of *common knowledge*, a key concept in the coordination and distributed systems literature [15]. The formalism presented here does not include a common knowledge operator, as it would be difficult to handle using regression – indeed, Batlag et al. [2] have shown that epistemic logic with both actions and common knowledge is more expressive than epistemic logic with common knowledge alone. The proper treatment of common knowledge requires a much richer language of *complex epistemic modalities*

in the style of van Benthem et al. [56]. In previous work we have introduced such modalities to the situation calculus for synchronous domains [18], and our ongoing research will unify that approach with the asynchronous formalism developed in this paper.

8. Conclusion

We have provided the first foundational account of epistemic reasoning in the situation calculus in asynchronous multi-agent domains. A principled axiomatisation of the observability of actions is used to explicitly define an agent's knowledge in terms of its local view. By reifying observations and views as terms in the logic, we are able to give a succinct definition of the dynamics of the knowledge fluent and prove that its behaviour matches our intuitive expectations.

The standard account of knowledge requires that domains be synchronous, so that the **Knows** operator need not consider arbitrarily-long sequences of hidden actions. Handling hidden actions requires a second-order induction axiom to universally quantify over situation terms, precluding the use of regression for effective automated reasoning. We have shown that this second-order axiom can be factored out of the regression rules, and replaced with a fixpoint calculation performed by the meta-level reasoning machinery. This fixpoint calculation requires theorem proving based on a very restricted set of axioms, offering a significant advantage over generic second-order theorem proving. Several decidable classes and tractable techniques can be realised as special cases of this generic result.

We have demonstrated that our account of knowledge is expressive enough to capture the standard account of knowledge based on public actions, as well as more complex formulations where the observability of actions depends on the state of the world. Moreover, it maintains its important theorems and the availability of a regression rule as more complex kinds of information-producing action are introduced.

Finally, and perhaps most importantly, we have shown that a simple modification to regression rules allows a situated agent to reason directly about its own knowledge using only its local view, rather than constructing a query that universally quantifies over all situations compatible with its view. Our new observation-based semantics thus provides a powerful account of knowledge suitable both for reasoning *about*, and for reasoning *in*, asynchronous multi-agent domains.

9. Acknowledgements

This work has been partially supported by an Australian Research Council (ARC) Discovery project (Foundations of human-agent collaboration: situation-relevant information sharing), grant No. DP130102825.

References

[1] Bradley Bart, James P. Delgrande, and Oliver Schulte. Knowledge and Planning in an Action-Based Multi-agent Framework: A Case Study. In *Advances in Artificial Intelligence*, volume 2056 of *LNAI*, pages 121–130. Springer, 2001.

[2] Alexandru Batlag, Lawrence S. Moss, and Slawomir Solecki. The Logic of Public Announcements and Common Knowledge and Private Suspicions. In *Proceedings of the 7th conference on Theoretical Aspects of Rationality and Knowledge (TARK'98)*, pages 43–56, 1998.

[3] Bernhard Beckert and Joachim Posegga. LeanT^{A}P: Lean Tableaue-based Deduction. *Journal of Automated Reasoning*, 15:339–358, 1995.

[4] Jens Claßen and 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)*, pages 589–599, 2008.

[5] Philip R Cohen and Hector J Levesque. Intention is choice with commitment. *Artificial intelligence*, 42(2):213–261, 1990.

[6] Giuseppe De Giacomo, Raymond Reiter, and Mikhail Soutchanski. Execution Monitoring of High-Level Robot Programs. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 453–465, 1998.

[7] Giuseppe De Giacomo, Yeves Lespérance, and Fabio Patrizi. Bounded situation calculus action theories and decidable verification. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, pages 467–477, 2012.

[8] James P Delgrande, Aaron Hunter, and Torsten Grote. On the representation and verification of cryptographic protocols in a theory of action. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 39–45. IEEE, 2010.

[9] Robert Demolombe and Maria del Pilar Pozos Parra. A Simple and Tractable Extension of Situation Calculus to Epistemic Logic. In *Proceedings of the 12th International Symposium on Foundations of Intelligent Systems (ISMIS'00)*, pages 515–524, 2000.

[10] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, Cambridge, Massachusetts, 1995.

[11] Melvin Fitting. LeanT^{A}P Revisited. *Journal of Logic and Computation*, 8(1):33–47, 1998.

[12] Christian Fritz, Jorge A. Baier, and Sheila A. McIlraith. ConGolog, Sin Trans: Compiling ConGolog into Basic Action Theories for Planning and Beyond. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 600–610, 2008.

[13] Hojjat Ghaderi, Hector Levesque, and Yves Lespérance. A Logical Theory of Coordination and Joint Ability. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 421–426, 2007.

[14] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[15] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[16] Ryan F. Kelly and Adrian R. Pearce. Towards High-Level Programming for Distributed Problem Solving. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'06)*, pages 490–497, 2006.

[17] Ryan F. Kelly and 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)*, pages 841–843, 2007.

[18] Ryan F. Kelly and Adrian R. Pearce. Complex Epistemic Modalities in the Situation Calculus. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 611–620, 2008.

[19] Ryan F. Kelly and Adrian R. Pearce. Property Persistence in the Situation Calculus. *Artificial Intelligence*, 174:865–888, 2010.

[20] Gerhard Lakemeyer and Yves Lespérance. Efficient reasoning in multiagent epistemic logics. In *Proceedings of the 20th biennial European Conference on Artificial Intelligence (ECAI'12)*, pages 498–503, 2012.

[21] Y. Lespérance, H. J. Levesque, and R. Reiter. A Situation Calculus Approach to Modeling and Programming Agents. In Rao A. and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*, pages 275–299. Kluwer, 1999.

[22] Y. Lespérance, H. J. Levesque, F. Lin, and R. B. Scherl. Ability and Knowing How in the Situation Calculus. *Studia Logica*, 66(1):165–186, October 2000.

[23] Yves Lespérance. On the Epistemic Feasibility of Plans in Multiagent Systems Specifications. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages*, volume 2333 of *Lecture Notes in Artificial Intelligence*, pages 69–85, 2001.

[24] Hector J. Levesque, Ray Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.

[25] F. Lin and H. Levesque. What robots can do: robot programs and effective achievability. *Artificial Intelligence*, 101:201–226, 1998.

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

[27] Fangzhen Lin and Ray Reiter. How to progress a database. *Artificial Intelligence*, 92:131–167, 1997.

[28] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[29] Robert C. Moore. Reasoning about Knowledge and Action. Technical Note 191, SRI International, October 1980.

[30] Rohit Parikh and R. Ramanujam. Distributed Processes and the Logic of Knowledge. In *Proceedings of the Conference on Logic of Programs*, pages 256–268. Springer-Verlag, 1985.

[31] Ron Petrick and Hector Levesque. Knowledge equivalence in Combined Action Theories. In *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR'02)*, pages 303–314, 2002.

[32] Ronald P. A. Petrick. *A Knowledge-level approach for effective acting, sensing, and planning*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 2006.

[33] Ronald P. A. Petrick. Cartesian Situations and Knowledge Decomposition in the Situation Calculus. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 629–639, 2008.

[34] Javier Pinto. Concurrent Actions and Interacting Effects. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 292–303, 1998.

[35] Javier Pinto. Concurrency and Action Interaction. Technical report, 2000.

[36] Javier A. Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1994.

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

[38] 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. Academic Press Professional, Inc., 1991.

[39] Ray Reiter. Proving Properties of States in the Situation Calculus. *Artificial Intelligence*, 64:337–351, 1993.

[40] Ray Reiter. Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 2–13, 1996.

[41] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[42] Sebastian Sardina and Stavros Vassos. The wumpus world in indigolog: A preliminary report. In *Proceedings of the Workshop on Non-monotonic Reasoning, Action and Change at IJCAI (NRAC-05)*, pages 90–95, 2005.

[43] Francesco Savelli. Existential assertions and quantum levels on the tree of the situation calculus. *Artificial Intelligence*, 170(6):643–652, 2006.

[44] Richard Scherl. Reasoning about the interaction of knowledge, time and concurrent actions in the situation calculus. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 1091–1098, 2003.

[45] Richard Scherl and Hector Levesque. Knowledge, Action, and the Frame Problem. *Artificial Intelligence*, 144:1–39, 2003.

[46] S. Schiffel and M. Thielscher. Reconciling Situation Calculus and Fluent Calculus. In *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference (AAAI'06/IAAI'06)*, pages 287–292, 2006.

[47] S. Shapiro and M. Pagnucco. Iterated Belief Change and Exogenous Actions in the Situation Calculus. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, pages 878–882, 2004.

[48] S. Shapiro, Y. Lespérance, and H. J. Levesque. Specifying Communicative Multi-Agent Systems. In *Agents and Multi-Agent Systems - Formalisms, Methodologies, and Applications*, volume 1441 of *Lecture Notes in Artificial Intelligence*, pages 1–14, 1998.

[49] S. Shapiro, Y. Lesperance, and H. Levesque. Goal Change in the Situation Calculus. *Journal of Logic and Computation*, 17:983–1018, 2007.

[50] Steven Shapiro and Yves Lespérance. Modeling Multiagent Systems with the Cognitive Agents Specification Language — A Feature Interaction Resolution Application. In *Intelligent Agents Volume VII — Proceedings of the 2000 Workshop on Agent Theories, Architectures, and Languages*, pages 244–259. Springer-Verlag, 2001.

[51] Steven Shapiro, Maurice Pagnucco, Yves Lespérance, and Hector J. Levesque. Iterated Belief Change in the Situation Calculus. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pages 527–538, 2000.

[52] Steven Shapiro, Yves Lespérance, and 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)*, pages 19–26, 2002.

[53] M. Thielscher. A Unifying Action Calculus. *Artificial Intelligence*, 175 (1):120–141, 2011.

[54] Johan van Benthem. Modal Logic meets Situation Calculus. Technical Report PP-2007-04, University of Amsterdam, 2007.

[55] Johan van Benthem and Eric Pacuit. The Tree of Knowledge in Action: Towards a Common Perspective. In *Advances in Modal Logic*, volume 6, pages 87–106, 2006.

[56] Johan van Benthem, Jan van Eijck, and Barteld Kooi. Logics of Communication and Change. *Information and Computation*, 204(II):1620–1662, 2006.

[57] Hans van Ditmarsch, Andreas Herzig, and Tiago de Lima. Optimal regression for reasoning about knowledge and actions. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1070–1075, 2007.

[58] Stavros Vassos and 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)*, pages 1004–1009, 2008.

Appendix A. Hunt the Wumpus Domain Axioms

The following are the complete axioms for the "Hunt the Wumpus" example domain from Section 6. Free variables are assumed to be universally quantified at outermost scope.

The background theory $\mathcal{D}_{bg}$ defines room adjacency, including symmetry of the predicate:

$$\begin{aligned}
Adjacent(r, r') \equiv\ & (r = R1 \wedge r' = R2) \vee (r = R1 \wedge r' = R4) \\
& \vee (r = R2 \wedge r' = R1) \vee (r = R2 \wedge r' = R3) \vee (r = R2 \wedge r' = R5) \\
& \vee (r = R3 \wedge r' = R2) \vee (r = R3 \wedge r' = R6) \vee (r = R4 \wedge r' = R1) \\
& \vee (r = R4 \wedge r' = R5) \vee (r = R4 \wedge r' = R7) \vee (r = R5 \wedge r' = R2) \\
& \vee (r = R5 \wedge r' = R4) \vee (r = R5 \wedge r' = R6) \vee (r = R5 \wedge r' = R8) \\
& \vee (r = R6 \wedge r' = R3) \vee (r = R6 \wedge r' = R5) \vee (r = R6 \wedge r' = R9) \\
& \vee (r = R7 \wedge r' = R4) \vee (r = R7 \wedge r' = R8) \vee (r = R8 \wedge r' = R5) \\
& \vee (r = R8 \wedge r' = R7) \vee (r = R8 \wedge r' = R9) \vee (r = R9 \wedge r' = R6) \\
& \vee (r = R9 \wedge r' = R8)
\end{aligned}$$

The background theory contains the usual unique names axioms, asserting that terms with different types or arguments are in fact different. The form is standard and we do not reproduce the full set here. Example axioms are:

$$\begin{aligned}
move(agt1, r1) \ &\neq\ shoot(agt2, r2) \\
move(agt1, r1) = move(agt2, r2) \ &\rightarrow\ agt1 = agt2 \ \wedge\ r1 = r2
\end{aligned}$$

It also captures the fact that only rooms adjacent to the Wumpus have a stench, using the following state invariant:

$$Stench(r, s) \equiv Adjacent(r, Wumpus(s))$$

The axioms $\mathcal{D}_{adp}$ define the precondition predicate $Poss$:

$$\begin{aligned}
Poss(c, s) \equiv\ & c \neq \emptyset \\
& \wedge \forall a, a' \in c : actor(a) = actor(a') \equiv a = a' \\
& \wedge move(agt, r) \in c \rightarrow Adjacent(r, Loc(agt, s)) \\
& \wedge shoot(agt, r) \in c \rightarrow Adjacent(r, Loc(agt, s)) \\
& \wedge alert(agt) \in c \rightarrow Stench(Loc(agt, s), s)
\end{aligned}$$

And the observation function Obs:

$$
\begin{aligned}
&Obs(agt, c, s) = o \equiv \\
&\quad move(agt1, r1) \in o \equiv move(agt1, r1) \in c \\
&\qquad \wedge [Loc(agt, s) = Loc(agt1, s) \vee Loc(agt, s) = r1] \\
&\quad \wedge shoot(agt1, r1) \in Obs(agt, c, s) \equiv shoot(agt1, r1) \in c \\
&\qquad \wedge Loc(agt, s) = Loc(agt1, s) \\
&\quad \wedge alert(agt1) \in Obs(agt, c, s) \equiv alert(agt1) \in c \\
&\qquad \wedge Loc(agt, s) = Loc(agt1, s) \\
&\quad \wedge footsteps \in Obs(agt, c, s) \equiv \exists agt1, r1 : move(agt1, r1) \in c \\
&\qquad \wedge [Adjacent(Loc(agt, s), r1) \vee Adjacent(Loc(agt, s), Loc(agt1, s))] \\
&\quad \wedge alert \in Obs(agt, c, s) \equiv \exists agt1 : alert(agt1) \in c \\
&\qquad \wedge Adjacent(Loc(agt, s), Loc(agt1, s)) \\
&\quad \wedge stench \in Obs(agt, c, s) \equiv \exists r1 : move(agt, r1) \in c \\
&\qquad \wedge Stench(r1, s) \\
&\quad \wedge scream \in Obs(agt, c, s) \equiv \exists agt1, r1 : shoot(agt1, r1) \in c \\
&\qquad \wedge r1 = Wumpus(s)
\end{aligned}
$$

The successor state axioms $\mathcal{D}_{ssa}$ define fluent change as follows:

$$
\begin{aligned}
r = Loc(agt, do(c, s)) &\equiv move(agt, r) \in c \\
&\quad \vee (r = Loc(agt, s) \wedge \neg \exists r' : move(agt, r') \in c) \\
Killed(do(c, s)) &\equiv Killed(s) \\
&\quad \vee \exists agt, r : shoot(agt, r) \in c \wedge r = Wumpus(s) \\
r = Wumpus(do(c, s)) &\equiv r = Wumpus(s) \\
Stench(r, do(c, s)) &\equiv Stench(r, s)
\end{aligned}
$$

The initial state axioms $\mathcal{D}_{S_0}$ state that the Wumpus is in room $R5$:

$$Wumpus(S_0) = R5$$

And that the hunters begin in room $R1$ with identical knowledge, knowing

the state of $R1$ but not of any other rooms:

$$Init(s) \rightarrow Loc(agt, s) = R1$$
$$Init(s) \rightarrow Wumpus(s) \neq R1 \wedge \neg Stench(R1, s)$$
$$K_0(agt, s', s) \equiv K_0(agt', s', s)$$
$$r \neq R1 \rightarrow \exists s : K_0(agt, s, S_0) \wedge Wumpus(s) = r$$
$$r \neq R1 \rightarrow \exists s : K_0(agt, s, S_0) \wedge Wumpus(s) \neq r$$
$$r \neq R1 \rightarrow \exists s : K_0(agt, s, S_0) \wedge Stench(r, s)$$
$$r \neq R1 \rightarrow \exists s : K_0(agt, s, S_0) \wedge \neg Stench(r, s)$$

Appendix B. Background State Invariants

We allow the background theory $\mathcal{D}_{bg}$ to include simple state invariants in the style of [26], as long as their maintenance is implied by the successor state axioms. That is, if they hold initially, then the successor state axioms must ensure that they will hold for all future situations.

Formally, let $\mathcal{D}_{inv} \subset \mathcal{D}_{bg}$ be the set of such invariants, each of which have the form:

$$(\forall s : \phi[s]) \in \mathcal{D}_{inv}$$

Let $\mathcal{D}_{inv0}$ be the same formulae applied to just the initial situations:

$$\mathcal{D}_{inv0} = \{Init(s) \rightarrow \phi[s] \quad | \quad (\forall s : \phi[s]) \in \mathcal{D}_{inv}\}$$

Then we require that:

$$\Sigma \cup \mathcal{D}_{ad} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{bg} \models \Psi$$
$$\text{iff}$$
$$\Sigma \cup \mathcal{D}_{ad} \cup \mathcal{D}_{ssa} \cup (\mathcal{D}_{S_0} \cup \mathcal{D}_{inv0}) \cup (\mathcal{D}_{bg} - \mathcal{D}_{inv}) \models \Psi$$

Such invariants thus do not imply any additional dynamics of the world, but merely provide an explicit encoding of static facts of the domain, such as fixed relationships between fluents.

Typical approaches to state invariants in the situation calculus, such as [26], would modify $\mathcal{D}_{ssa}$ and $\mathcal{D}_{S_0}$ to ensure maintenance of the constraints as above, then discard $\mathcal{D}_{inv}$ from the theory as redundant.

We keep the invariants as an explicit part of the background theory because they are useful when calculating the persistence condition. The algorithm for calculating $\mathcal{P}(\phi, \alpha)$ requires reasoning relative to $\mathcal{D} - \mathcal{D}_{S_0}$ and hence cannot take advantage of implicitly-encoded state constraints.

Appendix C. Complete Proofs

Lemma 1. *For situation terms s and s'', and agent agt:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models Legal(s) \wedge s \leq_{\mathbf{PbU}(agt)} s'' \rightarrow Legal(s'')$$

Proof. By induction on situation s''. The base case is $s'' = s$, for which the lemma is trivial. The inductive case is $s'' = do(c', s')$ where $s \leq_{\mathbf{PbU}(agt)} s'$ and $\mathbf{PbU}(agt, c', s')$. Expanding the definition of *Legal*, the inductive hypothesis gives $root(s') \leq_{Poss} s'$. Since $\mathbf{PbU}(agt, c', s')$ implies $Poss(c', s')$ we have $root(s') \leq_{Poss} do(c', s')$, which matches the definition of $Legal(s'')$ as required. □

Lemma 2. *For situation terms s and s'', and agent agt:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models s \leq_{\mathbf{PbU}(agt)} s'' \rightarrow View(agt, s) = View(agt, s'')$$

Proof. By induction on situation s''. The base case is $s'' = s$, for which the lemma is trivial. The inductive case is $s'' = do(c', s')$ where $s \leq_{\mathbf{PbU}(agt)} s'$ and $\mathbf{PbU}(agt, c', s')$. In this case we have that $\mathbf{PbU}$ implies $Obs(agt, c', s') = \emptyset$, so $View(agt, s'') = View(agt, s')$ by equation (5), and the inductive hypothesis of $View(agt, s') = View(agt, s)$ gives the required result. □

Lemma 3. *For situation terms s and s'', and agent agt:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models K(agt, s'', s) \rightarrow K_0(agt, root(s''), root(s))$$

Proof. By induction on situation s. In the base case of $Init(s)$, equation (9) ensures there is some s' such that $K_0(agt, s', s) \wedge s' \leq_{\mathbf{PbU}(agt)} s''$, and we have the desired implication using $root(s'') = root(s') = s'$.

For the $do(c, s)$ case, suppose that we have $K(agt, s'', do(c, s))$. Then by equation (8) there is some s' such that $s' \sqsubseteq s''$ and $K(agt, s', s)$. Then $root(s'') = root(s')$, and $K_0(agt, root(s'), root(s))$ by the inductive hypothesis, giving the required result. □

Lemma 4. *For situation terms s and s'', and agent agt:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models K(agt, s'', s) \rightarrow Legal(s'')$$

Proof. By induction on situation s. In the base case of $Init(s)$, using equation (9), if $K(agt, s'', s)$ then there must be an s' such that $K_0(agt, s', s)$ and $s' \leq_{\mathbf{PbU}(agt)} s''$. The foundational axioms ensure that $K_0(agt, s', s) \rightarrow Init(s')$, so all such s' are legal, making s'' legal by Lemma 1.

For the $do(c, s)$ case, the inductive hypothesis gives that $K(agt, s', s) \rightarrow Legal(s')$ and we split on $Obs(agt, c, s)$. Suppose $Obs(agt, c, s) = \emptyset$, then equation (8) gives $K(agt, s'', s)$ and $Legal(s'')$ follows immediately from the inductive hypothesis.

Alternately, suppose $Obs(agt, c, s) \neq \emptyset$, then equation (8) ensures there must be a c', s' satisfying three conditions: $K(agt, s', s)$, $Poss(c', s')$ and $do(c', s') \leq_{\mathbf{PbU}(agt)} s''$. The first of these yields $Legal(s')$ by the inductive hypothesis. Given that, the second yields $Legal(do(c', s'))$ by the definition of $Legal$. Given that, the third yields $Legal(s'')$ by Lemma 1 and we have the required result in both cases. □

Lemma 5. *For situation terms s and s'', and agent agt:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models K(agt, s'', s) \rightarrow View(agt, s'') = View(agt, s)$$

Proof. By induction on situation s. For the base case of $Init(s)$, using equation (9), $K(agt, s'', s)$ implies that there must be an s' such that $Init(s')$ and $s' \leq_{\mathbf{PbU}(agt)} s''$. Then $View(s') = View(s) = \epsilon$ by definition, and $View(s'') = View(s')$ by Lemma 2 to give the required result.

For the $do(c, s)$ case, suppose $Obs(agt, c, s) = \emptyset$. The definition of $View$ gives $View(agt, do(c, s)) = View(agt, s)$, while equation (8) gives $K(agt, s'', s)$ and hence $View(agt, s'') = View(agt, s)$ by the inductive hypothesis.

Alternately, suppose $Obs(agt, c, s) \neq \emptyset$, then equation (8) gives us s',c' where $do(c', s') \leq_{\mathbf{PbU}(agt)} s''$, $Obs(agt, c, s) = Obs(agt, c', s')$, and $K(agt, s', s)$. The definition of $View$ gives:

$$View(agt, do(c, s)) = Obs(agt, c, s) \cdot View(agt, s)$$
$$View(agt, do(c', s')) = Obs(agt, c', s') \cdot View(agt, s')$$

The inductive hypothesis gives us $View(agt, s') = View(agt, s)$ and hence $View(agt, do(c', s')) = View(agt, do(c, s))$, while Lemma 2 completes the equivalence by giving us $View(agt, s'') = View(agt, do(c', s'))$. □

Lemma 6. *For any situation s and agent agt:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathit{Legal}(s) \wedge \mathit{View}(agt, s) = \epsilon \rightarrow \mathit{root}(s) \leq_{\mathbf{PbU}(agt)} s$$

Proof. By induction on situation s. For the base case of $\mathit{Init}(s)$ the lemma is trivial since $\mathit{root}(s) = s$.

For the $do(c, s)$ case, the empty view implies $\mathit{Obs}(agt, c, s) = \emptyset$ and $\mathit{View}(agt, s) = \epsilon$, while $\mathit{Legal}(do(c, s))$ implies $\mathit{Poss}(c, s)$ and $\mathit{Legal}(s)$. This gives $s \leq_{\mathbf{PbU}(agt)} do(c, s)$, and using $\mathit{root}(s) \leq_{\mathbf{PbU}(agt)} s$ from the inductive hypothesis gives the required result. □

Lemma 7. *For any situation s, observations o, view v and agent agt:*

$$\begin{aligned}\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathit{Legal}(s) \wedge & \mathit{View}(agt, s) = o \cdot v \rightarrow \\ & \exists c', s' : \mathit{Legal}(s') \wedge \mathit{View}(agt, s') = v \\ & \wedge \mathit{Obs}(agt, c,' s') = o \wedge do(c', s') \leq_{\mathbf{PbU}(agt)} s\end{aligned}$$

Proof. By induction on situation s. For the base $\mathit{Init}(s)$ case we have $\mathit{View}(agt, s) = \epsilon$, so the LHS is never true and the lemma is trivial. For the $do(c, s)$ case we split on whether c is observable.

Suppose $\mathit{Obs}(agt, c, s) \neq \emptyset$. Then by definition of View we have that $\mathit{Obs}(agt, c, s) = o$ and $\mathit{View}(agt, s) = v$, and the $\exists c', s'$ on the RHS of the lemma is trivially satisfied by c and s themselves.

Alternately, suppose $\mathit{Obs}(agt, c, s) = \emptyset$. Then by definition of View we have $\mathit{View}(agt, do(c, s)) = \mathit{View}(agt, s)$. The inductive hypothesis gives us c' and s' satisfying the RHS of the lemma with $do(c', s') \leq_{\mathbf{PbU}(agt)} s$. Since the LHS gives us $\mathit{Legal}(do(c, s))$ we have $\mathit{Poss}(c, s)$, hence $\mathbf{PbU}(agt, c, s)$, hence $do(c', s') \leq_{\mathbf{PbU}(agt)} do(c, s)$. So the c', s' for the predecessor situation s also satisfy the RHS of the lemma for $do(c, s)$. □

Lemma 8 (From [37, 45])**.** *Suppose φ is a regressable formula of the situation calculus and $\mathcal{D}$ is a basic action theory. Then:*

$$\mathcal{D} \models (\forall)(\varphi \equiv \mathcal{R}[\varphi]),$$

where $(\forall)\phi$ denotes the universal closure of the formula ϕ with respect to its free variables.

Proof. This is a foundational result of the situation calculus, proven for example as Theorem 2 in [37] and as part of Theorem 7 in [45]. We repeat it here for completeness. □

Theorem 1. *For any agent agt and situations s and s″:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models K(agt, s'', s) \equiv$$
$$K_0(agt, root(s''), root(s)) \wedge Legal(s'') \wedge View(agt, s'') = View(agt, s)$$

Proof. The *only-if* direction is a trivial combination of Lemmas 3, 4 and 5. For the *if* direction we proceed by induction on situation s.

In the base case of $Init(s)$, we need some s' to satisfy the $\exists s'$ part of equation (9). Using $root(s'')$ is sufficient, since we have $View(agt, s'') = View(agt, s) = \epsilon$ and hence $root(s'') \leq_{\mathbf{PbU}(agt)} s''$ by Lemma 6.

For the inductive case with $do(c, s)$, we have two sub-cases to consider. Suppose $Obs(agt, c, s) = \emptyset$: then
$View(agt, s'') = View(agt, do(c, s)) = View(agt, s)$ and hence $K(agt, s'', s)$ holds by the inductive hypothesis, satisfying the equivalence in equation (8) and giving $K(agt, s'', do(c, s))$ as required.

Alternately, suppose $Obs(agt, c, s) \neq \emptyset$: then we have:

$$View(agt, do(c, s)) = Obs(agt, c, s) \cdot View(agt, s) = View(agt, s'')$$

And we need some s', c' to satisfy the $\exists s', c'$ part of equation (8).

Since s'' is legal, Lemma 7 implies there is some s',c' satisfying $Obs(agt, c', s') = Obs(agt, c, s)$, $View(agt, s') = View(agt, s)$ and $do(c', s') \leq_{\mathbf{PbU}(agt)} s''$. This is enough to satisfy equation (8) and so the equivalence holds as required. □

Theorem 4. *Given a basic action theory $\mathcal{D} \cup \mathcal{D}_K^{obs}$ and a uniform formula ϕ for which $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ exists:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(agt, \phi, s) \equiv \mathcal{R}(\mathbf{Knows}(agt, \phi, s))$$

Proof. To obtain this result, we must establish that our new regression rules in equations (13) and (14) are equivalences under the theory of action $\mathcal{D} \cup \mathcal{D}_K^{obs}$. The proof proceeds by cases on situation s and its mechanics mirror the analogous Theorem 7 in [45], but with the addition of a persistence condition application.

For notational clarity we define the abbreviation $\mathbf{PEO}(agt, \phi, o, s)$ (for "persists under equivalent observations") which states that ϕ holds in all legal futures of s compatible with observations o:

$$\mathbf{PEO}(agt, \phi, o, s) =$$
$$\forall c' : Obs(agt, c', s) = o \wedge Poss(c', s) \rightarrow \left[\forall s' : do(c', s) \leq_{\mathbf{PbU}(agt)} s' \rightarrow \phi[s']\right]$$

First, note that we need only consider two cases: $do(c, s)$ and S_0. We do not provide a regression rule for knowledge at other situation terms, so $\mathcal{R}(\mathbf{Knows}(agt, \phi, s))$ would leave the formula unchanged and the theorem is trivial.

In the $do(c, s)$ case, by definition of the **Knows** macro we have:

$$\mathbf{Knows}(agt, \phi, do(c, s)) \equiv \forall s'' : K(agt, s'', do(c, s)) \rightarrow \phi[s'']$$

Applying the successor state axiom from equation (8), introducing a variable to name $Obs(agt, c, s)$, and distributing $\forall s''$ across the two conjuncts gives us:

$$\begin{aligned}\mathbf{Knows}(agt, \phi, do(c, s)) \equiv \exists o : & \ Obs(agt, c, s) = o \\ & \wedge [o = \emptyset \rightarrow \forall s'' : (K(agt, s'', s) \rightarrow \phi[s''])] \\ & \wedge [o \neq \emptyset \rightarrow \forall s'' : (\exists c', s' : Obs(agt, c', s') = o \wedge Poss(c', s') \\ & \quad \wedge K(agt, s', s) \wedge do(c', s') \leq_{\mathbf{PbU}(agt)} s'') \rightarrow \phi[s'']]\end{aligned}$$

To make the third conjunct match **PEO** we rearrange using:

$$(\exists x : P(x)) \rightarrow Q \quad \Rightarrow \quad \forall x : (P(x) \rightarrow Q)$$

To give:

$$\begin{aligned}\mathbf{Knows}(agt, \phi, do(c, s)) \equiv \exists o : & \ Obs(agt, c, s) = o \\ & \wedge [o = \emptyset \rightarrow \forall s'' : (K(agt, s'', s) \rightarrow \phi[s''])] \\ & \wedge [o \neq \emptyset \rightarrow \forall c', s', s'' : (Obs(agt, c', s') = o \wedge Poss(c', s') \\ & \quad \wedge K(agt, s', s) \wedge do(c', s') \leq_{\mathbf{PbU}(agt)} s'' \rightarrow \phi[s''])]\end{aligned}$$

Pushing quantifiers over independent conjuncts and re-arranging using:

$$(A \wedge B) \rightarrow C \quad \Rightarrow \quad A \rightarrow (B \rightarrow C)$$

We obtain:

$$\begin{aligned}\mathbf{Knows}(agt, \phi, do(c, s)) \equiv \exists o :\ & Obs(agt, c, s) = o \\ & \wedge [o = \emptyset \rightarrow \forall s'' : (K(agt, s'', s)\ \rightarrow \phi[s''])] \\ & \wedge [o \neq \emptyset \rightarrow \forall s' : (K(agt, s', s) \rightarrow \forall c' : Obs(agt, c', s') = o \\ & \quad \wedge Poss(c', s') \rightarrow \forall s'' : do(c', s') \leq_{\mathbf{PbU}(agt)} s''\ \rightarrow \phi[s''])\big]\end{aligned}$$

Which matches the form of our **PEO** macro:

$$\begin{aligned}\mathbf{Knows}(agt, \phi, do(c, s)) \equiv\ \exists o :\ & Obs(agt, c, s) = o \\ & \wedge [o = \emptyset \rightarrow \forall s' : K(agt, s', s) \rightarrow \phi[s']] \\ & \wedge [o \neq \emptyset \rightarrow \forall s' : K(agt, s', s) \rightarrow \mathbf{PEO}(agt, \phi, o, s')]\end{aligned}$$

Since both conjuncts contain sub-formulae matching the form of the **Knows** macro, it can be substituted back in to give:

$$\begin{aligned}\mathbf{Knows}(agt, \phi, do(c, s)) \equiv\ \exists o :\ & Obs(agt, c, s) = o \\ & \wedge\ [o = \emptyset \rightarrow \mathbf{Knows}(agt, \phi, s)] \\ & \wedge\ [o \neq \emptyset \rightarrow \mathbf{Knows}(agt, \mathbf{PEO}(agt, \phi, o), s)]\end{aligned}$$

For $\mathbf{PEO}(agt, \phi, o, s')$ to legitimately appear inside the **Knows** macro it must be uniform in the situation variable s'. Applying the persistence condition and regressing to make the expression uniform, we develop the following equivalence:

$$\begin{aligned}&\mathbf{PEO}(agt, \phi, o, s) \equiv \\ &\qquad \forall c' :\ Obs(agt, c', s) = o \wedge Legal(c', s) \rightarrow \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt)), c')\end{aligned}$$

Suppressing the situation term in this uniform formula gives the regression rule from equation (13) as required.

For the base case of S_0, a straightforward transformation of equations (1) and (9) gives:

$$\mathbf{Knows}(agt, \phi, S_0) \equiv \forall s :\ K_0(agt, s, S_0) \rightarrow \left[\forall s' :\ s \leq_{\mathbf{PbU}(agt)} s' \rightarrow \phi[s']\right]$$

Applying the persistence condition operator, this can easily be re-written as:

$$\mathbf{Knows}(agt, \phi, S_0) \equiv \forall s \ : \ K_0(agt, s, S_0) \rightarrow \mathcal{P}(\phi, \mathbf{PbU}(agt))[s]$$

This matches the form of the definition for $\mathbf{Knows_0}$, which we can substitute in to give:

$$\mathbf{Knows}(agt, \phi, S_0) \equiv \mathbf{Knows_0}(agt, \mathcal{P}(\phi, \mathbf{PbU}(agt)), S_0)$$

Since all situations reachable by K_0 are initial, and since Lemma 8 ensures that regression preserves equivalence with respect to the original query, it is therefore valid to use $\mathcal{R}(\psi[S_0])^{-1}$ on the enclosed formula to give:

$$\mathbf{Knows}(agt, \phi, S_0) \equiv \mathbf{Knows_0}(agt, \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt))[S_0])^{-1}, S_0)$$

This is the regression rule from equation (14) as required.

Our regression rules are thus equivalences under the theory $\mathcal{D} \cup \mathcal{D}_K^{obs}$.

For base cases $Init(s)$ other than S_0 there is no explicit regression rule defined, so it would leave the formula unchanged in this case. The theorem thus holds for all situations s. □

Theorem 5. *Given a basic action theory $\mathcal{D}$ and a uniform formula ϕ for which $\mathcal{P}(\phi, \mathbf{PbU}(agt))$ exists:*

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \ \models \ \mathbf{Knows}(agt, \phi, v) \equiv \mathcal{R}(\mathbf{Knows}(agt, \phi, v))$$

Proof. Recall the definition of $\mathbf{Knows}(agt, \phi, v)$ as follows:

$$\mathbf{Knows}(agt, \phi, v) \stackrel{\text{def}}{=} \\ \forall s : \ \mathit{View}(agt, s) = v \wedge \mathit{root}(s) = S_0 \wedge \mathit{Legal}(s) \rightarrow \mathbf{Knows}(agt, \phi, s)$$

We also have the following simple corollary of Theorem 1:

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \forall s, s', s'' : \ \mathit{root}(s) = \mathit{root}(s') \wedge \mathit{View}(agt, s) = \mathit{View}(agt, s') \\ \wedge K(agt, s'', s) \rightarrow K(agt, s'', s')$$

The definition of $\mathbf{Knows}(agt, \phi, v)$ is thus equivalent to:

$$\mathbf{Knows}(agt, \phi, v) \ \equiv \ \neg\exists s : \ \mathit{View}(agt, s) = v \wedge \mathit{root}(s) = S_0 \wedge \mathit{Legal}(s) \\ \vee \ \exists s : \ \mathit{View}(agt, s) = v \wedge \mathit{root}(s) = S_0 \wedge \mathit{Legal}(s) \wedge \mathbf{Knows}(agt, \phi, s)$$

We thus need to find a single witness situation rather than examining all situations with that view. We proceed by induction over views. For the ϵ case, S_0 serves as an appropriate witness since it is always legal, $\mathit{View}(agt, S_0) = \epsilon$ and $\mathit{root}(S_0) = S_0$. Applying the regression rule for $\mathbf{Knows}(agt, \phi, S_0)$ gives us the same expression as applying the regression rule for $\mathbf{Knows}(agt, \phi, \epsilon)$. So if $\mathcal{R}(\mathbf{Knows}(agt, \phi, \epsilon))$ holds then so does $\mathbf{Knows}(agt, \phi, S_0)$. Using S_0 as a witness we conclude that $\mathbf{Knows}(agt, \phi, \epsilon)$ iff $\mathcal{R}(\mathbf{Knows}(agt, \phi, \epsilon))$ as desired.

For the inductive $o \cdot v$ case we split on whether there is any situation having that view. Suppose there is no such situation, then the definition of $\mathbf{Knows}(agt, \phi, o \cdot v)$ is trivially satisfied and the agent must know all statements. We need to show that the regression of $\mathbf{Knows}(agt, \phi, o \cdot v)$ is always entailed by the domain in this case. The regressed expression is:

$$\mathbf{Knows}(agt, \forall c : \mathit{Obs}(agt, c) = o \wedge \mathit{Poss}(c) \rightarrow \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt)), c), v)$$

If there is no situation having view v, then there is also no situation having view $o \cdot v$, and the above is entailed by the inductive hypothesis in this case.

Alternately, suppose there is a situation s having view v but no legal situation having view $o \cdot v$. Then all situations s' that have view equal to v must satisfy $\neg\exists c : \mathit{Obs}(agt, c, s') = o \wedge \mathit{Poss}(c, s')$, otherwise we could construct a situation with view $o \cdot v$. Since these situations s' are the only ones that can be K-related to s, the antecedent in the above implication is falsified at all such situations, and the regressed expression is equivalent to $\mathbf{Knows}(agt, \top, v)$ which is trivially entailed.

Finally, suppose there is a legal situation $do(c, s)$ having view $o \cdot v$. We can assume without loss of generality that $\mathit{Obs}(agt, c, s) = o$ and $\mathit{View}(agt, s) = v$. Regressing $\mathbf{Knows}(agt, \phi, do(c, s))$ in this case will produce:

$$\begin{aligned}\mathcal{R}(\mathbf{Knows}(agt, \phi, do(c, s))) = \; & \exists o' : \mathit{Obs}(agt, c, s) = o' \\ & \wedge [o' = \emptyset \rightarrow \mathbf{Knows}(agt, \phi, s)] \\ & \wedge [o' \neq \emptyset \rightarrow \mathbf{Knows}(agt, \forall c' : \mathit{Obs}(agt, c') = o' \\ & \quad \wedge \mathit{Poss}(c') \rightarrow \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt)), c'), s)]\end{aligned}$$

We have that $\mathit{Obs}(agt, c, s) = o$ and $o \neq \emptyset$, so this is equivalent to:

$$\mathbf{Knows}(agt, \forall c' : \mathit{Obs}(agt, c') = o \wedge \mathit{Poss}(c') \rightarrow \mathcal{R}(\mathcal{P}(\phi, \mathbf{PbU}(agt)), c'), s)$$

Since this matches the form of $\mathcal{R}(\mathbf{Knows}(agt, \phi, o \cdot v))$, and we have that the view of s is v, this will be entailed by the domain precisely when the regression of $\mathbf{Knows}(agt, \phi, o \cdot v)$ is entailed by the domain.

Thus if there is no legal situation with view v then $\mathcal{R}(\mathbf{Knows}(agt, \phi, v))$, is always entailed, while if there is such a situation s then $\mathcal{R}(\mathbf{Knows}(agt, \phi, v))$ is equivalent to $\mathcal{R}(\mathbf{Knows}(agt, \phi, s))$. The regression rules over observations are thus equivalences as desired. □