Proceedings, Eleventh International Conference on Principles of Knowledge Representation and Reasoning (2008)

Complex Epistemic Modalities in the Situation Calculus

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

Abstract

We develop a formal account of complex group-level epistemic modalities in the situation calculus, with a particular focus on reasoning about common knowledge. Expressions involving common knowledge cannot be handled by standard regression techniques and are thus difficult to reason about effectively. Taking our cue from recent promising work in dynamic epistemic logic, we overcome this limitation by increasing the expressive power of the epistemic language. The syntax of first-order dynamic logic is used to form complex epistemic modalities from the individual-level knowledge operators. Common knowledge is a special case of this syntax, using the unbounded iteration operator. We develop a regression rule for these complex modalities and demonstrate its use to reason about common knowledge in an example domain. The result is a rich multi-agent theory of knowledge and action in which complex group-level epistemic modalities are amenable to effective automated reasoning.

Introduction

In their landmark paper *Knowledge, Action, and the Frame Problem,* Scherl and Levesque (2003) adapt Reiter's solution to the frame problem in the situation calculus to allow reasoning about the knowledge of an agent using regression (Pirri & Reiter 1999). Extensions to concurrent actions (Scherl 2003), multiple agents (Shapiro, Lespérance, & Levesque 2002), and partial observability of actions (Kelly & Pearce 2007a) have produced an expressive logic of knowledge and action that is amenable to standard effective reasoning techniques of the situation calculus.

While powerful, the existing account has a shortcoming that limits its utility in multi-agent domains: it lacks a formal treatment of group-level epistemic modalities. Simple group-level modalities such as "everyone knows ϕ" can be expressed as finite combinations of individual-level knowledge operators, but more complex modalities such as "it is common knowledge that ϕ" cannot be handled in this way.

Common knowledge is typically introduced to the situation calculus as a separate axiom (Davis & Morgenstern 2005; Ghaderi, Levesque, & Lespérance 2007). While logically sound, this approach means regression can no longer be used for effective automated reasoning. Our work offers a new approach that combines a rich group-level epistemic language with a regression rule for effective reasoning.

To achieve this we must overcome a fundamental expressivity limitation, as the regression of common knowledge cannot be expressed in terms of common knowledge alone (Batlag, Moss, & Solecki 1998). We take our cue from recent promising work in the related field of dynamic epistemic logic, with the main idea due to van Bentham, van Eijck and Kooi (2006): increase the expressive power of the epistemic language so it is strong enough to formulate a proper regression rule. They have developed the Logic of Communication and Change (henceforth "LCC") using propositional dynamic logic to express epistemic modalities, and have shown that it allows reasoning about common knowledge using techniques akin to regression. We follow a similar approach in this paper and introduce complex epistemic modalities to the situation calculus.

While our development naturally parallels that of LCC, the richer ontology of the situation calculus also means there are substantial differences. Our formalism captures first-order effects, quantifying-in and de-dicto/de-re, and arbitrary sets of concurrent actions, while providing a regression rule for automated reasoning and integrating smoothly with existing techniques based on the situation calculus.

Following the tradition of (Scherl & Levesque 2003) we develop our formalism as a series of macro-expansions, with no changes to the underlying situation calculus theory. We adopt the language of first-order dynamic logic to construct complex epistemic paths. The macro $\mathbf{PKnows}(\pi, \phi, s)$ expresses knowledge using a path. Common knowledge between A and B can be expressed in this syntax using the iteration operator: $\mathbf{PKnows}((A \cup B)^*, \phi, s)$. Regression is modified to treat $\mathbf{PKnows}(\pi, \phi, do(c, s))$ as a primitive fluent, producing an equivalent formula $\mathbf{PKnows}(\mathcal{T}(\pi), \mathcal{R}(\phi), s)$. The result is a rich theory of knowledge and action in which group-level epistemic modalities are amenable to effective automated reasoning.

The paper proceeds as follows: Sections 2-3 review the necessary background material; Section 4 adopts dynamic logic as an epistemic path language and defines the macro $\mathbf{PKnows}$; Section 5 shows how to regress $\mathbf{PKnows}$ expressions; Section 6 demonstrates this technique with an example; and Sections 7-9 complete the paper with related work, future work and conclusions.

Background: The Situation Calculus

Our work utilises the situation calculus (Pirri & Reiter 1999) with multiple agents (Shapiro, Lespérance, & Levesque 2002) and concurrent actions (Reiter 1996), and we build on the standard account of knowledge due to (Scherl & Levesque 2003). Below is a brief overview.

The situation calculus is a many-sorted language of first-order logic augmented with a second-order induction axiom. It 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 sets of actions that occur simultaneously; SITUATION terms are histories of the actions that have occurred in the world; RESULT terms represent sensing results returned by actions; 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. An agent's *knowledge* is represented using the macro $\mathbf{Knows}$. We call the fluents that are directly affected by actions *primitive* fluents. Although defined as a macro, $\mathbf{Knows}$ is treated syntactically as a primitive fluent.

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

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

We assume a fixed $\mathcal{D}$ throughout the paper. There is a distinguished fluent predicate $Poss(c, s)$ indicating when it is possible to perform actions in a given situation. In simple domains it suffices to specify $Poss(a, s)$ for each individual action type, and define $Poss(c, s)$ by:

$$Poss(c, s) \equiv c \neq \{\} \wedge \forall a : a \in c \rightarrow Poss(a, s)$$

In more complex domains *precondition interaction* may be an issue, but techniques for handling this are well outside the scope of this paper; see (Reiter 1996).

Situations give a branching-time account of the world, with initial situations identified by $Init(s)$ and the function $do(c, s)$ constructing successor situations by performing sets of concurrent actions. The special initial situation S_0 represents the *actual* initial state of the world, while other initial situations model incomplete knowledge of the initial state.

The *uniform formulae* as defined in (Pirri & Reiter 1999) can be thought of as *properties* of the state of the world. They are basically logical combinations of primitive fluents referring to a common situation term. The meta-variable ϕ is used throughout to refer to an arbitrary uniform formula. As it is often useful to evaluate such formulae at several different situations, we will suppress the situation terms in uniform formulae to simplify the presentation. The notation ϕ^{-1} represents a uniform formula with the situation argument removed from all its fluents, while $\phi[s]$ represents a uniform formula with the particular situation s inserted into all its fluents.

The truth of a fluent is completely specified by defining its truth in the initial situation, and collecting the effects of the various actions into *successor state axioms*. Such axioms provide a monotonic solution to the frame problem. They have the following general form, asserting the truth of a fluent F in the successor situation $do(c, s)$ based on the current situation s and the actions c that were performed:

$$F(\bar{x}, do(c, s)) \equiv \Phi(\overrightarrow{x}, c, s)$$

Effective reasoning in the situation calculus depends on the regression meta-operator $\mathcal{R}_{\mathcal{D}}$ (Pirri & Reiter 1999), a syntactic manipulation whose behaviour can be summarised for our purposes as follows: it transforms a formula ϕ uniform in $do(c, s)$ into a formula $\mathcal{R}_{\mathcal{D}}(\phi)$ that is uniform in s and is equivalent to ϕ under the theory of action $\mathcal{D}$:

$$\mathcal{D} \models \phi \equiv \mathcal{R}_{\mathcal{D}}(\phi)$$

Regression operates by replacing primitive fluents $F(\bar{x}, do(c, s))$ with the body of the matching successor state axiom. It also replaces non-primitive fluents such as $Poss$ with their corresponding definitions. We will omit the subscript since $\mathcal{D}$ is constant throughout the paper, and will simplify the presentation of situation-suppressed formulae by using $\mathcal{R}(\phi, c)$ to denote $\mathcal{R}(\phi[do(c, s)])^{-1}$.

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

$$\mathcal{D} \models \phi[do(c_n, do(c_{n-1}, \ldots, do(c_1, S_0)))]$$
$$\text{iff}$$
$$\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \mathcal{R}^*(\phi)[S_0]$$

The axioms $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ are often in very simple forms or can be compiled to make reasoning more efficient. The trade-off is that the length of $\mathcal{R}^*(\phi)$ may be exponential in the length of ϕ. While an efficiency gain is not guaranteed, regression has proven very effective in practice.

Background: Epistemic Reasoning

This section presents epistemic reasoning specifically in the context of the situation calculus; for a comprehensive treatment of the wider field see (Halpern & Moses 1990).

Epistemic reasoning was first introduced to the situation calculus by (Moore 1980), and formalised extensively by (Scherl & Levesque 2003) whose paper is now the canonical reference for these techniques. It has been further extended to include concurrent actions (Scherl 2003), multiple agents (Shapiro, Lespérance, & Levesque 2002) and partial observability of actions (Kelly & Pearce 2007a). It is this family of techniques that we build upon in this paper.

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 knowledge 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 $\mathbf{Knows}$ is then defined as a shorthand for the standard possible-worlds definition of knowledge, stating that an agent knows something when it is true in all situations considered possible:

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

The sensing result function $SR(a, s)$ gives the result returned by the action a when executed in situation s, allowing additional information to be obtained at run-time. Partial observability of actions is accounted for by the function $Obs(agt, c, s)$, which specifies what information an agent perceives when actions are performed. A simple axiomatisation might have agents observe all actions performed, and all sensing results from their own actions:

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

We assume that the domain is synchronised, so agents always know when an action has occurred even if they are uncertain as to its precise details. In other words, we assume that $Obs(agt, c, s)$ is never the empty set. The dynamics of knowledge are specified by a successor state axiom for K:

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

This axiom ensures that s'' is considered a possible alternative to $do(c, s)$ when s'' is the result of doing actions c' in a situation s' that is considered a possible alternative to s, where c' was possible and produced the same observations as c. Thus an agent's knowledge after the occurrence of an action is completely determined by the combination of its knowledge before the action, and the information it observed about the action.

An important result of (Scherl & Levesque 2003) allows the regression operator to treat the $\mathbf{Knows}$ macro syntactically, as if it were a primitive fluent. This makes epistemic queries amenable to the standard reasoning techniques of the situation calculus, by reducing them to epistemic queries about the initial situation only. The regression rule is:

$$\begin{aligned} &\mathcal{R}(\mathbf{Knows}(agt, \phi, do(c, s))) \stackrel{\text{def}}{=} \\ &\quad \exists y : y = Obs(agt, c, s) \wedge \mathbf{Knows}(agt, \\ &\forall c' : \mathcal{R}(Poss(c') \wedge Obs(agt, c') = y) \rightarrow \mathcal{R}(\phi, c'), s) \end{aligned}$$

Note that this definition uses regression to replace $Poss(c')$ and $Obs(agt, c')$ with their definitions, which must be expressible using a finite combination of primitive fluents. The simplest way to ensure this is possible is by having finitely many action types.

Group-level epistemic modalities can now be defined in terms of individual-level knowledge. Let G be a finite group of agents. The basic group-level modality is "everyone knows ϕ", which is defined as:

$$\mathbf{EKnows}(G, \phi) \stackrel{\text{def}}{=} \bigwedge_{agt \in G} \mathbf{Knows}(agt, \phi)$$

Since the definition is finite, $\mathbf{EKnows}$ can be handled in the situation calculus using simple macro-expansion. To assert more complete knowledge by members of the group, one can say "everyone knows that everyone knows ϕ" by nesting $\mathbf{EKnows}$ operators. In general:

$$\mathbf{EKnows}^1(G, \phi) \stackrel{\text{def}}{=} \mathbf{EKnows}(G, \phi)$$

$$\mathbf{EKnows}^n(G, \phi) \stackrel{\text{def}}{=} \mathbf{EKnows}(G, \mathbf{EKnows}^{n-1}(G, \phi))$$

The higher the value of n, the stronger an assertion is made about the knowledge of the group. The strongest group-level modality is "it is common knowledge that ϕ", meaning that everyone knows ϕ, everyone knows that everyone knows ϕ, and so on ad infinitum. Common knowledge is extremely powerful and has deep implications for coordinated group behaviour. For example, the impossibility of the famous "Coordinated Attack" problem rests on an inability to obtain common knowledge (Halpern & Moses 1990). It can be defined variously via an infinite conjunction, a fixpoint or a transitive closure on $\mathbf{EKnows}$, e.g.:

$$\mathbf{CKnows}(G, \phi) \stackrel{\text{def}}{=} \bigwedge_{n \in \mathbb{N}} \mathbf{EKnows}^n(G, \phi)$$

Since its definition is second-order or infinitary, common knowledge cannot be reduced to individual knowledge during reasoning. The only formal treatments of common knowledge in the situation calculus and related literature introduce it via a separate, explicit axiom (Davis & Morgenstern 2005; Ghaderi, Levesque, & Lespérance 2007). This makes reasoning difficult as regression cannot be applied.

The problem boils down to a fundamental expressivity limitation, first discovered in the related field of dynamic epistemic logic:

> Epistemic logic with actions and common knowledge is more expressive than epistemic logic with common knowledge alone (Batlag, Moss, & Solecki 1998)

In our terminology: $\mathcal{R}(\mathbf{CKnows}(G, \phi, do(c, s)))$ cannot be expressed in terms of $\mathbf{CKnows}(G, \varphi, s)$. Given the deep similarities between the situation calculus and modal logic (van Bentham 2007), we can be confident that this limitation also holds in the situation calculus. Rather than trying to confirm it, we proceed directly with a technique for circumventing it. We follow the recent promising work of (van Benthem, van Eijck, & Kooi 2006), who use two important new ideas to produce a regression rule for common knowledge in their logic LCC:

- form more expressive epistemic modalities using the syntax of dynamic logic, interpreted epistemically
- apply regression within the modality as well as to the enclosed formula

We apply these ideas to perform epistemic reasoning in the situation calculus, allowing common knowledge to be handled using regression. While the development naturally parallels that of LCC, the much richer ontology of the situation calculus means there are also substantial differences. A full description of LCC would take us too far afield in this paper.

Complex Epistemic Modalities

In this section, we adopt the language of dynamic logic to express complex epistemic modalities. To deal gracefully with the many first-order aspects of the situation calculus we use a variant of *first-order dynamic logic,* adapted from (Kooi 2007) but with some simplifications.

First, let us specify the syntax of our new epistemic language. We use π to denote an arbitrary *epistemic path*:

Definition 1. *Let agt be an* AGENT *term, ϕ a uniform formula and x a variable name, then the epistemic paths π are the smallest set matching the following structural rules:*

$$\pi ::= agt \mid ?\phi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \mid \exists x$$

The test ($?$), sequence ($;$), choice ($\cup$) and unbounded iteration (*) operators are standard in dynamic logic, although test formulae may now contain variables that must be interpreted. The operator $\exists x$ allows the value of a variable to change by non-deterministically re-binding x to some value.

Let us reiterate the purpose of this path language. Despite using the syntax of dynamic logic, it is *not* related to actions in any way. Rather it expresses paths through the epistemic frame generated by the agents' individual K relations. We will shortly introduce a new macro $\mathbf{PKnows}(\pi, \phi, s)$ (read this as "path-knows") to express knowledge using these epistemic paths. To make this clear, here is how some different kinds of knowledge would be expressed using the standard account of knowledge, and how we intend to express them using epistemic paths:

$$\begin{aligned}
\mathbf{Knows}(A,\phi) &\equiv \mathbf{PKnows}(A,\phi)\\
\mathbf{Knows}(A,\mathbf{Knows}(B,\phi)) &\equiv \mathbf{PKnows}(A;B,\phi)\\
\mathbf{Knows}(A,\phi)\wedge\mathbf{Knows}(B,\phi) &\equiv \mathbf{PKnows}(A\cup B,\phi)\\
\mathbf{EKnows}(G,\phi) &\equiv \mathbf{PKnows}(\bigcup_{a\in G} a,\phi)\\
\mathbf{CKnows}(G,\phi) &\equiv \mathbf{PKnows}((\bigcup_{a\in G} a)^*,\phi) \qquad (1)
\end{aligned}$$

The semantics of epistemic paths are given by the macro $\mathbf{KDo}(\pi, s, s')$, which should be read "s' can be reached from s by following the epistemic path π". It expands to a rather complicated second-order formula in the base language of the situation calculus. What follows is a simplified presentation of the semantics of each operator; a complete definition of $\mathbf{KDo}$ can be found in the appendix.

Formulae of first-order dynamic logic are interpreted relative to both a "current world" and a "current variable binding". Variable bindings are represented by a first-order substitution μ, with $\mu(\phi)$ applying the substitution to the variables in ϕ and $\mu[x/z]$ setting the value of variable x to the term z. The semantics operate over pairs (μ, s).

Definition 2. *A situation s' is reachable from situation s via epistemic path π, denoted $\mathbf{KDo}(\pi, s, s')$, according to the following semantics. All introduced variables are fresh. RTC represents the standard second-order definition of reflexive transitive closure.*

$$\begin{aligned}
\mathbf{KDo}(\pi,s,s') &\equiv \exists\mu,\mu' : \mathbf{KDo}(\pi,\mu,s,\mu',s')\\
\mathbf{KDo}(agt,\mu,s,\mu',s') &\equiv \mu' = \mu \wedge K(agt,s',s)\\
\mathbf{KDo}(?\phi,\mu,s,\mu',s') &\equiv s' = s \wedge \mu' = \mu \wedge \mu(\phi)[s]\\
\mathbf{KDo}(\pi_1;\pi_2,\mu,s,\mu',s') &\equiv \exists\mu'',s'' :\\
&\qquad \mathbf{KDo}(\pi_1,\mu,s,\mu'',s'')\\
&\qquad \wedge\ \mathbf{KDo}(\pi_2,\mu'',s'',\mu',s')\\
\mathbf{KDo}(\pi_1\cup\pi_2,\mu,s,\mu',s') &\equiv \mathbf{KDo}(\pi_1,\mu,s,\mu',s')\\
&\qquad \vee\ \mathbf{KDo}(\pi_2,\mu,s,\mu',s')\\
\mathbf{KDo}(\exists x,\mu,s,\mu',s') &\equiv s' = s \wedge \exists z :\ \mu' = \mu[x/z]\\
\mathbf{KDo}(\pi^*,\mu,s,\mu',s') &\equiv \mathrm{RTC}[\mathbf{KDo}(\pi,\mu,\mathrm{s},\mu',\mathrm{s}')]
\end{aligned}$$

Like its single-agent counterpart $\mathbf{Knows}$, the macro $\mathbf{PKnows}$ is a straightforward encoding of the semantics of the modal box operator:

$$\mathbf{PKnows}(\pi,\phi,s) \stackrel{\text{def}}{=} \forall s' :\ \mathbf{KDo}(\pi,s,s') \ \rightarrow\ \phi[s']$$

An advantage of this macro-expansion approach is that $\mathbf{PKnows}$ expressions can be nested and quantified-into without any ontological difficulties, as it all simply expands to a formula of the base language. The de-dicto/de-re distinction is explicated by placing quantifiers inside/outside the scope of the macro. The situation calculus also settles other semantic issues that might arise in a modal formalism (e.g. rigid vs non-rigid designators).

These definitions provide the required expressiveness boost for our epistemic language. Of course, they should not be used directly for reasoning purposes as they expand to complicated second-order expressions. The next section will demonstrate that regression can treat these macros as if they were primitive fluents, facilitating an effective reasoning procedure for knowledge queries.

We close this section with a formal statement of the validity of our new formalism. The identifies in equation (1) demonstrate that our complex epistemic modalities subsume the modalities used in existing accounts of knowledge in the situation calculus.

Theorem 1. *The identities in equation* (1) *are entailed by any basic action theory $\mathcal{D}$, where CKnows is defined as the transitive closure of EKnows.*

Proof Sketch. Each is a straightforward matter of expanding the $\mathbf{PKnows}$ definition, using the relevant identities from Definition 2, and collecting back together components matching the form of the $\mathbf{Knows}$ macro. See the appendix for more details. □

Epistemic Path Regression

As with the standard **Knows** macro, we do not want to expand **PKnows** macros during reasoning. Instead, we need regression to treat **PKnows** syntactically as a primitive fluent. This can be achieved by regressing both the epistemic path π and the enclosed formula ϕ. Mirroring the notation of LCC, we introduce the meta-operator $\mathcal{T}$ for this purpose and define the following regression rule:

$$\mathcal{R}(\mathbf{PKnows}(\pi, \phi, do(c, s))) \stackrel{\text{def}}{=} \forall c' : \mathbf{PKnows}(\mathcal{T}(\pi, c, c'), \mathcal{R}(\phi, c'), s) \quad (2)$$

Let us consider the operation of the path-regressor $\mathcal{T}$ by analogy with $\mathcal{R}$. One can think of regression as a "pre-encoding" of the effects of an action: ϕ will hold in $do(c, s)$ iff $\mathcal{R}(\phi, c)$ holds in s. The path regressor $\mathcal{T}$ needs to lift this idea to epistemic paths: there is a π-path from $do(c, s)$ to $do(c', s')$ iff there is a $\mathcal{T}(\pi, c, c')$-path from s to s'.

To ensure this, every agt-step from s_1 to s_2 along the regressed path $\mathcal{T}(\pi)$ must correspond to an agt-step from $do(c_1, s_1)$ to $do(c_2, s_2)$ along the original path π. The path regressor uses a fresh variable x to track the action corresponding to the current situation. It adds several tests to encode the successor state axiom for K in the regressed path:

- action x must always be possible in the current situation
- the value of $Obs(agt, x)$ before making an agt-step must match the value of $Obs(agt, x)$ after making the step

Contrast with the regression rule for **Knows**, which encodes this information within the enclosed formula. Note that $\mathcal{R}$ is used to replace $Poss$ and Obs with their definitions, ensuring that the tests are proper uniform formulae. Note also that c and c' must be situation calculus variables rather than path variables, as they are used outside the path expression. $\mathcal{T}$ simply asserts that x is equal to c at the beginning of the path, and equal to c' at the end.

Definition 3. *The epistemic path regressor $\mathcal{T}(\pi, c, c')$ operates according to the definitions below, where x and z are fresh variables not appearing in π :*

$$\mathcal{T}(\pi, c, c') \stackrel{def}{=} \exists x\,;\, ?x = c\,;\, \mathcal{T}_a(\pi, x)\,;\, ?x = c'$$

$$\mathcal{T}_a(agt, x) \stackrel{def}{=} \exists z\,;\, ?\mathcal{R}(Obs(agt, x) = z)\,;\, agt\,;\, \exists x\,;?\mathcal{R}(Poss(x) \wedge Obs(agt, x) = z)$$

$$\mathcal{T}_a(?\phi, x) \stackrel{def}{=} ?\mathcal{R}(\phi, x)$$

$$\mathcal{T}_a(\exists y, x) \stackrel{def}{=} \exists y$$

$$\mathcal{T}_a(\pi_1; \pi_2, x) \stackrel{def}{=} \mathcal{T}_a(\pi_1, x)\,;\, \mathcal{T}_a(\pi_2, x)$$

$$\mathcal{T}_a(\pi_1 \cup \pi_2, x) \stackrel{def}{=} \mathcal{T}_a(\pi_1, x) \cup \mathcal{T}_a(\pi_2, x)$$

$$\mathcal{T}_a(\pi^*, x) \stackrel{def}{=} \mathcal{T}_a(\pi, x)^*$$

The following theorem states that these definitions behave as desired, respecting the semantics of epistemic paths:

Theorem 2. *For any epistemic path π:*

$$\mathcal{D} \models \mathbf{KDo}(\pi, do(c, s), s'') \equiv \exists c', s' :\ s'' = do(c', s') \wedge \mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s')$$

Proof Sketch. The proof proceeds by cases, covering each path operator in turn. The base cases agt, $?\phi$ and $\exists y$ follow from Definition 2 and the successor state axiom for K. The inductive cases are straightforward as $\mathcal{T}_a$ is simply pushed inside each operator. See the appendix for more details. □

With this result in hand, we can justify the use of equation (2) for regressing **PKnows** expressions:

Theorem 3. *For any epistemic path π, uniform formula ϕ and set of actions c:*

$$\mathcal{D} \models \mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall c' : \mathbf{PKnows}(\mathcal{T}(\pi, c, c'), \mathcal{R}(\phi, c'), s)$$

Proof Sketch. The mechanics of this proof mirror the corresponding proof from (Scherl & Levesque 2003): we expand the **PKnows** macro, apply Theorem 2 as a successor state axiom for **KDo**, re-arrange to eliminate existential quantifiers, then collect terms back into forms that match **PKnows**. See the appendix for more details. □

We are now in a position to reduce an epistemic query $\mathbf{PKnows}(\pi, \phi, s)$ at some future situation to an epistemic query $\mathbf{PKnows}(\mathcal{T}^*(\pi), \mathcal{R}^*(\phi), S_0)$ at the initial situation. While this is a significant gain for effective automated reasoning, it still remains to answer the regressed query. As with the work of (Scherl & Levesque 2003), we assume this will be handled by a special-purpose epistemic reasoning engine rather than by expanding the knowledge macros and reasoning directly in the situation calculus.

However, we should note that validity in first-order dynamic logic is undecidable; in fact it is Π^1_1-hard (Kooi 2007). As with previous work in the situation calculus, we must assume that axioms about the initial situation are in a restricted form amenable to effective reasoning. There are several special cases that can simplify answering the regressed query.

A common simplifying assumption is that the potential values of each variable can be finitely enumerated. In this case it is possible to translate our epistemic paths into propositional dynamic logic, which is decidable. The only difficulty is the elimination of variable bindings inside an iteration operator, which can be handled using a Kleene-style technique similar to the $\mathcal{K}$ translator of (van Benthem, van Eijck, & Kooi 2006).

Alternately, it may be that the initial situation is completely known and uncertainty is introduced only due to partial observability of actions. In this case the initial epistemic frame contains the lone situation S_0, and the regressed path can be reduced to a series of tests and variable re-bindings.

As we shall see in the next section, the epistemic paths resulting from regression can often be substantially simplified. In particular, if all actions are public then the variable x introduced by $\mathcal{T}$ can be simplified away as it will always contain the actions that were actually performed.

We are currently investigating further techniques for answering the regressed query in a more effective manner.

Example

In this section we give a small example to demonstrate our technique in action. Two agents, Alice and Bob, are attending a party. They know that its location, the functional fluent loc, is either Cathy's house or Dave's house, and they have just received an invitation telling them it is in fact at Cathy's house. The sensing action $read(agt)$ is publicly observable and returns the location of the party. All of this is common knowledge. This domain can be summarised as follows:

$$loc(S_0) = C$$
$$Poss(read(agt), s) \equiv true$$
$$SR(read(agt), s) = r \equiv r = loc(s)$$
$$\mathbf{PKnows}((A \cup B)^*, loc = C \vee loc = D, S_0)$$

Using the techniques developed in this paper, we can prove several interesting facts about this domain.

Example 1. *After Bob reads the invitation, he knows that the party is at Cathy's house:*

$$\mathcal{D} \models \mathbf{PKnows}(B, loc = C, do(read(B), S_0))$$

This example could be done equivalently using $\mathbf{Knows}$, but it is a good introduction to the mechanics of the regression procedure. To begin, note that loc cannot change and so is invariant under regression. Applying our new regression rule from equation (2), we get:

$$\forall c' : \mathbf{PKnows}(\mathcal{T}(B, read(B), c'), loc = C, S_0)$$

Expanding the definition of $\mathcal{T}$, the new epistemic path is:

$$\mathcal{T}(B, read(B), c') \Rightarrow$$
$$\exists x\,;\, ?x = read(B)\,;\, \exists z\,;\, ?\mathcal{R}(Obs(B, x) = z)\,;\, B\,;$$
$$\exists x\,;\, ?\mathcal{R}(Poss(x) \wedge Obs(B, x) = z)\,;\, ?x = c'$$

Inserting the definitions of $Poss$ and Obs and applying some straightforward simplifications, we get:

$$\mathcal{T}(B, read(B), c') \Rightarrow \exists z\,;\, ?z = read(B)\#loc\,;\, B\,;$$
$$?z = read(B)\#loc\,;\, ?c' = read(B)$$

Examining the test conditions in this path, we see that the value of loc after the B step must equal the value of loc before it. We can in fact simplify this to:

$$\mathcal{T}(B, read(B), c') \Rightarrow$$
$$\exists z\,;\, ?z = loc\,;\, B\,;\, ?z = loc\,;\, ?c' = read(B)$$

We can thus use the following when evaluating $\mathbf{PKnows}$ in the regressed query:

$$\mathbf{KDo}(\mathcal{T}(B, read(B), c'), S_0, s') \rightarrow loc(s') = loc(S_0)$$

Since $loc(S_0) = C$, the query is entailed.

Example 2. *After Bob reads the invitation, it is not common knowledge that the party is at Cathy's house:*

$$\mathcal{D} \not\models \mathbf{PKnows}((A \cup B)^*, loc = C, do(read(B), S_0))$$

Intuitively, since Alice does not observe the sensing results from the $read(B)$ action she should still consider $loc = D$ a possibility. Formally, the query regresses to:

$$\forall c' : \mathbf{PKnows}(\mathcal{T}((A \cup B)^*, read(B), c'), loc = C, S_0)$$

To calculate $\mathcal{T}((A \cup B)^*)$ we need the following result:

$$\mathcal{T}_a(A, read(B)) \Rightarrow$$
$$\exists z\,;\, ?z = read(B)\,;\, A\,;\, ?read(B) = z$$

The tests in this regressed path are *rigid* - they don't contain any fluents and so do not restrict the path in any way. It is thus equivalent to the original path A. This is as expected, since Alice does not observe any sensing results from $read(B)$ and $Poss(read(B))$ is always true, so her knowledge should not change. The regression of the entire epistemic path is then:

$$\mathcal{T}((A \cup B)^*, read(B), c') \Rightarrow$$
$$[A \cup \exists z\,;\, ?z = loc\,;\, B\,;\, ?z = loc]^*\,;\, ?c' = read(B)$$

Since this path permits A steps, and Alice considers $loc = D$ a possibility, the query is not entailed.

Example 3. *After Bob reads the invitation, it is common knowledge that Bob knows where the party is:*

$$\mathcal{D} \models \mathbf{PKnows}((A \cup B)^*, \exists x : \mathbf{PKnows}(B, loc = x), do(read(B), S_0))$$

We begin by regressing the inner expression over the fresh variable c', which produces:

$$\mathcal{R}(\exists x : \mathbf{PKnows}(B, loc = x), c') \Rightarrow$$
$$\exists x : \mathbf{PKnows}(B, loc = x) \vee c' = read(B)$$

So when regressing the outer expression, we get:

$$\forall c' : \mathbf{PKnows}(\mathcal{T}((A \cup B)^*, read(B), c'),$$
$$\exists x : \mathbf{PKnows}(B, loc = x) \vee c' = read(B), S_0)$$

From the previous example, $\mathcal{T}((A \cup B)^*, read(B), c')$ asserts that $c' = read(B)$. This is due to the public observability of the $read$ action. The enclosing $\forall c'$ can thus be simplified away and the result is a simple tautology.

If the $read$ action were not publicly observable, common knowledge would not be obtained; a proof of this would mirror example 2.

Example 4. *If Alice also reads the invitation, it becomes common knowledge that the party is at Cathy's house:*

$$\mathcal{D} \models \mathbf{PKnows}((A \cup B)^*, loc = C, do(read(A), do(read(B), S_0)))$$

Applying regression over the first action, and simplifying with $c' = read(B)$ as before, we reduce this query to:

$$\mathbf{PKnows}([\exists z\,;\, ?z = loc\,;\, A\,;\, ?z = loc \cup B]^*, loc = C, do(read(B), S_0))$$

When we apply regression a second time, the $\exists z$ and test components in this path are not modified. The result is:

$$\mathbf{PKnows}([\exists z\,;\, ?z = loc\,;\, A\,;\, ?z = loc \cup \exists z\,;\, ?z = loc\,;\, B\,;\, ?z = loc]^*, loc = C, do(read(B), S_0))$$

Or equivalently, the simpler form:

$$\mathbf{PKnows}([\exists z\,;\, ?z = loc\,;\, (A \cup B)\,;\, ?z = loc]^*, loc = C, S_0)$$

By a similar argument to the first example, this query is entailed by the domain. Again, the public observability of $read$ is the key to obtaining common knowledge in this case.

Related Work

Existing work using group-level epistemics in the situation calculus has, when common knowledge is included at all, used an explicit second-order axiom to define it and has thus forgone the use of regression for effective reasoning (Davis & Morgenstern 2005; Ghaderi, Levesque, & Lespérance 2007). By combining complex epistemic modalities with an effective reasoning procedure, our work allows reasoning about common knowledge in a wide range of application areas. Work that could immediately benefit from our approach includes: specification of multi-agent systems (Shapiro, Lespérance, & Levesque 2002); theories of coordination and ability (Ghaderi, Levesque, & Lespérance 2007); reasoning about the epistemic feasibility of plans (Lespérance 2001); analysing multi-player games (Bart, Delgrande, & Schulte 2001); and our own work on the cooperative execution of Golog programs (Kelly & Pearce 2006).

This paper clearly owes much to the heritage of dynamic epistemic logics in general (Halpern & Moses 1990; Batlag, Moss, & Solecki 1998) and the development of LCC in particular (van Benthem, van Eijck, & Kooi 2006). We choose the situation calculus for its much richer ontology, e.g. preconditions and effects are first order, while actions take arguments and may be performed concurrently. On one hand, this forces us to use a more powerful dynamic logic for our epistemic language and run the risk of undecidability. On the other, it actually simplifies some aspects of our presentation. We do not need explicit update frames, and the definition of our path regressor does not require an auxiliary Kleene-style operator to handle iteration.

In domains with a finite state-space the situation calculus may not offer a gain in expressiveness (e.g. the party-invitation domain could be formulated equivalently in LCC) but it can certainly provide a more succinct axiomatisation. Moving beyond such domains, our formalism offers the potential to incorporate other rich domain features that have been developed for the situation calculus, such as continuous time and actions with duration (Reiter 1996).

We echo the sentiment of (van Bentham 2007) and look forward to continued cross-pollination between these two related disciplines.

Future Work

The expressiveness of our epistemic language means that answering a regressed knowledge query can be difficult in general, and we are investigating ways to make reasoning easier while retaining first-order features. Starting from the expression for common knowledge, the path regressor $\mathcal{T}$ will generate only a fragment of the full epistemic language, e.g. it will not generate nested iteration operators. Restricting the domain can weaken this generated fragment, e.g. a relativised common-knowledge operator is sufficient for domains in which all actions are public (van Benthem, van Eijck, & Kooi 2006). Identifying restrictions such as this that can simplify reasoning in the epistemic language is a promising avenue for future research.

Our ongoing work extends the formalism to asynchronous domains, where some action occurrences may be completely hidden from an agent. Accounting for arbitrarily-long sequences of hidden actions requires a more complex reasoning procedure that builds on the techniques of (Kelly & Pearce 2007a; 2007b). We are also incorporating additional extensions to the situation calculus, such as continuous time (Reiter 1996). A journal paper presenting the extended version of our formalism is currently in preparation.

Conclusion

In this paper we have enriched the situation calculus with a comprehensive account of complex group-level epistemic modalities. Using first-order dynamic logic to construct complex modalities from the base knowledge operators of each agent, our formalism can capture a wide variety of group-level knowledge expressions, including the important case of common knowledge. It is formulated as a series of macro-expansions and additions to the meta-theoretical reasoning machinery, so it can be used directly to enrich existing action theories and should be compatible with other extensions to the situation calculus. By leveraging the situation calculus, we gain a very rich domain ontology without significant complications. In particular, concurrent actions can be handled in a straightforward manner.

Utilising the increased expressive power of our formalism, we have extended the regression meta-operator to do effective reasoning for epistemic queries. The most immediate benefit of this work, and indeed the impetus for its development, is that reasoning about common knowledge no longer requires a separate axiomatisation. As we have shown with a small example, it can now be handled using regression.

The end result is a rich multi-agent theory of knowledge and action in which complex group-level epistemic modalities are amenable to effective automated reasoning.

Appendix: Encoding Dynamic Logic

This appendix develops an encoding of first-order dynamic logic (henceforth "FODL") into the situation calculus via macro expansion. Unlike the simplified semantics presented in the body of the paper, these macros do not require substitutions to be encoded as terms in the logic. Our encoding operates by embedding FODL into Golog, a programming language built on the situation calculus.

This may seem like an unnecessary complication - why not expand FODL directly into sentences of the base situation calculus? We find the embedding into Golog simpler and clearer, and expect many readers familiar with the situation calculus will feel likewise. As Golog is well-understood, it relieves some our burden of proof that the embedding works as required. Since the semantics of Golog itself is based on macro-expansion, the end result is the the same: sentences of FODL macro-expand into sentences of the situation calculus. Finally, our modification to Golog so that it is interpreted over epistemic frames may further elucidate the intended operation of FODL in this paper.

We make one modification to the semantics of Golog, so that it is interpreted over epistemic frames. The base program component is no longer an action but an agent's epistemic relation. δ represents an arbitrary Golog program. Since we intend to interpret Golog over epistemic frames, we will use the macro $\mathbf{EDo}(\delta, s, s')$ to distinguish it from standard action-based Golog.

Definition 4. *The semantics of Golog over epistemic frames is given by the macro* $\mathbf{EDo}$ *defined as follows, where* P *names a predicate symbol:*

$$\begin{aligned}\delta ::= {} & agt \mid ?\phi \mid \delta_1;\delta_2 \mid \delta_1 \cup \delta_2 \mid \pi(x)\delta(x) \\ & \mid \delta^* \mid \mathbf{proc}\, P(\bar{x})\, \delta(\bar{x})\, \mathbf{end}\,;\, \delta \mid P(\bar{x})\end{aligned}$$

$$\begin{aligned}\mathbf{EDo}(agt, s, s') &\stackrel{def}{=} K(agt, s', s) \\ \mathbf{EDo}(?\phi, s, s') &\stackrel{def}{=} s = s' \wedge \phi[s] \\ \mathbf{EDo}(\delta_1\,;\,\delta_2, s, s') &\stackrel{def}{=} \exists s'' :\, \mathbf{EDo}(\delta_1, s, s'') \\ &\qquad \wedge\, \mathbf{EDo}(\delta_2, s'', s') \\ \mathbf{EDo}(\delta_1 \cup \delta_2, s, s') &\stackrel{def}{=} \mathbf{EDo}(\delta_1, s, s') \\ &\qquad \vee\, \mathbf{EDo}(\delta_2, s, s') \\ \mathbf{EDo}(\pi(x)\delta(x), s, s') &\stackrel{def}{=} \exists x :\, \mathbf{EDo}(\delta(x), s, s') \\ \mathbf{EDo}(\delta^*, s, s') &\stackrel{def}{=} \mathrm{RTC}[\mathbf{EDo}(\delta, s, s')] \\ \mathbf{EDo}(P(\bar{x}), s, s') &\stackrel{def}{=} P(\bar{x}, s, s')\end{aligned}$$

The final clause identifies a procedure call with arguments $\bar{x}$. Implementing procedure calls via macro expansion requires a second-order formula encoding the standard least-fixed-point semantics for recursive procedures. We will not repeat the definition here, but note that the invocation of a Golog program using procedure definitions appears as:

$$\{\mathbf{proc}\, P_1(\bar{x})\, \delta_1(\bar{x})\, \mathbf{end}\,; \ldots ;\, \mathbf{proc}\, P_n(\bar{x})\, \delta_n(\bar{x})\, \mathbf{end}\,;\, \delta\}$$

Clearly Golog is a very powerful language, so the question must be asked: could we use Golog as our epistemic path language, rather than FODL? Unfortunately not, as Golog has no notion of *state* - while the Golog operator $\pi(x)\delta(x)$ is similar to the FODL operator $\exists x$, its effect is localised to the contained program $\delta(x)$. FODL allows variable assignments to affect the entire remaining program. In order to simulate this via macro expansion, we use what is essentially a continuation-passing transformation to pass state among the operators.

First, note that any epistemic path π will contain only a finite number of FODL variables. Without loss of generality, suppose that π contains n such variables named x_1 to x_n. The idea is to translate each component of the path into a Golog procedure with n arguments, where the ith argument is used to pass in the current value of x_i. After performing the necessary operations to encode the semantics of that path component, it calls a continuation procedure representing the next path component. This translation is based on the macro $\mathbf{KDo}_c(\pi, N, C)$ which is passed the name to use for the procedure encoding the given path component (N) and the name of the continuation procedure (C).

Definition 5. *The embedding of FODL into Golog is given by the macros* $\mathbf{KDo}$ *and* $\mathbf{KDo}_c$ *defined as follows, where* P_i *are fresh procedure names and* $\bar{v}$ *are argument vectors of length* n*:*

$$\begin{aligned}\mathbf{KDo}(\pi, s, s') &\stackrel{def}{=} \mathbf{EDo}(\{\mathbf{KDo}_c(\pi, P, End) \\ &\qquad ;\, \mathbf{proc}\, End(\bar{v})\, ?\top\, \mathbf{end}\,;\, \pi(\bar{v})P(\bar{v})\}, s, s') \\ \mathbf{KDo}_c(agt, N, C) &\stackrel{def}{=} \mathbf{proc}\, N(\bar{v})\, agt\,;\, C(\bar{v})\, \mathbf{end} \\ \mathbf{KDo}_c(?\phi(\bar{x}), N, C) &\stackrel{def}{=} \mathbf{proc}\, N(\bar{v})\, ?\phi(\bar{v})\,;\, C(\bar{v})\ \mathbf{end} \\ \mathbf{KDo}_c(\exists x_i, N, C) &\stackrel{def}{=} \mathbf{proc}\, N(\bar{v})\, \pi(x) C(\bar{v}[v_i/x])\, \mathbf{end} \\ \mathbf{KDo}_c(\pi_1;\pi_2, N, C) &\stackrel{def}{=} \mathbf{KDo}_c(\pi_1, N, P)\,; \\ &\qquad \mathbf{KDo}_c(\pi_2, P, C) \\ \mathbf{KDo}_c(\pi_1 \cup \pi_2, N, C) &\stackrel{def}{=} \mathbf{KDo}_c(\pi_1, P_1, C)\,; \\ &\qquad \mathbf{KDo}_c(\pi_2, P_2, C)\,; \\ &\qquad \mathbf{proc}\, N(\bar{v})\, P_1(\bar{v}) \cup P_2(\bar{v})\, \mathbf{end} \\ \mathbf{KDo}_c(\pi^*, N, C) &\stackrel{def}{=} \mathbf{KDo}_c(\pi, P, N)\,; \\ &\qquad \mathbf{proc}\, N(\bar{v})\, C(\bar{v}) \cup P(\bar{v})\, \mathbf{end}\end{aligned}$$

This translation generates one procedure for each operator in the path, plus the procedure End used to successfully terminate execution. Most of these definitions are straightforward translations of equivalent operators in FODL to Golog. One interesting case is $\exists x_i$, which calls the continuation procedure with a fresh variable in position i.

The most complex case is π^*, which simulates iteration using a pair of mutually recursive procedures P and N. Procedure N can either terminate immediately (calling the continuation C) or call P. A call to P executes one iteration of π before continuing with another invocation of N. The possible executions for π^* are thus nil, π, $\pi;\pi$, etc as required by the semantics of FODL.

It should be clear that the expansion of each operator satisfies the relevant identity from Definition 2.

Appendix: Extended Proofs

Theorem 1. *The identities in equation* (1) *are entailed by any basic action theory $\mathcal{D}$, where CKnows is defined as the transitive closure of EKnows.*

Proof. Each is a straightforward matter of expanding the $\mathbf{PKnows}$ definition, using the relevant identities from Definition 2, and collecting back together components matching the form of the $\mathbf{Knows}$ macro. We take the $A \cup B$ case as an example:

$$\mathbf{PKnows}(A \cup B, \phi, s) \Rightarrow \\ \forall s' : \mathbf{KDo}(A \cup B, s, s') \rightarrow \phi[s']$$

Using the semantics of the $\cup$ operator:

$$\mathbf{PKnows}(A \cup B, \phi, s) \Rightarrow \\ \forall s' : [\mathbf{KDo}(A, s, s') \vee \mathbf{KDo}(B, s, s')] \rightarrow \phi[s']$$

Using the semantics of the agt operator, this expands to:

$$\mathbf{PKnows}(A \cup B, \phi, s) \Rightarrow \\ \forall s' : K(A, s, s') \rightarrow \phi[s'] \wedge K(B, s, s') \rightarrow \phi[s']$$

Which matches the form of $\mathbf{Knows}$, giving the required:

$$\mathbf{PKnows}(A \cup B, \phi, s) \Rightarrow \\ \mathbf{Knows}(A, \phi, s) \wedge \mathbf{Knows}(B, \phi, s)$$

□

Lemma 1. *For any epistemic path π:*

$$\mathcal{D} \models \mathbf{KDo}(\pi, do(c, s), s'') \equiv \\ \exists \mu, \mu', c', s' : \mu(x) = c \wedge \mu'(x) = c' \wedge \\ s'' = do(c', s') \wedge \mathbf{KDo}(\mathcal{T}_a(\pi, x), \mu, s, \mu,' s')$$

Proof. Proceed by cases, covering each path operator in turn. We will omit the use of $\mathcal{R}()$ in the definition of $\mathcal{T}_a$ to simplify the presentation. For the base case of an individual agent, we have:

$$\mathbf{KDo}(\pi, do(c, s), s'') \equiv K(agt, s'', do(c, s))$$

$$\mathcal{T}_a(agt, x) \Rightarrow \exists z\,;\, ?Obs(agt, x) = z\,;\, agt\,; \\ \exists x\,;\, ?Poss(x) \wedge Obs(agt, x) = z$$

Expanding $\mathbf{KDo}(\mathcal{T}_a(agt, x), \mu, s, \mu', s')$ thus produces:

$$\mathbf{KDo}(\mathcal{T}_a(agt, x), \mu, s, \mu', s') \equiv \\ \exists z : Obs(agt, \mu(x), s) = z \wedge \exists s'' : K(agt, s'', s) \wedge \\ Poss(\mu'(x), s'') \wedge Obs(agt, \mu'(x), s'') = z \wedge s'' = s'$$

Note that μ and μ' are never applied to a variable other than x. When we substitute this into the RHS of the hypothesis, $\mu(x)$ and $\mu'(x)$ are asserted to be c and c' respectively, so they can be simplified away to give:

$$\mathcal{D} \models K(agt, s'', do(c, s)) \equiv \exists c', s' : s'' = do(c', s') \\ \wedge K(agt, s, s') \wedge Poss(c', s') \\ \wedge Obs(agt, c, s) = Obs(agt, c', s')$$

This is the successor state axiom for K, which is trivially entailed by the domain.

For the $?\phi$ case, we have:

$$\mathbf{KDo}(?\phi, do(c, s), s'') \equiv \phi[do(c, s)] \wedge s'' = do(c, s) \\ \mathcal{T}_a(?\phi, x) \Rightarrow ?\mathcal{R}(\phi, x)$$

Giving:

$$\mathbf{KDo}(\mathcal{T}_a(?\phi, x), \mu, s, \mu', s') \equiv \\ \mathcal{R}(\phi, x)[s] \wedge s = s' \wedge \mu = \mu'$$

Substituting into the RHS of the hypothesis, this asserts that $c = c'$ and hence $s'' = do(c, s)$, so the hypothesis is clearly entailed.

The case for $\exists y$ is trivial as $\mathbf{KDo}(\exists y, s, s') \equiv s = s'$.

The inductive cases are straightforward as $\mathcal{T}_a$ is simply pushed inside each operator. We will take the π^* case as an example. The inductive hypothesis gives us:

$$\mathbf{KDo}(\pi, do(c, s), s'') \equiv \\ \exists \mu, \mu', c', s' : \mu(x) = c \wedge \mu'(x) = c' \wedge \\ s'' = do(c', s') \wedge \mathbf{KDo}(\mathcal{T}_a(\pi, x), \mu, s, \mu,' s')$$

We can apply RTC to both sides of this equivalence, along with two rearrangements: the LHS is expanded to put $\exists \mu, \mu''$ at its front, and the rigid tests on the RHS are taken outside the RTC operation. The result is:

$$\exists \mu, \mu'' : RTC[\mathbf{KDo}(\pi, \mu, do(c, s), \mu'', s'')] \equiv \\ \exists \mu, \mu', c', s' : \mu(x) = c \wedge \mu'(x) = c' \wedge \\ s'' = do(c', s') \wedge RTC[\mathbf{KDo}(\mathcal{T}_a(\pi, x), \mu, s, \mu,' s')]$$

Using the definitions of $\mathbf{KDo}$ and $\mathcal{T}_a$ we have:

$$\mathbf{KDo}(\pi^*, do(c, s), s'') \equiv \\ \exists \mu, \mu'' : RTC[\mathbf{KDo}(\pi, \mu, do(c, s), \mu'', s'')]$$

$$\mathbf{KDo}(\mathcal{T}_a(\pi^*, x), \mu, s, \mu', s') \equiv \\ RTC[\mathbf{KDo}(\mathcal{T}_a(\pi, x), \mu, s, \mu', s')]$$

Substituting these into the RTC of the inductive hypothesis completes the proof. □

Theorem 2. *For any epistemic path π:*

$$\mathcal{D} \models \mathbf{KDo}(\pi, do(c, s), s'') \equiv \\ \exists c', s' : s'' = do(c', s') \wedge \mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s')$$

Proof. Recall the rule for $\mathcal{T}(\pi, c, c')$:

$$\mathcal{T}(\pi, c, c') \Rightarrow \exists x\,;\, ?x = c\,;\, \mathcal{T}_a(\pi, x)\,;\, ?x = c'$$

Expanding $\mathbf{KDo}$ for this rule:

$$\mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s') \equiv \exists \mu, \mu' : \mu(x) = c \wedge \\ \mu'(x) = c' \wedge \mathbf{KDo}(\mathcal{T}_a(\pi, x), \mu, s, \mu', s')$$

We can trivially substitute $\mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s')$ into the RHS of Lemma 1 to get the required result. □

Theorem 3. *For any epistemic path π, uniform formula ϕ and actions c:*

$$\mathcal{D} \models \mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall c' : \mathbf{PKnows}(\mathcal{T}(\pi, c, c'), \mathcal{R}(\phi, c'), s)$$

Proof. The mechanics of this proof mirror the corresponding proof from (Scherl & Levesque 2003). First we expand the **PKnows** macro to produce:

$$\mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall s'' : \mathbf{KDo}(\pi, do(c, s), s'') \rightarrow \phi[s'']$$

Using Theorem 2 as a kind of pseudo-successor-state-axiom for **KDo**, we may write:

$$\mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall s'' : (\exists c', s' : s'' = do(c', s') \wedge \mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s')) \rightarrow \phi[s'']$$

Bringing the existential variables outside of the implication produces:

$$\mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall s'', c', s' : s'' = do(c', s') \wedge \mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s') \rightarrow \phi[s'']$$

The variable s'' is now doing no work in this formula, so it can be removed:

$$\mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall c', s' : \mathbf{KDo}(\mathcal{T}(\pi, c, c'), s, s') \rightarrow \phi[do(c', s')]$$

The use of variable s' now matches the form of the **PKnows** macro. Capturing it, and using regression on the ϕ term , we have the hypothesis as required:

$$\mathbf{PKnows}(\pi, \phi, do(c, s)) \equiv \forall c' : \mathbf{PKnows}(\mathcal{T}(\pi, c, c'), \mathcal{R}(\phi, c'), s)$$

□

References

Bart, B.; Delgrande, J. P.; and Schulte, O. 2001. Knowledge and planning in an action-based multi-agent framework: A case study. In *Advances in Artificial Intelligence*, volume 2056 of *LNAI*, 121–130. Springer.

Batlag, A.; Moss, L. S.; and Solecki, S. 1998. The logic of public announcements and common knowledge and private suspicions. In *Proc. of TARK'98*, 43–56.

Davis, E., and Morgenstern, L. 2005. A first-order theory of communication and multi-agent plans. *Journal of Logic and Computation* 15(5):701–749.

Ghaderi, H.; Levesque, H.; and Lespérance, Y. 2007. A logical theory of coordination and joint ability. In *Proc. AAAI'07*, 421–426.

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

Kelly, R. F., and Pearce, A. R. 2006. Towards high-level programming for distributed problem solving. In *Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 490–497.

Kelly, R. F., and Pearce, A. R. 2007a. Knowledge and observations in the situation calculus. In *Proc. AAMAS '07*, 841–843.

Kelly, R. F., and Pearce, A. R. 2007b. Property persistence in the situation calculus. In *Proc. IJCAI'07*, 1948–1953.

Kooi, B. 2007. Dynamic term-modal logic. In *Proc. Workshop on Logic, Rationality and Interaction*, volume 8 of *Texts in Computing*, 173–185. College Publ., London.

Lespérance, Y. 2001. On the epistemic feasibility of plans in multiagent systems specifications. In *Proc. 8th International Workshop on Agent Theories, Architectures, and Languages*, volume 2333 of *LNAI*, 69–85.

Moore, R. C. 1980. Reasoning about knowledge and action. Technical Note 191, SRI International.

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

Reiter, R. 1996. Natural actions, concurrency and continuous time in the situation calculus. In *Proc. KR'96*. Morgan Kaufmann. 2–13.

Scherl, R., and Levesque, H. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144:1–39.

Scherl, R. B. 2003. Reasoning about the interaction of knowledge, time and concurrent actions in the situation calculus. In *Proc. IJCAI'03*, 1091–1098.

Shapiro, S.; Lespérance, Y.; and Levesque, H. J. 2002. The cognitive agents specification language and verification environment for multiagent systems. In *Proc. AAMAS '02*, 19–26.

van Bentham, J. 2007. Situation calculus meets modal logic. Technical report, University of Amsterdam.

van Benthem, J.; van Eijck, J.; and Kooi, B. 2006. Logics of communication and change. *Information and Computation* 204(II):1620–1662.