Knowledge and Observation in the Situation Calculus

Ryan Kelly    Adrian Pearce

September 1, 2006
Outline

Motivation

Knowledge

Knowledge & Action

Observations

Reasoning

Conclusions
Motivation

What do we want?

- A powerful account of multi-agent knowledge
  - Concurrent Actions
  - Partial Observability
- that supports complex symbolic reasoning
- that allows agents to reason about their own world
- but is still computationally feasible
Motivation

What do we want?

- A powerful account of multi-agent knowledge
  - Concurrent Actions
  - Partial Observability
- that supports complex symbolic reasoning
- that allows agents to reason about their own world
- but is still computationally feasible

Our long-term goal: the cooperative execution of a shared Golog program by a distributed team of agents.
Knowledge is a modal operator

- Typically represented as: **Knows**(\(\phi\)) or **Knows**(Ryan, \(\phi\))
- \(\neg Knows(\phi) \not\rightarrow Knows(\neg \phi)\)

Knowledge is ”true belief”:

\[ Knows(\phi) \rightarrow \phi \]

Thus, it makes sense to approach knowledge using modal logic. Knowledge is given semantics in terms of ”possible worlds”.

Knowledge and Observation in the Situation Calculus
Possible Worlds Semantics

If an agent is unsure about the state of the world, there must be several different states of the world that it considers possible.

The agent knows $\phi$ iff $\phi$ is true in all possible worlds.
"Possible worlds" works well for reasoning about a static knowledge base, but what if the world itself is changing?

- How should the worlds considered possible change in response?
- What of all the associated reasoning-about-actions problems? (frame, quantification, ramification, ...)

Need to integrate with a powerful *theory of action*. 
Knowledge and Action

"Possible worlds" works well for reasoning about a static knowledge base, but what if the world itself is changing?

▶ How should the worlds considered possible change in response?

▶ What of all the associated reasoning-about-actions problems? (frame, quantification, ramification, ...)

Need to integrate with a powerful *theory of action*.

I happen to know just the thing...
Knowledge in the Situation Calculus

Why the situation calculus?

- Elegant monotonic solution to the frame problem
- Effective reasoning procedure
- "situations" provide a direct analog to "possible worlds"

Approach due (mainly) to Scherl and Levesque:
"Knowledge, Action and the Frame Problem", AI, 2003
Possible Situations

Recall that all statements about the world are relative to a situation. So, we must talk about the knowledge of an agent in a particular situation.

\[ \text{Knows}(\phi, s) \quad \text{or} \quad \text{Knows}(\text{agt}, \phi, s) \]
Possible Situations

Recall that all statements about the world are relative to a situation. So, we must talk about the knowledge of an agent in a particular situation.

\[ \text{Knows}(\phi, s) \text{ or } \text{Knows}(\text{agt}, \phi, s) \]

Replace "possible worlds" with "possible situations". Introduce a fluent \( K(s', s) \) meaning that "in situation \( s \), the agent considers it possible that the world may be in situation \( s' \)".
We can then define knowledge as a simple macro:

\[ \text{Knows}(\phi, s) \overset{\text{def}}{=} \forall s' \left[ K(s', s) \rightarrow \phi(s') \right] \]
The Frame Problem

To specify what the agent knows initially, restrict the properties of situations K-related to $S_0$:

$$\textbf{Knows}(\neg \text{Holding}(\text{Sandwich}, S_0)) \Rightarrow \forall s \ [K(s, S_0) \rightarrow \neg \text{Holding}(\text{Sandwich}, s)]$$
The Frame Problem

To specify what the agent knows initially, restrict the properties of situations $K$-related to $S_0$:

\[
\text{Knows}(\neg \text{Holding}(\text{Sandwich}, S_0)) \\
\Rightarrow \forall s [K(s, S_0) \rightarrow \neg \text{Holding}(\text{Sandwich}, s)]
\]

Since $K$ is just a fluent, we specify how it changes between situations with a successor state axiom:

\[
K(s'', \text{do}(a, s)) \equiv \exists s'. \ s'' = \text{do}(a, s') \land K(s', s) \land \text{Poss}(a, s')
\]

Automatically inherits the solution to the frame problem!
The Frame Solution

Knowledge and Observation in the Situation Calculus
Sensing

Agents can acquire additional information about their environment by performing "sensing actions" which return a result. The relationships between actions and their sensing results are captured by a new predicate $SR$:

$$SR(isNiceDay, s) = "YES" \equiv \neg\text{Raining}(s) \land \neg\text{Windy}(s)$$

These are easily incorporated into the successor state axiom:

$$K(s'', do(a, s)) \equiv \exists s'. s'' = do(a, s')$$
$$\land K(s', s) \land \text{Poss}(a, s') \land SR(a, s) = SR(a, s')$$
Effective Reasoning

The standard tool for effective reasoning in the S.C. is the regression operator $\mathcal{R}$. If fluent $F$ has the successor state axiom $F(do(a, s)) \equiv \Phi(a, s)$, then:

$$\mathcal{R}[F(do(a, s))] = \Phi(a, s)$$
Effective Reasoning

The standard tool for effective reasoning in the S.C. is the regression operator $\mathcal{R}$. If fluent $F$ has the successor state axiom $F(do(a,s)) \equiv \Phi(a,s)$, then:

$$\mathcal{R}[F(do(a,s))] = \Phi(a,s)$$

In general:

$$\mathcal{D} \models \phi(do(a,s)) \equiv \mathcal{R}[\phi(do(a,s))]$$

But the regressed term refers to $s$ rather than $do(a,s)$. 
Regression

By repeatedly applying \( R \), we reach a fluent expression referring only to \( S_0 \). This can be decided without many of the axioms in \( D \):

\[
\Sigma \cup D_{S_0} \cup D_{Poss} \cup D_{ssa} \cup D_{una} \models \phi(s)
\]

iff

\[
D_{S_0} \cup D_{una} \models R^*[\phi(s)](S_0)
\]

This is usually easier to answer, since regression has already done some of the reasoning for us.
Regressing Knowledge

Unfortunately, $\mathcal{R}$ cannot be applied to formulae that quantify over situations. We must therefore special-case $\mathcal{R}[\text{Knows}(\phi, s)]$.

Based on the successor state axioms for $K$, Scherl and Levesque develop the following regression rule:

\[
\mathcal{R}[\text{Knows}(\phi, do(a, s))] = \exists y. SR(a, s) = y \land \\
\text{Knows}(\text{Poss}(a) \land SR(a) = y) \rightarrow \mathcal{R}[\phi(do(a, s))], s)
\]

We can thus transform a query about knowledge in an arbitrary situation to a query about knowledge in the initial situation, which can (as before) be tackled more effectively.
Shortcomings

This is a very powerful framework...but:

▶ Does not handle concurrent actions
▶ Agents are assumed to be aware of *all* actions that occur
▶ Requires omniscient viewpoint
Shortcomings

This is a very powerful framework...but:

- Does not handle concurrent actions
- Agents are assumed to be aware of all actions that occur
- Requires omniscient viewpoint

In short, it’s not suitable for complex multi-agent domains.

Can we do better, while maintaining the theoretical and practical elegance of this approach?
Concurrent Actions

Basically, replace actions with sets of actions that all occur at the same instant:

$$\text{do}\{\text{eat}(John, Sandwich), \text{drink}(Ryan, Water)\}, s$$
Concurrent Actions

Basically, replace actions with sets of actions that all occur at the same instant:

\[
do(\{eat(John, Sandwich), drink(Ryan, Water)\}, s)
\]

Different agents may be aware of different subsets of the actions that have occurred.

When will an agent observe the occurrence of an action?
Observability

Well, let’s axiomatize it!

\[
\text{CanObs}(\text{agt1}, \text{eat}(\text{agt2}, \text{obj}), s) \equiv \\text{agt1} = \text{agt2} \lor \text{facing}(\text{agt1}, \text{agt2}, s)
\]

\text{CanObs} handles action observability in the same way that \text{Poss} handles action possibility.
Observability

Well, let’s axiomatize it!

\[ \text{CanObs}(\text{agt1}, \text{eat}(\text{agt2}, \text{obj}), s) \equiv \text{agt1} = \text{agt2} \lor \text{facing}(\text{agt1}, \text{agt2}, s) \]

\text{CanObs} handles action observability in the same way that \text{Poss} handles action possibility.

Could also ask, when will an agent observe the sensing outcome from an action?

\[ \text{CanSense}(\text{agt1}, \text{getTrainTimes}(\text{agt2}, \text{obj}), s) \equiv \text{withinEarshot}(\text{agt1}, \text{agt2}, s) \]
Observations

We can now formalise the Observations made by each agent when some set of concurrent actions is performed. It is a set of (action,result) pairs such that:

\[
Observations(agt, c, s) = o \equiv \forall a, r [ \langle a, r \rangle \in o \equiv \\
a \in c \land CanObs(agt, a, s) \land \\
(CanSense(agt, r) \rightarrow r = SR(a, s)) \land \\
(\neg CanSense(agt, a, s) \rightarrow r = "?" ) ]
\]
For Example

\[ \text{Obs}(Ryan, \{\text{eat}(John, \text{Sandwich}), \text{drink}(Ryan, \text{Water})\}, s) = \{< \text{eat}(John, \text{Sandwich}), "OK" >, < \text{drink}(Ryan, \text{Water}), "OK" >\} \]
For Example

\[ Obs(Ryan, \{\text{eat}(John, Sandwich), \text{drink}(Ryan, Water)\} , s) = \{< \text{eat}(John, Sandwich), "OK" >, < \text{drink}(Ryan, Water), "OK" >\} \]

\[ Obs(John, \{\text{eat}(John, Sandwich), \text{drink}(Ryan, Water)\} , s) = \{< \text{eat}(John, Sandwich), "OK" >\} \]
Observation Histories

We can also track the observations made by an agent as the world has evolved, by making a list of the observations made in each situation:

\[
\text{ObsHist}(agt, \epsilon, S_0)
\]

\[
\text{ObsHist}(agt, h, do(c, s)) \equiv
\exists o. \text{Observations}(agt, c, s) = o \land
\begin{align*}
o = \{\} & \rightarrow \text{ObsHist}(agt, h, s) \land \\
o \neq \{\} & \rightarrow \exists h'. h = o \cdot h' \land \text{ObsHist}(agt, h', s)
\end{align*}
\]
Observation Histories

Situations are a history of all actions that have happened in the world.

Observation histories track the subset of those actions that were observed by an agent.
Knowledge follows Observation

Halpern & Moses, 1990:
"an agent’s knowledge at a given time must depend only on its local history: the information that it started out with combined with the events it has observed since then"
Knowledge follows Observation

Halpern & Moses, 1990:
"an agent’s knowledge at a given time must depend only on its local history: the information that it started out with combined with the events it has observed since then"

Clearly, we require:

\[ \text{Knows}(agt, \phi, s) \equiv \exists h. \text{ObsHist}(agt, h, s) \land \forall s'. [\text{ObsHist}(agt, h, s') \rightarrow \phi[s']] \]
Knowledge follows Observation

Halpern & Moses, 1990:
"an agent’s knowledge at a given time must depend only on its local history: the information that it started out with combined with the events it has observed since then"

Clearly, we require:

\[ \text{Knows}(\text{agt}, \phi, s) \equiv \exists h. \text{ObsHist}(\text{agt}, h, s) \wedge \forall s'. [\text{ObsHist}(\text{agt}, h, s') \rightarrow \phi[s']] \]

\[ K(\text{agt}, s', s) \equiv \exists h. \text{ObsHist}(\text{agt}, h, s) \wedge \text{ObsHist}(\text{agt}, h, s') \]
K Graphically
SSA for Knowledge

First, let’s define:

\[ \text{Unobs}(\text{agt}, s, s'') \overset{\text{def}}{=} \forall c', s. s < \text{do}(c', s') \leq s'' \rightarrow \text{Observations}(\text{agt}, c', s') = \{} \]

Ryan Kelly (rfk@csse.unimelb.edu.au)
Knowledge and Observation in the Situation Calculus
SSA for Knowledge

First, let’s define:

\[ \text{Unobs}(agt, s, s'') \overset{\text{def}}{=} \forall c', s'. s < do(c', s') \leq s'' \rightarrow \text{Observations}(agt, c', s') = {} \]

Then the following successor state axioms captures the intended dynamics of knowledge update:

\[
K(agt, s'', do(c, s)) \equiv \text{Observations}(agt, c, s) = {} \land K(agt, s'', s') \lor \exists o, c', s'. \text{Observations}(agt, c, s) = o \land o \neq {} \land \text{Legal}(s'') \\
\land \text{Observations}(agt, c', s') = o \land K(agt, s', s') \land \text{Unobs}(agt, do(c', s'), s'')
\]
SSA Comparison

We’ve gone from this:

\[ K(s'', \text{do}(a, s)) \equiv \exists s'. \ s'' = \text{do}(a, s') \]
\[ \land K(s', s) \land \text{Poss}(a, s') \land \text{SR}(a, s) = \text{SR}(a, s') \]
SSA Comparison

We’ve gone from this:

\[
K(s'', do(a, s)) \equiv \exists s'. s'' = do(a, s') \\
\quad \land K(s', s) \land \text{Poss}(a, s') \land SR(a, s) = SR(a, s')
\]

To this:

\[
K(agt, s'', do(c, s)) \equiv \text{Observations}(agt, c, s) = \{\} \land K(agt, s'', s) \\
\quad \lor \exists o, c', s'. \text{Observations}(agt, c, s) = o \land o \neq \{\} \land \text{Legal}(s'') \\
\quad \land \text{Observations}(agt, c', s') = o \land K(agt, s', s) \land \text{Unobs}(agt, do(c', s'), s'')
\]
SSA Comparison

We’ve gone from this:

\[
K(s'', \text{do}(a, s)) \equiv \exists s'. \ s'' = \text{do}(a, s') \\
\land K(s', s) \land \text{Poss}(a, s') \land SR(a, s) = SR(a, s')
\]

To this:

\[
K(\text{agt}, s'', \text{do}(c, s)) \equiv Observations(\text{agt}, c, s) = \{\} \land K(\text{agt}, s'', s) \\
\lor \exists o, c', s'. Observations(\text{agt}, c, s) = o \land o \neq \{\} \land \text{Legal}(s'') \\
\land Observations(\text{agt}, c', s') = o \land K(\text{agt}, s', s) \land \textbf{Unobs}(\text{agt}, \text{do}(c', s'), s'')
\]

It’s messier, but it’s also hiding a much bigger problem...
Regression

Recall that $\text{Unobs}$ is defined by quantifying over situations. So regression cannot be applied to the RHS of our successor state axiom.

Observation-based knowledge cannot be approached using the standard regression operator.
Regression

Recall that Unobs is defined by quantifying over situations. So regression cannot be applied to the RHS of our successor state axiom.

Observation-based knowledge cannot be approached using the standard regression operator.

In fact, universal quantification over situations requires a second-order induction axiom. Must we abandon hope of an effective reasoning procedure?
Property Persistence

Suppose we could "factor out" the quantification. Then we could get on with the business of doing regression.

Define the persistence condition $\mathcal{P}[\phi, \alpha]$ of a formula $\phi$ and action conditions $\alpha$ to mean: assuming all future actions satisfy $\alpha$, $\phi$ will remain true.

$$\mathcal{P}[\alpha, \phi](s) \equiv \forall s''. s \leq s'' \land$$
$$\left( \forall c', s'. s < do(c', s') \leq s'' \rightarrow \alpha(c', s') \right) \rightarrow \phi(s')$$
$\mathcal{R}$ Graphically

\begin{center}
\includegraphics[width=\textwidth]{diagram.png}
\end{center}

Persistence

Regression

Ryan Kelly (rfk@csse.unimelb.edu.au)
Knowledge and Observation in the Situation Calculus
Regression using Persistence

It becomes possible to define the regression of our **Knows** macro:

\[
R[\text{Knows}(\text{agt}, \phi, \text{do}(c, s))] = \\
[\text{Observations}(\text{agt}, c, s) = \{\} \rightarrow \text{Knows}(\text{agt}, \phi, s)] \\
\wedge [\exists o. \text{Observations}(\text{agt}, c, s) = o \wedge o \neq \{\} \rightarrow \text{Knows}(\text{agt}, \forall c'. \text{Observations}(\text{agt}, c', s') = o \rightarrow R[\mathcal{P}[\phi, \text{UA}(\text{agt})](\text{do}(c', s'))], s)]
\]

**UA***(agt, c, s) ≜ Poss(c, s) ∧ Observations(agt, c, s) = \{\}
Calculating Persistence

Define $P^1[\phi, \alpha]$ to be ”persistence to depth 1”:

$$P^1[\phi, \alpha](s) \overset{\text{def}}{=} \phi(s) \land \forall c. \left[ \alpha(c, s) \rightarrow R[\phi(do(c, s))] \right]$$

We can assert that a property holds to depth 2, 3, ... by repeatedly applying $P^1$:

$$P^n[\phi, \alpha] = P^1[P^{n-1}[\phi, \alpha], \alpha]$$

We want persistence for any $n$: need the least-fixed-point of $P^1$. Fixed-point theory guarantees we can calculate this by trans-finite iteration.
Reasoning from Observations

We, the omniscient designer, can determine whether an agent knows something in any given situation:

\[ \mathcal{D} \models \text{Knows}(agt, \phi, s) \]

\[ \mathcal{D}_0 \cup \mathcal{D}_{una} \models \mathcal{R}^*[\text{Knows}(agt, \phi, s)] \]

But the agent cannot use this to reason about its own knowledge, as it doesn’t know the current situation. It needs to be able to reason from a local viewpoint:

\[ \mathcal{D} \models \text{Knows}(agt, \phi, h) \]
Regressing Over Observations

The regression operator can be modified to act over observation histories, instead of over situations:

\[
\mathcal{R}[\text{Knows}(agt, \phi, o \cdot h)] = \\
\text{Knows}(agt, \forall c'. \text{Observations}(agt, c', s') = o \rightarrow \\
\mathcal{R}[\mathcal{P}[\phi, \text{UA}(agt)](do(c', s'))], h)
\]

We can equip agents with a situation calculus model of their own environment.
**Demonstration**

Simple test domain: three agents and several objects which they can pickup, putdown and drop.

Agents can only observe their own actions.
Conclusions

What did we want?

- A powerful account of multi-agent knowledge
  - Concurrent Actions
  - Partial Observability
- that supports complex symbolic reasoning
- that allows agents to reason about their own world
- but is still computationally feasible

Future work:

- Extend implementation to handle concurrent actions
- Efficient calculation of persistence condition

Ryan Kelly (rfk@csse.unimelb.edu.au)
Knowledge and Observation in the Situation Calculus
Conclusions

What did we want?

- A powerful account of multi-agent knowledge
  - Concurrent Actions
  - Partial Observability
- that supports complex symbolic reasoning
- that allows agents to reason about their own world
- but is still computationally feasible

Future work:

- Extend implementation to handle concurrent actions
- Efficient calculation of persistence condition