Knowledge in the Situation Calculus

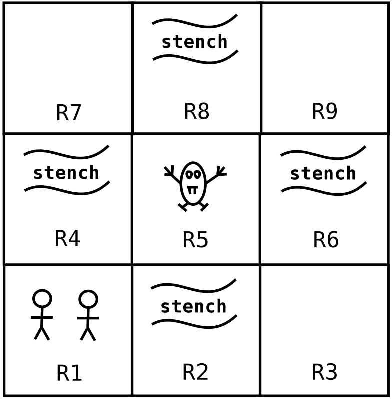
Adrian Pearce

joint work with Ryan Kelly

Department of Computing and Information Systems
The University of Melbourne

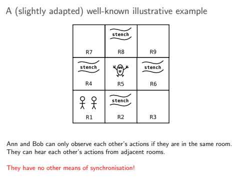
22 September 2014

A (slightly adapted) well-known illustrative example



Ann and Bob can only observe each other's actions if they are in the same room. They can hear each other's actions from adjacent rooms.

They have no other means of synchronisation!



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?

(Multi-agent) Hunt The Wumpus in the Situation Calculus

Action description axioms $\mathcal{D}_{ad}$:

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

Successor state axioms $\mathcal{D}_{ssa}$:

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

Initial situation (S_0): $Init(s) \rightarrow \forall agt : Loc(agt, s) = R1$
$Init(s) \rightarrow Wumpus(s) \neq R1 \wedge \neg Stench(R1, s)$
$Wumpus(S_0) = R5$

Knowledge in the Situation Calculus

Reasoning tasks

Extensions to the Situation Calculus for representing and reasoning about knowledge

- Epistemic reasoning [HalpernMoses92] (not this work)
  Epistemic reasoning about unrestricted forms of nested knowledge in K_n – Reasoning about group-level knowledge modalities
- Synchronous Knowledge [ScherlLevesque03] (builds on this work)
  Knowledge, action and the frame problem
- Asynchronous Knowledge [KellyPearce07] (our work)
  Asynchronous knowledge in the situation calculus — reasoning about knowledge with hidden actions

  Two aspects to knowledge

  1. incomplete information (through action can learn)
  2. lack of synchronisation (don't know how many actions have occurred)

Asynchronous Knowledge in the Situation Calculus?

Explanation closure assumes complete knowledge of $\mathcal{D}_{ssa}$

- Golog assumes complete knowledge of $\mathcal{D}_{ad}$ and $\mathcal{D}_{una}$ in S_0
- What if incomplete knowledge: $\mathbf{Knows}(\phi, s)$?

Limitation: Synchronicity This works well, but it depends on two assumptions:

- Complete knowledge (linear plan, no sensing)
- Synchronous domain (agents proceed in lock-step)

Nearly universal in the literature: *"assume all actions are public"*.

Challenge: Regression depends intimately on Synchronicity

Axiomatizing Observations

First, we must represent asynchronicity.

We reify the *observations* made by each agent, by adding the following action description function of the following form to D_{ad}:

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

If $Obs(agt, c, s) = \emptyset$ then the actions are completely hidden.

$View(agt, S_0) = \epsilon$
$Obs(agt, c, s) = \emptyset \rightarrow View(agt, do(c, s)) = View(agt, s)$
$Obs(agt, c, s) \neq \emptyset \rightarrow View(agt, do(c, s)) = Obs(agt, c, s) \cdot View(agt, s)$

Axiomatizing Observations

First, we must represent asynchronicity.

We reify the *observations* made by each agent, by adding the following action description function of the following form to D_{ad}:

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

If $Obs(agt, c, s) = \emptyset$ then the actions are completely hidden.

$View(agt, S_0) = \epsilon$
$Obs(agt, c, s) = \emptyset \rightarrow View(agt, do(c, s)) = View(agt, s)$
$Obs(agt, c, s) \neq \emptyset \rightarrow View(agt, do(c, s)) = Obs(agt, c, s) \cdot View(agt, s)$

- Reify (make concrete) the local perspective of each agent by explicitly talking abo what it has observed
- *when actions c are performed in situation s, agent agt will perceive observations o*
- Each agent makes a set of *observations*, each situation then corresponds to a loca *view* for that agent.
- Allowing the set of observations to be *empty* lets us model truly asynchronous domains.
- Definitions at bottom are added to foundational actions (since they do not chang from domain to domain).

Observations

In synchronous domains, everyone observes every action:

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

Sensing results can be easily included as action#sensing pairs:

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

And observability can be axiomatised explicitly

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

Observations

In synchronous domains, everyone observes every action:

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

Sensing results can be easily included as action#sensing pairs:

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

And observability can be axiomatised explicitly

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

- Sensing actions can be considered as private actions (but remember will have to handle situation that arbitrary many unobservable actions have occurred).
- CanObs is introduced as a new action description predicate.

Axiomatizing observations

Multi-agent Hunt the Wumpus

Observation axioms (added to $\mathcal{D}_{ad}$):

$$
\begin{aligned}
move(agt1, r1) \in Obs(agt, c, s) \equiv\ & move(agt1, r1) \in c \\
& \wedge [Loc(agt, s) = Loc(agt1, s) \vee Loc(agt, s) = r1] \\
shoot(agt1, r1) \in Obs(agt, c, s) \equiv\ & shoot(agt1, r1) \in c \\
& \wedge Loc(agt, s) = Loc(agt1, s) \\
alert(agt1) \in Obs(agt, c, s) \equiv\ & alert(agt1) \in c \\
& \wedge Loc(agt, s) = Loc(agt1, s)
\end{aligned}
$$

Axiomatizing observations

Multi-agent Hunt the Wumpus

OBSERVATION axioms:

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

Axiomatizing observations

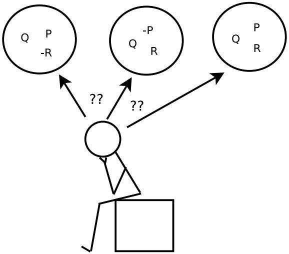
Action: global event changing the state of the world
Observation: local event changing an agent's knowledge

Situation: global history of actions giving current world state
View: local history of observations giving current knowledge

How can we let agents reason using only their local view?

Knowledge

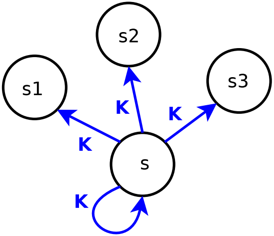
If an agent is unsure about the state of the world, there must be several different states of the world that it considers possible.
The agent *knows* ϕ iff ϕ is true in all possible worlds.



$$\mathbf{Knows}(Q) \;\wedge\; \neg\mathbf{Knows}(P) \;\wedge\; \neg\mathbf{Knows}(R) \;\wedge\; \mathbf{Knows}(P \vee R)$$

Knowledge

Introduce a possible-worlds fluent $K(agt, s', s)$:



We can then define knowledge as a simple macro:

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

Knowledge follows Observation

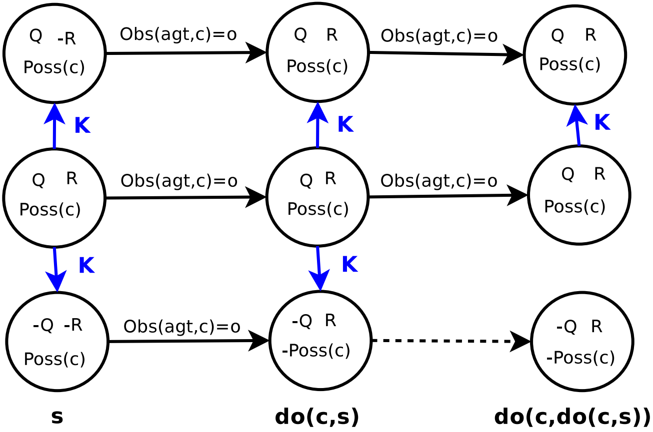
Halpern & Moses, 1990:
"an agent's knowledge at a given time must depend only on its local history: the information that it started out with combined with the events it has observed since then"

Clearly, we require:

$$K(agt, s', s) \ \equiv \ View(agt, s') = View(agt, s)$$

We must enforce this in the successor state axiom for K.

Knowledge: The Synchronous Case



Knowledge: The Synchronous Case

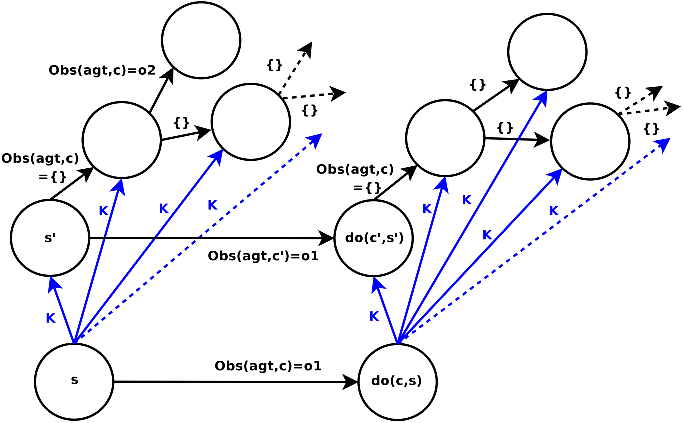
In the synchronous case, K_0 has a simple successor state axiom:

$$K_0(agt, s'', do(c, s)) \equiv \exists s', c' : \ s'' = do(c', s') \land K_0(agt, s', s) \\ \land Poss(c', s') \land Obs(agt, c, s) = Obs(agt, c', s')$$

And a correspondingly simple regression rule:

$$\mathcal{R}(\mathbf{Knows_0}(agt, \phi, do(c, s)) \stackrel{\mathsf{def}}{=} \exists o : Obs(agt, c, s) = o \\ \land \forall c' : \mathbf{Knows_0}(agt, Poss(c') \land Obs(agt, c') = o \rightarrow \mathcal{R}(\phi, c'), s)$$

Knowledge: The Asynchronous Case



Knowledge: The Asynchronous Case

First, some notation:

$$s <_\alpha do(c, s') \;\equiv\; s \leq_\alpha s' \wedge \alpha(c, s')$$

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

Then the intended dynamics of knowledge update are:

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

Sync vs Async

We've gone from this:

$$K_0(agt, s'', do(c, s)) \equiv \exists s', c' : \; s'' = do(c', s') \wedge K_0(agt, s', s) \\ \wedge Poss(c', s') \wedge Obs(agt, c, s) = Obs(agt, c', s')$$

To this:

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

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

Regressing Knowledge

Our new SSA uses $\leq_{PbU(agt)}$ to quantify over all future situations. Regression cannot be applied to such an expression.

An asynchronous account of knowledge cannot be approached using the standard regression operator.

In fact, this quantification requires a second-order induction axiom.
Must we abandon hope of an effective reasoning procedure?

Property Persistence

[KellyPearce2010]

Property persistence facilitates "factoring out" the quantification, this allows us to get on with the business of doing regression.

The *persistence condition* $\mathcal{P}[\phi, \alpha]$ of a formula ϕ and action preconditions α to mean: assuming all future actions satisfy α, ϕ will remain true.

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

Like $\mathcal{R}$, the idea is to transform a query into a form that is easier to deal with.

Property Persistence

[KellyPearce2010]

The persistence condition can be calculated using *fixpoint approximates* [CousotCousot79,Tarski55]

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

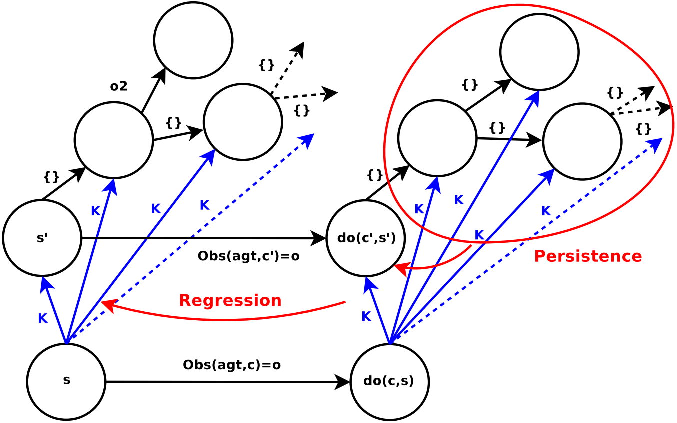
$$\mathcal{P}^n(\phi,\alpha) \stackrel{\text{def}}{=} \mathcal{P}^1(\mathcal{P}^{n-1}(\phi,\alpha),\alpha)$$

$$\left[\mathcal{P}^n(\phi,\alpha) \rightarrow \mathcal{P}^{n+1}(\phi,\alpha)\right] \Rightarrow \left[\mathcal{P}^n(\phi,\alpha) \equiv \mathcal{P}(\phi,\alpha)\right]$$

Corresponds to the *greatest fixpoint* – sound but not complete (might have to go beyond ω in case of infinite ground actions, as SO)

- This calculation provably terminates over complete, finite lattices (e.g. context-free case, STRIPS)
- Can be computed *offline* using *static domain reasoning* for non-disjunctive queries [DemolombePozosParra00]

Regressing Knowledge



Regressing Knowledge

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

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

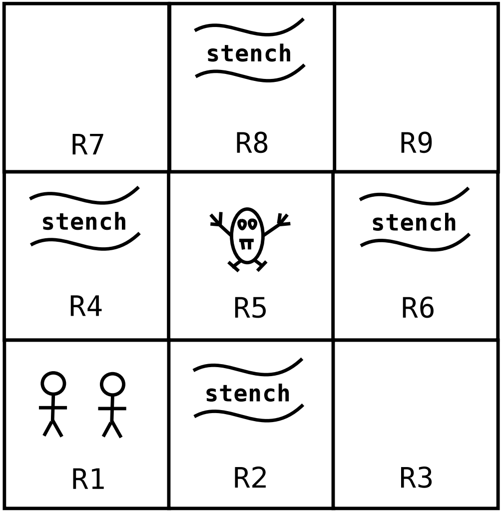
View-Based Reasoning

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

$$\mathcal{R}[\mathbf{Knows}(agt, \phi, o \cdot h)] = \mathbf{Knows}(agt, \forall c' : Obs(agt, c', s') = o \rightarrow \mathcal{R}[\mathcal{P}[\phi, PbU(agt)](do(c', s'))], h)$$

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

Multi-agent Hunt the Wumpus



Example 1

Initially, Ann does not know where the wumpus is:

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

Regression:

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

Fixpoint search terminates after single iteration:

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

Gives: $\mathcal{R}(\neg\exists r : \mathbf{Knows}(A, Wumpus() = r, S_0)) \Rightarrow$
$\neg\exists r : \mathbf{Knows_0}(A, \mathcal{R}((Wumpus() = r)[S_0])^{-1}, S_0)$

Example 1

Initially, Ann does not know where the wumpus is:

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

Regression:

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

Fixpoint search terminates after single iteration:

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

Gives: $\mathcal{R}(\neg\exists r : \mathbf{Knows}(A, Wumpus() = r, S_0)) \Rightarrow$
$\neg\exists r : \mathbf{Knows_0}(A, \mathcal{R}((Wumpus() = r)[S_0])^{-1}, S_0)$

No sequence of hidden actions could result in Ann learning Wumpus location:

- The notation $\phi[s']$ represents a uniform formula in which all fluents have their situation argument replaced with the particular situation term s'
- ϕ^{-1} represents a uniform formula with the situation argument removed from all its fluents.

Example 2

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

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

All initial situations have $\neg Stench(R1)$ and the background theory has $Adjacent(R2, R1)$, $Adjacent(R4, R1)$, 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 2

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

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

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.

It is therefore safe for him to move to an adjacent room.
Regressing as in the previous example gives the equivalent query:

Example 3

After Bob moves into room $R2$, he knows that it has a stench.

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

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

Expanding "$\exists o$" clause into a finite disjunction – only two possible values for $Obs(B, \{move(B, R2)\}, s)$, corresponding to 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:

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

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

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

The persistence condition calculation again terminates after one iteration:

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

So the query further simplifies to:

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

Since domain has 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.

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

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 $R2$.

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

Example 6

After moving to room $R4$ Ann observes a stench, knows that there is a stench in both $R2$ and $R4$:

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

And hence knows that the wumpus is in room $R5$:

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

Example 7

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

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

Example 8

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

$$\mathcal{D} \cup \mathcal{D}_K^{obs} \models \mathbf{Knows}(A, Killed, \\ do([\{move(B, R2)\}, \{alert(B)\}, \{move(A, R3)\}, \{shoot(A, R4)\}], S_0))$$

Example 9

Ann knows that Bob knows the wumpus is dead, as he will have heard the scream regardless of this location.

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

- At this point the hunters will have common knowledge that the Wumpus is dead.
- However, the current formalism is not rich enough to reason directly about common knowledge
- Although we have preliminary work on this topic in [KellyPearce08] that could be integrated with the approach taken here.

Applications of Knowledge in the Situation Calculus

What kind of applications?

- Public actions
- Private actions
- Guarded sensing actions
- Speech acts
- Explicit observability axioms
- Observability interaction
- Observing the effects of actions
- Delayed communication

Summary

A robust account of knowledge based on observations, allowing for arbitrarily-long sequences of hidden actions, in asynchronous settings

Allows agents to reason about their own knowledge using only their local information

- Main point: Facilitates first-order theorem proving for reasoning about hidden actions – Regression rule avoids SO logic by utilizing persistence condition
- Subsumes existing accounts of knowledge – equivalent to [ScherlLevesque03] in synchronous case
- Elaboration tolerant — compatible with existing techniques
- Preliminary Prolog implementation using modal variant of the LeanT^{A}P theorem prover [BernhardBeckertPosegga95,Fitting98], verified its operation on some simple examples

References

- Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Reasoning about Knowledge. The MIT Press, Cambridge, Massachesetts, 1995.
- Joseph Y. Halpern and Yoram Moses, Knowledge and Common Knowledge in a Distributed Environment, Journal of the ACM, Vol. 37, No. 3, pp. 549–587, 1990.
- Richard Scherl and Hector Levesque. Knowledge, Action, and the Frame Problem. Artificial Intelligence, 144:1-39, 2003.
- 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.

References

- 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.
- Ryan Kelly. "Asynchronous Multi-Agent Reasoning in the Situation Calculus", PhD Thesis, The University of Melbourne, 2008