Modeling Knowledge and Deliberation in the Situation Calculus

Yves Lespérance

Department of Computer Science
York University
Toronto, Canada

Ph.D. Seminar, Univ. of Rome, May 2002

Motivation

To do planning in IndiGolog, user provides non-deterministic program/plan skeleton, which is put in a “search block”. Interpreter must find plan/strategy that ensures successful execution of search block.

This is a very general version of the problem of *planning under incomplete information* with *sensing actions*.

Need specification. When is a plan a solution?

Can’t be just any program that achieves goal or reaches a final situation of the user supplied program. Should not require deliberation to execute.

Often, specifications require planner to return a program of a syntactically restricted form.

Our Approach To Formalizing Planning/Deliberation

Have developed account of planning under incomplete information that is more abstract, semantically-motivated.

Requires planner to return an *epistemically feasible* program, i.e., a program for which the agent always *knows* what step to take next no matter how sensing turns out.

Account framed as semantics for IndiGolog search/deliberation operator.

To characterize epistemically feasible plans, we will use an explicit model of knowledge within the action theory.

Before going over our semantics of search, we will look at how knowledge is modeled in logic, and in particular in the situation calculus.

E.g. Getting on Flight at Airport [Levesque 96]

Agent wants to board flight. Does not know gate in advance; must sense at airport.

To do planning to get on flight, execute IndiGolog program:

$$GetOnFlight_{Sketchy} \stackrel{\text{def}}{=} \Sigma(achieve(On\_plane(Flight123), True))$$

where

$achieve(Goal, GoodSit) \stackrel{\text{def}}{=}$

while $\neg Goal$ **do**

$\pi\, a[a; GoodSit(now)?]$

endWhile

E.g. Getting on Flight (cont.)

A fully detailed plan (user defined or returned by planner):

$GetOnFlight_{Detailed} \stackrel{\text{def}}{=}$

$go(Airport)$;
$check\_departures$; % sensing action
if $Parked(Flight123, GateA)$ **then**
  $go(GateA)$; $board\_plane(Flight123)$
else
  $go(GateB)$; $board\_plane(Flight123)$
endIf

Without sensing, goal cannot be achieved!

Lecture Outline

- Introduction
- Modeling Knowledge
- Formalization of Epistemically Feasible Programs
- Semantics of IndiGolog Deliberation Operator
- Cases where Syntactically Restricted Programs Are Sufficient
- Conclusion

Modal Logic of Knowledge

The first thing that logicians noticed about mental attitudes like knowledge, belief, desire, etc. is that they are not "truth-functional" like the propositional connectives ($\neg$, $\wedge$, etc.); whether **Know**$(Person, \phi)$ is true or false does not depend (mainly) on the truth-value of ϕ.

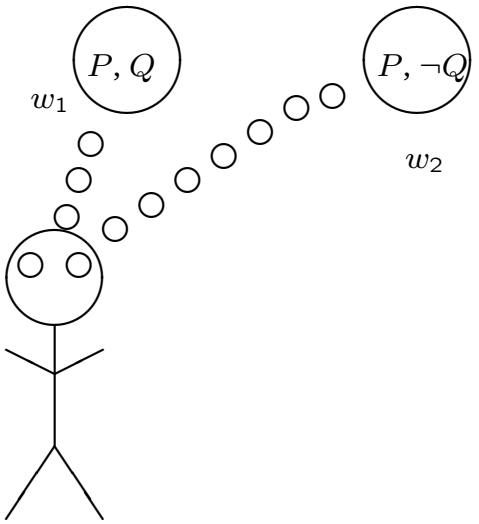
Reasoning about such notions has been formalized in *modal logics*. Sentence forming operators like **Know** are called modal operators.

Because modal operators are not truth-functional their semantics cannot be specified like that of the propositional connectives. Instead, the most common approach used is *possible world semantics*, which was developed by Kripke in the late '50s.

The basic intuition behind possible world semantics as applied to knowledge is to model what someone knows by enumerating the possible ways he or she thinks the world might be.

Modal Logic of Knowledge (cont.)

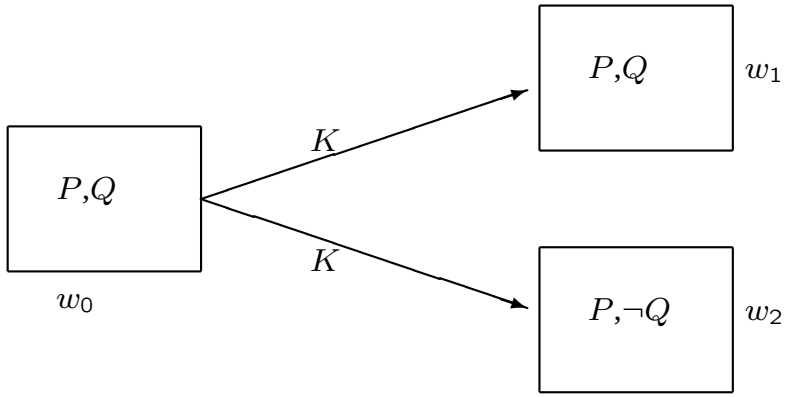
So for e.g. we might model an agent that knows that P is true but does not know whether Q is true or false as follows:



The “worlds” that the agent thinks are possible are said to be *accessible*. When an agent has more knowledge, he/she has fewer accessible worlds.

Modal Logic of Knowledge (cont.)

Mathematically, we model this with an accessiblilty relation K over worlds, e.g.:

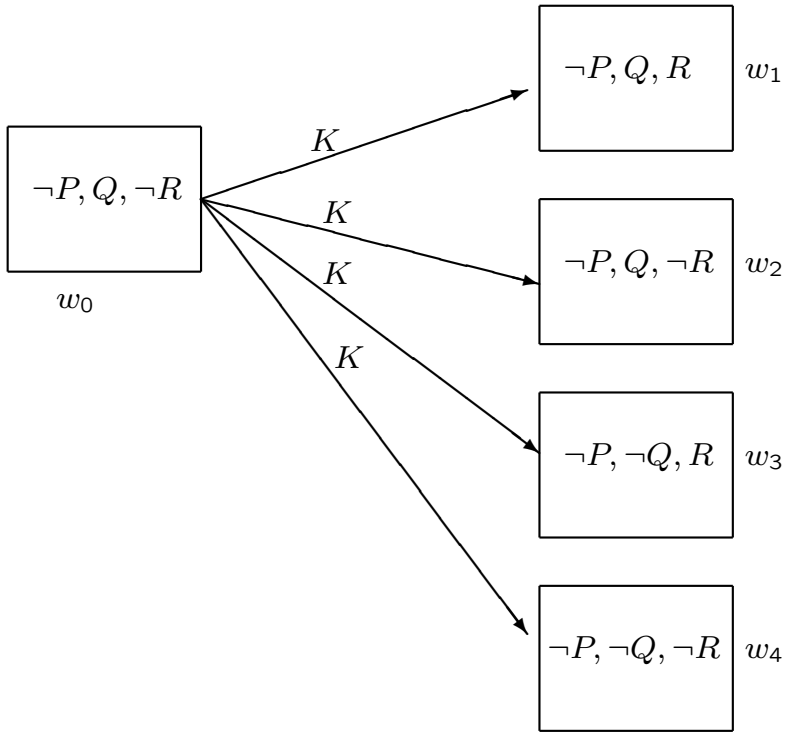


Then, the agent knows that ϕ in w iff ϕ is true in all w' that are accessible in w, i.e.,

$w \models$ **Know**(ϕ) iff
for all w' such that $K(w, w')$, $w' \models \phi$

Modal Logic of Knowledge (cont.)

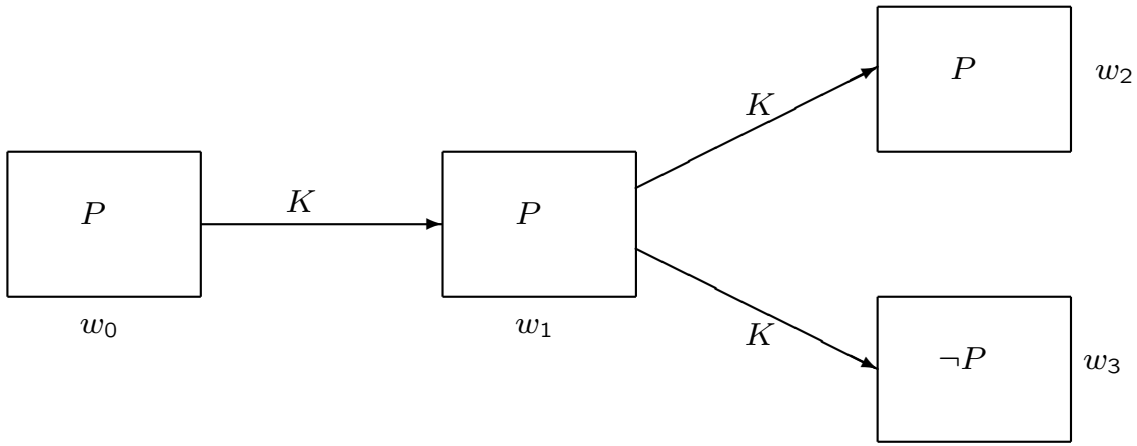
Another e.g., an agent that knows that P is false and does not know the truth value of Q and R:



Note that the number of possible worlds needed can grow exponentially with the number of atomic propositions.

Modal Logic of Knowledge (cont.)

One can also model nested attitudes in this approach. E.g., an agent that knows that P is true, but does not realize that this is the case:



$$\textbf{Know}(P) \wedge \neg\textbf{Know}(\textbf{Know}(P))$$

Modal Logic of Knowledge (cont.)

One of the main contributions of Kripke semantics is that one can force the model to satisfy certain interesting properties by imposing some constraints on the accessiblilty relation.

For e.g., by requiring that the accessiblilty relation K be transitive, we ensure that the following is valid:

$$\models \textbf{Know}(\phi) \supset \textbf{Know}(\textbf{Know}(\phi))$$

This is called the positive introspection principle.

Knowing Who/Knowing What

E.g. the bank manager knows what the combination of the bank's safe is; most other agents know that there is a combination for the safe but don't know what it is.

Knowing who/what can be represented by *quantifying into* the knowledge operator.

E.g. for the bank manager, we have

$$\exists c\, \mathbf{Know}(combination(BankSafe) = c).$$

For most other agents, have

$$\mathbf{Know}(\exists c\, combination(BankSafe) = c) \wedge$$
$$\neg \exists c\, \mathbf{Know}(combination(BankSafe) = c).$$

Knowledge in the Situation Calculus

Now, want to model how knowledge changes as actions are performed.

Can use the situation calculus to model action. First need to adapt the "possible world" model to the situation calculus [Moore].

Introduce accessibility relation as a fluent:

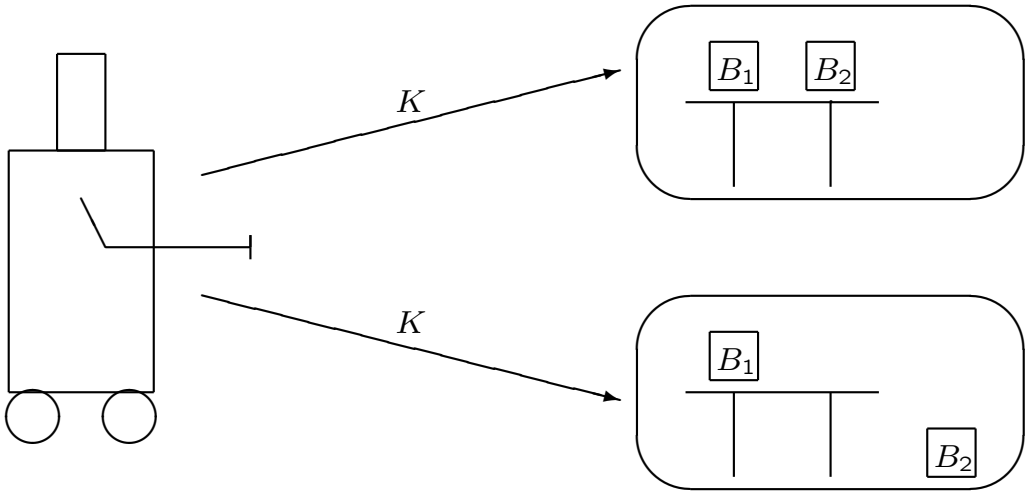
$K(s', s)$ holds
iff
in s, the agent thinks that s' may be the actual situation

Then treat knowledge as abbreviation:

$$\textbf{Know}(\phi, s) \stackrel{\text{def}}{=} \forall s'[K(s', s) \supset \phi(s')]$$

Knowledge in the Situation Calculus (cont.)

E.g.



$$\textbf{Know}(OnTable(b), s) \stackrel{\text{def}}{=} \forall s'[K(s', s) \supset OnTable(b, s')]$$

Knowledge and Action

Introduce primitive *knowledge-producing* actions

These affect agent's mental state, not the external world, e.g., after checking his sonars, a robot knows whether it is close to an obstacle.

Possible to extend solution to frame problem and obtain successor state axiom for knowledge fluent.

Knowledge and Action (cont.)

[Scherl & Levesque 93, Levesque 96]

Here, binary sensing actions only.

Information provided by sensing action represented by $SF(a, s)$. Have axioms:

$$SF(sense_\phi, s) \equiv \phi(s)$$

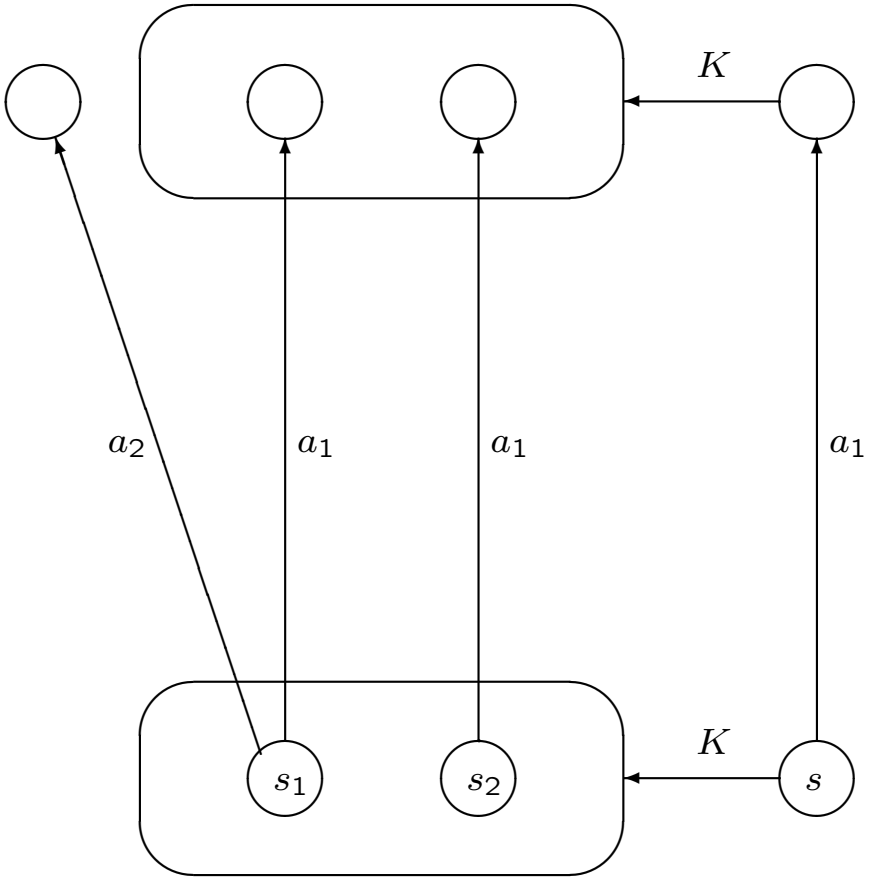
$$SF(nonSensingAct, s) \equiv True$$

Knowledge dynamics specified by successor state axiom for K:

$$\begin{aligned} K(s'', do(a, s)) \equiv \\ \exists s'[K(s', s) \wedge s'' = do(a, s') \wedge Poss(a, s') \wedge \\ (SF(a, s') \equiv SF(a, s))] \end{aligned}$$

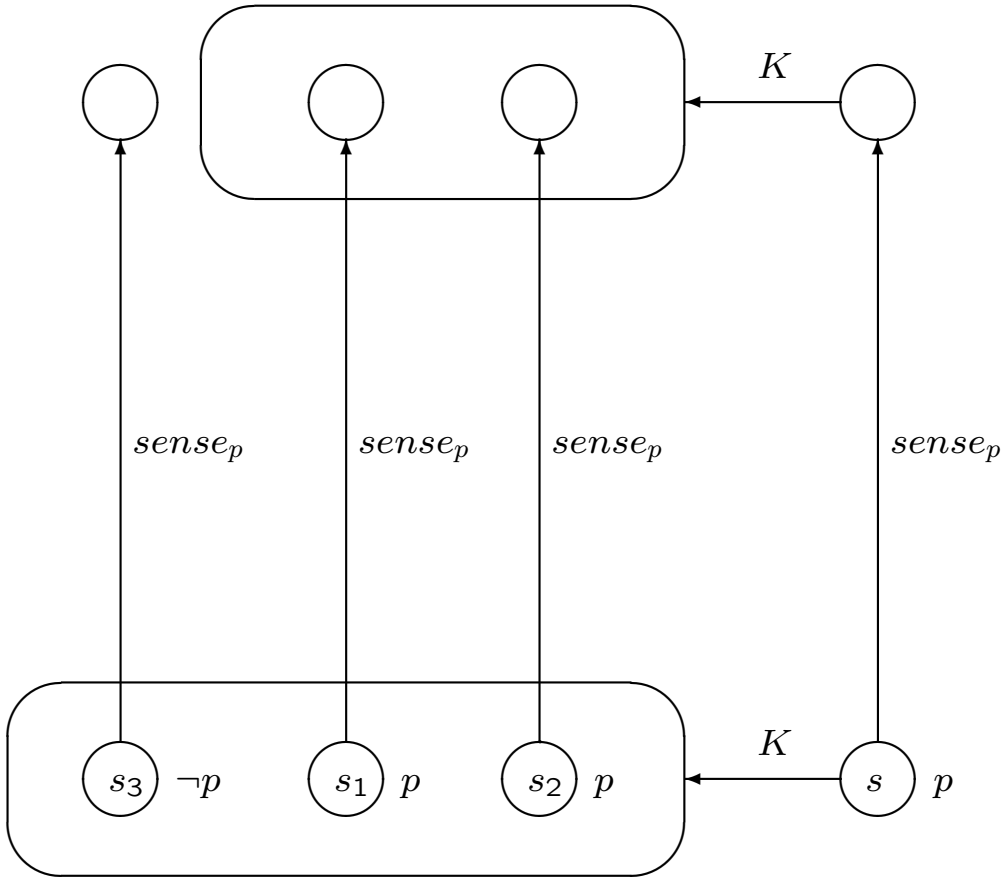
Knowledge and Action (cont.)

For non-sensing action act_1:



Knowledge and Action (cont.)

For sensing action $sense_p$:



Lecture Outline

- Introduction $\surd$
- Modeling Knowledge $\surd$
- Formalization of Epistemically Feasible Programs
- Semantics of IndiGolog Deliberation Operator
- Cases where Syntactically Restricted Programs Are Sufficient
- Conclusion

Epistemically Accurate Theories

We use *epistemically accurate theories*, i.e., theories where what is known accurately reflects what the theory says about the system. See paper for formal details.

For epistemically accurate theories we have:

Theorem 1 *For any objective sentence about situation* s, $\phi(s)$, $Axioms \cup \{Sensed[\sigma]\} \models \phi(end[\sigma])$ *if and only if* $Axioms \cup \{Sensed[\sigma]\} \models$ **Know**$(\phi, end[\sigma])$.

i.e., if some objective property of the system is entailed, then it is also known and vice-versa.

Epistemically Feasible Deterministic Programs

Programs that are deterministic and for which executor always has enough information to continue the execution, always knows what next step is. Formally:

$$EFDP(dp, s) \stackrel{\text{def}}{=} \forall dp', s'.Trans^*(dp, s, dp', s') \supset LEFDP(dp', s').$$

i.e., a program is an $EFDP$ in a situation if all reachable configurations involve a locally epistemically feasible deterministic program ($LEFDP$).

Epistemically Feasible Deterministic Programs (cont.)

Locally epistemically feasible deterministic programs:

$$
\begin{aligned}
&LEFDP(dp, s) \stackrel{\text{def}}{=} \\
&\quad \textbf{Know}(Final(dp, now), s) \vee \\
&\quad \exists dp'.\textbf{Know}(UniTrans(dp, now, dp', now), s) \vee \\
&\quad \exists dp', a.\textbf{Know}(UniTrans(dp, now, dp', do(a, now)), s)
\end{aligned}
$$

$$
\begin{aligned}
&UniTrans(dp, s, dp', s') \stackrel{\text{def}}{=} Trans(dp, s, dp', s') \wedge \\
&\quad \forall dp'', s''.Trans(dp, s, dp'', s'') \supset dp'' = dp' \wedge s'' = s'
\end{aligned}
$$

i.e., a program is a $LEFDP$ in a situation if the agent knows that it is currently $Final$ or knows what unique transition it can perform next.

Epistemically Feasible Det. Programs — Examples

Our original detailed plan to get on a flight is an $EFDP$:

$GetOnFlight_{Detailed} \stackrel{\text{def}}{=}$

$go(Airport)$;
$check\_departures$;
if $Parked(Flight123, GateA)$ **then**
  $go(GateA)$; $board\_plane(Flight123)$
else
  $go(GateB)$; $board\_plane(Flight123)$
endIf

But if delete the $check\_departures$ sensing action, no longer an $EFDP$.
Agent does not know what to do next at the test.

Epistemically Feasible Det. Programs — Properties

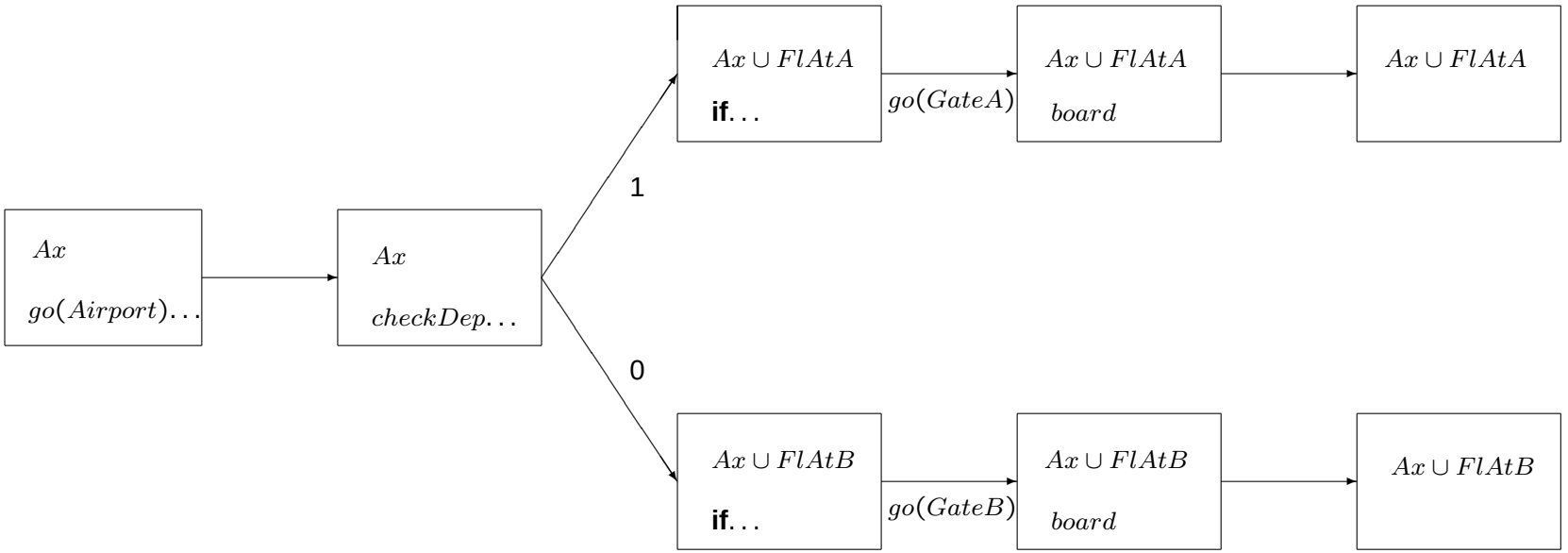
An $EFDP$ need not always terminate.

But if it is entailed that an $EFDP$ will terminate, then it can be successfully executed online whatever the sensing outcomes may be:

Theorem 2
Let dp be such that $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dp, end[\sigma])$. Then, $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dp, end[\sigma], s_f)$ if and only if all online executions of (dp, σ) are terminating.

Epistemically Feasible Det. Programs — Examples (cont.)

For $GetOnFlight_{Detailed}$ program, have the online executions:



Deliberation Operator — Semantics

$$Trans(\Delta_e(p), s, dp', s') \equiv$$
$$\exists dp.EFDP(dp, s) \wedge$$
$$\exists s_f.Trans(dp, s, dp', s') \wedge Do(dp', s', s_f) \wedge Do(p, s, s_f).$$
$$Final(\Delta_e(p), s) \equiv Final(p, s).$$

$(\Delta_e(p), s)$ can make a transition iff there is an $EFDP$ dp that reaches a $Final$ situation of the program provided p.

Thus $Axioms \cup \{Sensed[\sigma]\} \models Trans(\Delta_e(p), end[\sigma], dp', s')$ iff axioms entail that there is some $EFDP$ dp that reaches a $Final$ situation of the given program p no matter how sensing turns out (i.e., in every model).

Note: commits to the selected $EFDP$.

Deliberation Operator — Properties

Theorem 3 *If* $Axioms \cup \{Sensed[\sigma]\} \models Trans(\Delta_e(p), end[\sigma], p', s')$, *then*

1. *given program* p *can reach a* $Final$ *situation in every model, i.e.,* $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(p, end[\sigma], s_f)$,

2. $\Delta_e(p)$ *reaches a* $Final$ *situation in every model, i.e.,* $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(\Delta_e(p), end[\sigma], s_f)$, *and*

3. *All online executions from* $(\Delta_e(p), \sigma)$ *terminate,i.e.,* $\Delta_e(p)$ *can be successfully executed online whatever the sensing results are.*

Syntax-Based Accounts of Deliberation

In general, search/deliberation is very hard; amounts to planning where class of potential plans is very general.

Two interesting restricted classes:

- programs that do not perform sensing, i.e., *conformant plans*, and

- programs that are guaranteed to terminate in a bounded number of steps, i.e., *conditional plans*.

Have shown that for these 2 classes, one can restrict attention to simple syntactically-defined classes of programs without loss of generality.

Tree Programs

Class of *(sense-branch) tree programs* $TREE$ defined by:

$$dpt ::= nil \mid False? \mid a; dpt_1 \mid True?; dpt_1 \mid sense_\phi; \textbf{ if } \phi \textbf{ then } dpt_1 \textbf{ else } dpt_2$$

where a is non-sensing action, and $dpt_1, dpt_2 \in TREE$.

Includes conditional programs where tests only involve *conditions that have just been sensed* (or trivial tests).

Tree Programs — Properties

Theorem 4 *Let dpt be a tree program, i.e., $dpt \in TREE$. Then, for all histories σ, if $Axioms \cup \{Sensed[\sigma]\} \models \exists s_f.Do(dpt, end[\sigma], s_f)$ then $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dpt, end[\sigma])$.*

Whenever a tree program is executable, it is also epistemically feasible — agent will always know what to do next.

Finding a tree program that yields an execution of a program in a search block is our analogue of conditional planning (under incomplete information).

Tree Programs Can Express Any Bounded Strategy

Theorem 5 *For any program dp that is*

1. *an epistemically feasible deterministic program, i.e., $Axioms \cup \{Sensed[\sigma]\} \models EFDP(dp, end[\sigma])$ and*

2. *such that there is a known bound on the number of steps it needs to terminate, i.e., where there is an n such that $Axioms \cup \{Sensed[\sigma]\} \models \exists p', s', k.k \leq n \wedge Trans^k(dp, end[\sigma], p', s') \wedge Final(p', s')$,*

there exists a tree program $dpt \in TREE$ such that $Axioms \cup \{Sensed[\sigma]\} \models \forall s_f.Do(dp, end[\sigma], s_f) \equiv Do(dpt, end[\sigma], s_f)$.

Tree Programs Can Express Bounded Strategy (cont.)

So if restrict attention to $EFDP$s that terminate in bounded number of steps, then can further restrict attention to programs of very specific syntactic form, without any loss in generality.

Can be used to simplify implementation of deliberation.

Linear Programs

Also look at class of *linear programs*, i.e., sequences of primitive actions. Show that:

- Whenever a linear programs is executable, it is also epistemically feasible.
- Linear programs can express any strategy that does not involve sensing.

E.g. Implementation:
Search Operator Looking for Tree Program

```
trans(search_t(P),H,DPT1,H1):-
      buildTree(P,DPT,H), trans(DPT,H,DPT1,H1).
buildTree(P,[],H):- final(P,H).
buildTree(P,[(true)?|DPT],H):-
     trans(P,H,P1,H), buildTree(P1,DPT,H).
buildTree(P,[A,if(F,DPT1,DPT2)]):-
     trans(P,H,P1,[(A,_)|H]), senses(A,F),
     buildTree(P1,DPT1,[(A,1)|H]),
     buildTree(P1,DPT2,[(A,0)|H]).
buildTree(P,[A|DPT],H):-
     trans(P,H,P1,[(A,_)|H]), not senses(A,_),
     buildTree(P1,DPT,[(A,1)|H]).
buildTree(P,(false)?,H):- inconsistent(H).
```

Sound, but not complete wrt semantics.

Dealing with Non-Binary Sensing Actions

Formalization of $EFDP$ and deliberation handles sensing actions with any number of sensing outcomes.

Results about tree programs still hold for sensing actions with non-binary, but finitely many outcomes.

But with more than a few sensing outcomes, conditional planning becomes impractical without advice from programmer as to what conditions the plan should branch on.

[Sardiña 01] develops an implementation of deliberation in IndiGolog that uses such information.

Execution Monitoring

$\triangle_e$ commits to a particular $EFDP$. Bad idea if exogenous actions can make the selected $EFDP$ impossible to execute.

We also define another deliberation operator that monitors the execution of the selected $EFDP$ and replans when necessary.

Conclusion

New formal account of planning/deliberation with incomplete information; abstract and semantic-based.

Deliberator must produce *epistemically feasible deterministic program*, program for which agent, given initial knowledge and subsequent sensing, always *knows* what step to take next.

Characterized deliberation in IndiGolog in terms of this notion.

Shown that for certain classes of problems — conformant planning and conditional planning — search for epistemically feasible programs can be limited to programs of a simple syntactic form.

Much earlier work on epistemic feasibility. Ours is first to deal with expressive agent programming language and integrate with transition semantics.

Further Research

Relating 1st and 3rd person accounts of deliberation; compositional development of MAS.

Implementation of search/planning with non-binary sensing actions.

Representing and reasoning with incomplete knowledge.

More general accounts of sensing and knowledge change.

Multiagent planning.

Etc.

References

Hector J. Levesque and Gerhard Lakemeyer, *The Logic of Knowledge Bases*, MIT Press, 2001.

Raymond Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

G. De Giacomo, Y. Lespérance, H.J. Levesque, and S. Sardiña. On the Semantics of Deliberation in IndiGolog - From Theory to Implementation. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams (Eds.), *Principles of Knowledge Representation and Reasoning, Proc. of the 8th Int. Conf. (KR2002)*, Toulouse, France, April 22-25, 2002, 603-614, Morgan Kaufmann, 2002.