AI Planning for Autonomy

14. Epistemic Planning

What about the knowledge or belief of others?

Tim Miller

THE UNIVERSITY OF MELBOURNE

Winter Term 2017

Agenda

Agenda

Relevant Reading

- *Planning Over Multi-Agent Epistemic States: A Classical Planning Approach.* by Muise, Miller, McIlraith, Felli, Pearce, and Sonenberg, AAAI, 2015. Available from `http://people.eng.unimelb.edu.au/tmiller/pubs/aaai2015.pdf`

Removing Assumptions

Classical Planning

Recall the follow assumptions about classical planning:

- Deterministic events
- Environments change only as the result of an action
- Perfect knowledge (omniscience)
- Single actor (omnipotence)

In this lecture, we drop the assumption of perfect knowledge, and (start to) drop the assumption of single actor (but only a bit).
Why would we not want to assume perfect knowledge?

1. Because no autonomous entity ever has it.
2. Because many hard problems in autonomy & AI involve finding out information that we don't know; for example, locating survivors in a disaster and providing them with aid.
3. *Because we can never tell everything that another 'agent' (human or artificial) knows!*

Epistemic Planning

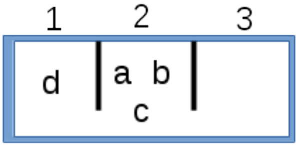
Epistemic Planning is a new and exiting field that removes the assumption that we have perfect information about the environment (it is *partially observable*), but more importantly, that different actors have different *beliefs* or *knowledge* about the environment and each other.

For example:

- Telling someone some new information that you have just learnt.
- Keeping secret, private, and/or classified information from others.
- Deceiving others.

Canonical example: Gossiping over a grapevine

There is a set of agents, and each has their own secret. Agents can share secrets with one another – not just their own secrets, but others as well. At the start, they know only their own secret. They can move freely between a series of rooms in a corridor, and *broadcast* any secret they currently believe to everyone in that room. Everybody inside of the room now believes that secret (thus, we assume strong trust!). Nobody outside of the room hears that secret.



Canonical example: Gossiping over a grapevine (cont'd)

There are three actions:

1. $share(i, secret_j, room_k)$: agent i shares secret j in room k (note that i and j may differ.
2. $move\_left(i)$: agent i moves left.
3. $move\_right(i)$: agent i moves right.

Some possible goals would be: everyone knows everyone else's secret; a subset of the agents known a subset of the secrets; or a subset of agents are deceived about a subset of secrets.

This problem is a simplification of well-known communication and network problems, and how to coordinate in team environments.

Example Solution: Consider a goal: agent a believes secret b is true, while it is NOT the case that agent b believes that agent a believes that secret b is true.

A solution:

$$[\ move\_right(a), share(b, secret_b, 1), move\_right(c), share(c, secret_b, 2)\ ]$$

Agenda

Epistemic Logic

Essentially, we are going to represent these as standard planning problems (e.g. classical or non-deterministic problems), except our state space and our actions can model *belief* or *knowledge* – hence the term “epistemic”.

We will use the following grammar as a simplified version of epistemic logic to represent epistemic atoms:

$$\phi ::= f \mid B\{ag\}\phi \mid \mathrm{not}(\phi)$$

Where:

- ag is a particular agent
- $f \in \mathcal{F}$: An original atom without belief
- E.g., $B\{Sue\}raining$: “Sue believes it is raining”
- E.g., $B\{Sue\}\mathrm{not}(B\{Bob\}at(Sue, 0, 1))$: “Sue believes that Bob does not believe she is at position (0, 1).”

Example actions: Sharing a secret

An agent can share a secret in a room if they are in that room and they believe that secret. If they execute the action, all agents in the room believe the secret.

```
(:action share_secret
    :parameters (?agent ?secret ?room)
    :precondition (and
          (at(?agent, ?room))
          (B{?agent}?secret))
    :effect (and
               (forall ?a2 - agent
                (when (at(?a2, ?room))
                    B{?a2}(?secret)))))
```

This is inherently and quite radically different to the action models we have seen so far in this subject. So far, we have looked at actions that can change the “environment”; e.g. by moving things, swapping things, etc. These are physical or *ontic* actions.

Here, we have an action that changes the mental state of another agent. These are *epistemic* or *doxastic* actions.

Epistemic Planning: Definition

An epistemic planning problem can be defined as:

- a state space S
- initial state $s_0 \in S$
- a set $G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each state $s \in S$
- a transition function $f(a, s)$ – this can be deterministic, non-deterministic, probabilistic.
- action costs $c(a, s) > 0$

What is different from classical/non-deterministic/MDP planning? From the definition above – nothing! However, the state space is over sets of belief atoms ϕ introduced two slides previously, rather than over just sets of atoms.

So what?!

Is this any harder? Can't we just write `B_sue_secret` as a normal atom and use a classical/non-deterministic/probabilistic planner? Sort of. This is almost what we do, except there are problems. Consider this:

The propositional atom `secret` can be false in one way: `not(secret)`.

The epistemic atom `B{Sue}secret` is *modal*, because we shift the mode (or perspective) to Sue. Thus, this atom can be false in two ways: (1) `not(B{Sue}secret)`; and (2) `B{Sue}not(secret)`.

That is, it is not the case that Sue believes the secret is true; or it is the case that Sue believes the secret is NOT true.

These two mean different things:

1. `not(B{Sue}secret)` specifies *absence* of believe of the secret.
2. `B{Sue}not(secret)` is stronger: it specifies the belief of the *opposite* (or negation) of the secret.

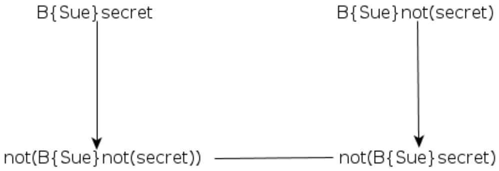
One would expect that if Sue believes the latter, then she should also believe the former (if we assume that Sue's beliefs are consistent).

Epistemic Modalities

Thus, we have four combinations here:

```
B{Sue}secret   B{Sue}not(secret)   not(B{Sue}secret)
                 not(B{Sue}not(secret)).
```

However, only two of them can be true at any point in time:



Problems

The use of epistemic atoms introduces three main problems that need to be solved during the search process:

1. How to update the state when an epistemic atom becomes true/false. This is known as the *belief update* problem.
2. How to update the state when an agent is aware that an event *may* have happened, but is unsure. This is known as the *uncertain firing* problem.
3. How to update the state to ensure that it is *complete*. This is know as the *closure* problem.

But first, a new concept: Conditional effects

To solve these problems, we are going to rely on a common planning method known as *conditional effects*. We already saw an example of this in the `share_secret` action:

```
    (when (at(?a2, ?room)) B{?a2}(?secret))
```

The `when (at(?a2, ?room))` is the condition on the effect. What this means is that when the action is executed, *if* the condition is true before the action was executed, then the effect will hold after the action is executed. If the condition is *not* true, the effect will only hold if it was already true before (that is, nothing changes).

A single action can have multiple such conditional effects.

For simplicity, we assume that an action that has no conditional effects is converted to one with a conditional effect of the form `when true` *effect*.

Open-world vs. closed-world modelling

A second concept to introduce is *open-world* vs. *closed-world* modelling.

Typically, planning models, such as the ones we have been discussing, make a closed-world assumption. Simply, this means that if an atom is not true, it must be false. In other words, if we do not know something is true, it must be false.

However, as we have seen earlier with mode/perspectives, epistemic modelling is open-world: we can be *unsure* of things.

To break the closed-world assumption, we need simply to use two atoms for each concept: one representing it being true; and one to represent it being false. For example, `secret` and `neg_secret` (negate secret). Then we can represent the four different perspectives:

```
secret   neg_secret   not(secret)   not(neg_secret).
```

`neg_secret` means that the secret is though to be false; whereas `not(secret)` means that the secret is not thought to be true; and `not(neg_secret)` means that the set is not thought to be false.

Agenda

Compiling away epistemic atoms

Recall that epistemic planning is just search over states of epistemic atoms. Our solution is use existing approaches to solve epistemic planning problems.

We *compile* a deterministic (non-deterministic/probabilistic) planning problem into a deterministic (non-deterministic/probabilistic) planning problem in three steps:

1. *Ground the epistemic atoms*. For every epistemic atom of the form `B{a}`ϕ, replace with a propositional atom `B_a_`ϕ', where ϕ' is the ground version of ϕ.
   For example: `B{Bob}B{Sue}secret` becomes the propositional atom `B_Bob_B_Sue_secret`.
2. *Ground the propositional atoms*. For every propositional atom, create its explicit negation.
   For example, `secret` has an opposite `neg_secret`, signifying that “we”, the planning agent believes the secret is false.
3. *Add new conditional effects*. For every conditional effect of an action, add a (possibly empty) set of new conditional effects that preserve the semantics of the epistemic actions.
   These new conditional effects handle the belief update, uncertain firing, and completeness problems defined earlier.
   We capture these as rules over the actions.

Belief update: Example

Consider the following example. An agent successfully executes the `share_secret` action in a room containing the agent Sue. Therefore, Sue should believe the secret.

During the search, we need to determine the next state. In a classical search, we would add the atom, and remove its negation *if the atom was false before this*.

In an epistemic problem, it is less straightforward. We should clearly add `B{Sue}secret`, but what should we remove?

`B{Sue}not(secret)`?	Remove: Sue should not believe the secret and its opposite.
`not(B{Sue}secret)`	Remove: it should not be that Sue believes the secret and does not believe the secret.
`not(B{Sue}not(secret))`	Keep: if Sue believes the secret, then she should also not believe the secret is false.

This latter can be captured as an invariant: $\texttt{B\{a\}}\phi \rightarrow \texttt{not(B\{a\}not(}\phi\texttt{))}$; which is known as the D axiom in epistemic logic.

Belief update: Formalism

We can formalise belief update as follows:

Belief update

For an action a, if a has an add effect, forget about its negation:

$$\text{“}\texttt{when}\ C\ \mathit{eff}\text{”} \in \mathit{Add}(a)$$
$$\Rightarrow$$
$$\text{“}\texttt{when}\ C\ \texttt{not}\,(\mathit{eff})\text{”} \in \mathit{Del}(a)$$

Example:

```
when at(a, room1) B{a}secret_b          ∈ Add(a)
⇒
when at(a, room1) not(B{a}secret_b)     ∈ Del(a)
```

So, if agent a believes $\mathit{secret\_b}$ is an add effect, then `not(B{a}secret_b)` should be a delete effect: agent a should no longer believe the secret.

Uncertain firing: example

Consider the following example Sue announces *secret* in room 3. We are unsure if Bob is in room 3 too.

Question: what should we believe about Bob's belief about *secret*?

Inherently, we are unsure about Bob's belief now (unless we believed that Bob believes this before the announcement anyway). We do not know if Bob believes the secret or he believes it is false. But! we can say something: we should not rule out the fact that Bob knows the secret.

So what should we add and remove? It gets tricky here!

`B{Bob}secret`?	Keep but do not add: if we believed beforehand that Bob believes the secret, we can safely continue to assume this, but we should not add this.
`B{Bob}not(secret)`?	Remove: we cannot be sure that Bob believes the secret is false.
`not(B{Bob}secret)`?	Remove: as above.
`not(B{Bob}not(secret))`?	Add: we should no longer believe that Bob believes the secret is false.

Uncertain firing: Formalism

We can formalise uncertain firing as follows:

Uncertain firing

For an action a, if a has an add effect, then we should be *aware* of any effect that may have fired:

$$\text{“}\texttt{when}\ C\ \mathit{eff}\text{”} \in \mathit{Add}(a)$$
$$\Rightarrow$$
$$\text{“}\texttt{when not}(\mathit{negate}(C))\ \texttt{not (}\mathit{eff}\texttt{)}\text{”} \in \mathit{Del}(a)$$

where $\mathit{negate}(C)$ takes each atom in C and returns its explicit negation; e.g $\mathit{negate}(\texttt{secret}) = \texttt{neg\_secret}$.
So, the rules states: add a new conditional effect that takes the conditions C that should hold, negates them, makes them conditions that should *not* hold, and add the negation as a delete effect.

This rule is *not* easy to understand without some thought and analysis! So please take the time to consider it.

Uncertain firing: Another Example

Example:

when at(a, room1) B{a}secret_b $\in Add(a)$
$\Rightarrow$
when not(neg_at(a, room1)) not(B{a}secret_b) $\in Del(a)$

So, if we are unsure whether the condition is true (`not(neg_at(a, room1))`), then we should *remove* (note this is a delete effect) that we do not believe that a believes the secret.

If you cannot get this rule, think through this example carefully until you understand it, then return to the rule.

Completeness: Example

This rule is the most simple. Consider an agent successfully executing the `share_secret` action in a room containing the agent Sue. Therefore, Sue should believe the secret. So, we should add `B{Sue}secret`.

However, what is the precondition of another action is: `not(B{Sue}not(secret))`?

Recall earlier D axiom: `B{a}`ϕ $\rightarrow$ `not(B{a}not(`ϕ`))`.

Therefore, this precondition should hold and the execution should be expanded in our search. Currently, it will not be, because executing our action will add the *grounded* atom `B_Sue_secret`, and our search algorithm does not know the D axiom, nor therefore that `neg_B_Sue_neg_secret` should be true.

Therefore, whenever we add something of the form `B{a}`ϕ, we should also add `not(B{a}not(`ϕ`))`.

Similarly, if we remove `not(B{a}not(`ϕ`))`, then we should remove `B{a}`ϕ.

Completeness: Formalism

Assume that for an epistemic atom eff, the term $\mathit{closure}(\mathit{eff})$ returns the set of epistemic atoms resulting from applying axiom D over and over to get the complete logical closure. We can formalise completeness with the following two rules:

Completeness

For an action a, if a has an add/delete effect, then complete these effects with their closure with these two rules:

$$\text{“}\texttt{when}\ C\ \mathit{eff}\text{”} \in \mathit{Add}(a)$$
$$\Rightarrow$$
$$\text{for } e \in \mathit{closure}(\mathit{eff}),\ \ \text{“}\texttt{when}\ C\ e\text{”}\ \in \mathit{Add}(a))$$

and

$$\texttt{when}\ C\ \mathit{eff} \in \mathit{Del}(a)$$
$$\Rightarrow$$
$$\text{for } e \in \mathit{closure}(\mathit{not}(\mathit{eff})),\ \ \text{“}\texttt{when}\ C\ \mathit{not}(e)\text{”}\ \in \mathit{Del}(a)$$

Completeness: Another example

Example 1:

`when true B{a}secret_b` $\in \mathit{Add}(a)$
$\Rightarrow$
`when true not(B{a}not(secret_b))` $\in \mathit{Add}(a)$

Example 2:

`when true not(B{a}secret_b)` $\in \mathit{Del}(a)$
$\Rightarrow$
`when true B{a}not(secret_b)` $\in \mathit{Del}(a)$

Recall that `B{a}secret_b` implies `not(B{a}not(secret_b))` and therefore: `B{a}not(secret_b)` implies `not(B{a}secret_b)`

Example

The compiled version of the example from Slide 11.

```
(:action share_secret
    :parameters (?agent ?secret ?room)
    :precondition (and
        (at(?agent, ?room))
        (B{?agent}?secret))
    :effect (and
         (forall ?a2 - agent
          (when (at(?a2, ?room))
              B{?a2}(?secret))
          (when (at(?a2, ?room))
              not(B{?a2}not(?secret)))     – (Completeness rule)
          (when (not(neg_at(?a2, ?room)))
              not(B{?a2}(?secret)))     – (Uncertain Firing rule)
          (when (not(neg_at(?a2, ?room)))
              B{?a2}not(?secret)))))     – (Completeness rule to the above
addition!)
```

Complexity

The compilation of an epistemic planning problem into a non-epistemic problem result in a problem that is exponentially larger than the original problem.

The complexity of the problem depends on the complexity of the underlying problem. For example, a deterministic epistemic planning problem will compile to an exponentially-larger deterministic classical planning problem. The resulting classical planning problem is a PSPACE problem.

However, as with FOND problems, the techniques discussed here manage to solve many large problems quickly!

Summary: Epistemic Planning

We introduced a new class of planning problem: epistemic planning, and should how to compile this down to a non-epistemic version that state-of-the-art planners solve these problems.

As with FOND planning, what is interesting is that we can use classical search techniques to solve this compilations. Thus, any major breakthroughs made in planning automatically apply to the epistemic planning domain.

What's next? Relaxing the assumption of single agents by looking at multiple actors, with a focus on *adversaries*.