

6.825 Techniques in Artificial Intelligence

Planning

- Planning vs problem solving
- Situation calculus
- Plan-space planning

Lecture 10 • 1

We are going to switch gears a little bit now. In the first section of the class, we talked about problem solving, and search in general, then we did logical representations. The motivation that I gave for doing the problem solving stuff was that you might have an agent that is trying to figure out what to do in the world, and problem solving would be a way to do that.

And then we motivated logic by trying to do problem solving in a little bit more general way, but then we kind of really digressed off into the logic stuff, and so now what I want to do is go back to a section on planning, which will be the next four lectures.

Now that we know something about search and something about logic, we can talk about how an agent really could figure out how to behave in the world. This will all be in the deterministic case. We're still going to assume that when you take an action in the world, there's only a single possible outcome. After this, we'll back off on that assumption and spend most of the rest of the term thinking about what happens when the deterministic assumption goes away.

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions

In planning, the idea is that you're given some description of a starting state or states; a goal state or states; and some set of possible actions that the agent can take. And you want to find the sequence of actions that get you from the start state to the goal state.

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions
- Can be cast as "problem-solving" problem

It's pretty clear that you can cast this as a problem solving problem. Remember when we talked about problem solving, we were given a start state, and we searched through a tree that was the sequences of actions that you could take, and we tried to find a nice short plan. So, planning problems can certainly be viewed as problem-solving problems, but it may not be the best view to take.

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions
- Can be cast as "problem-solving" problem
- But, what if initial state is not known exactly? E.g. start in bottom row in 4x4 world, with goal being C.

A	B	C	D
E			F
G			H
I	J	K	L

Actions: N,S,E,W

Just for fun, let's think about moving around on the squares of this grid. The goal is to get to state C, but all we know initially is that the agent is in the bottom row, one of states I, J, K, and L.

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions
- Can be cast as “problem-solving” problem
- But, what if initial state is not known exactly? E.g. start in bottom row in 4x4 world, with goal being C.
- Do search over “sets” of underlying states to find a set of actions that will reach C for any starting state.

A	B	C	D
E			F
G			H
I	J	K	L

Actions: N,S,E,W

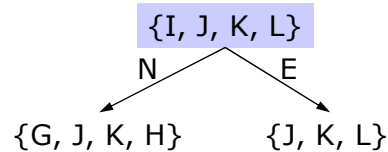
We could try to solve this using our usual problem solving search, but where our nodes would now represent sets of states, rather than single states. So, how would that go?

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions
- Can be cast as "problem-solving" problem
- But, what if initial state is not known exactly? E.g. start in bottom row in 4x4 world, with goal being C.
- Do search over "sets" of underlying states to find a set of actions that will reach C for any starting state.

A	B	C	D
E			F
G			H
I	J	K	L

Actions: N,S,E,W



Lecture 10 • 6

The agent starts in the node corresponding to states I,J,K,L, because it doesn't know exactly where it is.

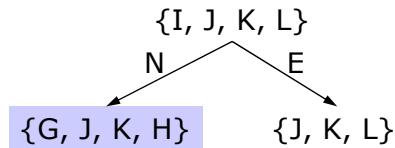
Now, let's say that it can move North, South, East, or West, but if it runs into an obstacle or the edge of the world, it just stays where it was.

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions
- Can be cast as “problem-solving” problem
- But, what if initial state is not known exactly? E.g. start in bottom row in 4x4 world, with goal being C.
- Do search over “sets” of underlying states.

A	B	C	D
E			F
G			H
I	J	K	L

Actions: N,S,E,W



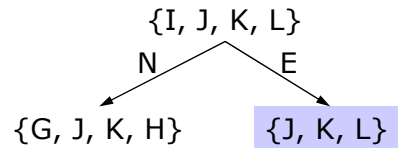
So, if it moves north, it could end up in states G, J, K, or H, so that gives us this result node.

Planning as Problem Solving

- Planning:
 - Start state (S)
 - Goal state (G)
 - Set of actions
- Can be cast as “problem-solving” problem
- But, what if initial state is not known exactly? E.g. start in bottom row in 4x4 world, with goal being C.
- Do search over “sets” of underlying (atomic) states.

A	B	C	D
E			F
G			H
I	J	K	L

Actions: N,S,E,W



And if it moves East, it could end up in J, K, or L.

You can see how this process would continue. And probably the shortest plan is going to be to go east three times (so we could only possibly be in state L) and then continue north until we get to the top, and then go west to the goal.

Planning as Logic

- The problem solving formulation in terms of sets of atomic states is incredibly inefficient because of the exponential blowup in the number of sets of atomic states.

So, we do have a way to think about planning in the case where you're not absolutely sure what your starting state is. But, it's really inefficient in any kind of a big domain because we're planning with sets of states at the nodes. Just to figure out how the state transition function works on sets of primitive states, we're having to go through each little atomic state in our set of states, and think about how it transitions to some other atomic state. And that could be really desperately inefficient.

Planning as Logic

- The problem solving formulation in terms of sets of atomic states is incredibly inefficient because of the exponential blowup in the number of sets of atomic states.
- Logic provides us with a way of describing sets of states.

One of our arguments for moving to logical representations was to be able to have a compact way of describing a set of states. You should be able to describe that set of states IJKL by saying, the proposition "bottom row" is true and you might be able to say, well, if "bottom row", and no obstacles above me, then when I move, then I'll be in "next to bottom row", or something like that.

Planning as Logic

- The problem solving formulation in terms of sets of atomic states is incredibly inefficient because of the exponential blowup in the number of sets of atomic states.
- Logic provides us with a way of describing sets of states.
- Can we formulate the planning problem using logical descriptions of the relevant sets of states?

That might mean that if you can set up the logical description of your world in the right way, you can talk about these sets of states, not by enumerating them, but by giving logical descriptions of those sets of states. And the idea is that a logical formula stands for all the interpretations, or world situations, in which it is true. It stands for all the locations that the agent could be in that would make the formula true.

Planning as Logic

- The problem solving formulation in terms of sets of atomic states is incredibly inefficient because of the exponential blowup in the number of sets of atomic states.
- Logic provides us with a way of describing sets of states.
- Can we formulate the planning problem using logical descriptions of the relevant sets of states?
- This is a classic approach to planning: **situation calculus**, use the mechanism of FOL to do planning.

Lecture 10 • 12

So, this leads us to the first approach to planning. It's going to turn out not to be particularly practical, but it's worth mentioning because it's a kind of a classical idea, and you can get somewhere with it. It's called **situation calculus**, and the idea is to use first-order logic, exactly as you know about it, to do planning.

Planning as Logic

- The problem solving formulation in terms of sets of atomic states is incredibly inefficient because of the exponential blowup in the number of sets of atomic states.
- Logic provides us with a way of describing sets of states.
- Can we formulate the planning problem using logical descriptions of the relevant sets of states?
- This is a classic approach to planning: **situation calculus**, use the mechanism of FOL to do planning.
- Describe states and actions in FOL and use theorem proving to find a plan.

Lecture 10 • 13

We have to figure out a way to talk about states of the world, and start states and goal states and actions in the language of first order logic, and then somehow use the mechanism of theorem proving to find a plan. That's the idea. It's appealing, right? It's always appealing, given a very general solution mechanism, if you can take a particular problem, and see how it's an instance of that general solution; then you can apply your general solution to the particular problem and get an answer out without inventing any more solution methods.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.

In situation calculus, the main idea is that we're going to "reify" situations. To reify something is to treat it as if it's an object. So, we're going to have variables and constants in our logical language that will range over the possible situations, or states of the world. This will allow us to say that world states have certain properties, but to avoid enumerating them all.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function:** a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$

So we can say things like "at robot Room Six" in "situations 9". More ordinarily, we would have said $\text{at}(\text{robot}, \text{room6})$, implicitly saying something about the current state of the world. But in planning, we need to be thinking about a bunch of different situations at the same time, and so we have to be able to name them.

For any proposition that can change its truth value in different situations, we'll add an extra argument, which is the situation in which we're asserting that the proposition holds. Then we can talk about how the world changes over time, by having the predicate or function "result". We can say the result of doing action "move north" in situation one is situation six.

Situation Calculus

- **Reify situations**: [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function**: a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$
- **Effect Axioms**: what is the effect of taking an action in the world

Now we need to write down what we know about S_6 because of the fact that we got it by moving north from S_1 . So, what do we know about S_6 ? In the textbook, Russell and Norvig use the wumpus world to illustrate situation calculus. We can describe the dynamics of the world using **effect axioms**. They say what the effects in the world are of taking particular actions.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function:** a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$
- **Effect Axioms:** what is the effect of taking an action in the world
 - $\forall x.s. \text{Present}(x,s) \wedge \text{Portable}(x) \rightarrow \text{Holding}(x, \text{Result}(\text{Grab}, s))$

Lecture 10 • 17

So, we might say that, for all objects x and situations s , if object x is present in situation s , and if object s is portable, then in the situation that results from executing the grab action in situation s , the agent is holding the object x .

Just to be clear, Grab is an action. So $\text{Result}(\text{Grab}, s)$ is a term that names the situation that results if the agent does a grab action in situation s . And what we're saying in the consequent here is that the agent will be holding the object x in that resulting situation.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function:** a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$
- **Effect Axioms:** what is the effect of taking an action in the world
 - $\forall x.s. \text{Present}(x,s) \wedge \text{Portable}(x) \rightarrow \text{Holding}(x, \text{Result}(\text{Grab}, s))$
 - $\forall x.s. \neg \text{Holding}(x, \text{Result}(\text{Drop}, s))$

Or, we might say that, for all objects x and situations s , that the agent is not holding object x in a situation that results from doing the drop action in situation s .

You can see the power of the logical representation in these axioms. We're able to say things about a very large, or possibly infinite set of situations, without enumerating them.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function:** a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$
- **Effect Axioms:** what is the effect of taking an action in the world
 - $\forall x.s. \text{Present}(x,s) \wedge \text{Portable}(x) \rightarrow \text{Holding}(x, \text{Result}(\text{Grab}, s))$
 - $\forall x.s. \neg \text{Holding}(x, \text{Result}(\text{Drop}, s))$
- **Frame Axioms:** what doesn't change

Lecture 10 • 19

It's not enough to just specify these positive effects of taking actions. You also have to describe what doesn't change. The problem of having to specify all the things that don't change is called the "frame problem." I think the name comes from the frames in movies. Animators know that most of the time, from frame to frame, things don't change.

And when we write down a theory we're saying, well, if there's something here, and you do a grab, then you're going to be holding the thing. But, we haven't said, for instance, what happens to the colors of the objects, or whether your shoes are tied, or the weather, as a result of doing the grab action. But, you have to actually say that. At least, in the situation calculus formulation, you have to say, "and nothing else changes" somehow.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function:** a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$
- **Effect Axioms:** what is the effect of taking an action in the world
 - $\forall x.s. \text{Present}(x,s) \wedge \text{Portable}(x) \rightarrow \text{Holding}(x, \text{Result}(\text{Grab}, s))$
 - $\forall x.s. \neg \text{Holding}(x, \text{Result}(\text{Drop}, s))$
- **Frame Axioms:** what doesn't change
 - $\forall x.s. \text{color}(x,s) = \text{color}(x, \text{Result}(\text{Grab}, s))$

Lecture 10 • 20

One way to do this, is to write frame axioms, where you say things like, "for all objects and situations, the color of X in situation S is the same as the color of X in the situation that is the result of doing grab. That is to say, picking things up doesn't change their color. Picking up other things doesn't change their color. But it can be awfully tedious to have to write all these out.

Situation Calculus

- **Reify situations:** [reify = name, treat them as objects] and use them as predicate arguments.
 - $\text{At}(\text{Robot}, \text{Room6}, S_9)$ where S_9 refers to a particular situation
- **Result function:** a function that describes the new situation resulting from taking an action in another situation.
 - $\text{Result}(\text{MoveNorth}, S_1) = S_6$
- **Effect Axioms:** what is the effect of taking an action in the world
 - $\forall x.s. \text{Present}(x,s) \wedge \text{Portable}(x) \rightarrow \text{Holding}(x, \text{Result}(\text{Grab}, s))$
 - $\forall x.s. \neg \text{Holding}(x, \text{Result}(\text{Drop}, s))$
- **Frame Axioms:** what doesn't change
 - $\forall x.s. \text{color}(x,s) = \text{color}(x, \text{Result}(\text{Grab}, s))$
 - Can be included in Effect axioms

It turns out that there's a solution to the problem of having to talk about all the actions and what doesn't change when you do them. You can read it in the book where they're talking about successor-state axioms. You can, in effect, say what the conditions are under which an object changes color, for example, and then say that these are the **only** conditions under which it changes color.

Planning in Situation Calculus

- Use theorem proving to find a plan

Assuming you write all of these axioms down that describe the dynamics of the world. They say how the world changes as you take actions. Then, how can you use theorem proving to find your answer?

Planning in Situation Calculus

- Use theorem proving to find a plan
- Goal state: $\exists s. \text{At}(\text{Home}, s) \wedge \text{Holding}(\text{Gold}, s)$

You can write the goal down somehow as the logical statement. For instance, you could say, well, I want to find a situation in which the agent is at home and is holding gold. But I don't want to just prove that such a situation exists; I actually want to find it.

Planning in Situation Calculus

- Use theorem proving to find a plan
- Goal state: $\exists s. \text{At}(\text{Home}, s) \wedge \text{Holding}(\text{Gold}, s)$
- Initial state: $\text{At}(\text{Home}, s_0) \wedge \neg \text{Holding}(\text{Gold}, s_0) \wedge \text{Holding}(\text{Rope}, s_0) \dots$

You would encode your knowledge about the initial situation as some logical statements about some particular situation. We'll name it with the constant s_0 . We might say that the agent is at home in s_0 and it's not holding gold, but it is holding rope, and so on.

Planning in Situation Calculus

- Use theorem proving to find a plan
- Goal state: $\exists s. \text{At}(\text{Home}, s) \wedge \text{Holding}(\text{Gold}, s)$
- Initial state: $\text{At}(\text{Home}, s_0) \wedge \neg \text{Holding}(\text{Gold}, s_0) \wedge \text{Holding}(\text{Rope}, s_0) \dots$
- Plan: $\text{Result}(\text{North}, \text{Result}(\text{Grab}, \text{Result}(\text{South}, s_0)))$
 - A situation that satisfies the requirements
 - We can read out of the construction of that situation what the actions should be.
 - First, move South, then Grab and then move North.

Lecture 10 • 25

Then, we could invoke a theorem prover on the initial state description, the effects axioms, and the goal description. It turns out that if you use resolution refutation to prove an existential statement, it's usually easy to extract a description of the particular s that the theorem prover constructed to prove the theorem (you can do it by keeping track of the substitutions that get made for s in the process of doing the proof).

So, for instance, in this example, we might find that the theorem prover found the goal conditions to be satisfied in the state that is the result of starting in situation s_0 , first going south, then doing a grab action, then going north. We can read the plan right out of the term that names the goal situation.

Planning in Situation Calculus

- Use theorem proving to find a plan
- Goal state: $\exists s. \text{At}(\text{Home}, s) \wedge \text{Holding}(\text{Gold}, s)$
- Initial state: $\text{At}(\text{Home}, s_0) \wedge \neg \text{Holding}(\text{Gold}, s_0) \wedge \text{Holding}(\text{Rope}, s_0) \dots$
- Plan: $\text{Result}(\text{North}, \text{Result}(\text{Grab}, \text{Result}(\text{South}, s_0)))$
 - A situation that satisfies the requirements
 - We can read out of the construction of that situation what the actions should be.
 - First, move South, then Grab and then move North.

Note that we're effectively planning for a potentially huge class of initial states at once. We've proved that, no matter what state we start in, as long as it satisfies the initial state conditions, then following this sequence of actions will result in a new state that satisfies the goal conditions.

Special Properties of Planning

- Reducing specific planning problem to general problem of theorem proving is not efficient.

Way back when I was more naive than I am now, when I taught my first AI course, I tried to get my students to do this, to write down all these axioms for the Wumpus world, and to try to use a theorem prover to derive plans. It turned out that it was really good at deriving plans of Length 1, and sort of OK at deriving plans of Length 2, and, then, after that, forget it--it took absolutely forever. And it was not a problem with my students or the theorem prover, but rather with the whole enterprise of using the theorem prover to do this job.

So, there's sort of a general lesson, which is that even if you have a very general solution, applying it to a particular problem is often very inefficient, because you are not taking advantage of special properties of the particular problem that might make it easier to solve. Usually, the fact that you have a particular specific problem to start with gives you some leverage, some insight, some handle on solving that problem more directly and more efficiently. And that's going to be true for planning.

Special Properties of Planning

- Reducing specific planning problem to general problem of theorem proving is not efficient.
- We will be build a more specialized approach that exploits special properties of planning problems.

We're going to use a very restricted kind of logical representation in building a planner, and we're going to take advantage of some special properties of the planning problem to do it fairly efficiently. So, what special properties of planning can we take advantage of?

Special Properties of Planning

- Reducing specific planning problem to general problem of theorem proving is not efficient.
- We will be build a more specialized approach that exploits special properties of planning problems.
 - Connect action descriptions and state descriptions [focus searching]

One thing we can do is make explicit the connection between action descriptions and state descriptions. If I had an axiom that said "As a result of doing the grab action, we can make holding true." and we were trying to satisfy a goal statement that had "holding" as part of it, then it would make sense to consider doing the "grab" action. So, rather than just trying all the actions and searching blindly through the search space, you could notice, "Goodness, if I need holding to be true at the end, then let me try to find an action that would make it true." So, you can see these connections, and that can really focus your searching.

Special Properties of Planning

- Reducing specific planning problem to general problem of theorem proving is not efficient.
- We will be build a more specialized approach that exploits special properties of planning problems.
 - Connect action descriptions and state descriptions [focus searching]
 - Add actions to a plan in any order

Another thing to notice, that's going to be really important, is that you can add actions to your plan in any order. Maybe you're trying to think about how best to go to Tahiti for spring break; you might think about which airline flight to take first. That maybe seems like the most important thing. You might think about that and really nail it down before you figure out how you're going to get to the airport or what hotel you're going to stay in or a bunch of other things like that. You wouldn't want to have to make your plan for going to Tahiti actually in the order that you're going to execute it, necessarily. Right? If you did that, you might have to consider all the different taxis you could ride in, and that would take you a long time, and then for each taxi, you think about, "Well, then how do I..." You don't want to do that. So, it can be easier to work out a plan from the middle out sometimes, or in different directions, depending on the constraints.

