COSC1125/1127 Artificial Intelligence

School of Computer Science and IT
RMIT University
Semester 1, 2019

Tutorial Sheet 7
Automated Planning

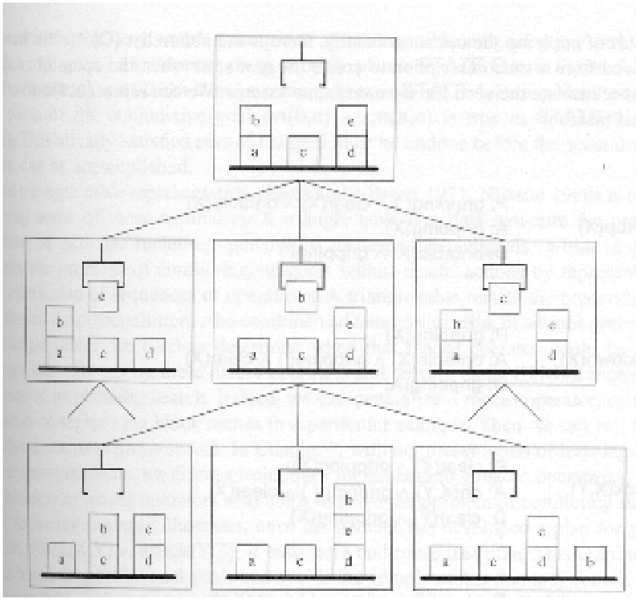
1. Explain informally but precisely:

   - Closed world assumption.
   - The frame problem.
   - How do you define an action using STRIPS (give an example).

Answer

- **Closed world assumption:** the assumption that when a proposition cannot be proven true, it is taken as false (like in a relational database: if the tuple is not in the table, it is not true).
- **The frame problem:** the challenge of representing the effects of action in logic without having to represent explicitly a large number of intuitively obvious non-effects. See this entry in Stanford Encyclopedia of Philosophy.
- How do you define an action using STRIPS (give an example): Example here

2. Consider the following blocks world scenario:



Suppose that we want to represent the blocks world scenario using "logical"[1] causal rules of the form $A \Rightarrow (E \Leftarrow C)$ with the intended meaning that "Action A will have effect E when condition C holds true". For example, $\forall x\ drop(x) \Rightarrow (Broken(x) \Leftarrow Fragile(x))$ states that dropping an object results in the object being broken when the object in question is fragile.

[1] We say "logical" because these are not logical formula per se, but just specification "rules" written with a logical flavour.

(a) Specify the initial state in logical representation, that is, using conjunction of atoms and relying on the close world assumption.

Answer

The initial state of the blocks world above may be represented by the following set of predicates:

$$Ontable(a) \wedge Ontable(c) \wedge Ontable(d) \wedge On(b, a) \wedge$$
$$On(e, d) \wedge Gripping(\cdot) \wedge Clear(b) \wedge Clear(c) \wedge Clear(e)$$

(b) Create the "logical" causal rules for the four operators in the blocks world domain, namely, actions $Pickup(x)$, $Putdown(x)$, $Stack(x, y)$, and $Unstack(x)$.

Answer

There are a number of rules (or operators) that operate on states in order to produce new states. These can be described (as per Luger, Chapter 8) as:

- $\forall x\ Pickup(x) \Rightarrow (Gripping(x) \Leftarrow (Gripping(\cdot) \wedge Clear(x) \wedge Ontable(x)))$
- $\forall x\ Putdown(x) \Rightarrow ((Gripping(\cdot) \wedge Ontable(x) \wedge Clear(x)) \Leftarrow Gripping(x))$
- $\forall x, y\ Stack(x, y) \Rightarrow ((On(x, y) \wedge Gripping(\cdot) \wedge Clear(x)) \Leftarrow (Clear(y) \wedge Gripping(x)))$
- $\forall x, y\, Unstack(x, y) \Rightarrow ((Clear(y) \wedge Gripping(x)) \Leftarrow (On(x, y) \wedge Clear(x) \wedge Gripping(\cdot)))$

Here, the rules take the form $A \Rightarrow (B \Leftarrow C)$. This means that operator A creates new state propertie(s) B when condition(s) C are true.

(c) Provide at least 5 frame axioms required in this domain. What issue can you see here when the domain is complex?

Answer

If we apply $Unstack(e, d)$ to the initial state from Question 2a the new state will be:

$$Ontable(a) \wedge Ontable(c) \wedge On(b, a) \wedge \boldsymbol{Gripping(e)} \wedge$$
$$Clear(b) \wedge Clear(c) \wedge Clear(e) \wedge \boldsymbol{Clear(d)}$$

In the above, we can see as a result of applying the new operator, new predicates have been added to the new state, whereas other predicates such as $On(e, d) \wedge Gripping(\cdot)$ have been deleted from the new state because they are no longer valid. Operatros such as this can also be described using STRIPS.
You will notice that predicates such as $Ontable(a) \wedge Ontable(c) \wedge Ontable(d)$ continue to remain true in the new state. The question here is how we can ensure this is indeed the case as we generate more and more new states. This is where frame axioms can be used.
Frame axioms are rules to tell what predicates describing a state are not changed by rule applications and are thus carried over intact to help describe the new state of the world. An example of a frame axiom is:

$$\forall x, y, z\ Unstack(y, z) \Rightarrow (Ontable(x) \Leftarrow Ontable(x))$$

The above rule says $Ontable$ is not affected by the $Unstack$ operator. By using this frame axiom and the operator $Unstack$, we can tell that when applying $Unstack(e, d)$ to state 1, we should continue to have $Ontable(a) \wedge Ontable(c) \wedge Ontable(d)$ in state 2, and update it with new predicates such as $Gripping(e) \wedge Clear(d)$. Even for a simple blocks world example like this, we need a number of other frame axioms. for example,

$$\forall x, y, z\ Unstack(y, z) \Rightarrow (Ontable(x) \Leftarrow Ontable(x))$$
$$\forall x, y, z\ Unstack(y, z) \Rightarrow (Clear(x) \Leftarrow Clear(x))$$
$$\forall x, y, z\ Pickup(x) \Rightarrow (On(y, z) \Leftarrow On(y, z))$$
$$\forall x, y\ Pickup(x) \Rightarrow (Ontable(y) \Leftarrow Ontable(y))$$
$$\forall x, y, z\ Putdown(x) \Rightarrow (On(y, z) \Leftarrow On(y, z))$$
$$\forall x, y\ Putdown(x) \Rightarrow (Ontable(y) \Leftarrow Ontable(y))$$
$$\ldots$$
etc.

After creating these axioms (which are straightforward and similar to those already created), we should question the complexity cost of adding this number of support axioms to a system in order to maintain a sound inference scheme. This issue can be addressed by using STRIPS, which maintains an *add* and *delete* list, to keep track of the new predicates that should be added to the new states as well as old predicates that should be deleted from the state.

3. Provide the STRIPS representation of all four actions.

Answer

$\underline{Pickup(x)}$:

Precondition:	$Gripping(\cdot), Clear(x), Ontable(x)$
Add List:	$Gripping(x)$
Delete List:	$Ontable(x), Gripping(\cdot)$

$\underline{Putdown(x)}$:

Precondition:	$Gripping(x)$
Add List:	$Ontable(x), Gripping(\cdot), Clear(x)$
Delete List:	$Gripping(x)$

$\underline{Stack(x, y)}$:

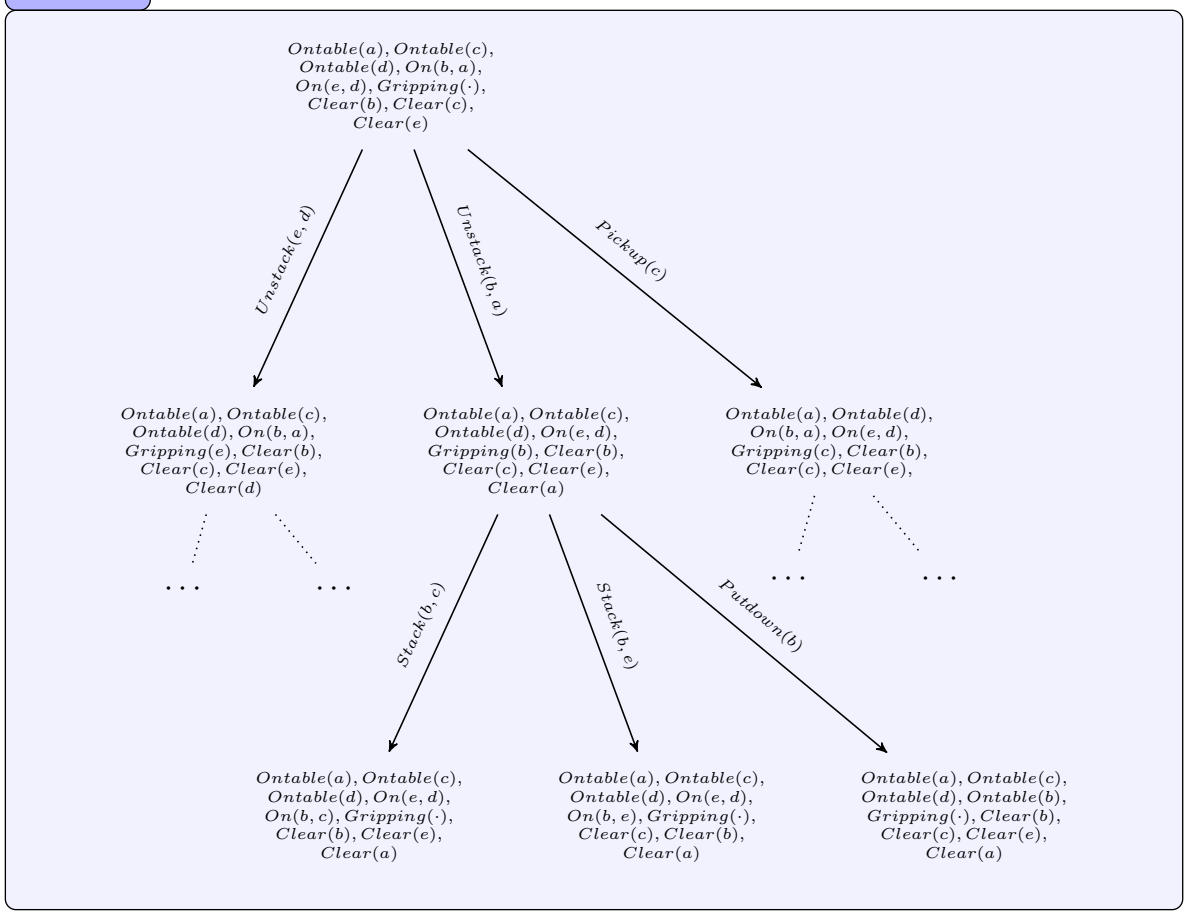
Precondition:	$Clear(y), Gripping(x)$
Add List:	$On(x, y), Gripping(\cdot), Clear(x)$
Delete List:	$Clear(y), Gripping(x)$

$\underline{Unstack(x, y)}$:

Precondition:	$Clear(x), Gripping(\cdot), On(x, y)$
Add List:	$Gripping(x), Clear(y)$
Delete List:	$Gripping(\cdot), On(x, y)$

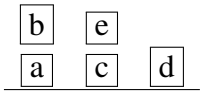
4. Use the operators and frame axioms of the previous question to generate the search state space of the blocks world given in the figure above. The root of the tree should be the logical initial state representation and edges should represent application of ground operators. You do not need to show the whole tree!

Answer



5. Show how forward search (that is, search from the initial state) and backward driven (that is, search from the goal state) can be used to find a plan for the following goal states:

(a) `On(e,c)`

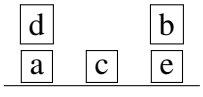


Answer

In a goal driven search, we know the goal state to start with. We can apply different operators to the goal state, and see what new states it can generate. In this case since the goal state is only two steps away from the initial state, you can easily identify that first $Unstack(e, c)$ and then secondly $Stack(e, d)$ will form a path to the initial state, therefore the reversed path would be the plan to achieve the required task. The plan is $Unstack(e, d);\ Stack(e, c)$.

In a data driven search, since we know the initial state for start, we can always try different operators, leading to different new states. From each new state we can try these operators again, eventually we can identify the goal state. Since this is a rather small search space, a Breadth-First-Search would be sufficient to find the path to the goal easily.

(b) `On(d,a)`



Answer

Similar to the above, however the search space is larger than the previous example (it takes at least 8 steps to get to the goal state). It would be more appropriate to use heuristic search (eg, measuring the distance of the current state from the goal state, and also checking if it is a repeated state), in order to guide the search towards the goal state.
Note for b) it can get really complicated in order to find the correct plan. For example we need to check if the preconditions of an operator are met or not, and the problem of having incompatible subgoals can also emerge. This goes beyond the scope of this subject.

6. Suppose that each block has a new property of colour. What needs to be done to your representation/encoding?

Answer

Nothing really, because the property of colour is not required for any preconditions of the operators.

7. Generate a plan to solve the following problem using

   - Data driven search (from initial state).
   - Goal driven search (from goal state).

Answer

A plan such as the following can be generated using goal-driven search:

$$Unstack(c, a)$$
$$Putdown(c)$$
$$Pickup(b)$$
$$Stack(b, c)$$
$$Pickup(a)$$
$$Stack(c, b)$$

You make use the STRIPS planner to see how the plan is generated.