Artificial Intelligence

Lecture 10 – Reasoning about Action II

Outline

- Planning a sequence of actions to achieve a given state
- STRIPS planning
- Planning and the frame problem
- Example: blocks world revisited
- Regression planning with STRIPS
- Least commitment planning

Change

- *Representing how things change* is one of the most important areas in knowledge representation
  - to learn we need to represent what happened in the past
  - to plan we need to represent hypothetical future states
- *Reasoning about change* is one of the most important kinds of reasoning - critical for selecting *actions*

Situation Calculus

- World is represented as a series of situations or 'snapshots'
- Each relation or property of our knowledge representation that can change is extended with an additional situation argument, e.g., e.g., $At(brian, C3, s_0)$, $At(brian, office, s_1)$
- Situations are simply constants - names for a particular state of the world
- All sentences which are true at a given point in time (situation) have the same situation argument

Results of Actions

- An action performed in a given situation s results in a *new situation*
- The function *result(a, s)* is used to denote the situation which results from performing the action *a* in the situation *s*, e.g.:
  - the term result *(pickup(x), s)* denotes the situation which results from picking up the block *x* in situation *s*
- Each action is described by two axioms: a *possibility axiom* and an *effects axiom*

Frame Axioms

- Effect axioms are not sufficient to keep track of which formulas are true in a given situation
- We also need to explicitly state which parts of the world are not changed by an action
- Axioms which describe which parts of the world are *not* changed by an action are called *frame axioms*
- Together, the effect and frame axioms provide a complete description of how the world evolves in response to actions

Planning

- Situation calculus tells us how the world *necessarily* changes as a result of performing an action
- *Planning* involves choosing a sequence of actions which, if executed, will achieve a given state of the world
- May be more than one sequence of actions which achieve the specified state
- Relies on localised representations and may use reasoning (e.g., situation calculus) to infer the consequences of possible actions

Planning in the Real World

- Planning is a hard problem
- Our knowledge of the initial state of the world is often incomplete or incorrect
- The world is continually changing and continues to change while we are planning
- Performing an action doesn't always have the intended effect - actions can fail or just have unpredictable outcomes
- We may make errors executing the plan

Classical Planning

- To make the problem tractable, we make some simplifying assumptions:
- We have *perfect knowledge* of the world, including the location and properties of all the objects in the world
- The world is *static* - i.e., it doesn't change unless we change it
- The world is *deterministic* - we know in advance the effect of performing an action in the world and each action has a single outcome
- Plan execution is *error-free*

Reasoning and Planning

- We can represent the problem in situation calculus and use standard logical inference to find plans
- For example, given a specification of a 'blocks-world' problem in the situation calculus: initial state, goal state, axioms describing the operators
- Given a goal, e.g, *On(a, b)* we can try to prove that there is a state in which, e.g., block *a* is on block *b*: ∃s *On(a, b, s)*
- The sequence of actions resulting in s gives the plan, e.g.,

$$result(put(a, b), result(pickup(a), s_0)))$$

- However this does not necessarily give a concise plan and can be very inefficient

Planning Problems

- Goal(s) to be achieved - complete or partial state description(s)
- Initial state(s) - again these may be complete or partial state descriptions; we may not know everything about the initial state or we may want a plan that works in a range of situations
- Operators specifying the preconditions and effects of actions
- Like situation calculus (and unlike problem-solving) the pre- and postconditions of an action are local or partial
- State descriptions localised rather than global, e.g., *on(blockA, blockB)* vs *<1, 0, 2, 0, 1, 2, 0, 0, 1>* in the eight puzzle

STRIPS

- States are represented as conjunctions of (function-free) ground literals
- Goals are represented by conjunctions of literals, possibly containing existentially quantified variables
- Actions are represented by *operators* which specify
  - the *name* of the action
  - the *precondition* - a conjunction of positive literals specifying what must be true for the action to be applicable
  - the *postcondition* - a conjunction of literals specifying how the situation changes when the operator is applied

STRIPS continued

- For example, an operator which stacks one block on top of another in the blocks world could be specified as

  [Clear(x), Clear(y)] STACK(x, y) [On(x, y), ¬Clear(y)]

- The precondition implicitly refers to the situation, *s*, immediately before the action, and the postcondition implicitly refers to the situation, *s′*, which results from performing the action
- In *s′*, all the positive literals in the postcondition hold, as do all the literals that held at *s*, except for those that are negative literals in the postcondition

Reasoning about Change

- When reasoning about how actions change the world, we need to consider:
  - when an action is *applicable* - the qualification problem
  - what the action *changes* - the ramification problem
  - what the action *does not change* - the representational and inferential frame problems

Change in the Situation Calculus

- In the situation calculus, these three problems are formalised using three sets of axioms:
  - possibility axioms say when an action is applicable
  - effects axioms say what an action changes
  - frame axioms say what an action does not change
- In STRIPS planning, operators take the place of possibility and effects axioms and there are *no frame axioms*

Planning and the Frame Problem

- Planning effectively pushes the frame axioms into the inference procedure
- The planner can assume that anything which is not explicitly listed as a postcondition of an action does *not* change
- This 'solves' the representational and inferential frame problems
- It *does not solve* the qualification problem or the ramification problem
- User must still provide complete descriptions of an action's pre- and postconditions

Example: Blocks World

- The blocks world domain consists of
  - a table, a set of cubic blocks and a robot arm
  - each block is either on the table, stacked on top of another block or being held by the arm
  - the arm can pick up a block and move it to another position either on the table or on top of another block
  - the arm can only pick up one block at a time, so it cannot pick up a block which has another block on top
- The goal is a plan to build one or more stacks of blocks, specified in terms of which blocks are on top of which other blocks

Example: Representing the Blocks World

- Blocks are represented by constants: *a*, *b*, *c*, ... etc.
- States are described using the following predicates:

On(*x*, *y*)	block *x* is on block *y*
OnTable(*x*)	block *x* is on the table
Clear(*x*)	there is no bock on top of block *x*
Holding(*x*)	the arm is holding block *x*
ArmEmpty	the arm is not holding any block

- Note that in STRIPS, the predicates (fluents) do not have a situation argument

Example: Blocks World Operators

[Holding(x), Clear(y)] ***STACK****(x, y) [On(x, y), ArmEmpty,*
¬Holding(x), ¬Clear(y)]

[On(x, y), Clear(x), ArmEmpty] ***UNSTACK****(x, y) [Clear(y),*
Holding(x), ¬On(x, y), ¬ArmEmpty]

[OnTable(x), Clear(x), ArmEmpty] ***PICKUP****(x) [Holding(x),*
¬OnTable(x), ¬ArmEmpty,]

[Holding(x)] ***PUTDOWN****(x) [OnTable(x), ArmEmpty, ¬Holding(x),]*

Example: Blocks World Axioms

- As in the situation calculus, we need some axioms to reason about the effects (ramifications) of actions

$\forall x\ (OnTable(x) \leftrightarrow \neg\exists y On(x, y) \wedge \neg Holding(x))$

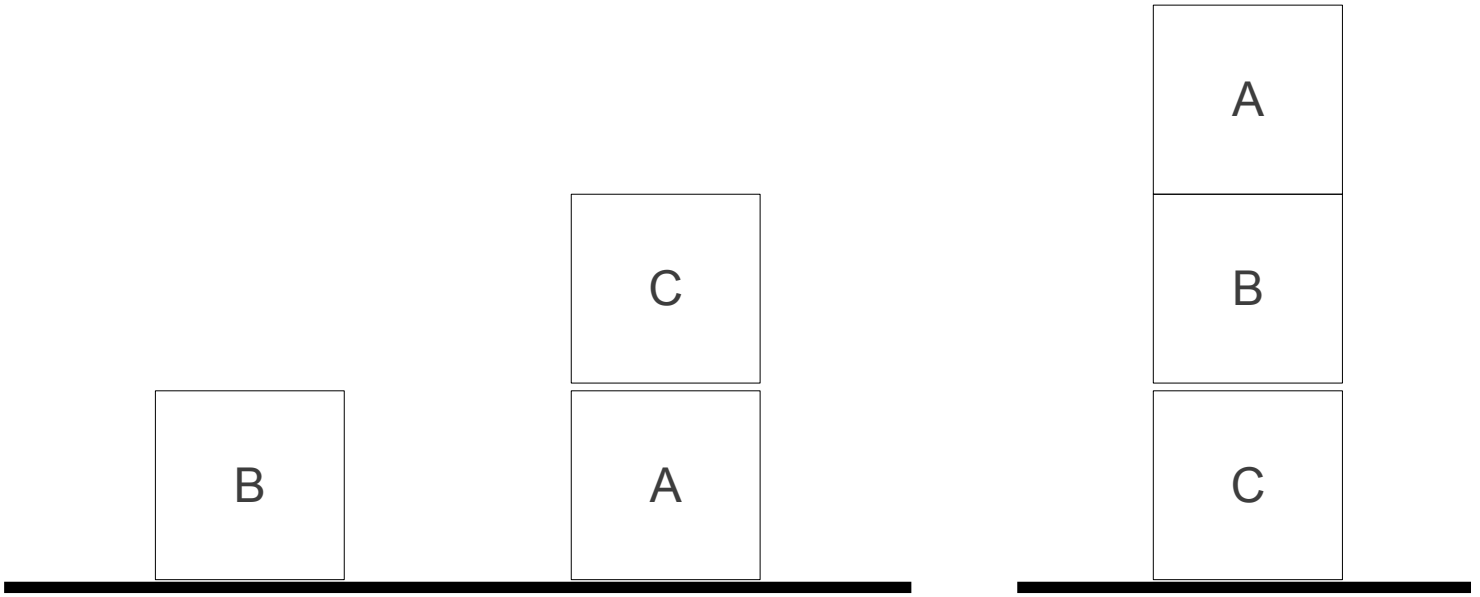
$\forall x\ (\exists y On(x, y) \leftrightarrow \neg OnTable(x) \wedge \neg Holding(x))$

$\forall x\ (Holding(x) \leftrightarrow \neg\exists y On(x, y) \wedge \neg OnTable(x))$

$\forall x\ (Clear(x) \leftrightarrow \neg\exists y\ On(y, x))$

$ArmEmpty \leftrightarrow \neg\exists x Holding(x)$

Example: Blocks World Problem



- Initial state:

  On(c, a) ∧ OnTable(a) ∧ OnTable(b) ∧ ArmEmpty ∧ Clear(b) ∧ Clear(c)

- Goal state: *On(a, b) ∧ On(b, c) ∧ OnTable(c)*

Regression Planning

- One way to solve STRIPS problems is to search backwards from the goal in world (situation) space
- Operator preconditions become *subgoals*—we stop when the operator preconditions are satisfied in the current state
- Resulting plan is a series of instantiated operators which, if applied in the initial state, result in the goal state
- Searching backwards often reduces the branching factor
- In typical problems the goal state has a small number of conjuncts, each of which is made true by a small number of operators, while there are many operators that can be applied in the initial state

Regression Planning in the Blocks World

- For example, we can decompose the blocks world problem into three subgoals: *On(a, b)*, *On(b, c)* and *OnTable(c)*
- The first subgoal, *On(a, b)*, is false in the initial situation, so we look for an operator which makes it true
- In this case, there is only one, ***STACK(a, b)***, which has preconditions: *Holding(a)*, which is false, and *Clear(b)*, which is true
- *Holding(a)* becomes a new subgoal, and we look for an operator to make it true - in this case there are two operators we can choose: ***UNSTACK*** and ***PICKUP***
- and so on …

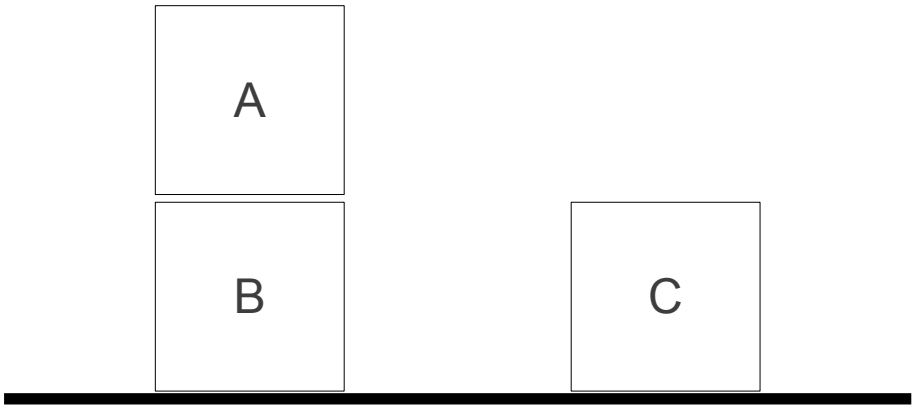
Exercise: STRIPS Planning

- Find a sequence of operator applications which achieves the subgoal *On(a, b)*
- Find a sequence of operator applications which achieves the goal

$$On(a, b) \wedge On(b, c) \wedge OnTable(c)$$

Clobbering

- With conjunctive goals it can be hard to ensure that steps in the plan don't interfere
- When planning to achieve G_1 ^ G_2 the postcondition of an action to achieve G_2 may make G_1 false
- For example, the state after achieving the first subgoal, On(a, b), could be as in (1)
- After achieving the second and third subgoals, *On(b, c)* and *OnTable(c)* the state could be as in (2)



(Figure 1) On(a, b) achieved

(Figure 2) On(b, c) and OnTable (a) achieved

Least Commitment

- Many planners adopt a principle of *least commitment*, which states that choices about plan step ordering and operator variable binding should only be made when necessary
- Reduces backtracking while planning
- Allows the agent to execute actions in parallel if it can
- A planner which returns plans in which some steps are ordered and other steps are unordered is called a partial order planner

Partial Order Planning

- A partial order plan consists of:
  - a set of steps (operator applications)
  - a set of ordering constraints on steps of the form $o_i < o_j$ (step *i* before step *j*)
  - a set of variable binding constraints, $v = x$, where v is a variable in some step, and x is a constant or another variable
  - a set of causal links, $o_i \xrightarrow{c} o_j$, (*i* achieves precondition *c* for *j*), which record the purpose of the step in the plan

Partial Plans

- Most partial order planners search in plan space rather than world (situation) space
- Planner progressively modifies an initial partial plan by adding steps and/or ordering or variable binding constraints
- Initial plan consists of two special operators:
  - *start* has no preconditions and its postcondition adds all the propositions that are true in the initial state
  - *end* has the goal state as its precondition and no effects
- and the ordering constraint *start* < *end*
- A plan is complete if every precondition of every step is achieved by some other step, and consistent if there are no contradictions in the ordering or binding constraints

Other Planning Approaches

- Hierarchical planning - using abstract operators to find an abtract plan which is then refined to the level of individual plan steps
- Planning with resource constraints (time, money etc.)
- Planning under incomplete information - plans contain *conditional operators* which incorporate sensing actions to select the action(s) to perform depending on the state of the world
- Interleaving planning and acting

Summary

- AI planning systems solve a particular kind of reasoning problem by
  - changing the state representation from explicit to attribute-based, allowing partial representations of states
  - changing the operator representation and inference procedure to avoid the need for explicit frame axioms
  - changing the space in which the search is carried out (from world space to plan space)