Administrative

- PS3: Due today
  - But no late penalty until Thursday

- PS4: Distributed this evening
Planning

- What is planning?
  - Find a sequence of actions that achieves a desired result
  - Haven’t we done planning already?

- Planning with TreeSearch
  - Initial state: our, well, initial state
  - Successor function: which actions can we perform?
  - isGoal: Have we achieved the desired state?

Specialized planning algorithms

- Planning with TreeSearch
  - Requires hand-written successors function
  - Very general, but generally slow.

- CSPs can be solved using TreeSearch, but we were able to do much better with specialized methods
  - Generic problem representation
  - Natural heuristics

- Can we do for Planning what we did for CSPs?
Rethinking Planning

- General structure
  - Actions change the *properties* of *objects*
  - First-Order Logic allows us to reason about the properties of objects...

Sketch of an idea

- Let’s say that we have some statements about the blocks world, like
  - On(B,A)
  - Clear(B)
  - On(A,T)
  - ~Clear(A)

- Can we “prove” that we can move block B?

\[ \text{On}(B,A) \land \text{Clear}(B) \Rightarrow \text{On}(B,T) \land \text{Clear}(A) \]

- But adding this to the KB introduces contradictions!
  - E.g., Clear(A) \land ~Clear(A)
What Went Wrong?

- We’re mixing our assertions about the world at different times!
- We need to augment representation with state/situation variables
  - On(B,A,S)
  - Clear(B,S)
  - On(A,T,S)
  - ~Clear(A,S)
- Now, we should be able to infer that there exists a new situation where B can be on the table (and A clear) if we move B from A to T:
  - On(B,A,S) ^ Clear(B,S) => On(B,T,NewSit) ^ Clear(A,NewSit)
- Better yet, create a general rule:
  - ∀x,y,s1. (On(x,y,s1) & Clear(x,s1)) => (∃s2. On(x,T,s2) & Clear(y, s2))
- How do we encode our goal state?
  - Ans(S) is true if S is a goal state

Situation Calculus Example 1

1. On(B,A,S)
2. On(A,T,S)
3. Cl(B,S)
4. ~Clear(A,S)
5. ∀x,y,s5. (On(x,y,s5) ^ Cl(x,y,s5)) => (∃t5. On(x,T,t5) ^ Cl(y,t5))

Convert 5 into CNF:

∀x,y,s5. ¬(On(x,y,s5) ^ Cl(x,y,s5)) v (∃t5. On(x,T,t5) ^ Cl(y,t5))
∀x,y,s5. (¬On(x,y,s5) v ¬Cl(x,y,s5)) v (∃t5. On(x,T,t5) ^ Cl(y,t5))
∀x,y,s5. (¬On(x,y,s5) v ¬Cl(x,y,s5)) v (On(x,T,PoT(x,y,s5)) ^ Cl(y,t5, PoT(x,y,s5)))
5a. ¬On(x5a,y5a,s5a) v ¬Cl(x5a,y5a,s5a) v On(x5a,T,PoT(x5a,y5a,s5a))
5b. ¬On(x5b,y5b,s5b) v ¬Clear(y5b, PoT(x5b,y5b,s5b))
Situation Calculus Example 1

1. On(B,A,S)
2. On(A,T,S)
3. Cl(B,S)
4. ¬Cl(A,S)
5a. ¬On(x5a,y5a,s5a) v ¬Cl(x5a,s5a) v On(x5a,T,PoT(x5a,y5a,s5a))
5b. ¬On(x5b,y5b,s5b) v ¬Cl(x5b,s5b) v Cl(y5b,PoT(x5b,y5b,s5b))

Now, let’s prove ∃s. Cl(A,s) and find the s!

6. ¬Cl(A,s6) v Ans(s6)
7. ¬On(x7A,s7) v ¬Cl(x7A,s7) v Ans(PoT(x7A,A,s7))
   from 6 & 5b, {x5b / A, s5b / PoT(x5b,y5b,s5b)}, standardize apart
8. ¬Cl(B,s) v Ans(PoT(B,A,s))
   from 7 & 1, {x7A / B, s7 / S}
9. Ans(PoT(B,A,s))
   from 8 & 3

Situation Calculus Example 2

1. On(B,A,S)
2. On(A,T,S)
3. Cl(B,S)
4. ¬Cl(A,S)
5a. ¬On(x5a,y5a,s5a) v ¬Cl(x5a,s5a) v On(x5a,T,PoT(x5a,y5a,s5a))
5b. ¬On(x5b,y5b,s5b) v ¬Cl(x5b,s5b) v Cl(y5b,PoT(x5b,y5b,s5b))

Now, let’s prove ∃s. On(A,B,s) and find the s!

6. ¬Cl(A,s6) v Ans(s6)
7. ¬On(x7A,B,s7) v ¬Cl(x7A,B,s7) v Ans(PoT(x7A,B,s7))
   from 6 & 5a, {x6a / A, y6a / B, s7 / PoB(x6a, y6a, s6a)}, standardize apart
8. ¬Cl(B,s) v Ans(PoB(B,A,s))
   from 7 & 1, {x7A / B, s7 / S}
9. Ans(PoB(B,A,s))
   from 8 & 3

? How do we know that A is still on T after we put B on the table in situation S???
Frame Axioms

- Need to say what continues to hold from one situation to another, as well as what stops holding.

- In our example:
  - Moving B from A onto the Table doesn’t change the fact that A is on the Table.
  - In any situation where a block is on something, then in a situation that arises when we move a different block, then the original block is still on whatever it was on.
  - If a block was clear in a situation, and then we move some other block anywhere but on top of it, then it is still clear in the situation after the other block was moved.
  - And so on...

PDDL

- Planning Domain Definition Language (PDDL)
  - Expresses typical frame axioms automatically
  - Database semantics
    - Closed world (fluents are false by default)
    - Two constants (Bob, Mr.Henderson) always refer to different objects
  - Based on STRIPS language (1971) used by Shakey

- Environment
  - fully observable, deterministic, finite, static, discrete

- Objectives
  - conjunctions of goal propositions
PDDL Action Schema

- Example:
  - **ACTION**: Fly(p, from, to)
  - **PRECOND**: At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)
  - **EFFECT**: \neg At(p, from) \land At(p, to)

- In comparison to FOL
  - FOL: Variables (use unification), Predicates, Functions, arbitrary connectives
  - PDDL: Variables (use unification), Predicates, No functions, conjunctions only

- Goals
  - Cannot have a vaccum which wants at least one clean room: Clean(Room1) \lor Clean(Room2)

- Effects
  - Sets values of propositions (overriding earlier values)
  - Implicitly at time+1: no state variables

A Blocks-World Problem

- Initial:
  - On(A,C) \land On(B,Table) \land On(C,Table) \land Clear(A) \land Clear(B)

- Goal:
  - On(B,C)

- Define the clear predicate?
  - Clear(x) \equiv \forall y. \neg On(y,x)
  - But we can’t: not part of the syntax of PDDL
  - Instead, values of Clear predicate are updated by actions
A Blocks-World Problem

Move(\(b,x,y\)) ("move \(b\) from \(x\) to \(y\")

\[
\text{precondition: } \quad \text{On}(b,x) \land \text{Clear}(b) \land \text{Clear}(y)
\]

\[
\text{effect: } \quad \text{On}(b,y) \land \text{Clear}(x) \land \neg \text{On}(b,x) \land \neg \text{Clear}(y)
\]

- Consider goal state: Move(A,C,B)
  - Can be unified with action Move(b,x,y)
    - \(\theta = \{b/A, x/C, y/B\}\)
  - Precondition satisfied?
  - Resulting state
    - \(\neg \text{On}(A,C) \land \text{On}(B,\text{Table}) \land \text{On}(C,\text{Table}) \land \text{Clear}(A) \land \neg \text{Clear}(B) \land \text{On}(A,B) \land \text{Clear}(C)\)

A small problem

Move(\(b,x,y\)) ("move \(b\) from \(x\) to \(y\")

\[
\text{precondition: } \quad \text{On}(b,x) \land \text{Clear}(b) \land \text{Clear}(y)
\]

\[
\text{effect: } \quad \text{On}(b,y) \land \text{Clear}(x) \land \neg \text{On}(b,x) \land \neg \text{Clear}(y)
\]

- What happens if we perform Move(b, x, Table)?
  - The table has a “special” property!
- How do we fix this?
Shakey the Robot

- Natural language interface
- STRIPS-style planner
  - Real-world implementation of blocks-world like problem

Shakey: The Movie

- Intro: 0:00 - 5:30
- STRIPS: 10:20 – 14:30
Searching for Plans

- Given a plan (sequence of actions) and an initial state, can test whether plan achieves goal

Q: How to generate solution plans?

A: search (as always...)

Forward State-Space Search

- Also called progression planning

- Planning as state space search:
  - Represent states by sets of positive ground literals
    - Literals not appearing are false or don’t matter
    - Initial state: given by planning problem
  - Action applicable in a state iff preconditions satisfied
  - Successor states generated by adding positive effect literals and deleting negative effect literals
  - Goal test succeeds iff state satisfies goal sentence
  - Step cost = 1 (typically)
Forward Search: Complexity

- In the absence of function symbols, the state space of a planning problem is finite
  - Therefore any complete graph-search algorithm will be a complete planning algorithm

- But will it be efficient?
  - Many irrelevant actions
    - all applicable actions are considered in each state
  - What is branching factor for blocks world with $N$ blocks?
  - Need good heuristic functions

Backward State-Space Search

- Also called regression planning
- Generates predecessors starting from goal state
  - Find action $A$ whose effect unifies with goal (or part)
  - New “goal” is set of conditions for this action to be applicable
  - Computing these conditions is called regressing the goal through the action.
  - Delete positive effects of $A$ that appear in goal
  - Add precondition literals of $A$

- Advantage: need only consider relevant actions
- Disadvantage: dealing with interactions among goal propositions
Regression Example

- Goal: On(B,C)
  - Choose action:
    - Move(B,Table,C)
      - Achieves On(B,C), has preconditions Clear(B), Clear(C), On(B,Table)
  - Choose action:
    - MoveToTable(A,C)
      - Achieves Clear(C), has preconditions Clear(A), On(A,C)
  - Remaining conditions satisfied in initial state

Next Time

- Planning Graphs and real-world planning