Planning

- Goal-based agent
- Determine a sequence of actions to achieve a goal
Planning Applications
Planning Approaches

- Search was one approach
  - Did not reason about actions (black boxes)
  - Inefficient when many actions

- Logic is another approach
  - Reasoning about change over time cumbersome (e.g., frame axioms)
  - Inefficient due to many applicable rules

- Can we combine the best of both?
Example: Blocks World

Start State

Goal State
Example: Blocks World

Init(On(A, Table) \land On(B, Table) \land On(C, A) \land Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C))

Goal(On(A, B) \land On(B, C))

Action(Move(b, x, y),
PRECOND: \ On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land (b \neq x) \land (b \neq y) \land (x \neq y),
EFFECT: \ On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y))

Action(MoveToTable(b, x),
PRECOND: \ On(b, x) \land Clear(b) \land Block(b) \land (b \neq x)
EFFECT: \ On(b, Table) \land Clear(x) \land \neg On(b, x))

[MoveToTable(C, A), Move(B, Table, C), Move(A, Table, B)]
Definitions

- Planning Domain Definition Language (PDDL)
- Initial state
- Actions
- Results
- Goal test
Conjunction of ground functionless atoms (i.e., positive ground literals)
- $\text{At(Robot1, Room1)} \land \text{At(Robot2, Room3)}$
- $\text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Bananas)} \land \ldots$
- $\text{At(Home)} \land \text{IsAt(Umbrella, Home)} \land \text{CanBeCarried(Umbrella)} \land \text{IsUmbrella(Umbrella)} \land \text{HandEmpty} \land \text{Dry}$

The following are not okay as part of a state
- $\neg \text{At(Home)}$ (a negative literal)
- $\text{IsAt(x, y)}$ (not ground)
- $\text{IsAt(Father(Fred), Home)}$ (uses a function symbol)

Closed-world assumption
- If don’t mention $\text{At(Home)}$, then assume $\neg \text{At(Home)}$
Goal
- Conjunction of literals (positive or negative, possibly with variables)
- Variables are existentially quantified
- A partially specified state

Examples
- \(\text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Bananas)} \land \text{Rich} \land \text{Famous}\)
- \(\text{At(x)} \land \text{Sells(x, Milk)}\) (be at a store that sells milk)

A state \(s\) satisfies goal \(g\) if \(s\) contains (unifies with) all the literals of \(g\)
- \(\text{At(Home)} \land \text{Have(Milk)} \land \text{Have(Bananas)} \land \text{Rich} \land \text{Famous}\) satisfies \(\text{At(x)} \land \text{Rich} \land \text{Famous}\)
PDDL: Actions

- Actions are modeled as state transformations
- Actions are described by a set of action schemas that implicitly define ACTIONS(s) and RESULT(s,a)
- Description of an action should only mention what changes (address frame problem)
- Action schema:

\[
\text{Action(Fly(p, from, to)}
\]
\[
\text{PRECOND: At(p, from) } \land \text{ Plane(p) } \land \text{ Airport(from) } \land \text{ Airport(to)}
\]
\[
\text{EFFECT: } \neg \text{At(p, from) } \land \text{ At(p, to)}
\]
PDDL: Actions

- **Precondition**: What must be true for the action to be applicable
- **Effect**: Changes to the state as a result of taking the action
- Conjunction of literals (positive or negative)

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))

Action(Fly(P1, SEA, LAX)) (ground action)
PRECOND: At(P1, SEA) \land Plane(P1) \land Airport(SEA) \land Airport(LAX)
EFFECT: \neg At(P1, SEA) \land At(P1, LAX)
PDDL: Actions

- Action a can be executed in state s if s entails the precondition of a
  - \((a \in \text{ACTIONS}(s)) \iff s \models \text{PRECOND}(a)\)
  - where any variables in a are universally quantified
- For example:

\[
\forall p, \text{from}, \text{to} \ (\text{Fly}(p, \text{from}, \text{to}) \in \text{ACTIONS}(s)) \iff s \models (\text{At}(p, \text{from}) \land \text{Plane}(p) \land \text{Airport}(\text{from}) \land \text{Airport}(\text{to}))
\]
The result of executing action a in state s is defined as a state s’
State s’ contains the literals of s minus negative literals in EFFECT plus positive literals in EFFECT
Negated literals in EFFECT called delete list or DEL(a)
Positive literals in EFFECT called add list or ADD(a)
RESULT(s, a) = (s − DEL(a)) ∪ ADD(a)
PDDL: Actions

- For action schemas any variable in EFFECT must also appear in PRECOND
  - RESULT(s, a) will therefore have only ground atoms
- Time is implicit in action schemas
  - Precondition refers to time t
  - Effect refers to time t+1
- Action schema can represent a number of different actions
  - Fly(Plane1, LAX, JFK)
  - Fly(Plane3, SEA, LAX)
Example: Air Cargo Transport

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ Cargo(C1) ∧ Cargo(C2) ∧ Plane(P1) ∧ Plane(P2) ∧ Airport(JFK) ∧ Airport(SFO))

Goal(At(C1, JFK) ∧ At(C2, SFO))

Action(Load(c, p, a),
    PRECOND:  At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    EFFECT:  ¬At(c, a) ∧ In(c, p))

Action(Unload(c, p, a),
    PRECOND:  In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    EFFECT:  At(c, a) ∧ ¬In(c, p))

Action(Fly(p, from, to),
    PRECOND:  At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
    EFFECT:  ¬At(p, from) ∧ At(p, to))

[Load(C1, P1, SFO), Fly(P1, SFO, JFK), Unload(C1, P1, JFK), Load(C2, P2, JFK), Fly(P2, JFK, SFO), Unload(C2, P2, SFO)]
Example: Car Navigation
**Complexity of Planning**

- **PlanSAT**: Does a plan exist
- **Bounded PlanSAT**: Does a plan exist with solution length \(< k\)
- Both are decidable
- Both are in complexity class $\text{PSPACE} \supset \text{NP}$
  - Can be solved by deterministic Turing machine using only polynomial space
- If disallow negative preconditions and effects, then PlanSAT in complexity class $\text{P}$
- Worst-case analysis
Planning in First-Order Logic

- Situation calculus
- The initial state is a situation
- A situation $\text{Result}(s,a)$ is the result of executing action $a$ in situation $s$
Planning in First-Order Logic

- Planning with situation calculus
- General approach, but inefficient
- Difficult, because FOL inference is difficult
Solving Planning Problems

- State-space search approach
- Determine a sequence of actions that when applied to the initial state yields a state which satisfies the goal
- Solve planning problems with any of the previous search algorithms
- Search tree usually large
  - Many instantiations of applicable actions
Planning As State–Space Search

- Forward (progression) state–space search
  - Start at the initial state and apply actions until the current state satisfies the goal
Problems

- Prone to exploring irrelevant actions
  - Example: Goal is Own(isbn), have action Buy(isbn), 10 billion books

- Planning problems often have large state spaces
  - Example: Cargo at 10 airports, each with 5 airplanes and 20 pieces of cargo
    - Goal: Move 20 pieces from airport A to airport B (41 steps)
    - Average actions applicable to a state is 2000
    - Search graph has $2000^{41}$ nodes

Need accurate heuristics

- Many real-world applications have strong heuristics
Planning As State–Space Search

- Backward (regression) relevant–states search
  - Start at goal and apply actions backward until we find a sequence of steps that reaches initial state
  - Only considers actions relevant to the goal (or current state)
Backward (Regression) Relevant-States Search (cont’d.)

- Works only when we know how to regress from a state description to the predecessor state description.

- Given a ground goal description g and a ground action a, the regression from g over a gives a state g’ defined by
  \[ g’ = (g - \text{ADD}(a)) \cup \text{PRECOND}(a) \]

Note: DEL(a) does not appear in the above, because we don’t know whether or not they were true before, so there’s nothing to be said about them.
We need to also deal with partially uninstantiated actions and states, not just ground ones

- For example: Goal is \( \text{At}(C2, \text{SFO}) \)
- Suggests action \( \text{Unload}(C2, p', \text{SFO}) \)

\[
\begin{align*}
\text{Action} (\text{Unload}(C2, p', \text{SFO}), \\
\text{PRECOND: } & \text{In}(C2, p') \land \text{At}(p', \text{SFO}) \land \text{Cargo}(C2) \land \text{Plane}(p') \land \\
& \text{Airport}(\text{SFO}), \\
\text{EFFECT: } & \text{At}(C2, \text{SFO}) \land \neg \text{In}(C2, p')
\end{align*}
\]

- The regressed state description is
  - \( g' = \text{In}(C2, p') \land \text{At}(p', \text{SFO}) \land \text{Cargo}(C2) \land \text{Plane}(p') \land \text{Airport}(\text{SFO}) \)
Backward (Regression) Relevant-States Search (cont’d.)

- Want actions that could be the last step in a plan leading up to the current goal
- At least one of the action’s effects (either positive or negative) must unify with an element of the goal
- The action must not have any effect (positive or negative) that negates an element of the goal
  - For example, goal is \( A \land B \land C \) and an action has the effect \( A \land B \land \neg C \).
For example

- Goal is `Own(0136042597)`
- An initial state with 10 billion ISBNs
- Single action scheme
- \( A = \text{Action}(\text{Buy}(i), \text{PRECOND: ISBN}(i), \text{EFFECT: Own}(i)) \)
- Unify goal `Own(0136042597)` with (standardized) effect `Own(i')`, producing \( \theta = \{i'/0136042597\} \)
- Regress over action `SUBST(\theta, A')` to produce the predecessor state description `ISBN(0136042597)` (which is part of the initial state)
Efficient planning (forward or backward) requires good heuristics

Estimate solution length

- Ignore some or all preconditions

\[
\text{Action}(\text{Fly}(p, \text{from}, \text{to}), \\
\text{PRECOND: } \neg \text{At}(p, \text{from}) \land \text{Plane}(p) \land \text{Airport}(\text{from}) \land \text{Airport}(\text{to}) \\
\text{EFFECT: } \neg \text{At}(p, \text{from}) \land \text{At}(p, \text{to}))
\]

- Ignore delete list

\[
\text{Action}(\text{Fly}(p, \text{from}, \text{to}), \\
\text{PRECOND: } \text{At}(p, \text{from}) \land \text{Plane}(p) \land \text{Airport}(\text{from}) \land \text{Airport}(\text{to}) \\
\text{EFFECT: } \neg \neg \text{At}(p, \text{from}) \land \text{At}(p, \text{to}))
\]
Estimate solution length
  ◦ Use state abstraction

  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))

  ◦ Assume subgoals independent
    • On(A,B) \land On(B,C)
State of the Art Planning

- International Planning Competition (IPC)
  - [http://ipc.icaps-conference.org](http://ipc.icaps-conference.org)

- Fast-Downward
  - Forward progression planner
  - Focus on good heuristics
  - Supports full PDDL
  - [http://www.fast-downward.org](http://www.fast-downward.org)
Summary: Planning

- Planning combines search and logic
- State-space search can operate in the forward direction (progression) or the backward direction (regression)
- State of the art approaches use combination of techniques
- Many real-world applications: mission planning, scheduling, navigation
"ADD TWO EGGS AND STIR."  
RIGHT.

THE RECIPE SAYS IT MAKES TWENTY PANCAKES, SO WE'LL EACH GET TEN.

NAH, THAT'S TOO MUCH TROUBLE.

WE'LL JUST MAKE ONE BIG PANCAKE AND CUT IT IN HALF.