CSC520 - Artificial Intelligence

Lecture 15

Dr. Scott N. Gerard

North Carolina State University

Mar 6, 2025

Agenda

- Classical Planning and PDDL
- PDDL State, Action Schema, Domain and Problem
- PDDL Problem Examples
- Algorithms and Heuristics for Planning

Classical Planning and PDDL

- Planning means finding a sequence of actions to accomplish a goal
- Problem solving agent can compute a plan
  - States, actions and goals are black boxes
  - Require domain-specific heuristics
- Knowledge-based agent can explicitly represent and reason with states and actions using a logical language
- Planning agent combines the ideas from problem solving agent and knowledge-based agents

Classical Planning

- Assumes a discrete, deterministic, static, and fully observable environment
- Planning domain definition language (PDDL) is a language for representing a planning problem
  - Initial state
  - Actions with preconditions and effects
  - Goal test

PDDL State

- Conjunction of ground and functionless fluents
  - Ground means having no variables
  - Fluent means an aspect of world that changes over time
- Follows database semantics
  - Unique-names assumption: Every constant symbol is a distinct object
  - Closed world assumption: Atomic sentences not known are false
  - Domain closure assumption: A model contains no more objects than the constant symbols

PDDL Action Schema

- Represents a family of ground actions
- Action schema contains the following
  - Action name
  - List of variables used in the schema
  - Preconditions of the action which is a conjunction of positive or negative fluents
  - Effects of the action which can be divided into add and delete lists

PDDL Action Schema Example

- Action schema

$$Action(Fly(p, from, to),$$
$$\text{PRECOND}: At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$$
$$\text{EFFECT}: \neg At(p, from) \wedge At(p, to))$$

- A ground action created by assigning constants to variables

$$Action(Fly(P_1, SFO, JFK),$$
$$\text{PRECOND}: At(P_1, SFO) \wedge Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK)$$
$$\text{EFFECT}: \neg At(P_1, SFO) \wedge At(P_1, JFK))$$

PDDL Action Precondition

- A ground action is *applicable* in a state if that state entails the action's precondition
  $(a \in Actions(s)) \Leftrightarrow s \models \text{PRECOND}(a)$
- Every positive literal in the precondition is in the state
- Every negative literal in the precondition is not in the state
- Example:

$$\begin{aligned}&Action(Fly(P_1, SFO, JFK),\\ &\quad \text{PRECOND}: At(P_1, SFO) \land Plane(P_1) \land Airport(SFO) \land Airport(JFK)\\ &\quad \text{EFFECT}: \neg At(P_1, SFO) \land At(P_1, JFK))\end{aligned}$$

$s = At(P_1, SFO) \land Plane(P_1) \land Airport(SFO) \land Airport(JFK) \land Plane(P_2) \land At(P_2, JFK)$

PDDL Action Effect

- Add list is the list of positive literals in the action effect
- Delete list is the list of negative literals in the action effect
- $s' = \text{RESULT}(s, a) = (s - \text{DEL}(a)) \cup \text{ADD}(a)$
- Example:

$$\begin{aligned}&Action(Fly(P_1, SFO, JFK),\\ &\quad \text{PRECOND}: At(P_1, SFO) \wedge Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK)\\ &\quad \text{EFFECT}: \neg At(P_1, SFO) \wedge At(P_1, JFK))\end{aligned}$$

$s = At(P_1, SFO) \wedge Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK) \wedge$
$Plane(P_2) \wedge At(P_2, JFK)$
$s' = Plane(P_1) \wedge Airport(SFO) \wedge Airport(JFK) \wedge Plane(P_2) \wedge$
$At(P_2, JFK) \wedge At(P_1, JFK)$

Planning Domain and Problem

- Planning *domain* is a set of action schemas
- Planning *problem* contains a domain with an initial state and a goal
- Initial state is a conjunction of ground fluents with closed world assumption
  - E.g., $Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO))$
- Goal is a conjunction of literals
  - Literals can be positive or negative
  - Can have variables
  - E.g., $Goal(At(C_1, SFO) \wedge \neg At(C_2, SFO) \wedge At(p, SFO))$

Air Cargo Transportation Problem in PDDL

$$
\begin{aligned}
&\mathit{Init}(\mathit{At}(C_1, \mathit{SFO}) \land \mathit{At}(C_2, \mathit{JFK}) \land \mathit{At}(P_1, \mathit{SFO}) \land \mathit{At}(P_2, \mathit{JFK}) \\
&\quad \land \mathit{Cargo}(C_1) \land \mathit{Cargo}(C_2) \land \mathit{Plane}(P_1) \land \mathit{Plane}(P_2) \\
&\quad \land \mathit{Airport}(\mathit{JFK}) \land \mathit{Airport}(\mathit{SFO})) \\
&\mathit{Goal}(\mathit{At}(C_1, \mathit{JFK}) \land \mathit{At}(C_2, \mathit{SFO})) \\
&\mathit{Action}(\mathit{Load}(c, p, a), \\
&\quad \text{PRECOND: } \mathit{At}(c, a) \land \mathit{At}(p, a) \land \mathit{Cargo}(c) \land \mathit{Plane}(p) \land \mathit{Airport}(a) \\
&\quad \text{EFFECT: } \neg \mathit{At}(c, a) \land \mathit{In}(c, p)) \\
&\mathit{Action}(\mathit{Unload}(c, p, a), \\
&\quad \text{PRECOND: } \mathit{In}(c, p) \land \mathit{At}(p, a) \land \mathit{Cargo}(c) \land \mathit{Plane}(p) \land \mathit{Airport}(a) \\
&\quad \text{EFFECT: } \mathit{At}(c, a) \land \neg \mathit{In}(c, p)) \\
&\mathit{Action}(\mathit{Fly}(p, \mathit{from}, \mathit{to}), \\
&\quad \text{PRECOND: } \mathit{At}(p, \mathit{from}) \land \mathit{Plane}(p) \land \mathit{Airport}(\mathit{from}) \land \mathit{Airport}(\mathit{to}) \\
&\quad \text{EFFECT: } \neg \mathit{At}(p, \mathit{from}) \land \mathit{At}(p, \mathit{to}))
\end{aligned}
$$

Air Cargo Transportation Problem Solution

Below are some possible plans.

$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$
$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)]$

$[Load(C_1, P_1, SFO),\ Load(C_2, P_2, JFK),$
$Fly(P_1, SFO, JFK),\ Fly(P_2, JFK, SFO),$
$Unload(C_1, P_1, JFK),\ Unload(C_2, P_2, SFO)]$

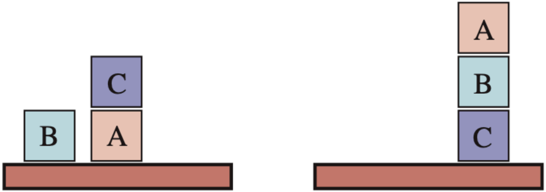
$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$
$Load(C_2, P_1, JFK), Fly(P_1, JFK, SFO), Unload(C_2, P_1, SFO)]$

$[Fly(P_1, SFO, JFK), Fly(P_1, JFK, SFO), Fly(P_1, SFO, JFK),$
$Fly(P_1, JFK, SFO), Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK),$
$Unload(C_1, P_1, JFK), Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO),$
$Unload(C_2, P_2, SFO)]$

Spare Tire Problem in PDDL

$Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(obj, loc),$
  PRECOND: $At(obj, loc)$
  EFFECT: $\neg At(obj, loc) \land At(obj, Ground))$
$Action(PutOn(t, Axle),$
  PRECOND: $Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle) \land \neg At(Spare, Axle)$
  EFFECT: $\neg At(t, Ground) \land At(t, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
    $\land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))$

Blocks-world Problem in PDDL



Start State

Goal State

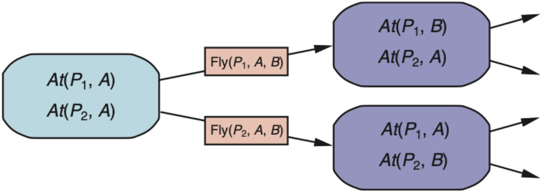
$$
\begin{aligned}
&Init(On(A, Table) \land On(B, Table) \land On(C, A) \\
&\quad \land Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C) \land Clear(Table)) \\
&Goal(On(A, B) \land On(B, C)) \\
&Action(Move(b, x, y), \\
&\quad \text{PRECOND: } On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land \\
&\qquad (b \neq x) \land (b \neq y) \land (x \neq y), \\
&\quad \text{EFFECT: } On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y)) \\
&Action(MoveToTable(b, x), \\
&\quad \text{PRECOND: } On(b, x) \land Clear(b) \land Block(b) \land Block(x), \\
&\quad \text{EFFECT: } On(b, Table) \land Clear(x) \land \neg On(b, x))
\end{aligned}
$$

Algorithms for Planning

- Use any of the state-space search algorithms
- Forward state-space search (Progression planners)
  - Start from the initial state, apply actions until goal state is reached
- Backward state-space search (Regression planners)
  - Start from the goal state, apply actions backward until initial state is reached

Forward State-space Search: Progression Planning

- From a state, apply all *applicable* actions until goal state is reached
- New state is obtained by adding positive literals and deleting the negative literals
  $s' = (s - \text{DEL}(a)) \cup \text{ADD}(a)$
- State space size may be too big for many problems



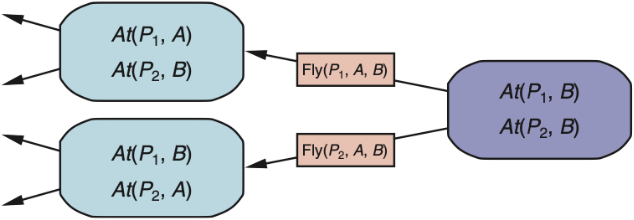
$Action(Fly(p, from, to),$
$\quad \text{PRECOND}: At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
$\quad \text{EFFECT}: \neg At(p, from) \wedge At(p, to))$

Backward State-space Search: Regression Planning

- Start at the goal state
- Apply the actions backward until we reach the initial state
- At each step, consider *relevant* actions
  - A *relevant* action's effect unifies with one of the goal literals and does not negates any goal literal
  - E.g., Consider a goal: $\neg Poor \wedge Famous$
    An action with effect *Famous* is relevant
    An action with effect $Poor \wedge Famous$ is not relevant
- Regression from goal g over action a gives a previous goal g'
  $\textsc{pos}(g') = (\textsc{pos}(g) - \textsc{add}(a)) \cup \textsc{pos}(Precond(a))$
  $\textsc{neg}(g') = (\textsc{neg}(g) - \textsc{del}(a)) \cup \textsc{neg}(Precond(a))$
- Benefits over forward search
  - Much lower branching factor
  - Only relevant actions are considered

Backward State-space Search: Regression Planning

- Regression from goal g over action a gives a previous goal g'
  $\textsc{pos}(g') = (\textsc{pos}(g) - \textsc{add}(a)) \cup \textsc{pos}(Precond(a))$
  $\textsc{neg}(g') = (\textsc{neg}(g) - \textsc{del}(a)) \cup \textsc{neg}(Precond(a))$
- Multiple relevant actions are OR children



$Action(Fly(p, from, to),$
$\quad \textsc{precond}: At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
$\quad \textsc{effect}: \neg At(p, from) \wedge At(p, to))$

Heuristics for Planning

- Need an admissible heuristic for using A^* search
- Recall that an admissible heuristic can be derived by defining a relaxed problem
- Domain-independent heuristics are possible due to factored state representation

Heuristics for Planning

- Ignore-preconditions drops all preconditions from actions
  - Every action becomes applicable in every state
  - Number of steps required is roughly equal to the number of literals in the goal
- Ignore-delete-lists removes all negated literals from action effects
  - No action will ever undo progress made toward the goal
- State abstraction heuristic groups multiple states together
  - State space has fewer states

Class Exercise

Find solution to the spare tire problem specified below.

$$
\begin{aligned}
&Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk))\\
&Goal(At(Spare, Axle))\\
&Action(Remove(obj, loc),\\
&\quad \text{PRECOND: } At(obj, loc)\\
&\quad \text{EFFECT: } \neg At(obj, loc) \land At(obj, Ground))\\
&Action(PutOn(t, Axle),\\
&\quad \text{PRECOND: } Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle) \land \neg At(Spare, Axle)\\
&\quad \text{EFFECT: } \neg At(t, Ground) \land At(t, Axle))\\
&Action(LeaveOvernight,\\
&\quad \text{PRECOND: }\\
&\quad \text{EFFECT: } \neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)\\
&\qquad\qquad \land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))
\end{aligned}
$$

Class Exercise

Action	State
	$state_0$ (initial)
$action_1$	$state_1$
$action_n$	$state_n$ (goal)