

Planning

Chapter 11

AI – A Modern Approach.

Lecturer:

Srividhya Rajendran

Website for slides download:

<http://cseweb.uta.edu/~srajendr>

Introduction

□ What is Planning?

- The task of coming up with a sequence of actions to achieve a given goal.

□ Applications:

- design and manufacturing
- military operations
- games
- space exploration

□ Can Problem Solving agents Do Planning?

- **YES**

Introduction

- Drawbacks of problem-solving agents
 - Scaling up to solve complex real planning problem.
 - Irrelevant actions.
 - Difficulty finding good heuristic function.
 - Cannot take advantage of problem decomposition.

What does this chapter cover?

- A new constrained language for representing planning problems.
- How the previously introduced search algorithms can use these representations.
- Planning algorithms that take advantage of the representation of the problem.

Classical Planning Environment

- Fully observable
- Deterministic
- Finite
- Static
- Discrete

STRIPS Language for Planning

- STRIPS (Stanford Research Institute Problem Solver).
- Uses representations that take advantage of logical structure of the problem.
- Representations like: States, Actions and Goals.

States

- Conjunction of positive literals.
- Literals should be ground and function free.
- Closed World Assumption:
Conditions not mentioned in state description are assumed false.

States

- Examples of State descriptions

Correct	Incorrect
At(Agent, Home)	At(x, y)
On(BlockA, BlockB)	On(Top(BlockA), BlockB)

- What is wrong in incorrect state descriptions?

Goals

- Conjunction of positive ground literals.
- Partially specified state.
- A state "s" satisfies a goal "g" if "s" contains all the atoms in "g".

Goals

□ Example:

■ Goal description:

Have (Banana) \wedge At (Agent, Home)

■ State description:

Have (Banana) \wedge At (Agent, Home) \wedge Drink (Agent, Coffee)

□ Here state description satisfies the goal as it contains all atoms in goal.

Actions

- Composed of 3 parts:
 - Action Name and Parameter List
 - Precondition
 - Representation- Conjunction of positive ground literals
 - Implies- All the conditions that must be true for this action to be executed.
 - Effect
 - Representation- Conjunction of function free literals
 - Describes- how the execution of given action changes the state.

Rules- Any variables in precondition and effect must appear in action parameter list.

Actions

□ Action Schema

- Op (ACTION: Go (there, here) ,
PRECOND: At (here) \wedge Path (here, there)
EFFECT: At (there) \wedge \neg At (here))

Semantics

Agent wants to go from Home to Supermarket

-Current State

**At (Home) \wedge Path (Home, Supermarket) \wedge
Path (Supermarket, Home)**

-Check to see if the action is applicable in given state

Precondition: At (here) \wedge Path (here, there)

by substituting {here/Home, there/Supermarket} we get see precondition is satisfied.

Semantics

- Take Action `Go(Home, Supermarket)`
- New State

`At(Supermarket) ∧ Path(Home, Supermarket) ∧ Path(Supermarket, Home)`

- The result of executing action `a` on a state `s` is `s'` that is same as `s` except that any positive literal `P` in effect of action is added to `s'` and any negative literal `¬P` is removed from `s'`.

Diagrammatic Representation

$\text{At}(\text{Home}) \wedge \text{Path}(\text{Home}, \text{Supermarket}) \wedge$
 $\text{Path}(\text{Supermarket}, \text{Home})$

$\text{Go}(\text{Home}, \text{Supermarket})$

$\text{At}(\text{Supermarket}) \wedge \text{Path}(\text{Home}, \text{Supermarket}) \wedge$
 $\text{Path}(\text{Supermarket}, \text{Home})$

Limitations of STRIPs

Representation

- Only positive & function free literals in states
- Closed World assumption
- Only conjunction of ground literals in goals.
- Effects are conjunction.
- No support for equality or types.

RESULT- Not expressive enough for some real world domains

Extension to STRIPS

- ADL (**A**ction **D**ependent **L**anguage)
 - Uses + and - ive literals in states.
 - Open World Assumption.
 - Effect $P \wedge \neg Q$ means add P and $\neg Q$ and delete $\neg P$ and Q .
 - Quantified literals in goals. Goals allow conjunction and disjunction.
 - Conditional effects are allowed.
 - Support for equality and types

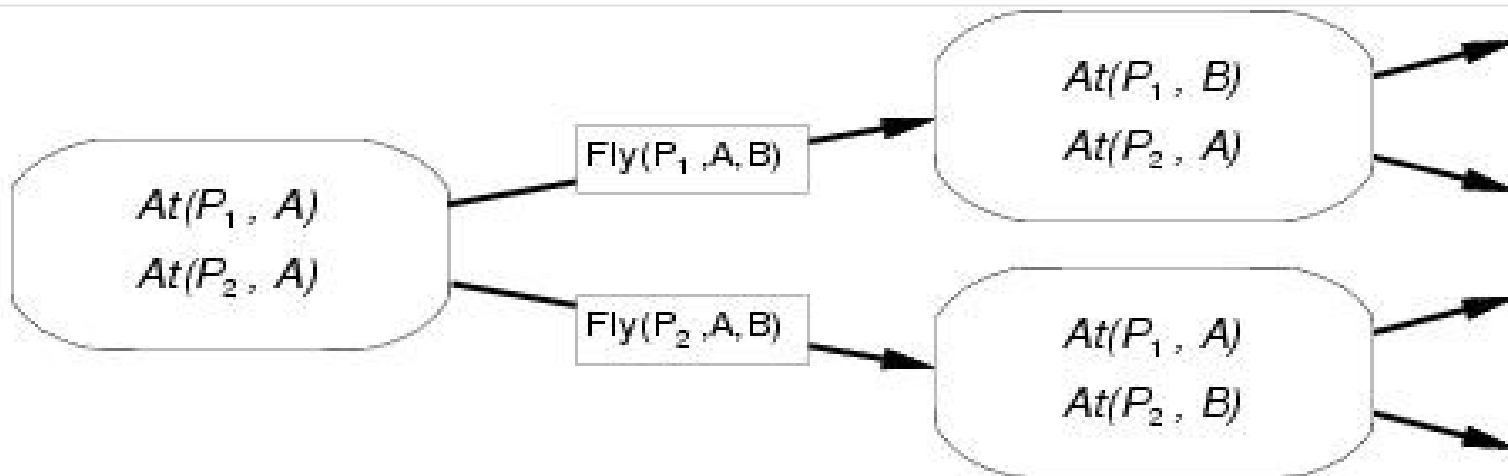
Standardization - **Planning domain definition language (PDDL)**

Planning with State-Space Search

- Planner that searches space of situations in the world.
- A path through this space from the initial state to goal state is Plan.
- Types:
 - Progression algorithm
 - Regression algorithm

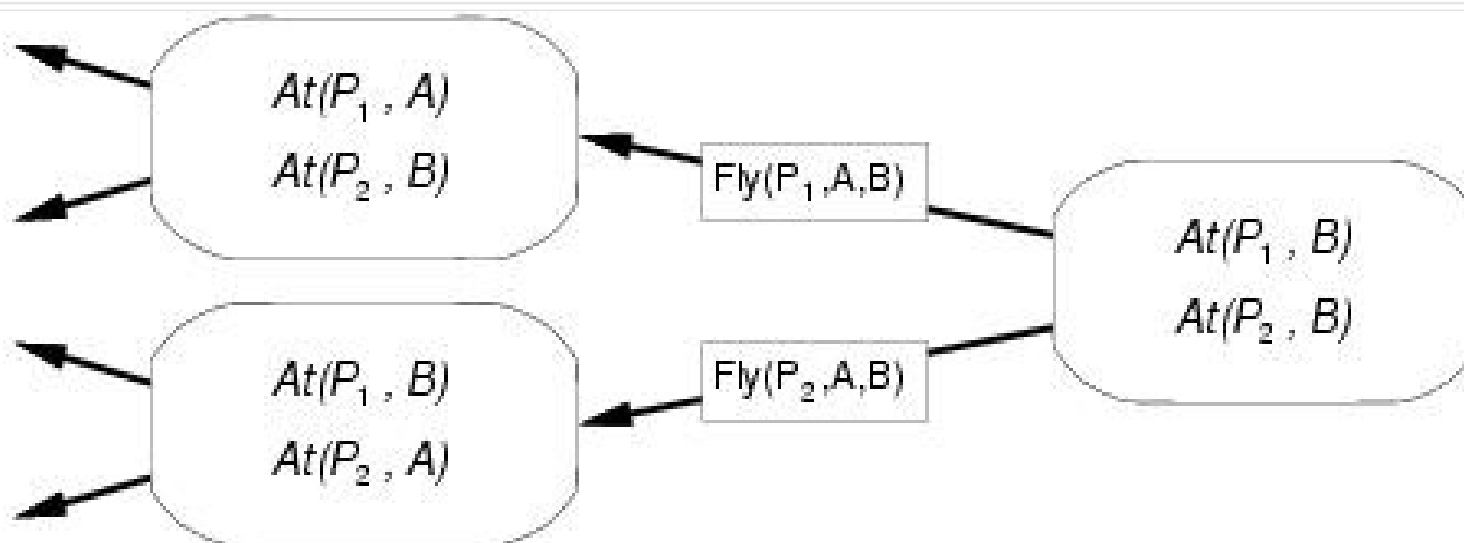
Progression Algorithm

- forward state-space search.
- Considers the effect of all possible actions in a given state.



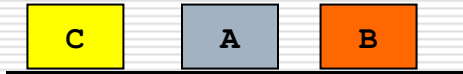
Regression Algorithm

- backward state-space search
- To achieve a goal, what must have been true in the previous state.

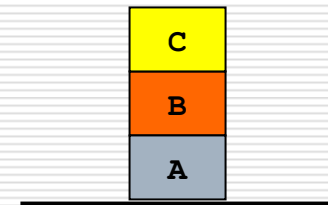


Blocks World Example

Initial State



Goal State



Init (On (A, Table) ^ On (B, Table) ^ On (C, Table)
^ Block (A) ^ Block (B) ^ Block (C)
^ Clear (A) ^ Clear (B) ^ Clear (C))

Goal (On (A, Table) ^ On (B, A) ^ On (C, B))

Blocks World Example

Op (Action : **Move (p, x, y)**)

Precond: $\text{On}(p, x) \wedge \text{clear}(y) \wedge \text{clear}(p)$

Effect : $\text{On}(p, y) \wedge \text{clear}(x) \wedge \neg \text{clear}(y) \wedge \neg \text{On}(p, x)$

Op (Action : **Movetotable (p, x)**)

Precond: $\text{On}(p, x) \wedge \text{clear}(p)$

Effect : $\text{On}(p, \text{Table}) \wedge \text{clear}(x) \wedge \neg \text{On}(p, x)$

Progression Algorithm:

- **Initial state:** Initial state of the planning problem.
- **Actions applicable at state s :** all actions whose precondition is satisfied.
- **Goal Test:** Goal of the planning problem.
- Step cost of action is 1.

([Blocks world Example with Progression Algorithm](#))

Drawbacks of Progression Algorithm

- Does not Address the irrelevant action problem.
- High branching factor.
- Bogs down without a good heuristic.

Regression Algorithm

- Initial State: Goal state of the planning problem.
- Predecessor State Construction: Ask what are the states from which taking any given action can lead to the current state.
 - Action Considered:
 - Should be relevant.
 - Consistent- Should not undo the desired literals.

Regression Algorithm

- The produced predecessor is as follows:
 - Any positive effects of A that appear in G are deleted.
 - Each precondition literal of A is added , unless it already appears.
- Goal State: When predecessor satisfies the Initial state of planning problem.

([Blocks world Example with Regression Algorithm](#))

Regression vs. Progression Algorithm

- Lower branching factor
- Considers only relevant actions.
- Neither are efficient without a good heuristic.
 - How many actions are needed to achieve the goal?
 - Exact solution is NP hard, find a good estimate.

Two approaches to find admissible heuristic

- The optimal solution to the relaxed problem.
 - Remove all preconditions from actions
- The subgoal independence assumption:
The cost of solving a conjunction of subgoals is approximated by the sum of the costs of solving the subproblems independently.

Totally Ordered Planner

- ❑ Explores strictly linear sequences of actions directly connected to start and goal.
- ❑ E.g. Progression and Regression algorithm.
- ❑ Does not take advantage of problem decomposition.
- ❑ Always makes decision on how to sequence actions from all the sub problem.

- **Require an approach that is flexible.**

Partial Order Planning (POP)

- ❑ Takes advantage of problem decomposition.
- ❑ Works on several subgoals independently.
- ❑ Solves each subgoal with several subplans and then combines the subplans.
- ❑ Has the flexibility in the order in which it constructs the plan.

Partially Ordered Planner

- It is called partially ordered planner because the planning algorithm places actions into the plan without placing a strict order on how actions are going to be executed.
- Does a search in the space of partial order plans.

Keywords used in POP

- **Least Commitment**: strategy of delaying the choice during the search.
- **Linearization**: Placing order on the partial order solution.
- **Ordering Constraint ($A \prec B$)**: Says that action A must be executed before action B, but not necessarily immediately before.

Keywords used in POP

- Causal Link ($A \xrightarrow{P} B$): Asserts that P is the effect of action A and a precondition of action B.
- Conflicts: Any action C that can be taken after action A and before action B and also negates the P is said to be conflicting with Causal link $A \xrightarrow{P} B$.
- Open Preconditions: A precondition not achieved by some action is said to be open.

Partial Order Plan Example

Problem: Putting on a Pair of Shoes.

Goal (RightShoeOn ^ LeftShoeOn)

Init ()

Action (**RightShoe**,
PRECOND:RightSockOn,
EFFECT:RightShoeOn)

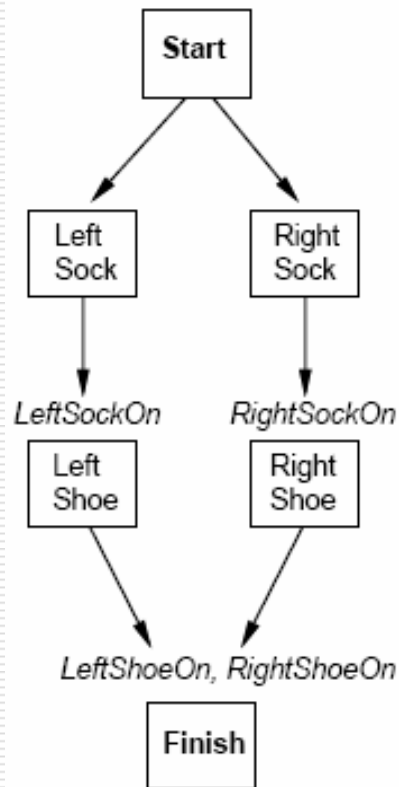
Action (**RightSock**,
EFFECT:RightSockOn)

Action (**LeftShoe**,
PRECOND:LeftSockOn,
EFFECT:LeftShoeOn)

Action (**LeftSock**,
EFFECT:LeftSockOn) .

Partial Order Plan Example

Partial-Order Plan:



Total-Order Plans:

